Ogong: Permissionless Verified Inference Without a Correctness Bond

A mechanism for trust-minimized language-model serving over untrusted hardware

Andrew Lee
andrew@joseon.com

June 2026

Abstract

Networks that buy inference from untrusted operators must detect a provider who substitutes a cheaper model, serves stale outputs, or returns degraded results. The prevailing designs either (i) pair cryptographic per-request verification with a large *slashable correctness bond*, raising a capital barrier that excludes casual supply, or (ii) abandon the bond for subjective quality scoring, which is gameable. We show this is a false choice forced by *expensive* verification. Recent verifiable-inference primitives, locality-sensitive activation commitments and log-probability spot-checks, verify a request at a small constant fraction ρ of its generation cost, cheap enough to audit *nearly every* request rather than a small sample. We prove that under escrowed payment and indistinguishable audits, honesty is a provider's best response precisely when the effective detection margin q satisfies $q \geq \Delta/(p+\Phi)$, where Δ is the per-request compute saved by cheating, p the fee, and Φ the franchise value of continued participation. Market viability already forces $\Delta < p$, so driving $q \to 1$ via full-coverage verification makes a *zero* bond and even a *zero* franchise sufficient: forfeiture of the escrowed fee on the cheated request alone deters. This frees stake to serve priority and availability, and reassigns Sybil resistance to a non-capital, hardware-rooted cost that bounds an operator's *fraction* of identities by its share of physical throughput, which we show reduces verifier-collusion to the standard honest-majority threshold. The construction is modality-agnostic: the same mechanism covers diffusion and codec models through a trajectory commitment whose single-step re-check preserves the small cost ratio, so the zero-bond result carries from text to image and audio. The same cheap verification applies *per segment*, and with no per-shard bond to multiply, a model too large for any single node is sharded across a *cohort* of untrusted commodity GPUs, each segment independently re-executed, signed, and paid for the layers it served: a bonded design multiplies its capital barrier by every shard, but at zero bond a cohort of casual nodes serves a frontier model trustlessly, which we build and validate end to end. We machine-check the economic claims in the Z3 SMT solver and the Lean proof assistant, model-check the deterrence game in PRISM-games, verify the settlement protocol in ProVerif and Tamarin, and implement and measure the full audited path on a production engine: an honest re-run reproduces the commitment exactly while a substituted model is rejected by both the log-probability and the sharper hidden-state signal. We claim no new cryptographic primitive; the contribution is the mechanism design that composes existing inference-verification primitives into a permissionless verified-inference network in which honest providers risk *zero capital*, a design we have not seen elsewhere.

1 Introduction

Trust-minimized inference is the defining problem of decentralized compute. A provider advertises a 70B-parameter model; an 8B substitute saves roughly an order of magnitude in compute while

emitting superficially fluent text. Detecting the substitution is difficult because language-model inference is *not* bitwise reproducible across hardware: concurrent batching changes the reduction order of floating-point accumulations, flipping near-tie arg max decisions that then cascade into divergent continuations. Consequently the naive protocol, re-run the request and compare the bytes, rejects honest providers and is unusable on any throughput-oriented server (§2).

The literature offers two postures, each carrying a cost we argue is unnecessary.

Crypto-verify-and-slash. Zero-knowledge proofs [26, 34], refereed re-execution [4], sampling games [59], and deterministic-inference attestation [3] each couple a verification mechanism to a *bonded stake*, in the restaking lineage [17], that is destroyed on a detected fault. The bond is sized to exceed the attacker's expected gain; honest operators must therefore lock substantial capital before serving a single request.

No-bond-but-subjective. Bittensor's Yuma consensus [39, 46] lets operators "turn on and earn": validators score outputs and rewards are *diluted* toward consensus rather than slashed. The verification is, however, subjective quality ranking, not a commitment that the claimed computation occurred, and is gameable by a sufficiently good cheaper model.

Our thesis is that the dichotomy is an artifact of verification cost. If verification is cheap enough to cover nearly all requests, the bond becomes redundant *for correctness*, stake is freed to buy the priority operators actually want, and a separate non-capital cost handles Sybil resistance. The primitives we use are prior work; the contribution is the mechanism that assembles them so the system attains a property, zero-capital permissionless serving, that none of the parts provides alone.

Contributions.

1. A verification abstraction parameterized by coverage, soundness, false-positive rate, and cost ratio (§3), with the security gradation made explicit: cryptographic (attestation), statistical (activation/log-probability), and future cryptographic-without-TEE (zkML).
2. A *deterrence theorem* (§4, Prop. 1) giving the exact honesty condition, and its corollary that escrowed-fee forfeiture under full-coverage verification deters with zero bond and zero franchise (Cor. 1).
3. A reassignment of stake to priority and availability, and a non-capital Sybil mechanism whose proof-of-distinct-GPU work doubles as verification duty and bounds an operator's identity fraction by its throughput share (§5, Prop. 6).
4. A concrete protocol instantiation (§7) that extends a deployed per-reply signature scheme into a signed model-identity-and-verification commitment, with the audit lifecycle and message formats given explicitly.
5. A *modality-general* verification (§3.1): a trajectory commitment with a single-step re-check ($\rho \approx 1/N$) extends the zero-bond result from text to diffusion and codec models, implemented and measured on audio, image, and video engines, with the single-step service endpoint and the validator's automatic dispatch wired end to end.
6. *Verified split inference* (§7): the same cheap per-segment verification, with no per-shard bond, lets a model too large for any single node be served by a *cohort* of untrusted commodity GPUs, each segment independently re-executed, signed, and paid for its layers. A bonded design cannot reach this at casual scale, since its capital barrier multiplies by every shard. Built and measured

end to end: a two-shard cohort reproduces the monolithic model at relative $\ell_2 \sim 10^{-5}$, with per-shard settlement enforced on-chain under a conservation invariant.

7. *Machine-checked* foundations (§6): the algebraic results are proved valid in the Z3 SMT solver and mechanized in Lean 4 (including the Ville maximal inequality behind the false-ejection bound), the deterrence game itself is model-checked in PRISM-games (the optimal policy reproduces the honesty boundary), and the settlement protocol's authentication and settle-once properties are verified in ProVerif and Tamarin.
8. An *implemented and measured* reference system (§6): the full audited path runs end to end, and on a live substitution pair an honest re-run (same build) reproduces the commitment while a substitute is rejected by the sharper hidden-state signal.
9. A positioning against eight contemporary systems showing the composition is, to our knowledge, unclaimed (§9).

What is proven, measured, trusted, and forecast. We separate these up front, because the contribution mixes registers and the reader should know which is which. *Proven* (machine-checked): the deterrence algebra, the per-segment (split-inference) deterrence corollary, and the martingale false-ejection bound (Z3, Lean 4), and the protocol's authentication and settle-once properties (ProVerif, Tamarin), none of which establishes that the served computation is correct; that is statistical. *Measured* (single reference engine, the regime stated at each claim): the verification separations s, ε, ρ, whose *cross-hardware* tail is the central deployment gate, not yet a settled result (§10.1). *Trusted* (external roots): TEE-tier soundness rests on the Intel/NVIDIA attestation chain; honest-majority on the validator committee. *Forecast* (behavioral/economic): adoption, the cold-start budget, and token value. The full ledger is §10.1; we flag it here so the machine-checked results are not read as covering the empirical and trust assumptions they do not.

2 Model

2.1 Actors and observables

The network, which we call Ogong,[1] is open and permissionless. It has three infrastructure roles, all permissionless and any of which a single operator may run:

- *Providers* serve inference behind a NAT-traversing tunnel. "Turning it on" is the entire onboarding step, which the zero-capital property of §4 makes viable.
- *Routers* are attested enclaves on the request hot path: they match a request to a provider and relay it, latency-optimized. They hold no consensus stake but are attested and slashable for misrouting. The routing and anonymization layer runs *inside* the enclave (Intel SGX / TDX), so the party operating a router runs verified code that cannot read the plaintext it routes: a consumer checks the router's attestation before trusting it, making prompt confidentiality a property of attested code rather than an operator's promise not to log. Routers are permissionless like the other roles, the founding operator runs the first ones to bootstrap and anyone may run more; this is the architecture from the outset, not an opt-in privacy mode added later.
- *Validators* are the security layer: attested enclaves that drive the randomness beacon, assign and adjudicate audits, and finalize verdicts, reputation, and slashing under consensus. They post the

[1] Ogong is the Korean name of the Monkey King, who sees through any disguise with his fiery golden eyes, the namesake of the *Golden Eyes* audit layer (§3). Subsystems take English names from the same source.

one slashable bond the system retains (§4); they are not on the latency-critical path and need no GPU (§7).

