PROVENONCE · BITTENSOR SUBNET 87

Proof of Assurance

An open Bittensor market for evidence-backed assurance of autonomous work

Will O’Brien, Founder & CEO, Provenonce

Chris Zacharia, Founder, bitstarter.ai — Contributor and Protocol Advisor

21 September 2026

v0.7.2 · Final Candidate

FIND WHAT EXECUTION MISSED. PROVE IT. IMPROVE THE NEXT RUN.

> *“To do right and live honourably and be just … or to do wrong and be unjust.”*
>
> PLATO · REPUBLIC IV, 445A–B · TRANS. DESMOND LEE

> *Quis custodiet ipsos custodes?*
> *Who will guard the guardians themselves?*
>
> JUVENAL · SATIRE VI.347–48

Reader’s Contract

Proof of Assurance sets out an open market for independently assuring autonomous work. Proposed by **Provenonce, Inc.**, the subnet is named **Provenonce** and identified on Bittensor as **Subnet 87 (SN87)**. The paper is organized in three parts:

- **Part I — Thesis** sets out the case for an assurance market.
- **Part II — Subnet Design and Operation** defines the protocol and its mechanics.
- **Part III — Adoption, Economics, and Evaluation** examines how the proposal could be adopted and the experiments by which it should be judged.

Technical appendices formalize the core objects and weight-setting path.

Unless explicitly identified as implemented or tested, the mechanisms in this paper are proposals. Their economic and performance outcomes remain hypotheses to be evaluated.

Problem

Advances in models, tools, memory, and orchestration are expanding the business-critical work entrusted to autonomous systems. Those systems may operate through frontier models, enterprise platforms, homegrown agents, or purpose-built operating systems. Their work leaves behind outputs and supporting material: traces, tool calls, decisions, artifacts, policies, approvals, outcomes, and omissions. Without a shared ontology and grammar, determining what happened, what governed it, and what should be compared can become an unbounded computational and interpretive effort. **The assurance gap** is the distance between work a system can produce and work an institution can independently trust.

Premise and hypothesis

Premise. Consequential autonomous work should be open to independent, repeatable assurance beyond the producing system’s self-attestation.

Hypothesis. An open contest among assurance methods can find material deficiencies that locked centralized baselines miss, under comparable evidence and resource budgets and at a cost the added assurance justifies. Part I develops the argument; Part III sets out how to test it.

Proposed system

The proposal combines a shared ontology and grammar for describing work, an Assurance Capsule for presenting permitted evidence, a Bittensor-native contest among assurance methods, and an Assurance Differential for returning what the evidence supports. Part I explains why these elements belong together; Part II specifies their design and boundaries. Bittensor interfaces and parameters can change. The planned **SN87 Technical Specification** and accompanying reference implementation, once published, will be the authoritative source for release-specific implementation detail; this paper does not claim that either is published or complete.

Questions and invitation

The proposed first experiment is deliberately narrow: a deterministic challenge family with self-verifying and reference-execution truth, compared with locked centralized baselines under criteria to be preregistered before testing. Broader claims require separate evidence.

The proposal remains accountable to empirical questions: whether hidden challenge supply remains credible; whether privacy controls preserve useful signal; whether independent methods discover material deficiencies beyond centralized baselines; whether scoring survives Goodhart pressure; and whether the value of avoided loss and improved decisions exceeds compute, latency, oracle, privacy, and coordination cost.

This paper makes no launch, token-price, revenue, performance, or universal-assurance promise. It makes the open questions concrete and invites the Bittensor community to validate, improve, challenge, and—where there is conviction—build the workstreams with us.

Abstract

Autonomous systems can produce increasingly capable work, yet the systems that generate the work are usually also asked to explain, evaluate, and defend it. Their outputs and supporting evidence may show what happened without establishing whether the right governed response was selected, the executed path respected policy and context, the evidence supports the finding, or a materially better continuation was missed. This is the assurance gap: intelligence creates the capacity to act; assurance tests whether a bounded execution deserves reliance. As autonomous work compounds, so does the cost of leaving its execution unreviewed.

Before an open market can reward assurance, autonomous work must first be made legible enough to challenge. Provenonce therefore proposes a **shared ontology and grammar of work**: the signal that required a response, the governing context, the constraints, the path actually taken, the execution trace, the evidence, and the policy boundary. From that grammar, a participating system may publish the minimum permitted account as an **Assurance Capsule**. The boundary is intended to remain open and provider-independent, with mapping loss and disclosure made explicit.

Provenonce proposes a Bittensor-native market around that common boundary. Validators issue hidden, class-specific challenges. Competing miners return an **Assurance Differential** declaring findings, no material deviation, or insufficient evidence/abstention; evidence and counterfactual paths are conditional. Validators score responses against declared oracle classes and submit Bittensor weights. The design asks whether open participation, plural methods, and programmable incentives can make independent assurance improve through contest.

PART I

Thesis

The Assurance Gap

Intelligence is not assurance

The current AI market is rapidly improving the intelligence available to an individual task. Models write, search, classify, code, plan, call tools, and operate interfaces. What remains unresolved is the **institution behind the action**: the agreement that authorized it, the customer context that made one branch valid, the policy that required review, the prior evidence that changed the decision, and the accountability that persists after the model response disappears.

That gap is easy to underestimate because the visible action is often simple. A click, a redline, a price, a deployment, a claim decision, or a customer email may be generated in seconds. Yet the correctness of that action may depend on years of accumulated institutional competence. Intelligence is not the same as dexterity: a model may be capable in the abstract and still fail to apply the institutional context behind the action.

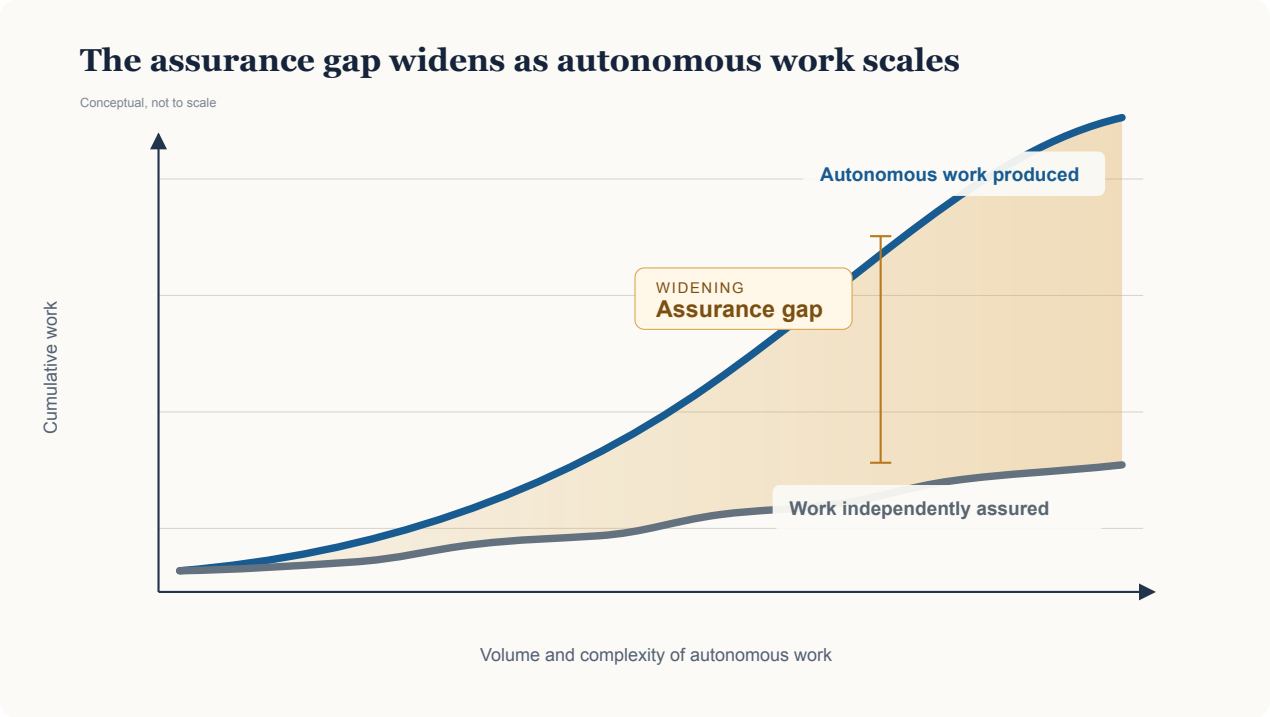
Observability helps. OpenTelemetry provides vendor-neutral conventions for traces, metrics, logs, and increasingly GenAI operations (OpenTelemetry Authors 2025, n.d.). Evaluation platforms can capture production traces, run deterministic scorers, use model judges, and incorporate human feedback (Braintrust n.d.-a, n.d.-b, n.d.-c). NIST's AI RMF organizes governance and risk work around Govern, Map, Measure, and Manage, and its GenAI profile supports more systematic testing, evaluation, verification, and validation (Tabassi 2023; Autio et al. 2024). These are real and useful capabilities.

They are not the same thing as an independent assurance market.

A trace says what the instrumentation recorded. A scorer says what a configured evaluator concluded. A model judge may say what another model found persuasive. A provider guardrail says what the provider chose to block. An internal audit says what the organization could reconstruct after the fact. None of these, by itself, creates a sustained contest among independent methods to discover what the execution missed, attach proof, calibrate uncertainty, and improve against hidden challenges over time.

Figure 1. The Assurance Gap

AI systems can produce work faster than institutions can independently trust it.



As the volume and complexity of autonomous work increase, work produced may outpace the work institutions can independently assure. The widening distance between the curves is the assurance gap. Conceptual, not to scale.

The Assurance Proposition

SN87 in one sentence

> **SN87 rewards miners for finding what execution missed – and proving it.**

The sentence contains four constraints:

1. **Finding** implies competition among methods, not rote verification alone.
2. **What execution missed** implies comparison against a declared space of requirements or alternatives.
3. **Proving** requires evidence, not persuasive language.
4. **Rewards** require a validator measurement function that Bittensor can turn into weights.

What SN87 is not

SN87 is not:

- a general-purpose workflow engine;
- a replacement for an enterprise platform, model provider, or homegrown loop;

- a general certification of an agent, model, organization, or collective;
- a certification of an entire work product or every semantic property of its content;
- a decentralized repository for enterprise secrets;
- a universal truth oracle;
- a promise that many miners automatically produce correctness;
- a token wrapper around a single provider's internal quality score; or
- a requirement that the customer's live business process depend on subnet availability.

Content is not excluded categorically. Content-sensitive challenges are admissible only when the disclosure mode permits them and the declared oracle makes the judgment defensible.

> **An axiom emerging from this work:** *Each node and every collective of nodes in a distributed system is self-sovereign.*

Provenonce proposes this axiom and applies it to two distinct domains. Originating systems retain authority over their work and evidence. Participating validators make their own evaluations, while the subnet collective applies its agreed consensus rules to the weights used for network incentives. Consensus can limit a weight's influence without transferring control of the originating work. For SN87, the division of responsibility is:

- The originating system performs the work and retains its source objects and evidence.
- The work owner decides whether and when to invoke external assurance and what the Capsule may disclose.
- Once invoked, SN87 is the independent assurance layer for that bounded execution: it issues the challenge and returns the Differential.
- The default integration may be asynchronous or in shadow, so subnet latency, unavailability, or abstention does not take the original work or evidence away from its owner.

Together, these boundaries separate voluntary submission from independent judgment: the work owner chooses whether to invoke SN87; once invoked, assurance is not a self-attestation by the originating system.

Why Bittensor

A precedent for open competition

Bitcoin showed that a decentralized network could coordinate digital value without a central clearing authority. Ethereum generalized that coordination into a programmable execution environment. Bittensor applies related market logic to machine intelligence: subnets define utility, miners compete to produce it, and validators translate local evaluation into weights (Nakamoto 2008; Buterin 2014; Rao n.d.). These systems are not interchangeable, and none proves that decentralized assurance will work. They do establish a useful historical pattern: when a scarce digital function can be specified, measured, and rewarded, open networks can invite more builders, more methods, and more adaptive competition than a single institution can direct.

Proof of Assurance asks whether that pattern can be extended one step further—from producing intelligent work to contesting whether a particular execution deserves reliance. The bet is not that decentralization makes judgment infallible. It is that a well-designed market can make disagreement productive, reward evidence over confidence, and let stronger assurance methods emerge in public competition.

The narrow reason

Bittensor supplies a protocol-native architecture for repeated competition among miners, validator ranking, stake-weighted consensus, and emissions. A subnet owns its request/response semantics and the implementation needed to participate; Bittensor supplies the network, identity, consensus, and incentive substrate (Bittensor n.d.-b, n.d.-d). The historical argument for peer-ranked intelligence markets appears in Rao's whitepaper; Timo develops the case for open, programmable incentive competition (Rao n.d.; Timo 2024). Neither source establishes SN87's performance.

That is the narrow reason SN87 belongs on Bittensor: the assurance function should improve through an open competition among methods rather than remain a permanently centralized evaluator.

The broader philosophical reason is risk concentration. If agentic work becomes institutional infrastructure, relying on one model provider, one orchestration vendor, or one internal scoring function concentrates the authority to declare that work trustworthy. Decentralization is not automatically correct, but it creates the possibility of geopolitical, methodological, and economic diversity. The market can reward different analytical approaches, different models, different toolchains, and different forms of proof.

