Ontological Fusion across Sandboxed Virtual Machines

Higher-topos semantics and conditional probabilistic integrity

Abstract. We model ontological fusion among distributed custom virtual machines as a homotopy colimit of local semantic contexts in an ∞-topos. Descent characterizes coherent provenance, and right adjoints to restriction give the largest fused predicate satisfying every local policy, even when those policies disagree. We give an exact certificate checker for a finite fragment and an abstract capability machine that separates candidate generation from authority to commit. For a fixed epoch of n validators, a certificate of size q for an inadmissible candidate requires at least q effectively compromised identities. Two conflicting certificates require at least $2q - n$. Sequential conditional bounds on domain compromise imply a Poisson-binomial upper bound, with separate terms for shared failures and sandbox violations. We also prove a sharp marginal-only bound. A three-guest QEMU implementation fuses timestamped integer vectors from two encodings. An independent oracle guest generates observations, freezes the fusers' rules, and scores their predictions against private labels. The fused rule explains 709 of 768 held-out observations in the recorded experiment.

Keywords. Ontology integration, ∞-topos, descent, capability confinement, Byzantine quorums, correlated failures.

1 Introduction

A distributed collection of virtual machines can use different vocabularies for the same entity, keep different evidence about it, and operate under different disclosure or update rules. A fusion system must specify which local records describe the same entity, which evidence for an identification survives the merge, and who may change shared state. For example, two sites may align a device identifier while authorizing different uses of its measurements. The alignment by itself supplies no authority to release either site's data.

We call the integration of semantic contexts, alignment witnesses and provenance *ontological fusion*. A context records a typed representation of claims and their dependencies. An alignment is an explicit map supplied by an authorized context publisher. Fusion concerns admissibility relative to these inputs. External facts such as whether a sensor actually measured a value require a separate premise about the source of the observation.

There are three mathematical layers. The semantic layer glues local contexts and computes the greatest predicate compatible with the local restrictions. The execution layer allows untrusted custom virtual machines to produce candidate updates and gives a precise meaning to confinement at the modeled interface. The certification layer determines how many distinct signing identities must depart from their specifications before an invalid candidate or a conflicting pair can be certified. Figure 1 shows where these layers meet.

The main cross-layer statement is a deterministic inclusion of failure events. Once that inclusion is proved, the stochastic assumptions can be changed independently of the semantic and execution models. This permits comparison of a sequential conditional model, arbitrary correlation with fixed marginals, and deployments in which one failure domain controls several signing identities. The distinction matters because more replicas can reduce some risks while leaving a shared dependency unchanged.

Section 7 instantiates the finite semantic model with timestamped vectors and continuously communicating QEMU guests. It separates agreement between the fusers from the independent evidence used to evaluate their shared rule.

2 Fusion, provenance and descent

Fix an ∞-topos $\mathcal{E}$, with all indexing diagrams small in a chosen universe. We use $\mathcal{E}_{/X}$ for the slice over X, $\mathrm{Map}_{\mathcal{E}}$ for mapping spaces, and $\mathrm{Sub}(X)$ for the poset of equivalence classes of (-1)-truncated maps into X. In the internal language, such a map is a proposition varying over X. General slice objects may have proof-relevant fibers.

Definition 2.1 (Fusion diagram)**.** Let I index sites and their specified overlaps or refinements. A fusion diagram is a functor $D\colon I \to \mathcal{E}$. Write $X_i = D(i)$, $X = \mathrm{colim}_I X_i$, and $j_i\colon X_i \to X$ for the colimit maps. An arrow $a\colon i \to k$ is part of the declared alignment and is interpreted by $D(a)\colon X_i \to X_k$. A provenance object is a map $V \to X$. Locally presented provenance consists of maps $V_i \to X_i$ with coherent equivalences $V_i \simeq D(a)^* V_k$ for the arrows of I.

An ordinary starting point is a span $X_L \leftarrow X_O \rightarrow X_R$. The overlap X_O represents those observations, aliases, or types for which the alignment supplies a correspondence. A general indexing category permits chains and cycles of alignments, with their coherences included in D. The

source can use an institution to express satisfaction under changes of signature [1]. The present semantics starts with the interpreted diagram in $\mathcal{E}$. A translation from an institutional presentation to D needs its own satisfaction-preservation argument.

Theorem 2.2 (Descent of fused provenance)**.** *For every fusion diagram,*

$$\mathcal{E}_{/X} \;\simeq\; \lim_{i \in I^{\mathrm{op}}} \mathcal{E}_{/X_i}, \tag{1}$$

where the transition functors are pullbacks. For $u\colon U \to X$ and $V \to X$, write $U_i = j_i^* U$ and $V_i = j_i^* V$. Then

$$\mathrm{Map}_{\mathcal{E}_{/X}}(U, V) \;\simeq\; \lim_{i \in I^{\mathrm{op}}} \mathrm{Map}_{\mathcal{E}_{/X_i}}(U_i, V_i). \tag{2}$$

Both equivalences are natural under base change in X.

Proof. Colimits in an ∞-topos satisfy descent [2, Proposition 6.1.3.9], which gives (1). Mapping spaces between two objects of a limit of ∞-categories are limits of the corresponding mapping spaces. Applying this fact to the restrictions of U and V proves (2). For $f\colon Y \to X$, the maps $Y_i = Y \times_X X_i \to Y$ form a colimit cocone since colimits are universal. Functoriality of pullback along the resulting squares gives the claimed naturality. □

Thus a local witness is usable globally when it participates in a coherent family. In a span, the right-hand side of (2) is a homotopy pullback of spaces. For a higher diagram it includes the higher compatibilities. Applying propositional truncation to the space of *global* witnesses expresses existence, as in homotopy type theory [3, Chapters 6–7]. Local truncations can forget precisely the transport conditions needed to form a global witness.

Example 2.3 (Local evidence with incompatible transport)**.** Work in $\mathcal{S}$. The homotopy pushout $* \amalg^h_{\{0,1\}} *$ has the homotopy type of a circle. Assign the two-element set as fiber over each vertex, and use the identity at one overlap point and the transposition at the other. This is coherent descent data for the span. Every local fiber admits a point. A global section would choose elements x and y with $y = x$ at the first overlap and $y = 1 - x$ at the second. The section space is empty. The construction realizes the nontrivial double cover of the circle. The set-valued colimit of the base span is a singleton, so it loses the loop responsible for the obstruction.