Consumers, arbitrary applications and their users, of which the authors' own hosted product is merely one, pay a per-request fee held in escrow and released to the provider only after the request survives its audit window. No actor is privileged: the protocol, not any operator, governs routing, settlement, and slashing.

2.2 Trust tiers

We distinguish two hardware regimes that yield qualitatively different guarantees and that we label honestly to users:

	Confidential (TEE) tier	**Verified tier**
Privacy	enclave (host blind)	host sees plaintext (labeled)
Correctness	hardware attestation	statistical model-identity
Per-request guarantee	hardware-attested[†]	probabilistic
Hardware	TDX / SEV-SNP / NVIDIA CC [44]	any GPU; Apple Silicon

[†] "Hardware-attested" is trust-rooted in the TEE vendor's (Intel/NVIDIA) signing infrastructure and an unbroken enclave (§10.1), a hardware-trust guarantee, not a bare cryptographic proof. The tier is a property of the *provider's attestation*, not a separate backend: a provider that presents a valid confidential-computing quote (NVIDIA CC under a dstack-style enclave) serves the TEE tier, and a plain GPU serves the verified tier. So a user's choice among a local machine, a confidential-enclave backend, and a verified-cloud backend is a choice over *one* permissionless provider pool, the members differentiated only by the attestation they present and the per-request verification it enables (§3). Any operator presenting a valid confidential-computing attestation joins as an ordinary provider; nothing in the protocol special-cases it, and the trust tier follows the serving provider's attestation rather than any brand.

2.3 Nondeterminism (measured)

On an Apple M1 Max serving a 26B mixture-of-experts model at greedy decoding, output is bit-identical run-to-run within a *fixed* batch context but changes when the request is co-batched with concurrent traffic: we observed a genuine arg max flip (*physical* $\rightarrow$ *mechanical*) induced solely by batch shape. Byte-exact verification of *served* output is therefore impossible on a batching server. All verified-tier checks below are rank- or distribution-based and survive reduction-order drift. We quantified the surviving honest drift directly under concurrent load (four slots, continuous batching): the logprob total variation stays at $\sim 10^{-4}$, and the hidden-state distance stays small. We commit the hidden state as a *sign-random-projection* sketch (a fixed bank of dense random directions, identical for provider and verifier) rather than the provider's magnitude-top-k components, which resolves two problems at once. A magnitude ranking is ill-conditioned at positions with no dominant outlier dimension: a different batch shape reshuffles the top-k, and a verifier re-selecting its own false-rejects an honest provider (we observed the per-window distance spike to 0.17, past a 0.1 threshold). And committing provider-*chosen* components lets a substitute match only those. Dense fixed projections have neither failure mode: every direction mixes all coordinates, so the comparison is well-conditioned *and* the checked subspace is not the provider's to choose. Over this sketch the honest distance is near-zero on the same build and sits orders of magnitude below a substitute (~ 1, measured Gemma-2 2B versus 9B).

2.4 Adversary

A provider is rational and risk-neutral, maximizing fees minus compute cost. The attack surface comprises model substitution, quantization fraud, stale/cached responses, partial execution, lazy or colluding verification, and Sybil identity inflation. We state the trust assumptions explicitly.

Assumption 1 (Validator honest majority)**.** A strict majority of validator stake is honest; consensus over routing weight, reputation, and slashing is taken under stake-weighted median aggregation, so a stake minority can neither frame an honest provider nor shield a cheater.

Assumption 2 (Unbiasable coordination randomness)**.** Audit-target selection and verifier assignment are driven by a randomness beacon that no single party can predict or bias before providers act, realized as a threshold-BLS signature over a committee key established by a no-dealer distributed key generation (with a per-validator VRF aggregate as a bootstrap fallback), so that no individual node (and no single compromised enclave) can steer it. Verdicts are written to a tamper-evident, on-chain-anchored log. The attested router enclave is *defense-in-depth*, not the sole root of trust; the consequences of enclave compromise, which degrades confidentiality but not this randomness, are analyzed in §10.

Assumption 3 (Escrow and audit window)**.** A request's fee is released only after an audit window during which the request may be selected for verification; a rejected request's fee is not released.

Assumption 4 (Audit indistinguishability)**.** A provider cannot, at serving time, distinguish a request that will be audited from one that will not. Audits may be *retroactive*: a logged past request is scored later.

Assumption 4 is what forbids the classic emissions-test cheat (serve the real model only when watched); it is realized by scoring randomly chosen logged requests and by making audit jobs wire-indistinguishable from ordinary requests, following the task-indistinguishability of Wang et al. [56]. The assumption is idealized, and the result degrades gracefully rather than cliff-edges under imperfection: if a provider can guess audited requests with advantage δ and cheats only on those it believes unaudited, the effective coverage falls from α toward $\alpha(1-\delta)$, so the honesty condition tightens to $q(1-\delta) \geq \Delta/p$ and is preserved for small δ by a little more coverage. Retroactive scoring of *logged* past requests is the structural defense, the audit decision is made after the fact from the signed record, so there is no serving-time signal to key on; honeypots and wire-indistinguishability raise δ's cost. Strong distinguishability (a reliable side channel) would break it, and is the assumption a deployment must engineer toward, not one we can prove away.

2.5 Where computation and trust reside

Inference is never on-chain. A blockchain replicates small state transitions across all validators; a forward pass over multi-gigabyte weights, emitting streaming tokens, is neither small nor cheaply replicable, and no degree of finality speed changes this. The chain carries only consensus-critical metadata, the provider registry and stake, payment escrow, audit verdicts, slashing, reputation, and an anchor (a hash) of each signed commitment of §7. None of this lies on the token-generation path, so settlement finality on the order of a second is immaterial against multi-second generation.

Routing, matching a request to a provider, is performed off-chain by the routers for latency, while audit selection and verifier assignment are driven by the validator beacon (Assumption 2) so they are auditable after the fact; the escrow lock and the eventual verdict are the on-chain events. No router or validator is individually trusted: their code is attested (a consumer verifies

what they run), they are slashable (routers for misrouting, validators for false verdicts), and any party may run either. The plaintext prompt never touches the chain, only a pointer and an escrow commitment do, so the chain observes metadata, not content, preserving the privacy posture of the TEE tier. We state plainly that a fully open validator set is the design target, not yet the deployed reality: bootstrapping necessarily begins with a small operator set and a governed model catalog, and decentralizes as independent attested validators stake in. The guarantees rest on Assumption 1, an honest validator-stake majority, not on any privileged operator.

Privacy versus on-chain verifiability. Confidentiality of the request, end-to-end encryption from consumer to provider, optionally onion-routed so no relay learns the consumer–provider pairing, is orthogonal to whether the *computation* is on-chain, and the two in fact pull against each other. Encrypting the prompt to a single provider is exactly what prevents the chain's validators from re-executing or natively verifying the output: they hold no plaintext. On-chain execution would instead require every validator to recompute the forward pass (wasteful, and by §2.3 not even reproducible) or to check a succinct proof of it (zero-knowledge inference [24, 26, 34], not yet practical at this scale); replicating multi-gigabyte encrypted requests into permanent consensus state is in any case the wrong use of the ledger. Our audit layer (§7) is the resolution: a sampled designated verifier, not chain-wide consensus, checks the commitment, so the system retains *both* confidentiality and verification. The chain may still *coordinate*, posting the request intent, escrow, and VRF assignment as transactions while the encrypted prompt travels off-chain to the assigned provider, which removes any trusted matching party at the cost of first-token latency; the forward pass remains off-chain in every variant.

3 Verification

We abstract a verification scheme by its security-relevant parameters and cite concrete instantiations for the constants, so the economic results below do not depend on any single primitive. That verification can be far cheaper than the computation it checks is a classic idea [21]; we quantify it for inference as the cost ratio ρ and show it is the lever that removes the bond.

Definition 1 ($(\alpha, s, \varepsilon, \rho)$-verifier)**.** A scheme is a pair $(\mathsf{Commit}, \mathsf{Verify})$. During inference Commit emits a commitment π. On audit, $\mathsf{Verify}(\text{prompt}, \text{output}, \pi, M)$ returns accept or reject at cost ρ times the cost of generating the output. It has *soundness* s against a class $\mathcal{C}$ of substitute computations: for an output produced by any $M' \in \mathcal{C}$ with $M' \neq M$, $\Pr[\mathsf{reject}] \geq s$ (a statistical bound over $\mathcal{C}$, not a worst-case bound over all bit-strings; the gradation is made precise below); *false-positive rate* ε: for an honest output, $\Pr[\mathsf{reject}] \leq \varepsilon$ over hardware nondeterminism; and is applied with *coverage* α, the fraction of requests audited.