The virtues of the contest

Independent assurance could become reusable infrastructure for autonomous work: a way to challenge execution without making every work owner build a closed evaluator. This is a proposed role, not a demonstrated outcome. It depends on useful findings, source-bound evidence, independent verification, privacy, and costs that the added assurance can justify. SN87 should therefore embody Bittensor's virtues rather than merely use its rails (Timo 2024):

- **Permissionless opportunity.** A useful assurance method should be able to enter the contest without first becoming a vendor to one company or adopting a particular work platform.
- **Bottom-up specialization.** Miners should be free to specialize by challenge class, domain, model family, evidence technique, or cost profile.
- **Plural judgment.** Validators may implement different defensible evaluation strategies, expose disagreement, and adapt locally while remaining auditable.
- **Programmable incentives.** Rewards should follow demonstrated utility under challenge—not pedigree, rhetoric, or institutional sponsorship.
- **Composability.** Capsules, Differentials, challenge schemas, and benchmark interfaces should be reusable by other subnets and work systems.
- **Resilience through diversity.** Independent code, infrastructure, models, and analytical approaches can reduce common-mode failure when that independence is measured rather than assumed.

This is also a cultural commitment. SN87 should welcome improvements that were not invented by its authors. Its credibility will grow when miners, validators, researchers, privacy engineers, operators, and work owners can challenge the design and make the resulting market harder to fool.

Bittensor provision and SN87 augmentation

Bittensor-native provision	SN87 use or augmentation	Boundary
Registered miners and validators; metagraph and chain-state discovery	Coordinates competing assurance methods and validator observations	Registration does not establish assurance quality.
Hotkey identities and authenticated requests	Binds participants to challenge and response records	Identity does not establish the truth of a response.
Validator weight submission	Converts reconstructable measurements into miner preferences	SN87 does not replace chain-side consensus.
Stake-weighted Yuma Consensus and validator incentives	Makes repeated competition economically persistent	Consensus can aggregate judgments; it cannot make an invalid scoring rule valid.
Chain-side Yuma and emissions mechanics	Supplies the incentive environment around submitted measurements	Market conviction does not prove product-market fit or assurance quality.

SN87 supplies the assurance challenge, ground-truth contract, privacy policy, measurement function, domain semantics, and evidence boundary. Bittensor supplies the protocol and incentive consensus around submitted measurements (Bittensor n.d.-a, n.d.-b, n.d.-d).

The limits of consensus

Yuma Consensus is designed for subjective utility networks. It applies stake-based consensus, clips unsupported excess weights, and uses validator-miner bonds to penalize manipulation and reward useful evaluation (Bittensor n.d.-d). Its security assumptions still matter. If validators share the same flawed evaluator, copy one another, collude, or receive leaked challenge truth, consensus can faithfully aggregate a bad measurement regime. Higher subjectivity variance also weakens guarantees.

SN87 should therefore push as much emission-driving evaluation as possible into **reproducible, independently auditable challenge classes**, while explicitly reporting the reliability of adjudicated components. Committed inputs and a pinned evaluator must reproduce the measurement. Uncalibrated changes to criteria can make scores reflect evaluator drift rather than improved assurance. Reproducibility does not establish truth: oracle validity and reliability remain separate requirements. Diversity should come from miner analysis and defensible challenge construction, not unexplained changes in the scoring function.

What SN87 evaluates

SN87 evaluates an execution—not the identity of the agent, the standing of the organization, or the total quality of the work product. Each challenge is bounded to the requirements, path, evidence, and oracle conditions declared for a particular run. The resulting Differential answers a narrower and more useful question: what does the permitted evidence establish about this execution under this challenge?

Shared meaning before a market

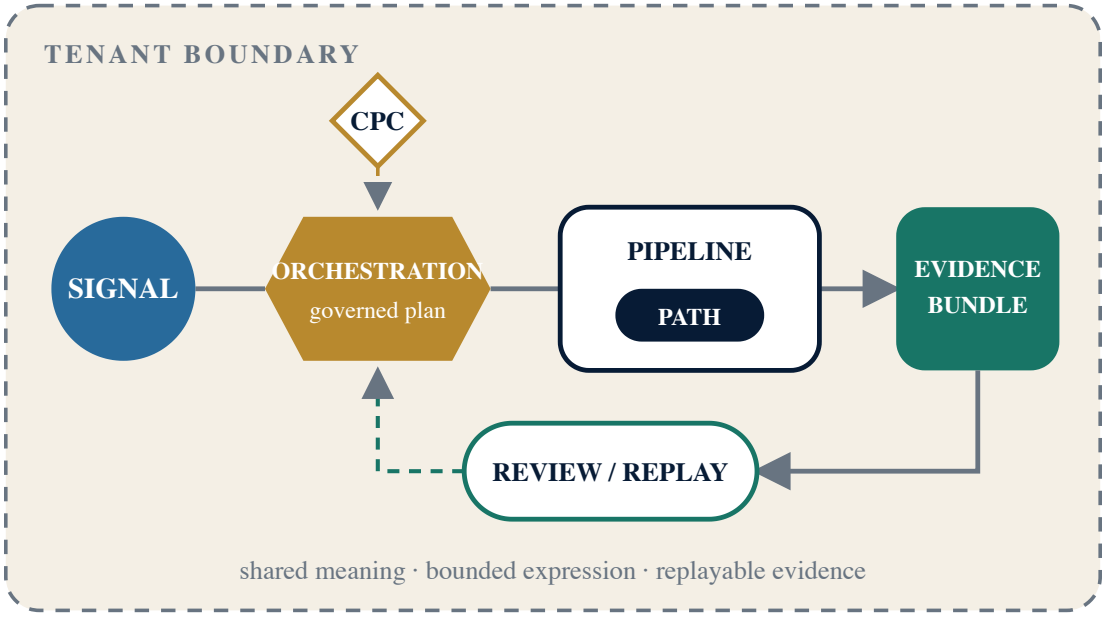
Independent assurance methods need both an **ontology** and a **grammar** for the work they compare. The ontology establishes the kinds of objects and relationships that carry meaning across systems. The grammar makes a bounded execution expressible in a form those methods can reconstruct and compare. Together they preserve shared meaning while exposing differences in naming, context, and disclosure instead of mistaking them for differences in execution quality.

The vocabulary below is drawn from **Provenonce Canon**, a reference vocabulary for governed work, and adapted here for a proposed, provider-independent assurance boundary. This paper introduces that vocabulary; it does not establish Provenonce Canon or imply that another system's objects are automatically equivalent. Other systems may map their own objects into the boundary, but must expose missing context and mapping uncertainty.

- A **Tenant** is the customer/account/workspace boundary.
- A **Signal** is real ingress or an event that requires response.
- An **Orchestration** is the tenant-bound governed plan and customer product context.
- A **CPC** is a durable Canonical Path Contract that defines a valid Path.
- A **Path** is a governed executable capability or a particular execution of that capability.
- A **Pipeline** is the run, the work. It is not the stored plan.
- An **Evidence Bundle** is structured proof of what happened, what was inferred, what was held, and what remains replayable.
- **Review / replay** turns execution into institutional memory and improvement.

Figure 2. A Shared Ontology and Grammar of Work

Stable objects make a bounded execution expressible across systems.



The ontology names the objects and relationships; the grammar makes their sequence expressible for review and replay.

These terms describe the originating work without transferring ownership of it. A participating system maps its permitted context into the shared boundary and declares any missing or uncertain correspondence. Part II specifies the mechanics of that mapping.

> **Reading the diagrams.** Blue marks source inputs; gold marks governed steps; green marks bounded results or review. Navy identifies system structure. Dashed lines mark a declared boundary, conditional path, or recurrence as labelled. Shapes and labels carry the same distinctions; color alone is not a guarantee.

Why the problem compounds

At small scale, an unreviewed execution is a local risk. At autonomous scale, traces, decisions, artifacts, and outcomes accumulate faster than institutions can interpret them. This is digital exhaust: the traces, decisions, artifacts, and outcomes of autonomous work, produced faster than they can be governed, compared, or learned from. Left unbounded, it becomes invisible debt—work that compounds without an equally scalable capacity to review and replay it.

The ontology and grammar make one execution legible. Their larger value emerges when comparable accounts accumulate: review can become reusable institutional memory rather than a series of isolated audits.

A single failed run is an incident. Repeated unreviewed runs become institutional drift.

Evidence should do more than support an audit. A meaningful run should leave replayable memory that improves the next decision. One Assurance Differential can catch a defect. Thousands of comparable Differentials can reveal weak branches, misleading confidence thresholds, recurring omissions, brittle handoffs, and successful novel Paths. The long-term opportunity is a better collective model of assured execution, built from reviewable findings and tested improvements. Part II turns that ambition into a bounded protocol: what may be disclosed, what a miner returns, and how a validator judges it.

PART II

Subnet Design and Operation

Terms used in this paper

The working vocabulary below follows the path from originating work to assurance. The full glossary preserves all definitions for reference; readers can return to it as needed. Provenonce Canon supplies one reference ontology. Other systems may map equivalent objects without adopting Provenonce or Oresund.

- **Work owner** controls the process and evidence and authorizes disclosure.
- **Canonical Path Contract (CPC)** defines a valid governed Path.
- **Path** is a governed capability or an execution of it.
- **Pipeline** is the run itself.
- **Evidence Bundle** preserves source-bound records for review and replay.
- **Assurance Adapter** maps permitted context and declares mapping loss.
- **Assurance Capsule** is the bounded account disclosed to SN87.
- **Assurance Challenge** declares the test, scoring policy, and oracle.
- **Assurance Differential** returns findings, no material deviation, or insufficient evidence/abstention.
- **Oracle** is the declared truth source against which a rule is judged.

The Assurance Capsule

Independent assurance requires a consistent account of the work being assessed. The Assurance Capsule expresses that account in a provider-independent structure while limiting disclosure to the evidence and context permitted by the work owner.

The work remains within the originating system. A source-side Assurance Adapter identifies the relevant objects, maps only the permitted context, and declares anything omitted, unavailable, inferred, or translated with loss. The following tuple describes the execution **within that originating work system** before disclosure:

$$x = (S, O, \mathcal{C}, \mathcal{P}_{\text{obs}}, L, E, \Gamma),$$

where:

- S is the Signal or Signal class;
- O is Orchestration context;
- $\mathcal{C}$ is the set of relevant CPC constraints;
- $\mathcal{P}_{\text{obs}}$ is the set or sequence of observed Path executions;
- L is the Pipeline trace;
- E is the admissible Evidence or Evidence digest; and
- Γ is the policy, entitlement, and environmental context necessary to interpret the run.

The Assurance Adapter does not export x by default. It applies a disclosure transform:

$$A_\pi(x) \to a,$$

where π is the Tenant's disclosure policy and a is the **Assurance Capsule** actually sent to a validator or miner. The transform may replace content with commitments, derived features, categorical factors, redactions, or attested execution references.

The Capsule preserves the distinction between source objects and their disclosed representation. The work owner's systems retain the Orchestration, Pipeline, Paths, and Evidence Bundle; SN87 receives only the account permitted by the disclosure policy.

Reconstructability

A capsule is useful only if a legitimate reviewer can reconstruct the claim it supports. Reconstructability does not require exporting every private byte. It requires that the Evidence chain and disclosure policy are sufficient to answer:

- What Signal and governed context were relevant?
- Which Path was taken?
- Which constraints applied?
- What did the system observe, infer, and hold?
- Which parts can be independently recomputed?
- Which parts rely on an external or adjudicated reference?
- What remains unknown?

This is the difference between telemetry and Evidence. Telemetry is recorded occurrence. Evidence is occurrence made legible against a claim.

The Digital Commodity: Assurance Differential

The Assurance Differential records the difference, if any, between an observed execution and its declared requirements. It declares exactly one response state: findings, no material deviation, or insufficient evidence/abstention. The name does not imply that a defect must exist. Its structured account makes a miner's bounded analysis inspectable, comparable, challengeable, and scorable without reducing it to an opaque number. Comparing Differentials across repeated challenges may reveal which methods discover material deviations, abstain responsibly, support findings with evidence, and propose useful counterfactuals. Each Differential records one analysis; the collection provides evidence for testing whether the network learns to assure work better.

Definition

For a challenge q, miner i returns:

$$x_i(q) = (\zeta_i, \Delta_i?, A_i?, \hat{\mathcal{P}}_i?, \mathbf{p}_i, \mathscr{R}_i),$$

where:

- ζ_i is exactly one response state: `FINDINGS`, `NO_MATERIAL_DEVIATION`, or `INSUFFICIENT_EVIDENCE_ABSTAIN`;
- Δ_i is a conditional set of claimed deviations, omissions, weak handoffs, or latent failure conditions;

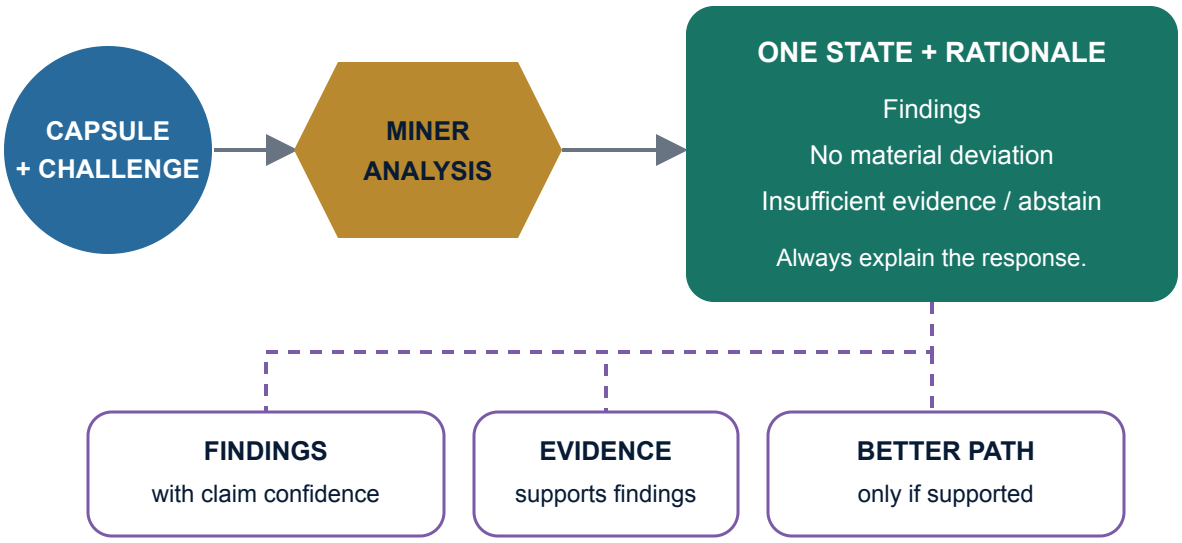
- A_i is supporting Evidence when a finding is asserted;
- $\hat{\mathcal{P}}_i$ is an optional counterfactual or superior Path continuation only where supported;
- $\mathbf{p}_i$ is a vector of calibrated confidences attached to claims; and
- $\mathcal{R}_i$ is a compact rationale sufficient for a reviewer to connect the response state, claim, Evidence, and recommended action.