2.1 Largest admissible predicate

Let each $P_i \in \mathrm{Sub}(X_i)$ express a local restriction for a particular operation, such as publication of a reconciled record. Overlaps with no additional constraint use $P_i = \top$. For any map $j\colon A \to B$, pullback has a right adjoint $\forall_j\colon \mathrm{Sub}(A) \to \mathrm{Sub}(B)$, obtained by restricting the dependent product in the slices to subobjects. Its internal interpretation is universal quantification over the fiber of j. Define

$$R = \bigwedge_{i \in I} \forall_{j_i} P_i \quad \text{in } \mathrm{Sub}(X). \tag{3}$$

Theorem 2.4 (Admissibility and change of context)**.** *For every $Q \in \mathrm{Sub}(X)$,*

$$Q \le R \quad\Longleftrightarrow\quad j_i^* Q \le P_i \;\; \textit{for every } i. \tag{4}$$

If the P_i agree by pullback along every alignment arrow, then $j_i^* R = P_i$ for all i. For any $f\colon Y \to X$, write $f_i\colon Y_i \to X_i$ and $j_i'\colon Y_i \to Y$ for the pullback projections. Then

$$f^* R = \bigwedge_i \forall_{j_i'} f_i^* P_i. \tag{5}$$

Proof. The meet in (3) and $j_i^* \dashv \forall_{j_i}$ give (4). If the local subobjects are compatible, (1) glues them to an object $P \to X$. Its diagonal is an equivalence since each restriction is an equivalence and the projections of the limit are jointly conservative. Thus P is a subobject. Equation (4) gives $P \le R$ and $j_i^* R \le P_i = j_i^* P$. Descent makes restriction order-reflecting on subobjects, so $R \le P$. Finally, f^* preserves meets and Beck–Chevalley for dependent products in each pullback square gives $f^* \forall_{j_i} = \forall_{j_i'} f_i^*$ on subobjects. This proves (5). □

For $u\colon U \to X$, u factors through R precisely when every $U_i \to X_i$ factors through P_i. Indeed, (5) and the adjunctions identify $u^* R = \top_U$ with satisfaction of each restriction on U_i. A policy violation somewhere above a point can therefore make the fused point inadmissible. A positive provenance assertion can have different variance. For instance, $\exists_j$ is image along j on subobjects. It can express that some site attested a property. A right adjoint is appropriate for the policy that every site covering the point must permit the operation.

2.2 A finite fragment and its checker

For the effective fragment take $\mathcal{E} = \mathcal{S}$ and apply $\tau_{\le 0}$ to the colimit of a finite diagram of finite sets. Here $\tau_{\le 0}$ denotes truncation to sets in $\mathcal{S}$. Its left adjoint property gives the colimit in sets, so the fragment uses the quotient of the disjoint union by the equivalence relation generated by the alignment maps. This fragment is computed separately from the proof-relevant setting in Example 2.3.

Let Z be the quotient. Use one vertex for each pair (i, x) with $x \in X_i$, and an edge $(i, x) \to (k, D(a)x)$ for each alignment generator $a\colon i \to k$. Connectivity in the underlying undirected graph determines the classes $z \in Z$. The checker computes the class and verifies $x \in P_i$ for every representative (i, x) of z. This is the finite evaluation of (3). If the class is admissible, let D_v be the finite set of proof records permitted at vertex v by the current evidence registry. For each edge $a\colon v \to w$ let

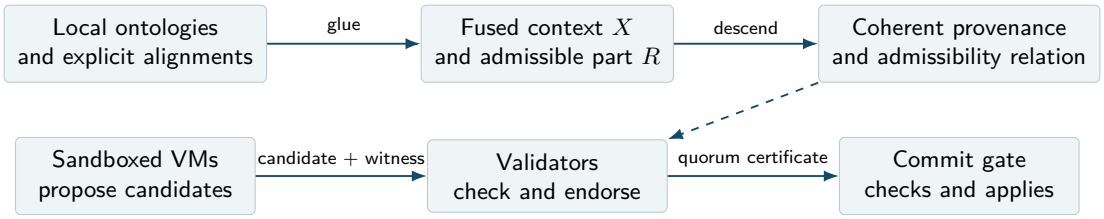


Figure 1: The semantic layer defines admissibility. The execution layer produces proposals, and the certification layer authorizes a particular state transition against a fixed policy and membership snapshot. The dashed arrow is the specification checked by validators.

$C_a \subseteq D_v \times D_w$ be a specified compatibility relation. The witness set for z is

$$W_z = \{(d_v)_{v\in z} \mid d_v \in D_v \text{ for every } v \in z, \\ (d_v, d_w) \in C_a \text{ for each } a\colon v \to w\}. \tag{6}$$

If a deployment uses relations beyond the specified edges, it lists their checks in the evidence registry. The finite provenance object is the family with fiber W_z over each $z \in R$. It yields descent data by restriction to the input sets. Conversely, if a deployment starts with coherent higher provenance, using this finite fragment requires a separate proof that its chosen finite witnesses preserve the intended validity relation.

Proposition 2.5 (Exact finite checking)**.** *Given the finite registry above, the checker accepts a purported witness exactly when its class is admissible and the supplied tuple belongs to W_z. An admissible witnessed class has an accepted encoding. Finding a witness is decidable.*

Proof. Connectivity gives the least equivalence relation containing the declared edges. A class is admissible precisely when each vertex lies in its local P_i, by the fiber interpretation of $\forall_{j_i}$. The checker uses its own enumeration of the vertices and edges in this class, rejects missing or repeated tuple indices, and checks the membership conditions in (6). These tests are necessary and sufficient. A witness in W_z encodes by listing its records in the fixed registry order. Exhausting the finite product of the D_v decides existence, including the case of an empty D_v. □

With the finite relations indexed for membership tests, graph traversal is linear in vertices plus declared edge instances. Checking a tuple adds one vertex membership test per representative and one relation test per edge. A smaller proof record must still establish all relevant restrictions, including the representatives that a candidate omitted. Exhaustive witness search can be exponential in the number of representatives.

Example 2.6 (Calibration records)**.** Two sites align aliases s_L and s_R of the same calibration instrument through a one-point overlap. Both restrict export to measured records with approved units. The left site also requires a current calibration endorsement. The class of the instrument can be exported only when all three vertices pass their declared predicates. Suppose the two sites each supply a permitted record, but the left record describes profile A and the right record describes profile B. If edge relations require identical profile identifiers, the class may satisfy R while its witness set is empty. Compatible provenance can witness the export of the class after the left restriction is met. The caller still requires the gate's explicit export permission.

3 An abstract capability machine

Each custom virtual machine runs an instruction language interpreted inside a sandbox. For the abstract machine it suffices to have pure local computation, authorized reads of immutable inputs, emission of candidate messages and halting. Languages with richer instruction sets may be used through a trace refinement to this interface. A proof for a particular bytecode semantics is distinct from a proof about the compiler, native runtime or host broker that realizes it, as illustrated by mechanized work on WebAssembly and seL4 [4, 5, 6].

Definition 3.1 (Machine interface)**.** In epoch e, a worker w has state (ℓ, c, g, Q) consisting of local store, control, fuel and outbox. The trusted broker holds a table $\mathsf{Cap}_{e,w}$ mapping handles to read scopes, with a scope specifying a snapshot and an object. Handles appearing in ordinary data have an effect only through lookup in this table. Define the modeled worker transition rules as follows.

1. A pure instruction changes only (ℓ, c).
2. A read consumes one instruction step and either returns immutable bytes within the current scope in $\mathsf{Cap}_{e,w}$ or returns a denial. The worker sees the bytes in its local store.
3. Emission appends a copied value to the worker's bounded outbox if the value belongs to the declared candidate format and fits the quota. The emission affects only the proposal queue.
4. Halt ends the worker run. An invalid instruction, a failed quota check for emission or exhausted fuel

produces a terminal trap. A terminal state has no worker step.