Instantiations. Activation-commitment hashing [45] achieves $\rho \approx 10^{-2}$ (verification is a teacher-forced prefill, up to 100× cheaper than autoregressive generation), commitment size $|\pi| \approx 258$ bytes per 32 tokens, and, empirically across models and across A100/RTX-4090 hardware, $s \approx 1$ and $\varepsilon \approx 0$ for model, precision, and prompt substitution. We confirm the separation on our own engine, teacher-forcing the claimed tokens through the reference and measuring argmax agreement over four prompts. Against honest output at 100%, a substituted model of a different family agrees 51%, a same-family smaller model (1B versus 3B) 84%, and the same model at a different quantization (the hardest case) 94%. Single-request argmax thresholding thus cleanly catches the profitable cheat ($s{=}1, \varepsilon{=}0$ at threshold 0.8 for the cross-family substitute) but, on this crudest metric, not quantization fraud, which sits near the noise floor (§10). Two finer checks resolve it. The full

distribution check of Inference.net [27], total-variation distance over the committed top-k logprobs, separates the quant case: a cross-backend quant cheat scores total-variation 0.054 against 0.014 for honest cross-backend drift (same model, Metal versus CPU), with no per-request overlap. We add the *Kolmogorov–Smirnov statistic* (the sup-norm distance between the two top-k CDFs [33]) as a complementary signal, used as a fixed-threshold divergence rather than a hypothesis test (a single deterministic re-run has no sampling model, so it inherits none of KS's distribution-free critical values; the threshold is calibrated empirically), since a localized probability-mass shift that the averaged TV can dilute still registers in the CDF maximum; validated on the live engine across a clean f16 quantization ladder [20, 38] (a provider declaring the f16 weights but serving a cheaper quant), the KS statistic scores $\approx 10^{-4}$ on an honest re-run (argmax agreement 100%) and rises monotonically with the quant gap (0.14 at near-lossless Q8 through 0.45 at Q4, and 0.71 for a gross size substitution), every cheat clear of its 0.10 rejection threshold. (Both TV and KS are taken over the committed top-k support, so substitution of tail mass below rank k is outside their reach, exactly the blind spot the activation check below covers.) Sharper still is the hidden-state commitment, where we *depart* from the magnitude-top-k sketch of Ong et al. [45]: we materialize the last hidden state (the language-model head input) for each generated token and commit a *dense random projection* of it: a fixed public bank of ± 1 (Rademacher) directions, committing the real projected values and scoring relative-L_2. This is a Johnson–Lindenstrauss sketch [29], in the binary-coin form of Achlioptas [1], which preserves L_2 distance; it is *not* a sign/Hamming locality-sensitive hash [12]. Because every direction mixes all coordinates, no substitute can match a hand-picked subspace the way it can under provider-chosen magnitude-top-k, and the comparison stays well-conditioned. Measured live (Gemma-2 2B versus 9B), an honest re-run scores relative-L_2 0.0 against ~ 1.1 for the substitute, clear of the 0.10 threshold by an order of magnitude, where the logprob argmax on the pair agrees 92%. On a clean quant ladder (Qwen2.5-0.5B, the provider declaring f16) the projection distance is monotone: 0.016 at Q8, 0.065 at Q5, 0.093 at Q4_K_M, rising to 0.15–0.30 for Q4_0/Q3/Q2 and 0.22 for a same-width but different-method Q4. So the activation check independently rejects the *coarse* quant band (Q4_0/Q3/Q2, ≥ 0.15) and gross substitution, while on the *near-lossless* band (Q8 through Q4_K_M, which the activation distance leaves below its 0.10 threshold) the binding check is the logprob KS/TV statistic above: the two are redundant on the coarse band and complementary on the near-lossless one, together spanning the spectrum. Matching the output distribution does not entail matching internal activations, so this is the check that closes distillation mimicry (§10), and its per-request margin is wide enough to need no aggregation. The sequential test of Prop. 3 then serves the residual roles of bounding lifetime false ejection under partial coverage and catching a sufficiently-frequent cheater (those above the decision boundary of Rem. 3), not of rescuing a weak per-request signal. The logprob check's known blind spot, decode-time substitution, is covered by the log-probability KS statistic at each decode position [27]. Seed-synchronized verification despite nondeterminism [31] and the lightweight empirical re-running of Wang et al. [56] give related guarantees.

A measurement caveat governs all of these numbers, and we state it plainly: they are single-engine, mostly cross-*backend* on one host (Metal versus CPU), over short replies and a handful of model pairs, the easy regime. The null that actually sets the production threshold is honest drift across *different operators' hardware*: distinct GPU architectures, flash-attention on or off, bf16 versus fp16 accumulation, KV-cache quantization, and, for mixture-of-experts models, routing that can shift with batch composition; and it is the *tail* of that distribution, not its mean, that bounds false ejection. Establishing that the honest cross-hardware tail sits clear of the near-lossless cheat signal (serving Q8 while billing for f16, the 0.14-KS regime) is per-backend-pair calibration we treat as the central deployment gate (§10.1), not a settled result. Until that matrix is measured, the verified-tier threshold is set conservatively per backend pair, and weight-declaration plus pricing

remains the backstop, which is also why the confidential (TEE) tier, whose guarantee does not depend on this numerical separation, is the stronger basis for a hard correctness claim today.

Soundness gradation. We are explicit that verified-tier soundness is *statistical*: $s \approx 1$ is an empirical bound, not a cryptographic one, and a sufficiently faithful substitute can in principle reduce the per-request margin (the quantization-fraud case of §10), though committing the hidden state and not only the logprobs raises that bar from matching the output distribution to matching internal activations. The TEE tier provides *cryptographic* soundness via attestation [44, 50]; zero-knowledge inference [24, 26, 34] would provide cryptographic soundness without a TEE but is not yet practical at language-model scale. The economic analysis below is stated for a generic $(\alpha, s, \varepsilon, \rho)$-verifier and instantiated with the statistical constants.

Layering and dispute (*Golden Eyes*). The verification subsystem, which Ogong calls *Golden Eyes*, runs audits cheapest-first: the commitment check [45], then the decode-position test [27], then, only on a disagreement, a refereed bisection [4, 49] that localizes the divergent operation using reproducible operators, identifying the liar without a full re-run. Assumption 2 resolves the "who verifies the verifier" regress: because the validator beacon assigns verifiers by an unbiasable VRF and verdicts are logged under consensus, a lazy verifier cannot selectively pass its confederate, and overlapping VRF-assigned audits yield peer-consensus without an infinite verification hierarchy. Output watermarking [32] is complementary but orthogonal: it embeds a provenance mark in generated text without certifying *which* model produced it, so it cannot by itself detect a substitution.

3.1 Beyond text: diffusion and codec models

The $(\alpha, s, \varepsilon, \rho)$-verifier of Def. 1 is modality-agnostic; only the per-request *check* is instantiated per architecture, and the economic results below depend only on ρ being small, not on the modality. Three cases:

- **Autoregressive token models** (text LLMs; also the autoregressive backbone of audio models such as a speech model's language-model stage): the logprob commitment above applies directly. This includes *image-input* vision-language models, which are autoregressive text generation conditioned on image embeddings, so no new primitive is needed: each generated token's hidden state attends over the image, hence the existing hidden-state commitment binds correct image processing. We confirmed this on a 2B vision-language model: the same image reproduces the committed hidden state exactly, while a different image moves the first generated token's hidden state by a substitute-level distance (0.30), so serving a different image, no image, or a weaker vision encoder is caught.
- **Diffusion / flow models** (image generation; audio diffusion) produce no per-token distribution. They iteratively denoise a continuous latent over N steps. The analog check is a *trajectory commitment*: the provider commits a Merkle root over $(\text{step}, \text{latent digest})$ at sampled denoising steps plus the final latent; a verifier, given committed latent z_t, runs *one* reference denoising step and checks $\hat{z}_{t+1} \approx z_{t+1}$ within a tolerance band, and checks $\mathsf{decode}(z_{\text{final}}) \approx$ output. The single-step re-check is the diffusion analog of the single-position teacher-forced check, at cost $\rho \approx 1/N$; the tolerance band plays the role ε does for text, calibrated per architecture (seeded scheduler + request-derived seed make the trajectory reproducible up to bounded numerical drift).
- **Non-autoregressive text diffusion** (e.g. block-diffusion language models that denoise a token canvas in parallel) emit neither a left-to-right distribution nor a continuous latent, so they

would need a hybrid of the two checks above: a *token-trajectory commitment*: a Merkle root over (step, canvas, per-position distribution) at sampled refinement steps, where a verifier re-runs one refinement step and checks that the locked tokens match and the per-position distribution stays within a total-variation tolerance, again at $\rho \approx 1/N$. This is a direct combination of the trajectory structure of the diffusion case with the discrete per-step check of the autoregressive case; we describe it for completeness but, unlike the audio, image, video, and (autoregressive) text modalities below, have not yet instrumented a block-diffusion text engine, so we leave its measurement to future work.