The miner does not merely return a scalar, and it is not structurally forced to invent a defect. A correct negative response is valuable on a sealed clean control; abstention is preferable to unsupported confidence but does not earn defect-detection credit.

The validator may derive a scalar from the structured response, but the commodity remains decomposable and auditable.

Figure 3. The Assurance Differential

A defensible response declares one state and explains it; claims require support.



Dashed branches are conditional; a defect is never required.

The Assurance Differential always declares one state and a rationale. Findings require Evidence and confidence. Counterfactuals appear only when supported; negative and abstaining responses need not invent a finding.

A concise formulation is:

$$\begin{aligned}\text{Assurance Differential} = \;& \text{State} + \text{Rationale} \\ & + \text{Conditional Findings} + \text{Evidence} \\ & + \text{Claim Confidence} + \text{Optional Better Path}\end{aligned}$$

Why not just an assurance score?

A scalar without decomposition is easy to market and hard to trust. Two miners could both return 0.91 for incompatible reasons. A customer cannot act responsibly unless the score exposes:

- what was found;
- which evidence supports it;
- which oracle or hidden truth judged it;
- how reliable the reference is;
- which alternative was proposed; and
- how uncertain the miner was.

For rules judged against an adjudicated reference, a score must distinguish objective accuracy from agreement with that reference and report the reference's reliability. The same distinction applies to the full miner response.

Evidence and economic value

Miners exposed to many possible Paths could learn which Evidence best demonstrates that a process did what it was intended to do. Their outputs could then become a reusable assurance commodity. The token must remain distinct from that Evidence: a token is an economic instrument; Evidence is a source-bound artifact. The proposed economic relationship is:

> $$\text{better assurance commodity} \Rightarrow \text{greater user demand} \Rightarrow \text{potential economic value},$$

not:

> $$\text{more tokens} = \text{more truth}.$$

Any durable network value would have to rest on the quality and reusability of assurance work. Representing Evidence as token quantity does not establish that value.

Miner specialization

The Differential's common response contract allows miners to specialize. One miner may excel at sequence anomaly detection, another at policy reasoning, another at counterfactual search, another at cryptographic proof verification, and another at domain-specific analysis. A strong validator does not require every miner to use the same model or method. It requires every response to satisfy the same output contract and face the same hidden challenge.

Independent methods may discover different material defects even when average accuracy is similar. That is a testable reason to compare decentralized competition with a centralized ensemble. Independence matters when evaluation shows that a method contributes nonredundant, material findings.

Assurance Roles and Message Flow

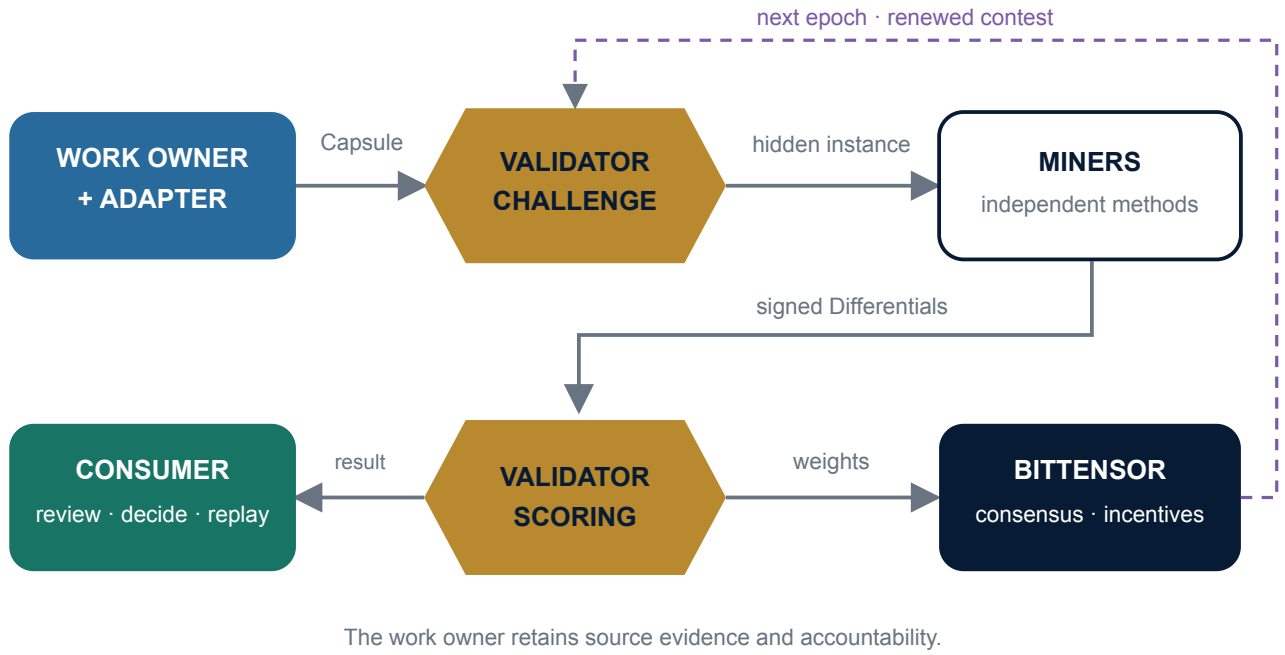
Roles

Role	Core responsibility	Boundary
Work owner	Controls the process, source data, disclosure policy, and accountable decision; chooses whether and when to request assurance.	Retains the work and source evidence within the relevant Tenant boundary.
Assurance Adapter	Observes the run, maps permitted context, binds evidence, and constructs the Capsule.	Declares omissions, inference, and mapping loss; does not transfer ownership.
Publisher	Defines versioned challenge classes, oracle bindings, admissibility rules, cost budgets, and benchmark material.	Its references remain subject to independent inspection and challenge.
Validator	Selects hidden challenge instances, verifies response integrity, scores against declared truth, aggregates performance, and submits weights.	Uses public scoring semantics while keeping instance truth sealed until the audit window.
Miner	Returns the Assurance Differential; may perform bounded analysis or re-execution where the challenge allows.	Does not run the customer’s live business process by default.
Bittensor / Subtensor	Registers participants, accepts validator preferences, applies configured consensus, and distributes incentives.	Does not define whether SN87’s assurance measurement is valid.
Assurance consumer	Reviews the Differential and decides whether to accept, investigate, replay, or alter future work.	Remains accountable for the resulting action.

Message flow

Figure 4. SN87 Assurance Market

Validators construct hidden challenges; miners compete to produce assurance intelligence.



SN87 assurance market loop. Validators measure the response and submit weights; Bittensor applies consensus and incentives. The assurance consumer receives a reviewable result, not a transfer of accountability.

An SN87 round is:

1. The work owner produces or authorizes an Assurance Capsule a under policy π.
2. Validator v combines a with a challenge class χ, nonce n, and hidden state h_v.
3. The validator sends an authenticated, versioned request that binds the challenge to its intended recipient and freshness conditions. Authentication must prevent substitution and cross-context replay.
4. Each miner returns a signed Differential $x_i(q)$ before the deadline.
5. The validator performs the integrity gate, rule-level scoring, challenge-level aggregation, and epoch update.
6. The validator transforms miner quality into a normalized weight vector W_v.
7. The validator submits a valid weight row through the network's supported mechanism.
8. Bittensor aggregates validator preferences through Yuma Consensus and applies network incentives.
9. The assurance consumer receives the Differential and its limitations. The work owner decides whether to investigate, replay, or improve future work; lessons may also inform later challenge design.

Challenge object

A complete challenge instance is:

$$q_v = (a, \chi, h_v, n, t_{\text{exp}}, \Omega),$$

where a is the capsule, χ the challenge class, h_v validator-hidden truth, n a nonce, t_{exp} the deadline, and Ω the scoring and admissibility declaration.

The visible scoring **schema** should be public. The individual hidden truth, perturbation seed, canary placement, and held-out instance should not be predictable before response submission. This preserves auditability without publishing the answer key.

Primary and secondary challenge families

Family	Miner task	Purpose and boundary
Primary — Assurance analysis	Analyze a Capsule and return a bounded Differential.	Default proposed commodity: findings, evidence, alternatives, and calibrated uncertainty.
Secondary — Bounded re-execution	Execute a sanitized or synthetic Path to produce a reference or counterfactual result.	Used only where objective comparison exists and disclosure policy permits it.
Secondary — Adversarial mutation	Detect controlled defects, omissions, reordered evidence, or misleading context.	Provides hidden test truth without claiming complete domain realism.
Outside the core proposal — Live enterprise execution	Execute full operational workflows.	Not required to test independent assurance; introduces materially greater privacy, reliability, and operational risk.

Privacy-Preserving Assurance

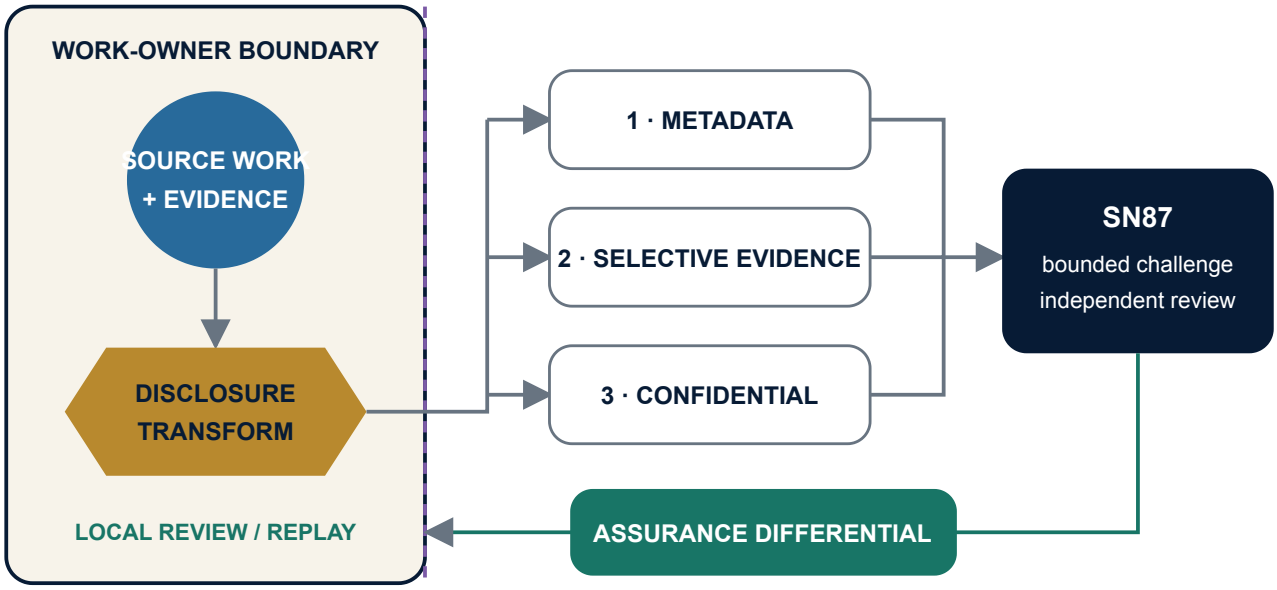
Privacy boundaries

OpenTelemetry's GenAI conventions explicitly warn that input messages, output messages, system instructions, retrieval queries, and tool arguments may contain sensitive or personally identifiable data (OpenTelemetry Authors n.d.). Instrumentation is not automatically safe because it is "only telemetry." Timing, graph shape, error patterns, model choice, document size, and derived features can also reveal trade secrets.

SN87 therefore defaults to **minimum disclosure** and separates three modes.

Figure 5. Privacy-Preserving Assurance Boundary

The work stays with its owner; policy controls what crosses the boundary.



The dashed boundary denotes policy, not a cryptographic guarantee.

A disclosure policy produces a bounded Capsule while the originating evidence remains with its owner. The Differential returns across the same governed boundary.

Mode 1: Metadata assurance

The default capsule may include:

- Signal class rather than raw Signal;
- Path identifiers or abstract transition classes;
- timing, retry, escalation, and outcome labels;
- evidence-presence commitments;
- policy and threshold identifiers;
- hashed or bucketed resource metrics; and
- cryptographic bindings to locally retained Evidence.

This mode can answer structural questions: were required stages present; did a handoff fail; was latency anomalous; did escalation occur; was the wrong branch taken under a declared rule; and did Evidence coverage meet policy?