Each executed instruction consumes one fuel unit and receives no additional fuel within that run. The model uses bounded-size values and treats each primitive as total. A trusted renewal starts a new run with a new fuel allocation. The immutable registry, validator keys and protected commit state are outside the worker's store and outbox.

The broker issues the scopes and checks them for the executing worker and the current snapshot. Read and emission rules preserve the scope tables. Validators use their own checker and registry view. The gate is the sole transition that can update the designated protected state. In the abstract interface, a worker may supply arbitrary bytes at its designated proposal boundary. This makes checker correctness independent of the worker's claimed language or its claimed execution trace.

Lemma 3.2 (Confinement of modeled worker steps)**.** *In every trace of worker steps, all returned broker reads have current read authority, emitted values use only the worker's outbox, and the protected state and capability tables are unchanged by worker steps. A run with initial fuel g executes at most g instructions.*

Proof. The read premise checks the scope in the trusted table. The denial path returns a distinguished denial value. Pure steps modify only local state. Emission changes only the outbox after its format and quota check. Halt and trap affect only the worker. Every rule preserves the broker's table. Induction over a finite interleaving of these rules proves the invariants, with tables interpreted at the snapshot used for each step. Fuel decreases by one for each executed instruction and is a nonnegative integer. □

Proposition 3.3 (Noninterference at the worker interface)**.** *Consider a deterministic worker program in two initial states with equal local store, control, fuel and outbox. Suppose its read capabilities have equal externally visible denial and byte responses in the two runs. Under the same sequence of worker scheduling decisions, the sequences of worker emissions and terminal outcomes agree.*

Proof. Induct over the common schedule. Each scheduled pure step sees equal local state, each read receives equal responses, and each emission performs the same format and quota tests on equal values and outboxes. Each halt, trap or fuel exhaustion therefore occurs at the same scheduled step and produces the same outcome. The invariant gives equality of the stated interface traces. □

The observation function in Proposition 3.3 contains message values and terminal outcomes under a fixed schedule. An implementation that exposes execution time, contention, devices or other shared state needs additional observations and a refinement proof for them. Likewise the fuel bound limits abstract instructions. Host time and memory guarantees require a bounded-cost realization of each primitive and bounded accounting for broker and checker work.

3.1 Refinement and enforcement

Let a concrete worker trace project to abstract worker steps and authorized boundary events, with concrete internal work allowed to stutter. The simulation obligation is that every concrete step preserves the relation or implements an abstract step with the same boundary observation. The relation must account for the concrete read result, commit-state changes, authorized outbox destination and epoch context. Starting from related initial states, induction gives an abstract trace for every concrete finite trace and transfers Lemma 3.2. An implementation proof must cover initial provisioning and the worker–broker interface in addition to instruction interpretation.

Confinement and certification have different trusted components. A local interpreter or host failure can violate confinement even if every validator remains honest. A worker compromise is an effective validator compromise only if it can control the relevant signing behavior. Placing a validator and its key behind the same isolation mechanism as a worker therefore changes the failure-domain accounting. A worker with no signing authority can emit malformed proposals while preserving the certificate invariant.

Example 3.4 (Two custom worker dialects)**.** A finite straight-line language operates on a value cell using instructions to read a named handle, apply a named total function from a fixed library, emit the current value, and halt. A stack dialect represents the cell as a stack of height one. Its read instruction replaces the top element, its map instruction applies a function to that element, and its emission copies that element to the broker. A register dialect uses a named accumulator and performs these operations on the accumulator. In either dialect a denial is represented by the same distinguished value.

Relate the two states by equating the stack element, the accumulator and the abstract cell, together with the instruction position, fuel and outbox. Under this relation each read uses the same handle and receives the same broker response. The pure function changes the represented value identically. Emission and halt preserve the correspondence of boundary events, including traps when the value cannot be emitted. Induction proves a trace refinement of each dialect to the abstract worker semantics. The dialects can normalize the two local calibration records of Example 2.6 and propose evidence records for the same context. The checker must still verify

profile compatibility and the gate must still verify export permission.

4 Certified updates

4.1 Epoch, slot and validity

An epoch e has a fixed authenticated membership registry with n_e distinct signing identities and threshold q_e, where $n_e/2 < q_e \leq n_e$. The immutable snapshot S_e specifies the fusion diagram, policies, evidence rules, effect rules, and the membership registry. Each protected resource has a canonical log, with sequence numbers assigned by the commit service. The slot is $s = (d, e, r, k)$ for deployment d, epoch e, resource log r, and sequence number k. Snapshot digests, view numbers and candidate hashes are payload fields when used. They never create a second slot for the same protected transition.

A candidate m contains its slot, snapshot commitment, requested operation, principal identifier, previous-state commitment, proposed effect, fused context reference and witness. The semantic predicate $\mathsf{Valid}_e(m)$ requires that the reference lies in the operation's R and that the supplied witness verifies in the declared provenance relation. It also requires that the proposed effect is the specified interpretation of that request against the named predecessor. For the finite checker of Proposition 2.5, the context reference is an identifier for z, the witness is a tuple in W_z, and the effect rule is a trusted deterministic function of the request and state. For example, the effect can write the unique canonical export record derived from z and the checked tuple. A checker for a richer presentation must prove the implication

$$\mathsf{Check}_e(m) = 1 \implies \mathsf{Valid}_e(m). \tag{7}$$

The gate checks the caller's permission separately against the authenticated principal and operation. An alignment, a provenance proof or a signature cannot create a principal's permission. A registry update that would change permissions requires its own operation and trust rule.

All parties use the same injective canonical encoding of the typed message and a signature domain specific to this deployment and endorsement protocol. Each endorsement authenticates the complete candidate commitment, for example by signing $h(\mathrm{enc}(m))$ in that domain. The snapshot commitment identifies S_e under the same binding assumption. Distinct fields and types are part of the encoding. A single immutable message links the candidate, slot, predecessor, policy, membership and witness checked by a validator.

Definition 4.1 (Honest validator and certificate)**.** An honest validator endorses only a message in the current epoch with $\mathsf{Check}_e(m) = 1$ and an authorized slot and registry context. It durably records its unique candidate commitment per slot before releasing an endorsement. It may retransmit that endorsement. It refuses to endorse a different candidate for that slot, including after a restart. A certificate for m consists of endorsements that verify for m under at least q_e distinct identities in the epoch registry.

Durability includes concurrent requests and recovery. A correct store uses an atomic operation for the first endorsement commitment and has a recovery procedure preserving the stored choice. The verified configuration binds each identity to one authorized verification key. An implementation that permits a rollback or loss of these records can invalidate the honest-validator premise even while the key stays secret.

The trusted commit gate verifies the certificate, the canonical slot, the active snapshot, the authenticated principal and its operation scope, and the expected predecessor. It checks and applies the protected update as one atomic transition. Repeated acceptance requests for a completed slot can return the recorded result. The protected update interface routes every such transition through the gate. In this model the gate delegates the full provenance check to validators. A configuration that also repeats the entire check at the gate can impose a stronger integrity property at the cost of giving the gate more trusted code.