- **Hybrid models** verify per stage: the autoregressive backbone by logprob commitment, the diffusion head and decoder by trajectory and decode checks.

With $\rho \approx 1/N$ small, full-coverage audits stay affordable and the zero-bond result (Cor. 1) carries across modalities: *only the per-step primitive is modality-specific, not the mechanism.* This requires emitting the engine's own trajectory, which a black box cannot, so it is available precisely because the provider runs an instrumented engine (§7). Image inference has been verified before, by heavyweight zero-knowledge proofs [30], by generic optimistic re-execution [13], and by output fingerprinting [57]; what is new here is the lightweight, diffusion-specific trajectory commitment whose single-step re-check catches step-skipping and substitution at $\rho \approx 1/N$. We have implemented and measured the audio case: instrumenting the denoising loop of our own ggml audio engine (a 3.5B diffusion-transformer flow model) to emit the per-step latent, an honest re-run reproduces the trajectory exactly (per-step relative $L_2 = 0$, so $\varepsilon = 0$), while a small (5%) perturbation of the conditioning, standing in for a substituted computation, diverges progressively through the trajectory to 0.27, cleanly separable at any reasonable tolerance. We also implemented the single-step re-check itself: from a committed latent z_i, one reference denoising step reproduces z_{i+1} to relative $L_2 = 0$ (against 0.52 for an unrelated step), confirming the $\rho \approx 1/N$ economics on the real engine, the validator re-runs one of N steps rather than the whole trajectory. (For a sampler with momentum across steps the re-check is exact at steps outside the guidance window and otherwise needs the committed sampler state.) We confirmed the same on the *image* modality with an independent engine: instrumenting the Euler sampler of a vendored Stable Diffusion implementation on a 1.5B latent-diffusion model, an honest re-run again reproduces the trajectory exactly (relative $L_2 = 0$ per step) while changing the prompt diverges progressively to 1.0. A third engine, a Wan latent video-diffusion model running on Apple Silicon, behaves identically: an honest re-run reproduces each committed step exactly and a fabricated step is rejected, so the trajectory primitive holds across three distinct diffusion engines and modalities (audio, image, video). As in the autoregressive case, these are same-build reproductions: a different GPU or kernel reproduces a latent to small *nonzero* drift, not literally zero, so the deployed accept threshold is set from the honest cross-backend tail rather than the same-build 0 reported here. We also measured the final-decode check on the audio decoder: decoding a committed final latent is deterministic and reproduces exactly (relative $L_2 = 0$), while a 2% perturbation of that latent changes the decoded output, so committing z_{final} binds the served audio. The validator-side adjudication of this check is implemented in the reference system on the same primitives as the text path: the verifier's per-step single-step re-check is bound to the committed trajectory root by Merkle inclusion before the tolerance is applied (§6), so a step scored against an uncommitted latent is rejected as tampered. The full audit loop is now implemented and validated end to end in the reference system, not only the adjudication. The diffusion engine exposes the single-step re-check as a service endpoint that an independent verifier drives *statelessly*, re-establishing the conditioning from the request alone and reproducing the committed step to relative $L_2 = 0$; the provider commits and serves the trajectory opening alongside the logprob commitment, for both job-based audio and synchronous video; and on audit selection the validator

dispatches the job to a verifier over the same transport as the text path, which re-runs the sampled steps and returns the per-step scores. End to end, an honest trajectory is audited and released while a tampered one is rejected. The full standalone treatment of this primitive (the threat model, the single-step soundness argument, the guidance-window subtlety, and the measurements across three independent engines) is the companion paper [36].

4 The economic mechanism

4.1 A per-request deterrence game

Fix a provider and a single request. The provider chooses $a \in \{\mathsf{H}, \mathsf{C}\}$: serve the reference model honestly at cost c_{H}, or cheat (substitute a cheaper model) at cost $c_{\mathsf{C}} < c_{\mathsf{H}}$. Write $\Delta := c_{\mathsf{H}} - c_{\mathsf{C}} > 0$ for the per-request compute saved. The fee is p, escrowed (Assumption 3). The request is audited with probability α by an $(\alpha, s, \varepsilon, \rho)$-verifier; by Assumption 4 the provider cannot condition a on whether this request is audited. On a reject verdict the escrowed fee is forfeited and the provider is ejected, forgoing a *franchise value* $\Phi \geq 0$: the present discounted net profit of remaining in the network. Providers are risk-neutral. We treat Φ as the equilibrium continuation value of honest play and analyze a single *one-shot deviation*, asking whether cheating on one request, given honest continuation thereafter, is profitable. This is the standard subgame-perfection criterion [22].

Proposition 1 (Honesty condition)**.** *Let $q := \alpha\,(s - \varepsilon)$ be the effective detection margin. Honest play is a best response if and only if*

$$q \;\geq\; \frac{\Delta}{p + \Phi}.$$

Proof. Including the continuation value, the expected payoff of honest play is

$$U_{\mathsf{H}} = (p - c_{\mathsf{H}}) + \Phi - \alpha\varepsilon\,(p + \Phi),$$

since with probability $\alpha\varepsilon$ an honest request is wrongly rejected, costing the fee p and the franchise Φ. Cheating yields

$$U_{\mathsf{C}} = (p - c_{\mathsf{C}}) + \Phi - \alpha s\,(p + \Phi),$$

as a cheat is rejected with probability αs. Then

$$U_{\mathsf{H}} - U_{\mathsf{C}} = (c_{\mathsf{C}} - c_{\mathsf{H}}) + \alpha(s - \varepsilon)(p + \Phi) = -\Delta + q\,(p + \Phi),$$

which is nonnegative iff $q(p + \Phi) \geq \Delta$, i.e. $q \geq \Delta/(p + \Phi)$. □

Remark 1 (Bonds merely substitute for the margin). If, additionally, a stake b is destroyed on rejection, the same computation gives the condition $q \geq \Delta/(p + \Phi + b)$, so $b \geq \Delta/q - (p + \Phi)$. A bond is therefore only a device to compensate for a *small* margin q; it is not fundamental. Bonded designs are large-b because expensive verification forced small α, hence small q.

4.2 Eliminating the bond

Market participation requires honest serving to be individually rational, $p \geq c_{\mathsf{H}}$ (otherwise no one serves). Since $\Delta = c_{\mathsf{H}} - c_{\mathsf{C}} \leq c_{\mathsf{H}} \leq p$ (strictly $\Delta < p$ whenever $c_{\mathsf{C}} > 0$ or $p > c_{\mathsf{H}}$, the only exception being the degenerate free-cheating, zero-margin point $c_{\mathsf{C}} = 0$, $p = c_{\mathsf{H}}$ where $\Delta = p$), we have $\Delta \leq p \leq p + \Phi$, so $\Delta/(p + \Phi) \leq 1$. The honesty condition is thus *satisfiable with* $b = 0$ whenever q can be pushed near 1. We stress that the relevant comparison is *per audited request*, not an average:

the zero-bond result needs $\Delta \le p$ on each request a provider might cheat, so a flat fee against query-dependent cost (Δ large on an expensive prompt served by a cheap substitute) can violate it locally. This is exactly why the marketplace quotes price per model and per output length (§4.8), tracking c_{H} so $\Delta \le p$ holds query by query; where pricing cannot track cost the residual margin is restored by a bond, the substitution of Rem. 3 that the cheap-verification regime otherwise removes.