It cannot honestly answer content questions whose truth is unavailable in the capsule. The output must state that boundary.

Mode 2: Selective evidence

A miner may need more than metadata to discriminate among hypotheses. SN87 can support bounded Evidence requests:

$$e_{ij} = \text{request}(\text{predicate}_j, \text{scope}_j, \text{privacy cost}_j).$$

The source-side adapter may return a redacted feature, derived fact, signed boolean, range proof, hashed excerpt, or minimal disclosure sufficient to evaluate the predicate. Each request is logged, priced, and policy-checked.

This creates a **proof economy**: miners that reach a correct conclusion with less sensitive disclosure and fewer evidence requests can be rewarded for epistemic efficiency.

Mode 3: Confidential challenge

Some semantic evaluation genuinely requires content. In that case, explicit policy may authorize encrypted data release only to an attested confidential-compute environment.

> **Illustrative technology — no provider or participant requirement.**
>
> AWS Nitro Enclaves can isolate enclave compute and support attestation-gated key release; Google Confidential VM exposes attestation evidence for a verifier. These examples show one present-day design pattern, not a technology selection, endorsement, or requirement for SN87 participants. (Amazon Web Services n.d.-a, n.d.-b; Google Cloud 2026)

A TEE changes the trust boundary; it does not prove that the code is semantically correct. SN87 must still specify:

- the approved image or measurement;
- the attestation verification policy;
- how keys are released;
- what output can leave;
- how side channels and operator risk are treated;
- what happens when attestation is unavailable; and
- whether the challenge remains reproducible across validators.

Confidential compute is a possible option for narrow challenge classes, but it should not become a blanket miner requirement. Mandatory TEEs would reduce participation, increase cost, and risk turning hardware availability into the dominant commodity.

Privacy budget

Let b_i be the normalized disclosure cost incurred by miner i through Evidence requests, where each request has policy-defined cost κ_j:

$$b_i = \sum_j \kappa_j \cdot \mathbf{1}[e_{ij} \text{ released}].$$

Privacy can then enter the efficiency penalty without allowing a privacy violation to be traded off against quality:

- policy violation $\Rightarrow$ hard gate failure;
- policy-compliant but disclosure-heavy analysis $\Rightarrow$ higher b_i and lower efficiency.

Data retention

The subnet should not become a raw Evidence warehouse. A bounded retention model is:

- network-facing: weight-related state and, where the selected implementation supports it, minimal commitments or references;
- validator-local: challenge truth and scoring artifacts for a bounded audit window;
- miner-local: only what policy and challenge require, with deletion attestations where feasible;
- Tenant-local: raw Evidence and full replayable history;
- shared benchmark store: sanitized or synthetic challenge corpora with explicit licenses.

This is a proposed allocation of responsibility, not a claim that a particular chain storage interface exists. The originating work system retains its source evidence independently of any external ledger. SN87 may verify or score only the permitted representation.

Challenge Classes and Oracle Taxonomy

The ground-truth problem

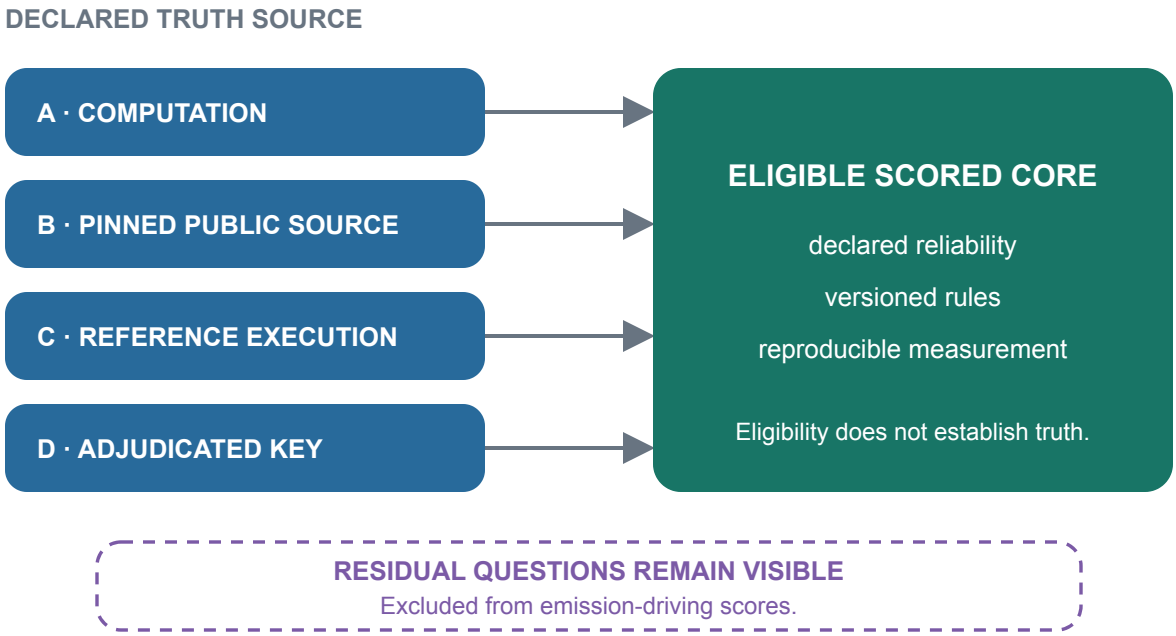
The central ground-truth question is: **how can a validator establish that one Assurance Differential is better than another?**

A universal answer does not exist. Some claims are mechanically verifiable; some depend on a pinned public source; some can be tested by re-execution; some depend on expert adjudication; and some remain irreducibly judgment-laden. SN87 must make this epistemic status explicit.

The proposed design uses four oracle classes, A–D, with reliability weighting for adjudicated references. The examples below illustrate classes, not selected domains or fixed implementation parameters.

Figure 6. Oracle Taxonomy and Epistemic Honesty

Only rules with a defensible truth source enter the scored core.



Types A–D define eligible oracle bindings. Residual uncertainty is reported, not silently scored.

Type A – self-verifying

A Type-A rule is checkable by computation over disclosed or committed data, without an external truth source. Examples:

- schema validity;
- required-field presence;
- arithmetic consistency;
- hash-chain validity;
- nonce and signature binding;
- deadline compliance;
- Path-transition legality;
- Evidence coverage declarations;
- edge equality between a committed output and the next consumed input; and
- resource-budget limits.

Type A is the strongest initial foundation because any honest validator running the same code on the same commitments should obtain the same result.

Type B – public authoritative oracle

A Type-B rule queries a pinned, auditable public reference:

- a regulatory threshold table at a declared version;
- an approved code dictionary;

- a public price or benchmark at a declared timestamp;
- a software vulnerability database snapshot;
- a signed policy or standard release.

SN87 must bind the oracle identity, version, query, response commitment, and caching policy. “Public” does not mean stable. A live web page without version pinning is not a reproducible oracle.

Type C – reference execution

A Type-C rule compares the miner response to a reference execution on held-out or generated inputs. Examples:

- deterministic mutation tests;
- a validator-generated synthetic workflow with known planted defects;
- a reference implementation for a bounded transform;
- a replay against a sealed environment;
- a known-optimal solution for a constrained problem.

Type C is powerful for adversarial challenge generation because the validator can know the answer without disclosing it in advance. It is also dangerous when the reference implementation becomes the hidden centralized product truth. The reference must be inspectable after the reveal window, and multiple implementations should be possible where feasible.

Type D – sealed adjudicated key

Some valuable fields have no objective oracle but do have expert-adjudicated references. Each Type-D oracle must declare:

- the answer-key commitment;
- the reveal window;
- epistemic/source reliability $\lambda_r \in (0, 1]$; and
- the method by which reliability was measured.

The Type-D claim is not “the miner is objectively correct.” It is:

> the miner agrees with an expert-adjudicated reference, within the measured reliability of that reference.

That is weaker, more honest, and more useful than hiding disagreement behind a single score.

A clinical-trial adverse-event example can separate hard, reproducible checks from adjudicated coding and causality, while reporting narrative medical coherence as residual uncertainty when no defensible oracle exists. An adjudicated reference set would use blinded multi-rater review, held-out sealed keys, and measured inter-rater agreement. The proportions depend on the domain and evaluation design; no numerical allocation is asserted here.

Residual uncertainty and scoring eligibility

Claims without a defensible oracle remain visible as residual uncertainty but do not contribute to emission-driving scores. They may support research or human review. A residual claim becomes eligible for scoring only when a versioned challenge class supplies a defensible oracle and declares its reliability.

Model judges are useful in centralized evaluation products for subjective criteria such as helpfulness, tone, and creativity (Braintrust n.d.-a). SN87 may use them as secondary signals, triage tools, or research features. They should not drive the integrity floor or the core score unless they are calibrated against a stronger oracle and their reliability is exposed.

Admissibility declaration

Each challenge class χ publishes an admissibility declaration:

Field	Required declaration
Class identity	Versioned name and the kinds of claims eligible for evaluation.
Oracle bindings	Rule source, pinned version or commitment, reliability, and the method used to warrant that reliability.
Residual fields	Questions to report without including them in emission-driving scores.
Disclosure	Permitted modes, evidence-request policy, and confidentiality limits.
Cost and time	Precommitted resource budget, deadline, and audit/reveal conditions.

A validator must reject a challenge instance that violates the class declaration. A miner must be able to inspect what type of claim it is being asked to make, even though the hidden instance truth remains sealed.

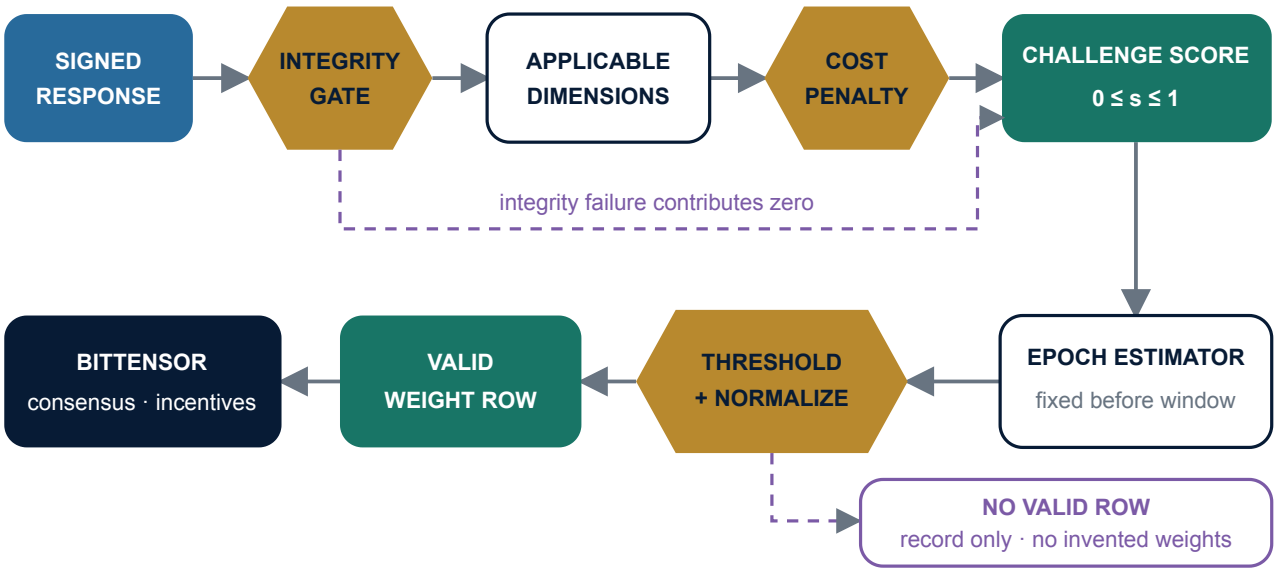
Bittensor-Native Assurance Mathematics

Overview and notation

The formulas below are scoring proposals, not calibrated constants. They define the path from a signed response to a validator weight row; Bittensor remains the chain-side consensus and emissions system.

Figure 7. From Responses to Bittensor Weights

SN87 defines the measurement path; Bittensor applies configured consensus.



An integrity failure scores zero; a window with no valid preference produces an explicit no-valid-row state rather than invented fallback weights.

Let V and $\mathcal{M}$ be the active validator and miner sets; $Q_v^{(t)}$ validator v's challenge set in epoch t; $q \in Q_v^{(t)}$ one challenge; $x_i(q)$ miner i's response; $h_v(q)$ hidden truth; $M_{kiq} \in [0,1]$ the score on applicable dimension k; s_{viq} the challenge score; $S_{vi}^{(t)}$ the epoch quality estimate; and $W_{vi}^{(t)}$ the submitted weight. The response rationale is $\mathcal{R}_i$; this symbol is not reused for recall. All emission-driving values must be reproducible from the challenge commitment, canonical response bytes, oracle reveal, and pinned scoring implementation.

Notation family	Interpretation
Upper-case V; calligraphic M	Sets of validators and miners.
Lower-case v, i, q, k, r, t	Validator, miner, challenge, dimension, rule, and evaluation-window indices.
Upper-case M with dimension subscripts	A bounded dimension score; distinct from the miner set.
Lower-case s; upper-case S	One challenge score; an epoch quality estimate.
Upper-case G, W	Non-compensable integrity gate; validator preference row.
Lambda by rule; lambda by cost	Source reliability; separately indexed cost coefficients.

Parameters and numerical behavior are fixed by the scoring specification before a window begins. The equations state a candidate mechanism, not a claim that its robustness or economic behavior has been established.