4.2 Deterministic fault thresholds

The adversary may submit arbitrary candidate bytes, control delivery order, withhold messages, and control the behavior of compromised identities. A signing identity is *effectively compromised* in an epoch if the adversary has control over its endorsement behavior during the period in which that epoch's endorsements are usable. A key exposure or a failure of a local implementation of (7) or Definition 4.1 is charged to the responsible identity, or to a shared failure event if it is common to the deployment. The count includes every identity effectively controlled at any time in this period. It includes capabilities carried in from previous epochs when they can still affect a current endorsement. Thus a bound on simultaneous control alone is insufficient.

Let B_e be the set of effectively compromised identities. Let G_e be the event that a designated shared premise fails, including the integrity of the shared trust roots, registry semantics, commit gate, or a common checker defect. Let K_e be the event of accepted cryptographic binding failure for this epoch. Outside $G_e \cup K_e$, an endorsement accepted under an uncompromised identity was issued by that identity for the exact candidate and snapshot under consideration.

Write I_e for the event of an invalid certificate accepted for an authorized slot, meaning that $\mathsf{Valid}_e(m)$ is false. This includes an invalid certified commit because the gate requires such a certificate. Write C_e for the existence of two cryptographically verified certificates for distinct

canonical messages in the same authorized slot under the epoch registry. This property includes certificates delivered to different readers and is stronger than observing two updates at a single correctly serialized gate.

Theorem 4.2 (Certificate thresholds)**.** *For every trace outside $G_e \cup K_e$,*

$$I_e \implies |B_e| \geq q_e, \tag{8}$$
$$C_e \implies |B_e| \geq 2q_e - n_e. \tag{9}$$

Consequently, with $k_e = 2q_e - n_e$,

$$I_e \cup C_e \subseteq G_e \cup K_e \cup \{|B_e| \geq k_e\}. \tag{10}$$

Proof. An honest endorsement of an invalid candidate contradicts (7). Every signer in its certificate therefore belongs to B_e, which proves (8). If two certificates have signer sets A and A', both have cardinality at least q_e and $|A \cup A'| \leq n_e$. Thus $|A \cap A'| \geq 2q_e - n_e$. Each identity in the intersection has endorsed distinct messages in one slot, so each belongs to B_e by Definition 4.1. The inclusion follows because $0 < k_e \leq q_e$. □

The thresholds are sharp for the abstract certificate rules when the corresponding proposals can be formed. With q_e controlled identities, their signatures alone can form a certificate for an inadmissible candidate at the signature layer. For a conflict at $k_e = 2q_e - n_e$, take two sets of q_e with exactly k_e identities in common. Those identities endorse both candidates. Each other signer can endorse its own admissible candidate once. The second construction presumes two distinct admissible messages for the same slot. A state machine with unique deterministic admissible content for every slot can have a stronger property.

Corollary 4.3 (Availability threshold and quorum choice)**.** *To meet the cardinal conditions for conflict safety against at most f compromised identities and for a certificate from only the remaining $n-f$ identities, the threshold must satisfy*

$$\frac{n+f}{2} < q \leq n - f. \tag{11}$$

For integer n, f, some integer q satisfies (11) exactly when $n \geq 3f + 1$. At $n = 3f + 1$, the choice is $q = 2f + 1$.

Proof. The intersection threshold exceeds f if $2q - n > f$. Endorsements from the remaining identities can reach q only if $q \leq n - f$. Taking the largest allowed q shows feasibility if and only if $n > 3f$. Substitution yields the boundary choice. □

The conditions describe safety and the arithmetic capacity to make progress. They provide no procedure for assembling votes when honest validators split their first endorsements. A deployment needing progress must supply a coordination protocol with an appropriate network model, such as a suitable Byzantine state-machine replication protocol [7]. Any such integration must show that the final endorsement rules still imply Theorem 4.2.

4.3 Replays and epoch changes

Binding the epoch, snapshot and predecessor to the message prevents treating an old endorsement as an endorsement of a new candidate under the binding premise. The atomic gate additionally compares the active epoch and predecessor with protected state. To transfer the result across membership changes, one sufficient design is a single trusted handoff per resource log, atomic with respect to its protected writes. The handoff closes the old epoch, records its unique committed head, and anchors the new epoch at that head. Slots continue monotonically across epochs.

With that handoff premise, induction on protected transitions gives a single committed chain across the epoch boundary. Every ordinary transition consumes the current predecessor atomically. A handoff preserves the current head, changes the active epoch exactly once for that handoff and makes the old epoch ineligible for further writes. The safety of a distributed implementation of the handoff is an additional obligation. The certificate-intersection proof alone only compares certificates under one fixed membership and slot.

5 Finite-horizon security bounds

5.1 Adversary model and accounting

Fix a finite horizon of H epochs and an adversary class $\mathcal{A}$ with specified limits on time, interaction count, and any initial privileges or information. The adversary may adapt its proposals, scheduling and requests to its observed history. Probabilities are over the experiment chosen for the deployment, including cryptographic randomness and any explicitly modeled randomized provisioning or fault conditions. All bounds below must hold for each adversary in $\mathcal{A}$ with the same claimed budget. A fixed deterministic defect can give an event probability of one under this experiment.

Let $\mathcal{F}_{e-1}$ be the information fixed at the start of epoch e. Use the conditional budget

$$\mathbb{P}(G_e \mid \mathcal{F}_{e-1}) \leq \beta_e, \tag{12}$$

where β_e is known at that start. Set $\mathbb{P}(\bigcup_{e \leq H} K_e) \leq \gamma_H$. The parameter γ_H is a computational security bound in the selected cryptographic model. For example, count the honest signing keys and bound the chance of a forgery under an uncompromised key by a multi-user signature bound, then add a bound for collisions in the commitments actually used. When invoking a single-key chosen-message bound instead, the conversion must account for the number of keys, signing-oracle queries, and adaptive exposures. Standards specify primitives [8, 9]. The deployment supplies the security assumptions and parameters for the experiment.

On the baseline event G_e^c, assume one signing identity per modeled failure domain for now. Fix a domain order

at epoch start and let $Z_{e,i}$ indicate effective compromise of identity i in the epoch. These indicators can describe correlated outcomes. Assume that for all prefix histories of positive conditional probability,

$$\mathbb{P}(Z_{e,i} = 1 \mid \mathcal{F}_{e-1},\, G_e^c,\, Z_{e,1}, \ldots, Z_{e,i-1}) \le p_{e,i}. \tag{13}$$

This is a conditional law given the initial history and the good baseline. A zero-probability baseline history needs no such law. The $p_{e,i} \in [0,1]$ are known at epoch start. The domain order reveals outcomes for the proof. It need not be the chronological order in which an adversary acts. Equation (13) controls risk even after seeing previous compromise outcomes and is stronger than a collection of marginal bounds. It is an assumption about the model after conditioning on G_e^c.

5.2 Domination and the main bound

For $\mathbf{p} = (p_1, \ldots, p_n)$, define the Poisson-binomial tail

$$T_{\mathbf{p}}(k) = \mathbb{P}\left(\sum_{i=1}^{n} Y_i \ge k\right). \tag{14}$$

Here the comparison variables Y_i are independent with $Y_i \sim \text{Bernoulli}(p_i)$.