Corollary 1 (Zero-bond, zero-franchise deterrence under full coverage)**.** *Suppose fees are escrowed (Assumption 3) and the verifier satisfies $\alpha(s-\varepsilon) \ge \Delta/p$. Then honesty is a best response with bond $b = 0$ and franchise $\Phi = 0$. In particular, with $\varepsilon \to 0$, $s \to 1$ and full coverage $\alpha \to 1$, forfeiture of the escrowed fee on the cheated request alone deters, independent of ejection or any Sybil cost.*

Proof. Set $\Phi = 0$ in Prop. 1: the condition becomes $q \ge \Delta/p$, which holds by hypothesis. Since $\Delta < p$, the right side is below 1 and is reached by $q = \alpha(s-\varepsilon) \to 1$. □

Corollary 2 (Per-segment deterrence: sharding preserves the bound)**.** *Let a model be served by a cohort whose shards tile its layers, and let a shard run fraction $w \in (0, 1]$ of them. Cheating that shard saves $w\Delta$ (the compute saved scales with the layers it runs) and forfeits the proportional fee share wp (the cohort settlement splits the provider fee by layers, §7), against the* same *franchise Φ (a shard's stake is not split). Then the whole-model honesty condition implies the per-segment one:*

$$q(p+\Phi) \ge \Delta \implies q(wp+\Phi) \ge w\Delta.$$

Full-coverage verification that deters the monolith therefore deters every shard, at zero per-shard bond*: sharding does not weaken deterrence.*

Proof. From $\Delta \le q(p+\Phi)$, $q(wp+\Phi) - w\Delta \ge q(wp+\Phi) - w\,q(p+\Phi) = q\Phi(1-w) \ge 0$, using $q \ge 0$, $\Phi \ge 0$, $w \le 1$. The slack $q\Phi(1-w)$ shows a *smaller* shard is more deterred per unit of gain, because the franchise is not split across the cohort. Machine-checked in `proofs.smt2` (Cor 2). □

The standalone treatment of this result, with the machine-checked proofs (Z3 and Lean 4) and the non-capital, dual-use Sybil-resistance result, is the companion paper [37].

Proposition 2 (Dominant-strategy honesty; uniqueness; no provider collusion)**.** *Under audit indistinguishability (Assumption 4), the deterrence margin $U_{\mathsf{H}} - U_{\mathsf{C}} = -\Delta + q(p+\Phi)$ of Prop. 1 depends only on a provider's own action and the fixed audit parameters, not on any other provider's choice. Hence whenever $q \ge \Delta/(p+\Phi)$ honest play is a* weakly dominant *strategy, strictly so when the inequality is strict, and,* at a fixed audit margin *q, the all-honest profile is the unique dominant-strategy equilibrium: there is no collusive equilibrium in which providers jointly cheat at fixed q. Collusion that instead drives down the* shared *audit margin (by degrading verifier coverage, making q endogenous) is the separate verifier-coverage game, bounded by the throughput majority of Cor. 3.*

Proof. A provider cannot condition on whether its request is audited (Assumption 4) and audits are drawn per request, so U_{H} and U_{C} are functions of that provider's action alone; the sign of their difference is independent of the profile of other providers. A strategy that is a best response against *every* opponent profile is dominant (strictly, under strict inequality), and a strictly dominant strategy gives a unique equilibrium. The collusion-free claim is the contrapositive: a joint deviation leaves each member's per-request margin unchanged, so no coalition gains, an indistinguishable per-request audit cannot be relaxed by agreement. This is dominant-strategy incentive compatibility, the strongest such notion, and is why the result needs no equilibrium-selection or belief assumption. □

Remark 2 (Comparative statics of required coverage). The minimal deterring coverage is $\alpha^\star = \Delta/\big(p(s-\varepsilon)\big)$ (set $\Phi = 0$, $q = \alpha(s-\varepsilon) = \Delta/p$). It increases in the cheat profit Δ and the false-positive rate ε and decreases in the fee p and the soundness s: $\partial\alpha^\star/\partial\Delta > 0$, $\partial\alpha^\star/\partial\varepsilon > 0$, $\partial\alpha^\star/\partial p < 0$, $\partial\alpha^\star/\partial s < 0$. A sharper verifier (higher s, lower ε) or a higher fee thus buys deterrence at less coverage, and a more profitable cheat demands more, each sign as intuition predicts; the four monotonicities are machine-checked in `proofs.smt2`.

Why cheap verification is the enabler. Auditing every request ($\alpha = 1$) costs the network a fraction ρ of its total compute. The asymmetry is hardware-dependent but always small: $\rho \approx 10^{-2}$ on datacenter GPUs [45], and we *measure* $\rho \approx 0.05$ on Apple Silicon (M1 Max, Metal; gemma-4 26B-A4B Q4, 128-token replies: prefill ≈2.4 ms/token vs decode ≈49 ms/token, a ∼20× gap), the weakest backend we target.[2] So full coverage is a 1–5% overhead, and the regime $q \to 1$ of Cor. 1 is affordable, exactly what bonded designs built around expensive ($\rho \approx 1$) re-execution could not assume. This is the step that Alves et al. [3], starting from the same cheap-verification premise, explicitly declined: it *reduces* but does not eliminate slashing.

4.3 Bounding honest ejection

With $\alpha = 1$ and single-strike ejection, honest providers are ejected at rate ε per request, which is unacceptable unless ε is provably 0. We instead aggregate audits with a sequential probability ratio test [54, 55].

Proposition 3 (Controlled false ejection)**.** *Let each audit emit an independent log-likelihood ratio for the hypotheses "honest" (reject w.p. ε) versus "cheating" (reject w.p. s). An SPRT with ejection threshold chosen for target false-ejection probability β ejects an honest provider with probability at most β over its lifetime, while the expected number of audited requests to eject a persistent cheater is $O\big(\log(1/\beta)\,/\,D(s\|\varepsilon)\big)$, where D is the binary Kullback–Leibler divergence.*

Proof sketch. Under H_0 the accumulated likelihood ratio is a nonnegative martingale of unit initial value; Ville's inequality [53] bounds the probability that it *ever* reaches $1/\beta$ by β, so ejecting only when it does gives a lifetime false-ejection probability $\le \beta$ (this is the open-ended, never-resetting bound, stronger than a per-episode one). Under H_1, Wald's identity [54] gives, for a cheater *persisting* at induced reject-rate s, the expected stopping time $\mathbb{E}[N \mid H_1] \approx \log(1/\beta)/D(s\|\varepsilon)$, with $D(s\|\varepsilon) = s\log\frac{s}{\varepsilon} + (1-s)\log\frac{1-s}{1-\varepsilon}$ large for $s \approx 1, \varepsilon \approx 0$. □

Thus full coverage deters (Cor. 1) while honest ejection stays below a chosen β and a *persistent* cheater is caught in a handful of audited requests. The two bounds rest on different things: the false-ejection bound holds unconditionally under H_0 (it needs only that honest re-runs reject with probability $\le \varepsilon$), whereas the catch-time bound assumes a fixed cheat rate s; an *intermittent* cheater whose induced reject-rate lands in the indifference region is deterred per-request by Cor. 1, not by ejection (Rem. 3). The martingale bound assumes the per-audit likelihood ratios are *independent*; the windows within a single reply share co-batch numerical noise and are not, so the independent unit is the *reply* (one audit decision per `reply_id`, formed by aggregating that reply's window scores into a single verdict) rather than the per-window scores treated as separate trials.

Remark 3 (Decision boundary and intermittent cheating). The one-sided test ejects once the per-provider reject rate exceeds the zero-drift boundary $p^\star = \frac{\log\frac{1-\varepsilon}{1-s}}{\log\frac{s}{\varepsilon}+\log\frac{1-\varepsilon}{1-s}}$ (for the measured

[2] Measured with the `verify_commit` harness in the reference implementation; the prefill that scores a claimed reply is the teacher-forced verifier's cost, the decode is the original generation's.

$s = 0.9, \varepsilon = 0.02,\ p^\star \approx 0.37$). A persistent or sufficiently-frequent cheater rejects above $p^\star$ and is ejected as above; an *occasional* cheater whose induced reject rate lands in $(\varepsilon, p^\star)$ sits in the indifference region and is ejected only with intermediate probability, not the β guarantee (which holds at the honest rate ε). Tightening H_1 toward the suspected cheat rate, or a CUSUM, narrows this gap; in the measured hidden-state regime ($s \to 1, \varepsilon \to 0$) the boundary collapses toward 0, so even a single audited reject is conclusive and the indifference region vanishes. We verified this behavior in the validator's sequential test (honest providers bounded by β, cheaters above $p^\star$ ejected almost surely, the gradient between).

Remark 4 (Operative deterrent under the SPRT). Single-strike ejection (Prop. 1, where one reject costs Φ) and the SPRT are two regimes of one design. In the deployed regime a detected cheat *withholds that request's fee* immediately (Cor. 1, $\Phi = 0$) but does *not* eject; ejection occurs only when the SPRT statistic crosses threshold, at which point the franchise Φ is lost. Hence the *per-request* deterrent is the fee-forfeiture bound $q \geq \Delta/p$, while Φ is the cumulative reputational stake backing the eventual ejection of a persistent cheater; the two collapse to Prop. 1 exactly when the SPRT is set to eject on the first strike.