Hard integrity gate

Define the non-compensable integrity gate:

$$G_{viq} = G_{\text{schema}}\, G_{\text{policy}}\, G_{\text{signature}}\, G_{\text{nonce}}\, G_{\text{evidence}}\, G_{\text{deadline}} \in \{0,1\}. \tag{1}$$

Malformed responses, policy violations, invalid signatures or nonces, broken commitments, missing required attestations, late responses, or undeclared oracle dependencies set $G_{viq} = 0$ and therefore $s_{viq} = 0$. This G is distinct from chain-side miner incentive.

Rule scores: reproducibility is not truth

For each applicable rule r, let $u_r \in [0,1]$ be agreement, $w_r > 0$ its precommitted importance, $\delta_r \in \{0,1\}$ whether repeated evaluation is deterministic under pinned inputs, and $\lambda_r \in (0,1]$ the warranted epistemic reliability of its truth source. Determinism does not force $\lambda_r = 1$. A Type A or C rule may be reproducible while inheriting a faulty specification or reference implementation. Reliability equals one only when the source warrant justifies it.

For applicable rules $\mathcal{A}_k(q)$:

$$M_{kiq} = \frac{\sum_{r \in \mathcal{A}_k(q)} w_r \lambda_r u_r}{\sum_{r \in \mathcal{A}_k(q)} w_r \lambda_r}. \tag{2}$$

If $\mathcal{A}_k(q)$ is empty, M_{kiq} is omitted, not set to zero. Objective/reproducible and adjudicated subtotals use the same conditional rule: report the quotient only when its denominator is positive; otherwise report NA. Every later aggregate renormalizes over applicable dimensions. SN87 reports δ_r separately from λ_r so repeatability cannot masquerade as truth.

Reliability changes a rule's relative influence in this quotient; it does not impose an absolute ceiling on the quotient. Agreement with every low-reliability reference can still produce a score of one. Reference reliability must therefore remain separately visible. The score measures warranted agreement under the declared rules, not a certified probability of truth.

Dimension 1: defect detection and clean controls

Let $\Delta^*(q)$ be the hidden defect set. Each true defect d has precommitted severity $\omega_d > 0$. Each unmatched claimed finding $f \in FP_i$ incurs a precommitted false-positive cost

$$c_{\text{FP}}(f) = \max\{c_{\min}, c_{\text{class}}(\hat{s}_f), c_{\text{val}}(f,q)\} > 0,$$

where $c_{\min}$ is a nonzero floor, c_{class} is the challenge-class schedule for claimed severity $\hat{s}_f$, and c_{val} is the validator's sealed rule-based severity assessment. A miner therefore cannot reduce its penalty merely by understating severity. Let TP_i be matched true positives.

$$\mathrm{Prec}_{iq} = \begin{cases} \dfrac{\sum_{d \in TP_i} \omega_d}{\sum_{d \in TP_i} \omega_d + \sum_{f \in FP_i} c_{\mathrm{FP}}(f)}, & |TP_i| + |FP_i| > 0, \\ 1, & |TP_i| + |FP_i| = 0, \end{cases} \tag{3}$$

$$\mathrm{Rec}_{iq} = \frac{\sum_{d \in TP_i} \omega_d}{\sum_{d \in \Delta^*(q)} \omega_d} \quad \text{when } |\Delta^*(q)| > 0; \quad \text{otherwise NA.} \tag{4}$$

For defect-bearing cases:

$$M_{iq}^{\mathrm{det}} = \frac{(1+\beta^2)\mathrm{Prec}_{iq}\mathrm{Rec}_{iq}}{\beta^2 \mathrm{Prec}_{iq} + \mathrm{Rec}_{iq}}, \tag{5}$$

with the value zero when precision and recall are both zero. Clean negative controls use $M_{iq}^{\mathrm{clean}} = 1$ only for an explicit `NO_MATERIAL_DEVIATION` response. Abstention earns zero detection credit; a `FINDINGS` response receives a precommitted monotone penalty derived from $\sum_f c_{\mathrm{FP}}(f)$. The matching ontology, equivalence classes, partial credit, c_{FP}, ω_d, and β are committed before responses are observed. The precision/recall tradeoff parameter is strictly positive, and every clean-control score remains within zero and one.

Response states and anti-spam

Every response declares exactly one state: `FINDINGS`, `NO_MATERIAL_DEVIATION`, or `INSUFFICIENT_EVIDENCE_ABSTAIN`. Findings and counterfactuals are conditional fields. A correct negative response on a sealed clean control earns the clean-control score. A false `NO_MATERIAL_DEVIATION` on a defect-bearing case receives zero detection recall. Abstention avoids fabricated findings but earns no detection credit; calibration and abstention quality may be reported separately. Repeated indiscriminate abstention cannot clear the minimum-quality threshold. Unsupported findings incur false-positive cost, preventing verbosity from becoming a winning strategy.

Dimensions 2–4: evidence, calibration, and utility

For matched true findings, Evidence quality is:

$$M_{iq}^{\mathrm{ev}} = \frac{\sum_{d \in TP_i} \omega_d \, \mathcal{V}(A_{id})}{\sum_{d \in TP_i} \omega_d}, \tag{6}$$

when TP_i is nonempty; otherwise it is omitted. The evidence verifier returns a value between zero and one. It may use a bounded geometric decomposition of source binding, relevance, sufficiency, and reproducibility, with all active exponents normalized to sum to one.

For $n_{iq} > 0$ scored confidence claims, define the Brier score and calibration quality:

$$\mathrm{BS}_{iq} = \frac{1}{n_{iq}} \sum_{j=1}^{n_{iq}} (p_{ijq} - y_{ijq})^2, \qquad M_{iq}^{\mathrm{cal}} = 1 - \mathrm{BS}_{iq}. \tag{7}$$

Where a defensible utility function J_q is oriented so that higher is better, and the reference improvement is strictly positive:

$$M_{iq}^{\mathrm{util}} = \mathrm{clip}\left(\frac{J_q(\hat{P}_i) - J_q(P_{\mathrm{obs}})}{J_q(P^*) - J_q(P_{\mathrm{obs}})}, 0, 1\right). \tag{8}$$

Otherwise utility is omitted. No counterfactual is required for a negative or abstaining response, and no invented utility fills a missing dimension.

Non-circular robustness

Let $\mathcal{K}_0(q)$ be applicable quality dimensions excluding robustness and efficiency. For each committed perturbation $h \in \mathcal{H}_q$, first compute a base score:

$$b_{viq}(h) = \exp\left(\sum_{k \in \mathcal{K}_0(q)} \bar{a}_k(q) \ln\left(\max\{M_{kiq}(h), \epsilon\}\right)\right), \quad \sum_k \bar{a}_k(q) = 1. \tag{9}$$

Because every $M_{kiq} \in [0,1]$, this construction remains in $(0,1]$ and an all-perfect vector equals one. Robustness is then calculated from the already-fixed base scores:

$$M_{iq}^{\mathrm{rob}} = \mathrm{clip}\left(1 - \frac{1}{|\mathcal{H}_q|} \sum_{h \in \mathcal{H}_q} |b_{viq}(h) - b_{viq}(\mathrm{id})|, 0, 1\right). \tag{10}$$

This calculation requires a nonempty perturbation family and at least one grounded base dimension. Otherwise robustness is omitted. The smoothing constant lies strictly between zero and one; all active dimension weights are nonnegative and sum to one. Perturbations must preserve the scoring meaning they are intended to test.

Rank stability across equivalent mutations is also reported as a preregistered statistic. Robustness is inserted only after equation (10), so it cannot depend on a composite that already contains itself.

Response-indexed cost and final composite

Let c_{iq}, ℓ_{iq}, b_{iq}, and o_{iq} be normalized compute, latency, disclosure, and external-oracle costs for miner i on challenge q. All costs and their penalty coefficients are nonnegative:

$$\eta_{iq} = \exp\left(-\lambda_c c_{iq} - \lambda_\ell \ell_{iq} - \lambda_b b_{iq} - \lambda_o o_{iq}\right). \tag{11}$$

Let $\mathcal{K}(q)$ be the applicable final dimensions with renormalized weights $a_k(q)$ summing to one. The bounded challenge score is:

$$s_{viq} = G_{viq} \exp\left(\sum_{k \in \mathcal{K}(q)} a_k(q) \ln\left(\max\left\{M_{kiq}, \epsilon\right\}\right)\right)\eta_{iq}. \tag{12}$$

Inapplicable dimensions are omitted and weights renormalized; integrity failures remain zero. If no grounded quality dimension is applicable, the challenge cannot earn quality credit: it receives zero and is flagged for challenge-class review. An empty product must not create a perfect score. A required field missing from a response is a failure, not an inapplicable dimension.

Experiments should compare this proposal with arithmetic and constrained-linear alternatives under the same budgets and attacks.

Epoch quality and exploration

With precommitted finite difficulty $d_q > 0$ and finite, nonnegative freshness ϕ_q, use those quantities as weights on bounded challenge scores, not as multipliers that can raise a score above one. A concrete candidate is the following weighted winsorized mean:

$$S_{vi}^{(t)} = \frac{\sum_{q \in Q_{vi}^{(t)}} d_q \phi_q \operatorname{clip}\left(s_{viq}, L_{vi}^{(t)}, U_{vi}^{(t)}\right)}{\sum_{q \in Q_{vi}^{(t)}} d_q \phi_q}. \tag{13}$$

Here the indexed challenge set contains the instances actually assigned to that miner. The lower and upper limits are weighted quantiles at a precommitted tail fraction and its complement. For positive tail fractions below one half, each quantile is the smallest observed score whose cumulative normalized weight reaches its declared fraction; ties share the same score. At tail fraction zero, use the observed minimum and maximum, giving the ordinary weighted mean. Zero-weight observations do not affect the estimate.

Missing, late, or invalid assigned responses enter as zero. Winsorization may soften isolated failures, so the report must also expose their raw rate; a precommitted eligibility floor can disqualify persistent failure. The estimator is bounded between zero and one, but adversarial resistance is a hypothesis to test, not a property established by its name.

An empty assignment set, zero total weight, or failure to meet the precommitted sample and eligibility floors yields an unavailable estimate. Such a miner receives no positive thresholded contribution. The tail fraction, quantile convention, assignment rule, sample floor, and missing-response policy are fixed before evaluation and recorded with the scoring specification.

Exploration may use $\hat{\mu}_{vi} + \kappa_e \hat{\sigma}_{vi}$ for routing, but exploration is not an emission bonus.

Piecewise normalization and the no-valid-weight state

Apply threshold θ between zero (inclusive) and one (exclusive), and sharpness $\gamma > 0$, to each available estimate:

$$z_{vi} = \max\left(S_{vi}^{(t)} - \theta, 0\right)^{\gamma}, \qquad Z_v = \sum_{j \in \mathcal{M}} z_{vj}. \tag{14}$$

Then:

$$W_{vi}^{(t)} = \begin{cases} z_{vi}/Z_v, & Z_v > 0, \\ \perp, & Z_v = 0. \end{cases} \tag{15}$$

The $\perp$ state means “no valid preference row.” The validator records the state and does not fabricate equal weights or use epsilon normalization. An unavailable estimate contributes zero to the thresholded total, without being reported as a measured quality of zero. The submission policy must distinguish a valid row from no valid row and follow the network’s supported behavior.

Relationship to Bittensor consensus

SN87 produces a validator-to-miner preference row. Bittensor applies its own finalized clipping, stake-weighted consensus, incentive, bonds, dividends, and emissions behavior. Conceptually:

$$(\mathrm{Inc}^{(t)}, \mathrm{Div}^{(t)}, \mathrm{Bond}^{(t+1)}) = \mathrm{Yuma}\left(W^{(t)}, \sigma^{(t)}, \mathrm{Bond}^{(t)}; \Theta^{(t)}\right), \tag{16}$$

where $\sigma^{(t)}$ is chain-derived validator stake and $\Theta^{(t)}$ the finalized chain configuration. These symbols are deliberately distinct from detection, Brier, rationale, integrity, and oracle-reliability notation.

These equations describe the division of responsibility. Mutable network interfaces and deployment configuration belong in the separately versioned SN87 Technical Specification, planned for publication alongside the reference implementation. It must pin SDK and runtime versions, request and response authentication, weight submission and commit-reveal behavior, canonical serialization and commitments, and the chain configuration verified for each release. These are requirements for that companion specification, not claims of a completed or published artifact.

Why this is Bittensor-native

The full path is:

$$\begin{aligned}\text{Execution} \quad &\rightarrow \text{Challenge} \rightarrow \text{Differential} \\ &\rightarrow s_{viq} \rightarrow S_{vi}^{(t)} \rightarrow W_{vi}^{(t)} \\ &\rightarrow \text{Yuma Consensus} \rightarrow \text{Emissions}\end{aligned}$$

The subnet's intellectual contribution is the assurance challenge and measurement function. Bittensor's contribution is the open market and consensus that make repeated competition economically persistent.

Parameter governance

The following symbolic coefficients express the scoring design; this paper does not prescribe calibrated numerical values:

$$\beta, \quad a_k(q), \quad \lambda_c, \lambda_\ell, \lambda_b, \lambda_o, \quad \theta, \quad \gamma, \quad \kappa_e,$$

Each scoring specification defines its match tolerances and sampling policy before evaluation begins. Those settings remain fixed within the evaluation window, are recorded with the scoring implementation, and change through versioned migration. Their values depend on the challenge class and evaluation design.

Challenge difficulty functions, the estimator's tail fraction and eligibility floors, and the stability of Type-D references remain research questions. Experiments must establish how those choices affect ranking, gaming resistance, and the reliability of the resulting scores.

A weight function that changes silently is not assurance. It is governance by hidden configuration.