Lemma 5.1 (Sequential domination)**.** *If binary variables $Z_1, \ldots, Z_n$ satisfy the conditional caps in (13) under a fixed starting conditional law, then*

$$\mathbb{P}\left(\sum_i Z_i \ge k\right) \le T_{\mathbf{p}}(k) \tag{15}$$

for every integer k. The same coupling bounds nonnegative weighted sums.

Proof. For a prefix $z_{<i}$ of positive probability, write $a_i(z_{<i})$ for the conditional chance that $Z_i = 1$ and assign $a_i = 0$ on null prefixes. On a fresh probability space take independent uniform U_i on $[0,1]$. Recursively set $\widehat{Z}_i = \mathbf{1}_{\{U_i < a_i(\widehat{Z}_{<i})\}}$ and $Y_i = \mathbf{1}_{\{U_i < p_i\}}$. Successive conditioning gives $(\widehat{Z}_i)_i$ the joint law of $(Z_i)_i$. The caps imply $\widehat{Z}_i \le Y_i$ for each i pointwise. Any nonnegative weighted sum preserves this inequality. Taking tails proves the claims. □

Theorem 5.2 (Conditional certified integrity)**.** *Under (12)–(13), for every adversary in $\mathcal{A}$,*

$$\mathbb{P}\left(\bigcup_{e=1}^{H} (I_e \cup C_e)\right) \le \min\left\{1,\ \gamma_H + \sum_{e=1}^{H} \mathbb{E}[\beta_e + T_{\mathbf{p}_e}(k_e)]\right\}. \tag{16}$$

For invalid certificates alone, replace k_e by q_e. Deterministic budgets remove the expectations. A deterministic upper bound valid for all adversaries also bounds the supremum over $\mathcal{A}$.

Proof. By (10), a failure in the horizon is contained in the union of $\bigcup_e K_e$, the events G_e, and the events $\{|B_e| \ge k_e\} \cap G_e^c$. Conditional on the epoch start and G_e^c, apply Lemma 5.1 to $|B_e| = \sum_i Z_{e,i}$. Multiplying the tail bound by $\mathbb{P}(G_e^c \mid \mathcal{F}_{e-1}) \le 1$ gives an upper bound of $T_{\mathbf{p}_e}(k_e)$ on the joint event given $\mathcal{F}_{e-1}$. The tower property, (12), and the union bound give (16). Equation (8) gives the invalid-only threshold. No independence across epochs, or between the cryptographic and shared failure events, is used. □

A failure budget ε for the horizon means that the right-hand side of (16) is at most ε, and the two certificate properties then hold with probability at least $1 - \varepsilon$ in that experiment. Even a small constant per-epoch failure probability $p > 0$ gives eventual failure probability approaching one in the independent repetition experiment, since the probability of no failure in H trials is $(1-p)^H$. A bound uniform in the horizon requires, for example, an appropriate summable sequence of risk bounds and a uniform cryptographic budget for the modeled experiment.

5.3 Shared failure domains

Suppose a deployment has d_e modeled domains. On G_e^c, let $Z_{e,i}$ indicate compromise of domain i, and let $w_{e,i}$ bound the number of signing identities it can control. The mapping from domains to identities must cover every effective compromise, so $|B_e| \le \sum_{i=1}^{d_e} w_{e,i} Z_{e,i}$. Overlapping domains can make this bound conservative. If (13) holds for domain indicators, (16) remains valid with

$$T_{\mathbf{p}_e, \mathbf{w}_e}(k) = \mathbb{P}\left(\sum_{i=1}^{d_e} w_{e,i} Y_{e,i} \ge k\right). \tag{17}$$

This follows by the weighted conclusion of Lemma 5.1. If one domain can control k_e signers, the comparison tail for conflicts includes that domain's own compromise probability. Deploying different interpreters under the same privileged administrator leaves that authority in the same modeled domain or in G_e. Experimental work on multiversion programming also shows why distinct implementations alone do not justify independent failure rates [10].

5.4 What marginal probabilities imply

Proposition 5.3 (Marginal-only bound and sharpness)**.** *Under any fixed conditional law, if $\mathbb{P}(Z_i = 1) \le p_i$ and $1 \le k \le n$, then*

$$\mathbb{P}\left(\sum_i Z_i \ge k\right) \le \min\left(1, \frac{\sum_i p_i}{k}\right). \tag{18}$$

For a common cap $p_i = p$, this is sharp for every $p \in [0,1]$ and every such k.

Proof. On the tail event, $\sum_i Z_i \geq k$, so $\mathbb{E}[\sum_i Z_i] \geq k\mathbb{P}(\sum_i Z_i \geq k)$. This proves the bound. For sharpness, put $r = \min(1, np/k)$. With probability r, choose a uniformly random k-element subset of the identities and set its indicators to one. With probability $1-r$, set all indicators to zero. Each marginal is $rk/n \leq p$, and the tail probability equals r. □

Using (18) in place of the Poisson-binomial tail gives a version of Theorem 5.2 for marginal information alone, conditional on the epoch start and G_e^c. A simple common-cause case sets $Z_i = Z$ for all i, with Z Bernoulli of parameter p. The chance of crossing any positive threshold up to n is then p. Conditioning on one observed compromise makes the next indicator equal one with certainty. This is precisely the situation that the stronger cap in (13) excludes for $p < 1$.

5.5 Quorum families and compromise cut sets

Some deployments require a fixed mix of validators from different administrative groups. Let $\mathcal{Q}$ be a nonempty family of allowed signer sets in a finite registry. A certificate must contain some member of $\mathcal{Q}$. The deterministic argument of Theorem 4.2 says that an invalid certificate requires the compromised set to contain a quorum. A conflict requires it to contain an intersection $A \cap A'$ for $A, A' \in \mathcal{Q}$. These are upward-closed conditions on the compromised set. For an upward-closed condition with inclusion-minimal sets $\mathcal{M}$, the coupling of Lemma 5.1 bounds its probability by the probability of that condition for the independent comparison set $\{i : Y_i = 1\}$. In particular, a union bound gives the estimate

$$\min\left(1, \sum_{M \in \mathcal{M}} \prod_{i \in M} p_i\right). \tag{19}$$

For failure domains controlling several identities, first take the inverse image of the upward-closed condition along the monotone map sending compromised domains to the union of their affected identity sets. Then use the minimal domain sets in (19). With two groups of three validators each and a quorum requiring two from each group, every quorum intersection contains a member of each group. If a single local domain controls all signers of its group, a conflict on the good baseline requires compromising both domains, even though either group alone has three signers. The generalized condition retains this group structure.