4.4 Stake reassigned

With the correctness bond removed, stake is repurposed:

- **Priority (pure upside).** Higher stake raises routing weight, hence volume and earnings; it is optional, and a zero-stake provider still earns on routed work.
- **Availability as a verified emission input.** Any node (provider, validator, or router) that is online and passes random liveness challenges earns a share of emission even before traffic is routed (§4.6); disconnection and missed service-level targets forfeit that share of *rewards* while offline (cf. 17), never principal.
- **Validators alone post a slashable bond**, for false verdicts; Yuma-style stake-weighted-median consensus [46] governs routing, reputation, and slash-on-consensus. Consumers pay in fiat; OGONG rewards are issued separately and only for *verified* contribution (§4.6), so the per-request correctness deterrent stays fiat-denominated regardless of the emission schedule.

4.5 Fee distribution

Each request's fee divides among the parties that produce the service: the bulk to the serving provider (which bore the GPU cost), a routing fee to the router, a settlement fee to the handling validator (sized so honest income exceeds a mis-verdict bribe), a verification reserve, a protocol cut (the vig, funding the treasury and development), and a small routing-privacy license fee (below). Two rules keep the split incentive-compatible. First, *verification is paid as work, not for verdicts*: a verifier earns a flat fee per audit it performs, *independent of the outcome*, so it is neutral as to whether it accepts or rejects. This, with the capped catch-bounty and false-accuser slashing of §8, removes any incentive to fabricate rejections. Second, a fee withheld from a caught cheat (Cor. 1) is *refunded to the consumer*, the wronged party, not paid to the catcher; the provider's loss, hence the deterrent of Prop. 1, is unchanged, while no party profits from a reject. Validators earn additionally through consensus alignment under Assumption 1: verdicts matching the stake-weighted consensus are rewarded, deviating ones diluted or slashed. Income is fees plus, during bootstrap, halving OGONG emissions for verified work (§4.6); as the network matures fees dominate, so steady-state revenue tracks real demand rather than inflation.

Model-creator attribution and royalty (deferred to governance). The model-id namespace `ogong/⟨tier⟩/⟨maker⟩/⟨model⟩` carries a creator attribution, and the settlement program reserves a maker-royalty slot in the split so a model's creator could share in the fees it earns. We deliberately leave that slot *inactive* (the maker share is zero at launch) because activating it turns on a question the protocol cannot answer mechanically and that no single operator should decide: *who may legitimately claim a model as theirs.* A naive permissionless royalty invites a leaker to register someone else's proprietary weights and collect on them, and the obvious fix, a trusted registry that vets uploads, reintroduces precisely the centralized gatekeeper the network exists to remove. A genuinely permissionless alternative is sketchable (stake-backed registration with royalties held in a time-locked escrow, challengeable by anyone who shows the model is a near-copy of an earlier registration using the same activation-similarity machinery the audit layer already runs, the challenger slashing the impostor's stake), but it protects only against on-network plagiarism, not against a clean leak of weights that were never registered. Because that residual, and the provenance policy it implies, is a community value judgment rather than an engineering default, we leave the maker slot for future on-chain governance to activate and parameterize, and ship attribution-without-payment in the interim.

Routing-privacy license fee. Every router practices the same proprietary primitive, hardware-enclave anonymization of network traffic (patent pending, VP.net), which is what makes the confidentiality of §2.5 a property of attested code rather than operator promise. The network licenses it from the inventor as a small, *time-bounded* fee of 2 basis points (0.02%) of settled volume, routed to VP.net in OGONG. We size and bound it deliberately so it cannot become a rent that undercuts the network's permissionlessness: it is a *fee*, taken from settlement and never from issuance, so the fixed cap of §4.6 is untouched and the token is not inflated; it scales with usage as a license should; and governance may only *lower* it, never raise it, in the limit to zero. Crucially it *sunsets*: a license to a patent should not outlive the patent, so the fee tapers automatically to zero at the earliest of patent expiry, the method being dedicated to the public, or the inventor ceasing to defend and develop it, and never runs past the patent term. Patents run roughly two decades, so the license retires on about the same horizon as the emission taper of §4.6: both are bootstrap-era scaffolding that the mature, fee-funded network sheds, not standing claims. It funds defending and developing the primitive every router depends on while the patent is live, and it confers no protocol control, governance remains token-weighted under Assumption 1. We flag two honest caveats. A payment to one party, even a bounded and sunsetting one, sits in tension with the no-privileged-operator posture of §2.5; the smallness, the automatic sunset, and the governance cap that lets token holders reduce it sooner (in the limit to nothing), are what keep it a bounded license rather than an open-ended tap. And a standing token-denominated revenue stream is relevant to the characterization of §4.7; a license fee paid *in* OGONG for a specific proprietary method is a B2B royalty to the patent holder, not a distribution to token holders, hence a narrower instrument than a claim on holders' returns, but the distinction is a legal matter, not one we resolve here.

4.6 The native token (OGONG): a settlement currency

OGONG (ticker $OGONG) is the network's single native token and its *settlement currency*: providers, routers, and validators are paid in OGONG, and it is simultaneously (i) the slashable validator bond and the dilutable provider priority stake (§4), (ii) the priority-allocation weight, and (iii) the governance asset over the catalog and parameters (α, the challenge rate λ, audit thresholds, fee splits). Consumers need not hold it: fiat gateways collect fiat (e.g. by card) and pay the network in OGONG on the user's behalf, preserving a zero-crypto-exposure path, while users who prefer

may pay in OGONG directly. Settlement flows through OGONG, so every unit of usage is a unit of demand for it.

Hard cap, no burn, "Bitcoin, but the work is useful". OGONG has a fixed cap and a Bitcoin-style [42] halving emission, issued *only for verified contribution*: verified work (inference that passed Golden Eyes, audits performed, requests correctly routed) *and* verified availability (random liveness challenges a *node* passes by being online and ready: a provider ready to serve its catalog, a validator ready to verify, a router ready to route). Emission accrues to all three operator roles, not providers alone. The availability term is what lets any operator turn its node on and begin earning immediately, before traffic is routed to it, the continuous-issuance property of a Bitcoin- or Bittensor-style schedule in which block rewards accrue regardless of instantaneous demand; we admit it as an emission input precisely because it is *verified*, not asserted. A liveness challenge is an ordinary request the prover cannot distinguish from real traffic (Assumption 4), audited on the same terms, so a node claiming readiness it does not have fails the challenge exactly as a cheat fails an audit. Faking either work or availability is therefore exactly cheating, which the audit layer catches, so emission-farming reduces to the attack the protocol already defeats: rewards follow proven contribution, not stake alone, avoiding the stake-captures-rewards failure documented for prior designs [39]. The availability share is drawn from the same finite emission tranche and so *decays with the halving*: it is a bootstrap subsidy that keeps spare capacity online while demand is too thin to pay for it, not a perpetual entitlement. As the schedule tapers, providers are funded by real service fees rather than issuance, and the network deliberately declines a perpetual tail emission: a fixed cap means the seigniorage paid by holders for that capacity is bounded and front-loaded, not a standing tax. We deliberately use *no* buyback-burn: under a fixed cap, burning is redundant (the cap already supplies scarcity) and, combined with growth, drives supply toward zero rather than the cap, a "disappearing-gold" artifact (`tokenomics_model.py`, regime A). Without burn, supply rises and *asymptotes* to the cap (the Bitcoin shape, regime A*: $S_\infty = S_{\text{genesis}} + E_0 H/\ln 2$, here S_{genesis}=1B, the 20% allocation, and the emission tranche $E_0 H/\ln 2$=4B, summing exactly to the 5B cap), and because OGONG is the settlement currency, value accrues from settlement demand against the fixed cap (the equation of exchange, below), so no burn is needed for value capture. The result is sound, scarce money whose scarcity is backed by intrinsically useful work and which, being the medium of exchange, *circulates* rather than merely being hoarded.