Validator Design

Open code, hidden instances

Miners can reverse-engineer any visible metric and optimize for the score rather than the intended work. Fully secret scoring code would undermine auditability and participation. Fully predictable challenges would invite proxy gaming. The proposed design addresses this tension by separating:

- **open scoring semantics** – dimensions, formulas, tolerances, admissibility, and challenge-class construction;
- **hidden challenge instances** – seeds, canaries, held-out truth, perturbations, and sampling choices; and
- **post-window reveal** – enough information for independent audit without making future challenges trivial.

Validator pipeline

For each response:

1. verify miner identity, signature, nonce, version, and deadline;
2. validate schema and disclosure-policy compliance;
3. verify Evidence commitments and required attestations;
4. reveal or query the challenge oracle;

5. match findings against hidden defects and grade Evidence;
6. score calibration, counterfactual utility, and robustness where supported;
7. apply the efficiency and disclosure penalty;
8. update the miner's robust epoch estimate;
9. construct normalized weights; and
10. submit a valid preference row using the network's supported mechanism, or record the explicit no-valid-row state.

```
function score_response(challenge q, response x_i):
    G = integrity_gate(q, x_i)
    if G == 0: return 0

    truth = resolve_oracles_after_commit(q)
    detection = score_state_and_findings(x_i.state, x_i.findings, truth)
    evidence = verify_matched_evidence(x_i, truth, q.policy) if matched_findings else OMIT
    calibration = brier_quality(x_i.confidences, truth.outcomes) if scored_claims else OMIT
    utility = score_counterfactual(x_i.paths, truth.utility) if supported else OMIT
    robustness = score_perturbations(i, q) if eligible_perturbations else OMIT

    active_dimensions = declared_applicable_dimensions(
        detection, evidence, calibration, utility, robustness)
    if no_grounded_quality_dimension(active_dimensions): return 0
    quality = bounded_geometric_mean(active_dimensions, omit_missing=True)
    penalty = exp(-response_indexed_costs(i,q))
    return quality * penalty
```

Validator heterogeneity

A healthy validator set should differ in:

- challenge sampling;
- benchmark slices;
- perturbation seeds;
- miner sampling and exploration;
- infrastructure and geography;
- implementation language where possible; and
- independent operational judgment about which challenge classes deserve weight.

Validators should not differ in the meaning of a committed rule. Reproducible rules create convergence; independent challenge construction creates discovery.

Validator audit

Validators themselves require evaluation. Candidate mechanisms:

- canary miners or known reference responses;
- delayed comparison to revealed ground truth;
- challenge-coverage audits;
- challenge leakage tests;
- weight-vector anomaly detection;
- disclosure-policy compliance audits;

- public scoring-version attestations; and
- analysis of whether network incentives actually reward accurate evaluation rather than copying or common-mode behavior.

A validator that simply runs subnet-owner code can still provide distribution and liveness, but not meaningful epistemic independence. The long-term network should reward validators that improve challenge quality, not merely deploy binaries quickly.

Commit-reveal scope

Commit-reveal can conceal a weight row until its reveal condition, reducing the opportunity for direct copying where the Bittensor configuration enables it. This defense does not establish the quality of the underlying measurement and does not prevent:

- miners learning challenge instances;
- validators sharing hidden truth;
- a publisher constructing biased challenge pools;
- common-mode scoring bugs; or
- collusion outside the chain.

SN87 must therefore treat commit-reveal as one defense, not the threat model.

Ground Truth and Benchmark Construction

Benchmarks are network capital

Without renewable challenge supply, rewards can outlast useful evaluation. Repetitive public benchmarks invite memorization. SN87 therefore needs a continuing process that converts real assurance needs into sealed, versioned, auditable challenge families.

The benchmark itself becomes durable network capital when it contains:

- representative process positions;
- hidden defects and counterfactuals;
- oracle bindings;
- measured ambiguity;
- privacy-safe distributions;
- adversarial variants;
- cost baselines; and
- post-challenge Evidence that can improve the next version.

Challenge-source ladder

Level	Source	Strength	Principal limitation
0	Synthetic planted defects	Fully known truth; fast mechanism debugging.	Low realism.
1	Replayed historical executions	Realistic traces and known outcomes.	Selection bias and privacy.
2	Shadow comparisons	Compares alternative Paths without affecting live work.	Convergence and confidence do not prove production benefit.
3	Design-partner gold sets	Sealed domain references with measured reliability.	Cost, adjudication variance, and limited domain scope.
4	Temporal outcome challenges	High economic relevance after outcomes arrive.	Slow feedback and confounding.
5	Live adversarial assurance	Highest realism and discovery value.	Highest operational, privacy, and safety risk.

The decomposition test

Before admitting a domain, decompose "correct execution" into atomic claims:

1. define the business outcome and accountable decision;
2. enumerate atomic correctness claims;
3. assign each claim to Type A, B, C, D, or residual;
4. estimate weights and severity;
5. measure or declare oracle reliability;
6. calculate objective, adjudicated, and residual coverage;
7. define privacy mode and cost; and
8. reject the domain if the emission-driving core is too subjective or too expensive to verify.

A clinical-trial adverse-event example illustrates a possible application, not a universal conclusion. Regulated workflows may be attractive precisely because much of their value rests on completeness, validity, consistency, timing, and audit integrity – properties that can be mechanically checked – while expert judgment is isolated and reliability-labeled. Other domains may fail the decomposition test.

Difficulty calibration

Challenge difficulty should be estimated from empirical miner performance, not author intuition alone. Candidate model:

$$d_q = \operatorname{clip}\left(\delta_0 + \delta_1 H_q + \delta_2 A_q + \delta_3 P_q + \delta_4 O_q, d_{\min}, d_{\max}\right), \tag{17}$$

where:

- H_q = hidden-defect severity and count;

- A_q = ambiguity / adjudicated share;
- P_q = perturbation depth; and
- O_q = oracle cost or complexity.

The lower and upper difficulty bounds must be finite and strictly positive. After enough observations, item-response or Bayesian estimates may replace this initial model. Difficulty must not become a reward multiplier that incentivizes publishers to produce inscrutable tasks. A hard challenge is valuable only if its truth remains defensible.

Dataset splits and leakage

Minimum partitioning:

- public examples for implementers;
- development set for local testing;
- validator-private challenge pool;
- sealed canary set;
- temporal holdout;
- retired audit set released after useful life.

Leakage controls:

- salted commitments;
- generated variants;
- frequent semantic-preserving perturbations;
- publisher separation;
- access logging;
- delayed reveal;
- challenge fingerprints excluded from miner inputs; and
- penalties for reused or copied Evidence traces.

Benchmark governance

A benchmark publisher chooses what the network learns to value. That role must be explicit and economically bounded. Proposed controls:

- publisher bond;
- challenge-class review and versioning;
- oracle and residual disclosure;
- fee proportional to verification cost;
- challenge performance reports;
- retirement for leakage or saturation;
- competing publishers; and
- no publisher-specific private oracle that validators cannot independently audit.

This preserves open authorship without turning the task publisher into a hidden sovereign.

Security and Threat Model

Threat model premise

The adversary is not only a malicious miner. SN87 must assume that:

- miners optimize every measurable proxy;
- validators may be lazy, correlated, compromised, or economically captured;
- benchmark publishers may bias tasks toward their own systems;
- enterprises may submit misleading or incomplete capsules;
- adapters may mis-canonicalize foreign workflows;
- secrets can leak through metadata and derived features;
- common-mode bugs can cause honest validators to agree on the wrong score; and
- market attention can reward a compelling narrative before the commodity is useful.

Security is therefore a layered property of source integrity, privacy, challenge design, scoring, validator independence, and economic feedback.

Threat matrix

Threat	Failure mode	Primary defenses	Residual risk
Goodhart / proxy gaming	Miner optimizes visible metric without useful assurance	hidden instances, multi-dimensional score, canaries, perturbations, post-window audit	public challenge family can still saturate
Output spoofing	Miner returns plausible findings without analysis	Evidence contract, nonce binding, hidden defects, proof verification	sophisticated fabricated traces may pass weak checks
Evidence forgery	Hash-consistent but false local logs	attestations where justified, sampled re-execution, source commitments, external corroboration	a hash proves immutability, not truth of origin
Canonicalization error	Adapter maps foreign workflow into the wrong CPC/Path semantics	local review, confidence on mapping, reversible source links, shadow mode	confident analysis of a bad representation
False-positive flooding	Miner produces alarming but low-value findings	severity-weighted precision, reviewer-cost penalty, evidence sufficiency	novel true defects can initially resemble noise
Benchmark leakage	Miner learns hidden answer keys	rotation, generated variants, delayed reveal, access controls, temporal splits	insider leakage remains possible
Validator copying	Validators mirror consensus instead of evaluating	weight commit-reveal where supported, independent challenge seeds, audit against revealed truth	shared scoring code can create common-mode correlation

Threat	Failure mode	Primary defenses	Residual risk
Validator collusion	Stake coalition favors miners or leaks truth	stake diversity, audit canaries, anomaly detection, public challenge reports, Yuma clipping	honest-majority / utility assumptions remain load-bearing
Publisher capture	Task author defines a captive private oracle	admissible-oracle rules, publisher bond, independent audit, competing publishers	domain-specific source access may remain concentrated
Oracle poisoning	Public or adjudicated reference is wrong	version pinning, reliability measurement, multiple sources, appeal / retirement	authoritative sources can be systematically wrong
Privacy leakage	Raw or derived data reveals secrets	minimum disclosure, policy gates, local retention, privacy budgets, confidential compute	metadata inference and side channels cannot be eliminated
TEE compromise / misuse	Attested code is vulnerable or operator controls inputs/outputs	measured image, key-release policy, minimal output, patch governance	attestation proves environment identity, not semantic correctness
Denial of service	Expensive challenges exhaust miners or validators	cost budgets, sampling, publisher fees, rate limits, timeouts	adversary can still reduce liveness during spikes
Sybil / capital concentration	Economic power dominates evaluation diversity	native registration and stake mechanics, delegation diversity, challenge audits	Bittensor remains stake-weighted, not one-person-one-vote
Narrative / market capture	Token value outruns assurance utility	separate market and assurance reporting, paid-demand metrics, benchmark scorecards, bounded claims	markets can remain reflexive for long periods

What cryptographic integrity proves

Hashes, nonces, Merkle commitments, and tamper-evident Evidence protect the integrity of recorded bytes. They do not establish the truth of the recorded account:

> **A valid hash proves that committed bytes have not changed. It does not prove that the bytes faithfully describe reality.**

Cryptographic integrity supports Evidence. It does not replace source authentication, execution attestation, oracle quality, or independent recomputation.

The following commitment construction is a design proposal. Before it can be normative, the SN87 Technical Specification must define canonical serialization, domain separation, field order, length-prefix encoding, Merkle leaf and node rules, and rejection of nonconforming commitments. No accepted implementation is established here. In the proposed construction, `LP(x)` denotes an unsigned-length prefix followed by the canonical bytes of field `x`; the specification must fix the encoding and field order for each version:

$$D_E = \text{ASCII(SN87:EVIDENCE-COMMITMENT:<spec-version>)} \parallel \text{LP (capsule version)} \tag{18a}$$

$$K_E = \text{LP (Pipeline ID)} \parallel \text{LP (canonical Path events)} \parallel \text{LP (source commitments)} \tag{18b}$$

$$M_E = D_E \parallel K_E \parallel \text{LP (policy)} \parallel \text{LP (nonce)} \parallel \text{LP (timestamp)}, \qquad C_E = \text{SHA256}\,(M_E). \tag{18c}$$

The specification must require rejection of duplicate keys, ambiguous number formats, unnormalized text, and noncanonical timestamps. A Merkle root may replace a large event list only once versioned leaf and node domains and a canonical tree rule are defined. Revealed leaves must bind the run, field, and ordinal context needed to prevent substitution. Those rules and their reference vectors remain companion-specification work; the equations alone do not define an interoperable encoding.

Canonicalization as an attack surface

A source-system adapter is strategically useful and technically dangerous. A homegrown loop may not have an explicit Orchestration, CPC, or Path vocabulary. The adapter must infer structure from telemetry. If it infers incorrectly, SN87 may deliver a mathematically precise assurance result about the wrong process.

The adapter should therefore emit a mapping manifest:

Element	What the reviewer needs to know
Source and mapping version	Which work system produced the record and which mapping rules were applied.
Source links	Which observed events support each mapped Signal, Path, or other object.
Mapping warrant	Whether a field was observed, inferred, or confirmed locally, and how confidence was assessed.
Omissions and loss	Which events remain unmapped and which distinctions the representation cannot preserve.
Approval and disclosure	Whether consequential interpretations were confirmed and what the policy permits a reviewer to inspect.

Low-confidence or high-consequence mappings should require local confirmation before emission-driving assurance. The source-system trace must remain reversible from the canonical representation where policy permits.

Appeals and dispute resolution

Assurance findings can affect regulated, financial, employment, or safety decisions. SN87 needs a dispute path:

1. work owner challenges a finding or score;
2. validator releases the committed challenge and oracle evidence after the confidentiality window;
3. independent validators re-score;
4. adjudicated rules expose reliability and disagreement;
5. the challenge class may be corrected, deprecated, or reweighted; and
6. the original artifact remains immutable but gains a linked disposition.

SN87 should never rewrite history to appear correct. It should make correction reconstructable.

PART III

Adoption, Economics, and Evaluation

Provenonce, Oresund, and the Open Standard

The proposed scoring path is only one part of an assurance market. Adoption depends on systems being able to submit bounded evidence, consumers finding the result useful, and experiments showing value beyond existing review methods.