5.6 Sandbox violations and the joint guarantee

Let J_e be the event of a modeled worker boundary violation in epoch e, such as an unauthorized broker read, a write outside the designated outbox or an unauthorized protected-state write by a worker. Let $L_{e,w}$ denote failure of the worker-specific simulation premise of Section 3. Include shared failures of that premise, such as a shared isolation layer failure, in G_e. Include the applicable cryptographic binding failures in K_e if the broker uses them. Lemma 3.2 and trace refinement give

$$J_e \subseteq G_e \cup K_e \cup \bigcup_w L_{e,w}. \tag{20}$$

Suppose the worker roster is fixed at epoch start and $\mathbb{P}(L_{e,w} \mid \mathcal{F}_{e-1}, G_e^c) \leq \delta_{e,w}$ for each worker. For a dynamic roster, the same statement can be applied to a fixed budget of run slots with unused slots contributing zero. Applying the union bound directly to (10) and (20) yields

$$\mathbb{P}\left(\bigcup_{e=1}^{H} (I_e \cup C_e \cup J_e)\right) \leq \min\left\{1,\ \gamma_H + \sum_{e=1}^{H} \mathbb{E}\left[\beta_e + T_{\mathbf{p}_e}(k_e) + \sum_w \delta_{e,w}\right]\right\}. \tag{21}$$

For a worker-only guarantee, omit the quorum tail. Equation (21) counts the common event budget once per epoch for the two protections. A single local defect can violate a worker boundary, so this term generally has a first-order union bound even when the quorum tail is small. The observation model from Section 3 determines which boundary violations the guarantee covers.

6 Numerical results and evidence

6.1 Exact evaluation of the comparison tail

The polynomial

$$F(z) = \prod_{i=1}^{d} (1 - p_i + p_i z^{w_i}) \tag{22}$$

has coefficient $[z^j]F(z) = \mathbb{P}(\sum_i w_i Y_i = j)$. This follows from multiplying the generating functions of the independent comparison variables. One can initialize $a_0(0) = 1$ and $a_0(j) = 0$ otherwise, and use the recurrence

$$a_i(j) = (1 - p_i)a_{i-1}(j) + p_i a_{i-1}(j - w_i) \tag{23}$$

with negative indices set to zero. The desired tail is $\sum_{j \geq k} a_d(j)$. An implementation interested only in the tail can merge indices $j \geq k$ into one absorbing bin. Unit weights give the ordinary Poisson-binomial distribution, and equal caps reduce it to the binomial.

Table 1 sets $n = 3f + 1$ and $q = 2f + 1$ for several f. For $n = 10, q = 7$, the conflict threshold is four and the invalid-certificate threshold is seven. Suppose the same ten-domain cap applies in each of $H = 100$ epochs and

n	q	$2q-n$	Conflict tail	Invalid tail
4	3	2	5.92030×10^{-4}	3.97000×10^{-6}
7	5	3	3.39625×10^{-5}	2.06515×10^{-9}
10	7	4	2.00128×10^{-6}	1.16878×10^{-12}
13	9	5	1.20373×10^{-7}	6.89609×10^{-16}
16	11	6	7.34800×10^{-9}	4.17146×10^{-19}

Table 1: One-epoch comparison tails for unit weights and hypothetical sequential caps $p_i = 0.01$. Shared and cryptographic terms are additional. Values use the exact binomial distribution and are rounded here.

the shared term is $\beta_e = 10^{-5}$. Equation (16) gives an upper bound of approximately $1.20013 \times 10^{-3} + \gamma_{100}$ on the chance of any invalid or conflicting certificate. This illustrative calculation uses stipulated budgets.

With twenty workers and a stipulated $\delta_{e,w} = 10^{-6}$ per worker per epoch, (21) adds 2.0×10^{-3} to that horizon bound. If only the marginal signer caps $p_i = 0.01$ are supported, Proposition 5.3 gives a sharp worst-case one-epoch probability of 0.025 for reaching the conflict threshold under those caps. With one domain controlling four of the ten identities and six domains each controlling one, the comparison tail becomes

$$0.01 + 0.99\,\mathbb{P}\big(\mathrm{Bin}(6, 0.01) \geq 4\big), \tag{24}$$

which is approximately 0.01000015. This is the conditional comparison with a cap of 0.01 for each domain, including the domain of weight four. Both alternatives change the bound by orders of magnitude without changing the nominal number of signing identities.

6.2 Deployment evidence

The system claim has three proof and evidence interfaces.

1. A semantic soundness argument gives a versioned ontology diagram, a specified validation relation, and a checker proof such as Proposition 2.5. Examples should exercise an incompatible overlap, a missing witness and a valid coherent fusion.

2. A refinement argument identifies the exact worker runtime, broker, key custody, validator lock, membership snapshot and commit gate. In particular, validators need a common interpretation of the signed bytes and a durable first endorsement. End-to-end refinement has to account for initialization and recovery paths.

3. The deployment must justify β_e, γ_H, the conditional caps or marginal caps, and any local boundary budgets against the declared adversary class. The domain map records common roots such as shared privileged services, code and operational control. A demonstrated deployment-independent proof may eliminate a term for the modeled component. A quantitative assumption must otherwise remain visible in the bound.

For scale, suppose a testing distribution produces N independent, identically distributed trials with no failures. An exact one-sided $(1-\alpha)$ binomial upper limit is $1 - \alpha^{1/N}$, obtained by solving $(1-p)^N = \alpha$ [11]. For an upper limit of 10^{-6} at 95% confidence, this requires at least 2,995,731 such trials. This calculation concerns the failure probability under the trial distribution. A uniform bound on adaptive adversaries or on every compromise prefix in (13) requires a separate argument connecting the experiment to that guarantee.

The principal exclusions are properties with distinct specifications. Availability depends on network and coordination assumptions. Timing and other channels depend on the observation model. Truth of external attestations depends on the sources of the data. Privacy of a witness disclosed to the validator set depends on additional access and disclosure rules. Within the modeled interface, a safety guarantee applies to protected transitions and to the particular boundary events in (20).

7 Integer streams in three QEMU guests

The experiment concerns two streams of abstract integer vectors. One QEMU guest generates both streams and evaluates the predictions. Two separate QEMU guests receive one stream apiece and exchange observations at every logical tick. Each fuser independently reconstructs the same ontology and selects a rule from the same declared class. The generator and evaluator is called the *oracle*. It retains the held-out labels while the fusers communicate, and its score is independently recomputed from the transcripts.

7.1 Typed observations and the fusion carrier

Let $\mathbb{B} = \{0, 1\}$ be the discrete Boolean type. A site i emits occurrences (i, t, a) with $t \in \mathbb{N}$ and $a \in A_i$, where A_i is a 0-truncated observation type. The tick is a logical timestamp. The cumulative record at t is a finite set of occurrences, so the prefix operation is a functor from the ordered set of ticks to the poset of finite evidence sets. This presentation applies to any homotopically discrete A_i with the needed alignment maps. The implementation uses $A_L = A_R = \mathbb{Z}^4$, represented by four signed 32-bit integers.

The shared typed context is $D = \mathbb{B} \times \mathbb{Z} \times \mathbb{B}$, with coordinates (g, x, y). Here g is a binary feature, x an

integer feature and y the observed label. The declared embeddings of D into the left and right languages are

$$b_L(g, x, y) = (1, g, x, y), \tag{25}$$
$$b_R(g, x, y) = (7, 2x + 7, y, 1 - g). \tag{26}$$

On the recognized tags, the right decoder reverses the permutation and solves $x = (u - 7)/2$ for its second raw coordinate u. Integer parity and bit constraints determine the domains of the decoders. In the implementation only values representable in the source encoding are emitted. The oracle also generates records with unfamiliar tags 9 and 99. They remain in the evidence set under their original source and tick.