Value: the equation of exchange. Since OGONG is the network's settlement currency and unit of account (§4.8), its value is grounded in the medium-of-exchange identity $M \cdot V = R$ [19], not in a heuristic and not in a cash-flow claim. Writing R for the *real* (USD-measured) value of settlement throughput per period, the numeraire for purchasing power even though transactions denominate in OGONG, and V for velocity, the network value is $M = R/V$ and the unit price is $R/(V\,S_{\text{float}})$, where the float $S_{\text{float}} = S(1-\sigma)$ is circulating supply net of the fraction σ locked in provider priority stake and validator bonds. We address the standard velocity objection [48] head-on: a pure medium-of-exchange token is bought and immediately sold, so V is high and the price suppressed. The design answers it with *velocity sinks*, priority staking and validator bonds remove supply from the float and raise value by $1/(1-\sigma)$ at fixed throughput, while the fixed cap means growth in R accrues to a bounded supply. The comparative statics, value rising in throughput R, falling in velocity V, and rising in the staked fraction σ, are checked in `value_model.py`. We are careful about what this is: it is the token's *purchasing power as money*, the worth of holding a settlement currency in order to transact, set by usage demand against a fixed cap, exactly the equation of exchange that prices any monetary medium. It is *not* a forecast of price, a promise

of appreciation, or a representation of investment return; any movement in purchasing power is incidental to monetary demand, not a distribution from anyone's efforts. OGONG is a consumptive *utility* token under this model rather than a dividend instrument: the fee streams (the 2 bps license fee, the vig) accrue to specific parties, not pro-rata to holders, so a holder has no claim on others' earnings. The characterization is taken up directly in §4.7.

Proposition 4 (Endogenous velocity: the staking equilibrium)**.** *The locked fraction σ is not a free parameter but the equilibrium of a lock-or-hold choice. A holder keeps tokens liquid (transactional velocity V) or locks them as priority stake or a validator bond; stakers receive a share ϕ of settlement revenue R as priority returns, and the marginal staker locks until the staking yield $\phi R/(\sigma S\,price)$ equals the opportunity cost of capital r. With the equation of exchange $S\,price = R/(V(1-\sigma))$ on the liquid float, eliminating the price gives*

$$\sigma^\star = \frac{\phi V}{r+\phi V} \in (0,1), \qquad M = \frac{R}{V(1-\sigma^\star)} = \frac{R}{V}\Big(1+\frac{\phi V}{r}\Big).$$

So $\sigma^\star$ rises in the staker revenue share ϕ and the transactional velocity V and falls in the opportunity cost r ($\partial\sigma^\star/\partial\phi > 0$, $\partial\sigma^\star/\partial V > 0$, $\partial\sigma^\star/\partial r < 0$), velocity is pinned by primitives rather than assumed, and endogenizing the lock decision multiplies the medium-of-exchange value R/V by the staking multiplier $1+\phi V/r > 1$.

Proof. The two equilibrium conditions are the yield equality $\phi R/(\sigma S\,\text{price}) = r$ and the exchange identity $S\,\text{price} = R/(V(1-\sigma))$. Substituting the second into the first gives $\phi V(1-\sigma)/\sigma = r$, hence $\sigma^\star = \phi V/(r+\phi V)$; the value expression follows by substitution. The three sign claims and $\sigma^\star \in (0,1)$ are machine-checked in `proofs.smt2`, and `value_model.py` confirms the two value derivations agree. □

Remark 5 (The staking return is work income, not interest). The "return" ϕ to staking is not a passive yield paid on capital: staking buys *priority*, higher routing weight, hence more service volume, and the staker is compensated by the fees on the additional work it performs (§4.8). It is a positional benefit in a two-sided marketplace, economically a queue position that lets one sell more service, not interest on a deposit. This distinction is material to the characterization of §4.7.

Why cheaper inference is a tailwind, not a deflation trap. Inference is priced in OGONG, and its *real* price tracks the competitive unit cost of serving it (§4.8), a cost that falls secularly as energy, hardware, and model efficiency improve. This is the network's adoption flywheel, cheaper inference is more inference, but it is worth being precise about how it reaches the token, since a falling unit price taken alone would *shrink* settlement throughput R and bleed value. Writing demand as iso-elastic, $Q = A\,P^{-e}$, throughput is $R = P\,Q = A\,P^{1-e}$, so as the price P falls R *rises* exactly when the demand elasticity $e > 1$, is flat at $e = 1$, and falls only when demand is inelastic ($e < 1$). The $e > 1$ regime is the *Jevons paradox* [2, 28], the empirically robust pattern by which a cheaper general-purpose input is consumed more than proportionally, observed across coal, electricity, bandwidth, and cloud compute, and visible again in the recent record of inference, where order-of-magnitude annual falls in cost per token have been met with faster growth in tokens served. We are deliberate about the status of this claim: it is an *empirical* regularity, not a theorem, and it is regime-dependent, a mature, saturated market can turn inelastic, at which point this particular channel flattens. Two things keep that from threatening the design. First, the token *functions* without it: a fixed-cap settlement currency is sound money whenever R merely *persists* against bounded supply, so elasticity is what makes value *appreciate*, not what makes the system *work*, and flat R at a fixed cap still holds value. Second, R has a growth channel the elasticity argument

ignores, the *extensive* margin: even if demand for one modality saturates, settlement grows as the network adds modalities and use-cases (image, video, audio, agents, §3.1), so breadth of demand substitutes for depth. The supply side closes the loop, cheaper energy preserves provider margins at the lower real prices, so providers keep serving as prices fall and the two-sided market survives the decline (the cold-start subsidy bridges the early window before fees mature, `bootstrap_model.py`). The net claim is therefore the careful one: cheaper inference *plus* elastic-or-broadening demand raises the token's value, and cheaper inference alone, against the fixed cap, at minimum preserves it. Both the elastic and inelastic regimes are checked in `value_model.py`.

Genesis allocation and emission. OGONG has a fixed cap of 5,000,000,000 (5B) units. The dominant tranche, 80%, is *never pre-allocated*: it is emitted only for verified contribution on the halving schedule above, over roughly two decades, so the large majority of supply is *earned* through audited work and availability rather than granted or sold. The same permissionless emission is open to anyone, the founding team included, which holds *no* privileged emission and earns on identical terms to any other operator. The remaining 20% is the launch allocation:

Bucket	**Supply**	**Share**	**Terms**
Earned emissions	4,000M	80%	∼20yr halving; earned for verified work and availability by providers/validators/routers (anyone, team included)
Public liquidity	125M	2.5%	Bonding curve 2% (100M) + CEX listing reserve 0.5% (25M); permissionless curve → AMM, *proceeds locked*
Core team & advisors	625M	12.5%	3-yr vest; *non-controlling*
Foundation	250M	5%	3-yr vest; *independent* steward; transparent on-chain budget; no founder control

The team and Foundation allocations both vest linearly over *three years* on the same wall-clock schedule; the team allocation is additionally handled as a company-side compensation matter, legally distinct from the public distribution. Neither the team nor the Foundation allocation carries *any* control rights.

The launch distribution, stated plainly. Unlike a pure mined launch, the 2.5% liquidity tranche is made available to the public through a *permissionless* launch mechanism, a constant-product curve that migrates to an AMM, whose purpose is to *bootstrap settlement velocity*: a working medium-of-exchange market in which earned OGONG can circulate and be spent from day one, so that providers will *accept* OGONG as payment for work and consumers can *acquire* it to settle inference, establishing its function as the network's unit of account and settlement currency at launch. This is a liquidity and price-discovery bootstrap for a *usable* currency, not a capital raise and not a vehicle for holder appreciation. We do not characterize it as "not a sale." Two properties define it and bear on §4.7: (i) the *proceeds are locked*, they capitalize the on-chain liquidity pool and are never withdrawn by the issuer, the company, or the Foundation, so no party is *capitalized* by the distribution; and (ii) the mechanism is *permissionless and not issuer-seeded*, deployable by any independent party, with no solicitation of or negotiated terms with purchasers. Emissions release against per-epoch verified-contribution tallies (served work plus passed liveness challenges) finalized by validator consensus (§6), so the contribution-backed majority inherits the honest-majority security of the verdicts.

We state the premine plainly: a 20% launch allocation funds liquidity, listings, an independent Foundation, and the team, and we do not overclaim a "fair launch." The property we *do* claim, verifiable on-chain, is narrower and stronger: the dominant tranche (80%) is unspendable except

as settlement for verified contribution (served work and passed liveness challenges) the audit layer accepts; the launch proceeds are locked and capitalize no party; the team and Foundation allocations vest and carry no control; and the founding team holds *no* privileged emission, earning beyond its vested allocation only as an *operator* on the same permissionless terms open to anyone. The team's principal upside is *company* equity and the arm's-length IP license (§4.5), rather than a controlling token stake, so no participant holds unearned controlling supply at launch. OGONG launches as a capped-supply SPL token on Solana with staking, emission, and slashing in an audited program, and is designed to migrate to a dedicated chain via a canonical lock-and-mint bridge. Vesting is enforced by an audited token-lock program on a wall-clock schedule (not a block count); the exact terms and the legal and securities treatment are deployment matters that a launch will finalize.