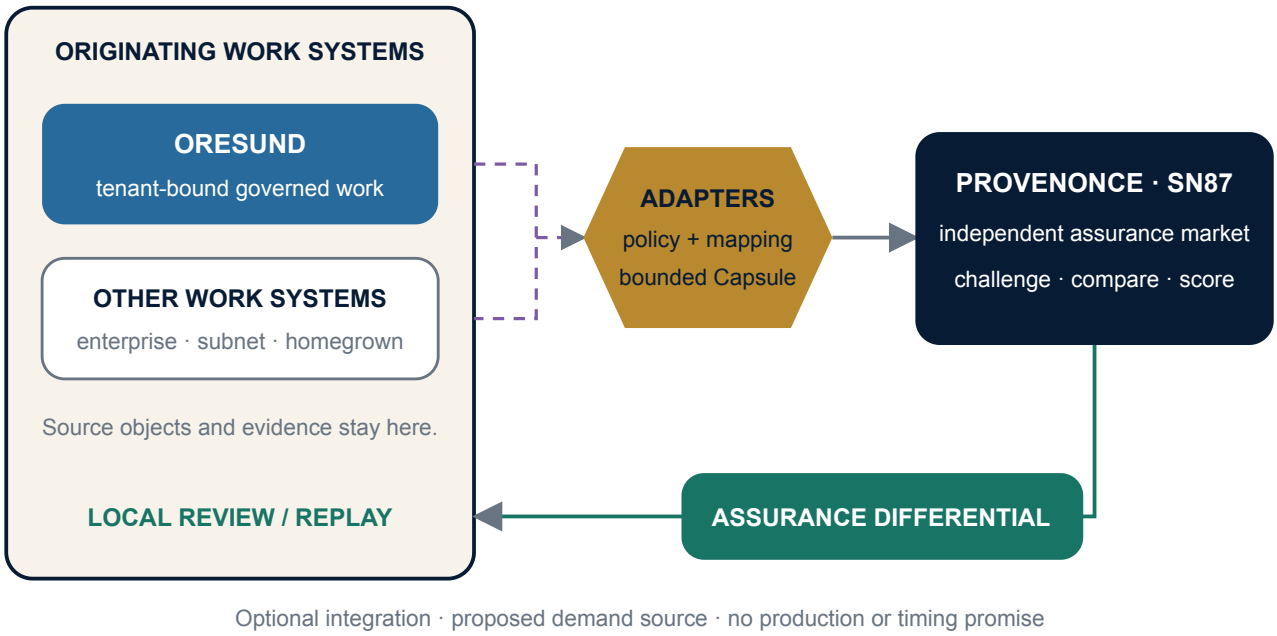
Provenonce proposes Proof of Assurance for Bittensor Subnet 87 and contributes **Provenonce Canon** as one reference vocabulary for governed work. The shared Capsule, Differential, challenge, oracle, privacy, benchmark, and scoring interfaces are intended to remain open to other work systems and assurance methods. Neither the company nor any one application controls access to SN87.

In addition to SN87, Provenonce operates a **System of Operations** for the agentic enterprise, with **Oresund** as its kernel. Oresund is a reference implementation for tenant-bound Orchestrations, Canonical Path Contracts, Paths, Pipelines, and evidence-bearing runs. Provenonce intends to connect Oresund to SN87 as one upstream source of enterprise demand for independent assurance. That intention is not a promise of production integration, demand, performance, or timing.

Oresund retains and governs the originating work. A permitted Capsule may cross the assurance boundary; an Assurance Differential may return to review and replay. Other enterprise platforms and homegrown systems may use the same open boundary through their own adapters. Every integration must expose omissions, inference, and mapping loss rather than claiming automatic equivalence.

Figure 8. Oresund Integration Example

Oresund is one source of governed work and proposed enterprise demand—not a gateway to SN87.



Oresund and other work systems may use the same open assurance boundary while retaining their source objects and accountability.

The standard is the shared assurance boundary, not one enterprise architecture. Capsule and Differential schemas, challenge and oracle interfaces, benchmark interfaces, privacy modes, and reference scoring should be open. An originating system may retain its own runtime, private data, and remediation process. The planned reference implementation and SN87 Technical Specification are companion artifacts; naming them here does not establish public availability.

Market Signal, Network Value, and Demand

Current mechanics and parameters must be reverified against official Bittensor documentation and the live target before any public claim about them. The SN87 Technical Specification must record the release-specific versions, configuration, and verification evidence. Historical papers support the architectural argument, not current network settings.

Subnet asset markets create economic signals. Those signals are not direct measurements of assurance quality. Price, liquidity, stake, attention, and emissions may diverge from benchmark hardness, truth quality, miner diversity, calibration, useful defect yield, privacy performance, and recurring paid demand.

SN87 should therefore report two ledgers. The market ledger observes network conditions. The assurance ledger measures the assets the assurance market seeks to compound: fresh challenge supply, credible ground truth, independent method diversity, explained validator disagreement, calibrated confidence, cost per material finding, and decision impact. Separating these ledgers makes it possible to ask whether market attention is accompanied by useful assurance.

A challenge class is economically defensible only when the combined value of avoided loss, independent discovery, reusable benchmark capital, and learning exceeds compute, latency, oracle, privacy, and coordination cost. Candidate buyers include enterprises, other subnets, benchmark publishers, auditors, insurers, regulated operators, and Provenonce customers, but these are proposed demand sources. No revenue, token, fee-volume, buyback, treasury, or price outcome is promised.

For each candidate class, compare human review, one frontier model, a centralized multi-model ensemble, and SN87 competition. Report severity-weighted recall, false-positive cost, evidence sufficiency, calibration, unique material issue yield, latency, oracle cost, disclosure cost, and total cost per defensible finding. Durable demand is established only by recurring willingness to pay for results that change a decision or control outcome.

Adoption and Interoperability

SN87 should feel like an assurance capability, not a requirement to learn network identities, evaluation windows, or validator routing. The adapter observes an approved process boundary, constructs a Capsule, enforces disclosure policy locally, negotiates supported challenge classes, and returns a finding with evidence and recommended action.

OpenTelemetry and platform-native traces can provide transport-level inputs, but they do not by themselves encode the governing orchestration, valid path, policy context, evidence sufficiency, or disclosure boundary. SN87's proposed assurance grammar maps those semantics into the Capsule. A source that records them explicitly may require less reconstruction; an adapter must disclose where it infers context or cannot establish an equivalent field. Oresund, enterprise platforms, and homegrown loops must each make that mapping and its uncertainty inspectable. A native vocabulary alone does not prove interoperability or assurance quality.

Capability advertisements are claims, not proof. Validators should test supported challenge classes, disclosure modes, domain labels, response limits, and attestation with canaries. Mapping confidence and the provenance of every derived field belong in the Capsule.

An Open Invitation

This paper defines a proposed design and the questions needed to test it. Its common objects, challenge boundary, response contract, oracle classes, scoring path, privacy modes, and evaluation gates give specialists a concrete basis for criticism and collaboration.

Workstream	Work to undertake
Incentive mechanism	Test whether rewards select for material, evidence-backed discovery rather than verbosity, imitation, or strategic abstention.
Validation and ground truth	Build hidden, renewable challenge supply; reliable oracle classes; benchmark governance; and transparent disagreement analysis.
Market and liquidity mechanics	Explore how demand, fees, stake, emissions, and cost budgets can support useful assurance without turning price into a proxy for truth.
Security and anti-gaming	Pressure-test collusion, leakage, Sybil behavior, evaluator copying, common-mode models, adversarial Capsules, and canonicalization attacks.
Mining methods	Develop diverse approaches to defect discovery, evidence construction, calibration, counterfactual generation, abstention, and cost control.
Ecosystem openness	Make adapters, schemas, privacy boundaries, and benchmark interfaces usable across enterprise systems, subnets, model providers, and homegrown agents.

Privacy engineering and oracle design cut across every workstream. Neither can be added after the market is functioning.

The invitation has three levels. **Validate** the problem statement, assumptions, and experiments. **Improve** the grammar, threat model, benchmarks, and incentive design. **Undertake** a workstream when there is enough shared conviction to build and test it. Participation does not require agreement with the proposed design; useful criticism is part of the assurance mechanism being proposed.

> **Do not centralize the answer. Decentralize the contest that discovers it.**

Proposed Economics

The economics are an experiment in paying for independent assurance, not a token or revenue forecast. A candidate challenge fee may include expected compute, oracle/reference cost, privacy overhead, storage, settlement, and audit cost. The publisher declares the budget and the validator samples within it; costly

disclosures require a separate policy gate.

Decentralized evaluation duplicates some computation. It cannot be assumed cheaper than a single call. The stronger hypothesis is that a shared market can amortize benchmark, validator, and method-development costs across work owners and reward methods that find different high-severity defects. The relevant unit is cost per material, defensible finding—not cost per model call.

Any relationship between service demand, subnet asset value, and participant economics requires its own evidence. It must not substitute for evidence that the service loop works. Proposed parameters, aggregation choices, and fee functions remain uncalibrated hypotheses.

Experimental Program and Evaluation

The first experiment must be able to reject the central hypothesis. Structured request/response exchange is necessary but insufficient: competition must produce reliable assurance value beyond centralized baselines.

Testing the assurance hypothesis

The program must be capable of falsifying the hypothesis. The proposed first test uses a deterministic challenge family with self-verifying and reference-execution truth; its criteria and stopping rules must be preregistered before results are observed.

The hypothesis must be tested against locked centralized baselines under comparable evidence, compute, and disclosure budgets. Preregistered criteria must cover uncertainty, clean-control false positives, privacy, cost and latency, operational independence, and stopping conditions. Detection without supported evidence, additional verbosity, or a gain explained by a larger resource budget is not evidence that independent competition improves assurance.

Challenge instances and answer keys must remain hidden during evaluation. A reproducible report must identify the challenge distribution, source and mutation provenance, evaluator and baseline identities, applicable oracle reliability, missing observations, and known attacks. It must report incremental material yield and uncertainty alongside costs and failure modes, including cases in which the centralized baseline performs better.

The hypothesis is weakened or rejected when gains disappear under leakage controls or independent replication, false-positive or privacy costs exceed the preregistered limits, rankings are unstable, or methods optimize the proxy without improving the intended work. A result supports only the tested challenge classes and disclosure conditions; wider domain, privacy, and economic claims require separate evidence.

Public Limitations

Cost and variance. Multiple miners, validators, oracle queries, and privacy controls can add latency and cost. Some tasks are better served by a deterministic local check or one trusted evaluator.

Privacy. Metadata can reveal business patterns; selective evidence can expose sensitive facts; confidential computing carries implementation and side-channel risks. A TEE can attest to an environment, not semantic correctness.

Stake concentration and common-mode behavior. Bittensor consensus is stake-weighted. Independent node count does not prove independent judgment, and shared code or infrastructure can create correlated failure.

Adaptation pressure. Open scoring encourages improvement but also benchmark overfitting. Hidden fresh instances, mutation families, temporal splits, canaries, and challenge retirement are continuing operational requirements.

Domain portability. The Differential can be general while truth and scoring remain domain-specific. Success in deterministic software or infrastructure tests does not establish medical, legal, financial, or other semantic correctness.

Open questions for collaboration

The following research questions remain open:

1. Which deterministic challenge families produce the highest unique material yield per unit cost?
2. Which aggregation rule best resists proxy gaming without hiding failure behind averages?
3. How should validator challenge quality and independence be measured?
4. What disclosure-price function causes minimally sufficient evidence requests?
5. What evidence would justify extending assurance beyond the challenge classes already tested?

These questions concern assurance performance and design; no commercial outcome is implied.

Conclusion

Autonomous capability is advancing faster than institutional assurance. The proposed response is to make bounded execution legible, preserve evidence, and create an independent contest around three questions: what did this execution miss, what proves the claim, and what should happen next?

Provenonce proposes that contest for SN87 as a Bittensor-native open standard and market. Work stays in its originating system. A Capsule exposes only the approved semantics. Miners return evidence-backed Differentials. Validators test them against declared truth classes and submit weights. Bittensor supplies the coordination and incentive substrate; SN87 remains responsible for challenge integrity, scoring, privacy, benchmark governance, and useful outcomes.

Provenonce contributes the reference vocabulary and this assurance proposal. Oresund is one possible source and consumer of assurance; other systems may use the same boundary. Digital exhaust is the motivating externality. SN87's proposed object of assurance is a bounded execution account—not the exhaust itself.

Advancement requires evidence from preregistered comparisons, independent replication, and explicit accounting for uncertainty, false positives, privacy, cost, and latency. A successful experiment supports only the tested conditions. Broader privacy, adjudication, market, and domain claims remain hypotheses until separately evaluated.

> **Find what execution missed. Prove it. Improve the next run.**

That is the proposed commodity. Whether it deserves a durable market is the experiment.

Acknowledgements and provenance

This paper draws on Provenonce Canon, the reference vocabulary for governed work, and a body of protocol design, oracle analysis, and experimental planning preserved in the project's governed source record. Will O'Brien developed the manuscript and underlying assurance proposal. Chris Zacharia contributed editorial review and protocol advice. The source record preserves the lineage of these contributions; private materials are not offered as public evidence of implementation or results. Responsibility for this manuscript rests with its author.

Technical Appendices

Formal Object Model

Core objects

Assurance Capsule

$$a = A_{\pi}(S, O, \mathcal{C}, \mathcal{P}_{\text{obs}}, L, E, \Gamma).$$

Required properties:

- versioned;
- source-bound;
- policy-bound;
- minimally disclosed;
- reconstructable within policy;
- explicit mapping confidence for external systems; and
- compatible with one or more declared challenge classes.