The fused semantic context is the homotopy pushout $X = A_L \amalg^h_D A_R$ in spaces. Because both maps are injections of discrete types, each component of this pushout consists of a single unglued vertex or of two vertices joined by one edge. Every component is contractible. Thus X is equivalent to the ordinary set pushout. It has one copy of D and the two source-tagged complements. The provenance key (i, t) is retained separately from the quotient. Two occurrences can have the same semantic value while still contributing two distinct pieces of evidence.

Proposition 7.1 (Replica agreement at a complete prefix)**.** *Suppose both fusers have the same versioned alignment, candidate rule ordering, and training cut. If both have received all observations through a common tick with unique payloads per provenance key, the two local unions give the same evidence set. The fusers therefore derive the same ontology at that cut. Exact duplicate deliveries have no effect.*

Proof. The first assertion follows because both replicas contain the union of the same two finite source prefixes. Set union is associative, commutative and idempotent, as in the grow-only replicated set [17]. The quotient interpretation is a deterministic function of the source-tagged raw values, and the rule selection below is a deterministic function of the resulting training multiset of occurrences. Determinism gives the same result at both replicas. Adding an already present occurrence leaves that multiset unchanged. The implementation rejects unequal payloads for the same key. □

7.2 Ontology induction and identification

An exported ontology specifies the two embeddings, the typed carrier, the residual records and a relation that predicts y from (g, x). The embeddings are supplied as part of the schema. The predictive relation is learned. Define $H_\theta(x) = \mathbf{1}_{\{x \geq \theta\}}$ for $\theta \in \{-8, -4, \ldots, 24\}$. The ordered hypothesis class contains the constants $0, 1$, the projection g, and three ordered families H_θ, $g \wedge H_\theta$, and $g \vee H_\theta$. Thresholds within a family are ordered numerically. There are thirty rules. The order provides a reproducible tie break, with the single-feature rules ahead of the two-feature rules.

For a set of participating sources S, let T_S be the successfully decoded training occurrences from those sources. The chosen rule is the lexicographic minimizer of

$$\left(\sum_{(g,x,y) \in T_S} \mathbf{1}_{\{h(g,x) \neq y\}}, \operatorname{rank}(h) \right). \tag{27}$$

The sum is indexed by occurrences, including repetitions of the same decoded value at different keys. A selected rule determines the subobject $R_h = \{(g, x, y) \in D \mid y = h(g, x)\}$ of the fused carrier. Pulling R_h back along either local inclusion gives the records of that language that the same rule explains. The experiment scores the fused fit and both single-source fits on the same complete holdout. An observation is explained when the rule predicts its label, contradicted when it predicts the other label, and residual when no decoder applies. The explanation rate divides matches by the total raw holdout count, including residuals.

The oracle draws a run-level threshold $\theta_* \in \{4, 8, 12\}$. The left stream has $g = 1$ and cycles x through $-8, -4, \ldots, 20$. In the right stream g alternates after each group of four ticks and x cycles through $12, 16, 20, 24$. The noiseless label is $g \wedge H_{\theta_*}(x)$. The oracle flips some labels and changes some tags using separate pseudorandom draws, with frequencies close to $1/19$ and $1/29$ in the generator. Label perturbations and residual tags remain part of the evidence. They can lower explanatory coverage.

Proposition 7.2 (Complementary support)**.** *Assume that the noiseless training data contain a full cycle from each source. The fused fit uniquely identifies $g \wedge H_{\theta_*}$. The left-only fit selects H_{θ_*}, and the right-only fit selects g.*

Proof. On the left support, $g = 1$ and the full grid contains θ_* and $\theta_* - 4$. An exact threshold on that grid must be θ_*. Both H_{θ_*} and its conjunction with g are exact. The constants, the gate alone and the disjunctions cannot express the left transition. The tie break selects H_{θ_*}. Every right signal satisfies $x \geq \theta_*$ and both values of g occur at each right signal. The gate alone predicts this stream exactly and ranks ahead of the other exact rules. A bare threshold fails at the same right signal with different gates. The only rule exact on both supports is consequently the conjunction with threshold θ_*. □

The same mechanism has a finite perturbation condition. On the decoded training rows, let $d(h, h_*)$ count disagreements between the predictions of an alternative h and the ideal rule h_*. If at most r of those labels are flipped and $d(h, h_*) > 2r$ for all alternatives, then h_* remains the unique error minimizer. Its observed error is at most r, while any alternative makes at least $d(h, h_*) - r > r$ errors. Residual rows can affect this condition by removing separating examples.

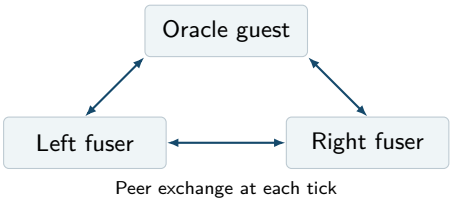


Figure 2: The oracle supplies observations and receives checked predictions over two separate serial links. A third link directly connects the fusers.

7.3 Independent evaluation with sealed labels

In each training window the oracle discloses complete local vectors to the corresponding fuser. Each fuser sends the observation to its peer. The oracle independently fits all three models on the original labeled prefix. At the training cut it accepts a declaration of the three rule identifiers from each fuser only if both declarations equal the oracle's computed identifiers. It stores this result before releasing any held-out features.

For each later tick the oracle retains two full vectors and sends each fuser its corresponding vector with the label coordinate replaced by -1. This sentinel is outside $\mathbb{B}$. The fusers exchange these masked vectors and return predictions from the frozen rules for their own input. The oracle checks each prediction against the frozen rule on its own disclosed features and scores it against the original held-out value. An unknown tag requires abstention and counts as residual. The oracle emits original records to its separate host console only after checking every prediction in the window. The fusers receive neither those records nor a score feedback channel.

Theorem 7.3 (Integrity of the reported score)**.** *Assume the oracle and its isolation are correct, the holdout is generated by the declared source, and the two fusers have no access to the oracle's private state. The fusers may coordinate their messages arbitrarily. For every accepted window, the reported score equals the evaluation of the training-prefix minimizer on the complete oracle holdout. A substituted prediction that differs from that rule cannot affect an accepted score.*

Proof. The oracle fixes the rules by (27) using the labeled training prefix and accepts both declarations before transitioning to the holdout phase. The later state contains the fixed identifiers, and the protocol expects one prediction per source at each known held-out tick. It compares each prediction with the frozen rule's value or the declared abstention. The counters update only for the current complete observation as held by the oracle. Induction on the held-out ticks gives the exact count for each outcome. Failure to supply a matching prediction or to complete the window prevents that window's report. □

Fit	Explained	Contrad.	Resid.	Rate
Fused	709	30	29	92.32%
Left only	545	194	29	70.96%
Right only	534	205	29	69.53%

Table 2: Oracle evaluation over the same 768 held-out vectors for all three fits. The denominator includes residuals.

In the reference run the host passes a fresh 256-bit seed only to the oracle guest. The oracle expands it with the ChaCha20 block function [20]. The seed and original labels are outside the fuser inputs. The completed run records the seed for repeatability. A replay using that disclosed seed is a reproduction of a completed experiment. The evaluation theorem depends on the trusted host and oracle, and the two fusers can still coordinate on all information released to them. Faults that stop a guest can prevent a report.

7.4 Implementation and recorded run

The implementation boots two freestanding 32-bit x86 fusers and one freestanding oracle under QEMU's system emulator. Each guest has 32 MiB of assigned RAM and boots a Multiboot ELF image [18, 19]. The fusers share the fuser image, with the source selected at boot. Only the oracle image contains the observation generator. There are no guest NICs or shared guest disks. Each link in Figure 2 is a pair of host FIFO pipes connected to two QEMU serial backends. The host creates the links, releases a startup barrier, and captures the three consoles. Observation generation and rule selection execute inside the guests.

The serial wire format uses fixed-size 32-byte frames with a type, source, tick, four signed coordinates and a CRC. The replica log indexes occurrences by source and tick. It treats scheduled identical replays as idempotent and rejects a differing replay. Round synchronization bounds outstanding work, and each finite window yields one immutable verified result. The current ontology is exported atomically as a JSON object after the host auditor checks the three guest records. The full reports retain their timestamp range. A continuous mode repeats the protocol on successive windows until interrupted, with one active ontology at each verified cut.

The independent host auditor reconstructs the training prefix from oracle records, recomputes the three fits, and checks equality of both fusers' training and masked holdout transcripts. It recomputes the predictions and scores from the recorded full vectors and verifies the fusers' sealed summaries. The serial CRC and the rolling prefix fingerprint check framing and consistency. The score proof relies on the oracle's own records and the private topology.

The recorded run used QEMU 10.0.13 with TCG. It completed three windows of 128 training and 128 held-out ticks each, with two observations per tick. Each fuser received 768 distinct peer observations and processed 45 replays. Both guests selected $g \wedge H_8$ in every window. The left-only fit selected H_8 and the right-only fit selected g. The fused model explained 358/384 left records and 351/384 right records. Table 2 gives the combined counts. In a repeat run with the recorded seed, the three console transcripts and the result files matched byte for byte. The source bundle includes the program, the guest images, the transcripts and the independent audit procedure.

A separate validation run used modified copies of both fusers that substituted a different prediction at the first held-out tick. The oracle rejected the prediction and no result for that window was exported. A continuous run was also stopped after two verified windows. The saved prefix could be audited from the three consoles.

This experiment isolates learning under an explicit ontology and a known hypothesis class. It measures synthetic held-out coverage and the behavior of the evaluation protocol. Applying the architecture to other discrete observation objects requires a decoder, alignments and an explanatory hypothesis class for those objects. Establishing operational sandbox integrity requires the implementation refinement and risk evidence of Sections 3 and 5.

8 Relation to prior work

Goguen and Burstall's institutions give an abstract satisfaction condition for changing signatures [1]. Goguen applies colimits to the integration of heterogeneous information and sheaves to concurrent systems [12, 13]. Functorial data migration likewise uses the adjoints of restriction to express migration and query operations [14]. The present construction uses ∞-topos descent to preserve higher provenance across a gluing diagram and the right adjoint on subobjects to retain every local restriction. In particular, (3) remains defined for incompatible policy assignments, while (2) detects when local evidence has no global section.

Complete mediation and least privilege motivate the broker boundary [15]. Proof-carrying code illustrates how untrusted producers can supply evidence for a trusted checker [16]. Here the evidence establishes admissibility of a proposed state transition. Worker confinement is a separate refinement property. The deterministic intersection argument has the standard Byzantine quorum structure [7]. Its role here is to transfer the semantic checker condition to a lower bound on the number of effectively compromised endorsers required for a specified failure. This makes the probabilistic input replaceable and exposes the effect of shared failure domains.

9 Conclusion

A fused ontology admits a largest subobject satisfying all local policy restrictions, and an admissible update still needs coherent provenance. A capability sandbox can let custom virtual machines propose such updates without giving them commit authority. Under sound checking and authenticated endorsement, a quorum converts the semantic condition into explicit fault thresholds. Those thresholds give a finite-horizon risk bound only when the deployment supplies defensible conditional, shared and cryptographic budgets. The sandbox boundary has its own risk bound, because it can fail before any quorum is affected.

In the integer-stream experiment, the two fusers recover a common rule because their observations supply complementary constraints. Private oracle scoring keeps model agreement and explanatory coverage distinct. The empirical counts characterize the scripted workload, while the probability theorem concerns deployments satisfying its stated fault and refinement premises.

Evidence and disclosure. This manuscript supplies mathematical proofs, exact numerical formulas and a reproducible synthetic QEMU experiment. The source bundle includes guest programs and images, the independent transcript checker and the recorded evidence. The manuscript and implementation were prepared with assistance from OpenAI ChatGPT. The submitting researcher or group supplies the responsible byline.

References

[1] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1), 95–146, 1992. DOI: 10.1145/147508.147524.

[2] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies 170, Princeton University Press, 2009. Author's text.

[3] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. Open book and source.

[4] C. Watt. Mechanising and verifying the WebAssembly specification. *Proceedings of Certified Programs and Proofs (CPP)*, 53–65, 2018. Author's paper.

[5] T. Murray, D. Matichuk, M. Brassil, P. Gammie, T. Bourke, S. Seefried, C. Lewis, X. Gao, and G. Klein. seL4: From general purpose to a proof of information flow enforcement. *IEEE Symposium on Security and Privacy*, 415–429, 2013. Open paper.

[6] seL4 Foundation. What the proofs assume. Verification documentation.

[7] M. Castro and B. Liskov. Practical Byzantine fault tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, 173–186, 1999. USENIX publication.

[8] National Institute of Standards and Technology. *Digital Signature Standard*. FIPS 186-5, 2023. NIST publication.

[9] National Institute of Standards and Technology. *Secure Hash Standard*. FIPS 180-4, 2015. NIST publication.

[10] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1), 96–109, 1986. DOI: 10.1109/TSE.1986.6312924.

[11] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4), 404–413, 1934. DOI: 10.1093/biomet/26.4.404.

[12] J. A. Goguen. Information integration in institutions. Author manuscript. UC San Diego copy.

[13] J. A. Goguen. Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, 2(2), 159–191, 1992. Cambridge University Press.

[14] D. I. Spivak. Functorial data migration. *Information and Computation*, 217, 31–51, 2012. Author's preprint.

[15] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), 1278–1308, 1975. Open paper.

[16] G. C. Necula. Proof-carrying code. *Proceedings of the 24th ACM Symposium on Principles of Programming Languages (POPL)*, 106–119, 1997. Open paper.

[17] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. *A Comprehensive Study of Convergent and Commutative Replicated Data Types*. Inria Research Report RR-7506, 2011. Open report.

[18] QEMU Project. System emulator invocation, including the two-way pipe character device. QEMU documentation.

[19] GNU Project. *Multiboot Specification*, version 0.6.96. Specification.

[20] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*. RFC 8439, 2018. RFC Editor.