Assurance Challenge

$$q_{v} = (a, \chi, h_{v}, n, t_{\text{exp}}, \Omega).$$

Required properties:

- challenge-class version;
- hidden-state commitment;
- nonce and deadline;
- oracle bindings;
- scoring parameters;
- disclosure modes;
- cost budget; and
- reveal / audit policy.

Miner Differential

$$x_{i}(q) = (\zeta_{i}, \Delta_{i}?, A_{i}?, \hat{\mathcal{P}}_{i}?, \mathbf{p}_{i}, \mathcal{R}_{i}).$$

Required properties:

- exactly one response state;
- structured findings only for `FINDINGS`;
- Evidence references when findings are asserted;

- confidence per claim;
- alternative Paths only where supported;
- first-class `NO_MATERIAL_DEVIATION` and `INSUFFICIENT_EVIDENCE_ABSTAIN` states;
- a nonempty rationale for every state; and
- signed response metadata.

Assurance Finding

A user-facing finding should identify the challenge, state the material claim, distinguish severity from confidence, name the oracle class, link supporting Evidence, disclose limitations, and recommend an action within the consumer's authority. A counterfactual Path is optional.

For example, a structural challenge might establish that required human review did not occur before an external action. Its Evidence would bind the missing review gate to the relevant trace; its limitations would state whether content semantics were assessed. The consumer could then investigate and decide whether replay from the review gate is appropriate. This illustrates a finding's structure, not a configured severity or confidence value.

Proposed Signed Request / Response Contract

A transport-independent contract separates the content being signed from the mechanism that carries it:

Object	Required content	Conditional content
Request	Protocol version, challenge and class identifiers, Capsule, visible scoring policy, hidden-state commitment, nonce, expiry.	Additional evidence access permitted by the disclosure policy.
Response	Protocol version, challenge identifier, exactly one response state, nonempty rationale, authenticated identity and freshness binding.	Findings and their Evidence/confidence; supported counterfactuals; evidence-request record; additional commitments.

Authentication binds sender, intended recipient, request context, canonical content, and freshness so a valid record cannot be substituted or replayed in another context. An implementation must follow the network's supported authentication mechanism rather than infer one from this conceptual contract (Bittensor n.d.-c).

Large Evidence stays in the originating system or policy-controlled storage; the response carries commitments and bounded material. Schema invariants are explicit: `rationale` must be nonempty for every response state; `findings` must be nonempty if and only if `response_state=FINDINGS`; `findings` must be null or empty for both `NO_MATERIAL_DEVIATION` and `INSUFFICIENT_EVIDENCE_ABSTAIN`; and every `FINDINGS` response must provide either confidence on each finding or a nonempty `confidence_report`. Findings without Evidence references fail the integrity gate. Counterfactual paths remain conditional and may be absent when unsupported.

Proposed Weight-Setting Algorithm

```
inputs:
    challenge responses for epoch t
    revealed oracle truth
    protocol parameters θ

for each miner i:
    scores_i = []
    weights_i = []
    for each challenge q assigned to i:
        weights_i.append(q.difficulty * q.freshness)
        if response_missing(i,q) or integrity_gate(i,q) fails:
            scores_i.append(0)
            continue
        rule_scores = evaluate_declared_rules(i,q)
        dimension_scores = reliability_weighted_dimensions(rule_scores)
        if no_grounded_quality_dimension(dimension_scores):
            scores_i.append(0)
            flag_challenge_for_review(q)
            continue
        quality = bounded_geometric_aggregate(
            applicable_dimensions_only(dimension_scores),
            smoothing="max(score, epsilon)")
        penalty = exp(-response_indexed_costs(i,q))
        scores_i.append(quality * penalty)

    if sample_or_eligibility_floor_fails(i) or sum(weights_i) == 0:
        S_vi = UNAVAILABLE
        z_vi = 0
    else:
        S_vi = weighted_winsorized_mean(scores_i, weights_i, tail_fraction)
        z_vi = max(S_vi - theta, 0) ** gamma

Z_v = sum_j(z_vj)
if Z_v > 0:
    W_vi = z_vi / Z_v
    submit_weight_row(W_v, finalized_chain_configuration)
else:
    record NO_VALID_WEIGHT_ROW
    follow pinned safe policy; do not fabricate or epsilon-normalize weights
```

Implementation requirements:

- deterministic fixed-point or numerically specified behavior;
- versioned normalization;
- omit missing dimensions and renormalize only applicable dimension weights;
- explicit epsilon values used only inside bounded logarithms, never normalization;
- reproducible random seeds for audit;
- tests for empty oracle classes, correct clean controls versus abstention, false-positive cost, absent quality dimensions, empty perturbation families, missing responses, quantile ties, zero denominators, insufficient samples, and no-valid-weight behavior;
- submission behavior consistent with the network's supported rules;
- no dependence on local wall-clock except committed deadlines; and
- public reference vectors.

Glossary

The full reference vocabulary is preserved below. Return to the working vocabulary.

Group	Term	Meaning in this paper
Work	**Work system**	A model, platform, agent loop, or governed operating system that executes a bounded process.
Work	**Work owner**	The party that controls the process and its evidence, authorizes disclosure, and remains accountable for resulting decisions.
Work	**Tenant**	The customer, account, or workspace boundary within which governed work and its access rules are organized.
Work	**Signal**	An event or ingress that requires a governed response.
Work	**Orchestration**	The tenant-bound plan and institutional context governing a response.
Work	**Canonical Path Contract (CPC)**	A durable contract defining a valid Path.
Work	**Path**	A governed executable capability, or one execution of that capability.
Work	**Pipeline**	The run itself—not the stored plan.
Work	**Evidence Bundle**	Source-bound material showing what happened, what was inferred, what was held, and what remains replayable.
Work	**Review / replay**	Inspection of a recorded execution or reconstruction under declared conditions, used to learn from and improve work.
Assurance	**Execution evidence**	Records, traces, and artifacts made interpretable against declared requirements, provenance, and a disclosure boundary.
Assurance	**Assurance Adapter**	A source-side component that maps permitted context into the shared boundary and declares omissions, inference, and mapping loss.
Assurance	**Assurance Capsule**	The minimum structured account of an execution that the work owner permits SN87 to inspect.
Assurance	**Assurance Challenge**	A bounded test with a declared challenge class, scoring policy, disclosure mode, and oracle binding.
Assurance	**Challenge class**	A versioned family of tests sharing claim types, oracle rules, scoring semantics, and disclosure limits.

Group	Term	Meaning in this paper
Assurance	**Challenge instance**	One test drawn from a class, with its own input, hidden truth, nonce, and deadline.
Assurance	**Assurance Differential**	A structured response declaring findings, no material deviation, or insufficient evidence/abstention, plus only the conditional fields the evidence supports.
Assurance	**Counterfactual**	A better Path or continuation proposed only where the evidence and challenge class support it.
Roles	**Publisher**	An author of versioned challenge classes, truth bindings, admissibility rules, and benchmark material.
Roles	**Miner**	A participant that analyzes a challenge and returns an Assurance Differential.
Roles	**Validator**	A participant that samples challenges, verifies responses, measures performance, and submits a miner preference row.
Roles	**Assurance consumer**	The party that receives the result and decides whether to accept, investigate, replay, or change future work.
Evaluation	**Oracle**	A declared truth source: computation, pinned public source, reference execution, or adjudicated key.
Evaluation	**Oracle binding**	The versioned connection between a rule and the truth source used to judge it.
Evaluation	**Residual uncertainty**	A visible question without a defensible oracle; it is excluded from emission-driving scores.
Evaluation	**Clean control**	A challenge instance intentionally containing no material defect.
Evaluation	**Calibration**	The measured relationship between stated confidence and observed correctness.
Evaluation	**Hidden state**	Instance truth and sampling information withheld during the contest and disclosed for audit under the declared policy.
Evaluation	**Canary**	A controlled test or reference response used to detect leakage, copying, or scoring failure.
Evaluation	**Goodhart pressure**	The risk that optimizing a rewarded measure stops improving the property it was meant to represent.
Privacy	**Disclosure policy**	The work owner’s rules for what information may leave the originating system and under what conditions.

Group	Term	Meaning in this paper
Privacy	**Disclosure mode**	The method of access authorized for a challenge: metadata, selective evidence, or confidential computation.
Privacy	**Commitment**	A cryptographic binding to bytes retained elsewhere; it proves integrity, not truth.
Privacy	**Nonce**	A challenge-specific value used with identity and freshness checks to prevent reuse of a response in another context.
Privacy	**Attestation**	Cryptographic evidence about a measured execution environment, assessed against a verifier’s policy.
Privacy	**Trusted execution environment (TEE)**	An isolated execution environment intended to limit access to code and data; its guarantees depend on the platform and threat model.
Network	**Hotkey**	A Bittensor operational signing identity for a participant.
Network	**Metagraph**	A view of registered subnet participants and associated network state.
Network	**Subtensor**	The blockchain layer that records Bittensor state and applies its network rules.
Network	**Epoch**	A bounded evaluation window over which responses are aggregated.
Network	**Weight row**	A validator’s normalized preference over miners, or an explicit no-valid-row state.
Network	**Yuma Consensus**	Bittensor’s chain-side mechanism for aggregating validator preferences under its consensus rules.
Network	**Emissions**	Network-distributed incentives; they are not a measure of truth or assurance quality.
Network	**Commit-reveal**	A mechanism that commits to a value before revealing it, used where supported to reduce direct copying.

References

Amazon Web Services. n.d.-a. “Cryptographic Attestation.” *AWS Nitro Enclaves User Guide*. Accessed September 21, 2026. https://docs.aws.amazon.com/enclaves/latest/user/set-up-attestation.html.

Amazon Web Services. n.d.-b. “What Is Nitro Enclaves?” *AWS Nitro Enclaves User Guide*. Accessed September 21, 2026. https://docs.aws.amazon.com/enclaves/latest/user/nitro-enclave.html.

Autio, Chloe, Reva Schwartz, Jesse Dunietz, Shomik Jain, Martin Stanley, Elham Tabassi, Patrick Hall, and Kamie Roberts. 2024. *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. NIST AI 600-1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.600-1.

Bittensor. n.d.-a. “Commit-Reveal-Weights-Enabled Hyperparameter.” *Bittensor Documentation*. Accessed September 21, 2026. https://www.bittensor.com/docs/hyperparameters/commit-reveal-weights-enabled.

Bittensor. n.d.-b. “Emissions.” *Bittensor Documentation*. Accessed September 21, 2026. https://www.bittensor.com/docs/concepts/emissions.

Bittensor. n.d.-c. “Signed Requests.” *Bittensor Documentation*. Accessed September 21, 2026. https://www.bittensor.com/docs/guides/signed-requests.

Bittensor. n.d.-d. “Yuma Consensus.” *Bittensor Documentation*. Accessed September 21, 2026. https://www.bittensor.com/docs/internals/consensus.

Braintrust. n.d.-a. “Evaluate Output Quality with Scorers and Classifiers.” *Braintrust Documentation*. Accessed September 21, 2026. https://www.braintrust.dev/docs/evaluate/write-scorers.

Braintrust. n.d.-b. “Observe Your Application.” *Braintrust Documentation*. Accessed September 21, 2026. https://www.braintrust.dev/docs/observe.

Braintrust. n.d.-c. “Set Up Human Review.” *Braintrust Documentation*. Accessed September 21, 2026. https://www.braintrust.dev/docs/annotate/human-review.

Buterin, Vitalik. 2014. “A Next-Generation Smart Contract and Decentralized Application Platform.” *Ethereum Whitepaper*. https://ethereum.org/whitepaper/.

Google Cloud. 2026. “Confidential VM Attestation.” *Confidential Computing Documentation*. Updated September 18. Accessed September 21, 2026. https://docs.cloud.google.com/confidential-computing/confidential-vm/docs/attestation.

Juvenal. n.d. *Satires*, VI.347–48. Latin text. The Latin Library. Accessed September 21, 2026. https://www.thelatinlibrary.com/juvenal/6.shtml.

Nakamoto, Satoshi. 2008. “Bitcoin: A Peer-to-Peer Electronic Cash System.” https://bitcoin.org/bitcoin.pdf.

OpenTelemetry Authors. 2025. “Documentation.” *OpenTelemetry*. Last modified August 29. Accessed September 21, 2026. https://opentelemetry.io/docs/.

OpenTelemetry Authors. n.d. “GenAI.” *Semantic Convention Attribute Registry*. Accessed September 21, 2026. https://opentelemetry.io/docs/specs/semconv/registry/attributes/gen-ai/.

Plato. 1974. *Republic*. Translated by Desmond Lee. 2nd ed. Harmondsworth: Penguin.

Rao, Yuma. n.d. “Bittensor: A Peer-to-Peer Intelligence Market.” *Bittensor Whitepaper*. Accessed September 21, 2026. https://www.bittensor.com/whitepaper.

Tabassi, Elham. 2023. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1.

Timo. 2024. “The Bittensor Standard: Towards P2P Computational Capitalism.” *Substack*. March 22. Accessed September 21, 2026. https://timo37.substack.com/p/the-bittensor-standard. Also reproduced by Bittensor: https://www.bittensor.com/content/the-bittensor-standard.

How to cite

To cite this version:

> O’Brien, Will. 2026. *Proof of Assurance: An Open Bittensor Market for Evidence-Backed Assurance of Autonomous Work*. Version 0.7.2, Final Candidate, September 21. Provenonce, Inc. Contributor and Protocol Advisor: Chris Zacharia, Founder, bitstarter.ai.

Use a section heading and, for the PDF, a page number when referring to a specific passage.