Circulating supply. The unlock-and-emission schedule is modeled month by month in `token_schedule.py` (re-parameterized to the allocation above), composing the launch liquidity, the vesting of the team and Foundation allocations, and the halving emission. The launch float is the public-liquidity tranche (the 2.5%: the bonding curve plus the listing reserve as listings are confirmed), on the order of ∼2–3%; the team and Foundation allocations vest on a wall-clock schedule, and the earned majority accrues over the ∼20-year halving, so circulating supply rises from this modest base as emissions and vesting proceed and asymptotes to the cap. The script checks the schedule against the claims above, that the allocation sums to the 5B cap, that circulating supply is monotone and never exceeds it, and that its asymptote equals $S_{\text{genesis}} + E_0 H/\ln 2 = 5\text{B}$.

Protocol-controlled funds and stewardship. The network has a single protocol-controlled pool, the *vig-treasury*, funded by the protocol vig (§4.5), held on-chain and disbursed by token-weighted governance through a budgeted, public, on-chain process, never to a personal wallet. The *Foundation* holds the 5% steward allocation under an *independent* mandate, its signers and directors not controlled by the founders, and deploys it transparently for ecosystem purposes on a public on-chain budget; it does not direct the network. A steward whose role is thus limited and ministerial, coordinating and supporting rather than managing, does not reintroduce the efforts-of-others prong, the basis on which staff declined to recommend enforcement against a comparably situated network foundation [51]. No entity both controls the treasury and directs the protocol: governance over parameters, the catalog, and fee splits rests with token-weighted consensus from launch (§4.7). The cold-start provider subsidy is the *emission availability floor* itself (§4.6): a provider earns for being online and passing liveness challenges before fee revenue exists, so emission, not a discretionary treasury bucket, bootstraps capacity. We are candid that with a modest Foundation allocation and no large reserve, exploit make-whole at launch is limited; the escrow and no-double-spend guarantees (Prop. 5) bound the blast radius, and governance may direct the treasury to incidents once fees accrue.

Funding. Pre-revenue costs are met by VP.net and the founders from ordinary company financing (equity, e.g. a private placement to accredited investors under the usual exemptions, a company security distinct from OGONG), and the usage-funded *vig-treasury* (§4.5) becomes the steady-state budget as the network matures, the baton pass of §4.6. VP.net captures value as the network's arm's-length *IP licensor*, earning the routing-privacy license fee and the vig (§4.5), a company revenue stream rather than a OGONG holding. The vig accrues to VP.net for a fixed *five-year term* from launch and then reverts entirely to the on-chain vig-treasury under token-weighted governance, so this company revenue stream is time-limited by construction and the network's steady-state funding detaches from its originator, so the network's ongoing operation does not rest

on the company's continuing efforts. The founding team's upside therefore derives principally from company equity and the perpetual IP license rather than a controlling token stake or an indefinite protocol cut. The public launch proceeds are locked into on-chain liquidity and capitalize no party (§4.6), so the network's funding neither routes through an issuer selling OGONG for its own account nor depends on the continuation of any one company: OGONG is earned and used on a network that *lives on its own*.

4.7 Token characterization

OGONG is engineered for a *consumptive-utility* characterization: a settlement currency consumed for inference, not an investment contract. The prong-by-prong legal analysis (the US *Howey* test, the EU MiCA regime, the functional-network / no-identifiable-promoter-effort posture) is a matter for counsel, maintained separately. We note here only that we engineer for a utility characterization, do not represent the determination as settled, and review each jurisdiction for securities-law compliance before launch.

4.8 Pricing and the provider marketplace

Prices are *denominated and settled in OGONG*, the network's unit of account: a provider quotes in OGONG and is paid in OGONG, the same unit, so there is no quote-versus-settlement currency split and no spot conversion on the settlement path. This is deliberate. OGONG is the money of the network, and pricing *in* it, rather than quoting in dollars and settling in the token, frames it as a medium of exchange rather than a dollar-tracking instrument, the consumptive-utility posture of §4.7. A provider that wishes to insulate its margin, and the real value of the per-request deterrent of Cor. 1, from OGONG volatility may *peg* its OGONG price to a USD target through a price oracle, re-quoting as the rate moves; a rational provider does so, which keeps the forfeited-fee deterrent tracking the real resource value a cheat would save, so the honesty condition holds at unchanged coverage. The peg is the provider's *choice*, not a protocol denomination: the unit of account is OGONG. Providers set their own prices, so routing is a two-sided market and, crucially, *lowest price does not always win*. The router scores candidates on price, measured performance (latency, throughput, uptime), reputation (Golden Eyes audit-pass history and SLA record), and stake, optimizing to a consumer-expressed preference (cheapest / fastest / highest-quality / a blend). A provider with a clean audit history and strong latency can thus command a premium and still win routing for quality-seeking demand: the verification layer doubles as a *market signal*, honest high-performance providers earn more, and the race-to-the-bottom of a pure price auction is avoided.

4.9 Payment and fast settlement

Per-request micropayments are far too frequent, small, and latency-sensitive to settle on-chain individually, even at Solana finality, so the network meters off-chain and settles net on-chain, in the manner of payment channels. A consumer (or gateway) deposits OGONG into an *on-chain escrow*; the validator quorum *admits* a spending allowance by reserving it against that escrow (one consensus round); the consumer then *streams* nonce-ordered draws against the reserved allowance at sub-millisecond latency; net balances settle to Solana in periodic batches, with the audit window setting the cadence. Economic finality is immediate: a provider holds a reserved, consumer-signed claim enforceable against on-chain escrow, so it need not await a block, while chain finality ($\sim$0.4 s) applies only to the off-critical-path batch.

Money is not rooted in the TEE. Funds are custodied by the on-chain escrow program, never by an operator or enclave; a debit requires the consumer's signature; and net settlement is finalized by honest-majority validator consensus (Assumption 1) anchored on-chain. The TEE only *accelerates* (ordering and low-latency relay) and is not in the custody path. A compromised enclave can thus degrade liveness (censor, mis-order) or privacy but *cannot move or forge funds*; its worst-case financial impact is bounded to a single request's fee (below).

Proposition 5 (No double-spend of escrow)**.** *Under consensus-sequenced admission, the total reserved against a consumer's escrow E satisfies $\sum \text{reserved} \le E$ at all times, so every* admitted *payment is fully payable: no provider that served an admitted payment is left unpaid, irrespective of how many authorizations the consumer signs or how quickly. Surplus authorizations are rejected (unserved); a provider that serves an* un-admitted *authorization bears at most one request's fee, covered by the escrow buffer or its bond.*

Proof. The reserved total increases only on an admission, which the sequencer performs iff reserved + amount $\le E$; hence reserved $\le E$ is an invariant. The admitted payments are a subset whose sum equals reserved $\le E$, so all are payable. Verified over randomized concurrent double-spend attacks in `payment_model.py`: admission holds stiffed providers at 0% where a naive signature-only scheme stiffs up to 78%. □

5 Sybil resistance without capital

Corollary 1 deters cheating on *audited* requests via fee forfeiture. Two roles remain for an identity cost: protecting the integrity of audit sampling against an operator who controls many identities, and supplying margin when one chooses $\alpha < 1$. We discharge both without capital.

What Sybil actually buys. Under Assumption 2 verifier pairing is an unbiasable VRF draw, so idle identities earn nothing and cannot be *aimed* at one's own jobs. An operator profits only by holding a large *fraction* of all identities, so that self-pairing (verifying one's own served request) is likely. Hence the cost need not make a single identity expensive; it must bound an operator's identity *fraction*.

Timed proof-of-distinct-GPU. A validator issues to each identity, as an independent Poisson process of rate λ, a challenge requiring W floating-point operations answered within wall-clock T on the router's clock. We stress what we may *not* assume: identities are unlinkable, so the router cannot synchronize an operator's challenges, and an adversary may time-share one card across staggered challenges. The bound below is therefore an *average-load* (queue-stability) argument that needs no synchronization, not an instantaneous-capacity one.

Proposition 6 (Identity bound by sustained throughput)**.** *An operator of aggregate throughput F (FLOP/s, net of its serving load) that meets every challenge deadline can sustain at most*

$$k \;\le\; \frac{F}{\lambda W}$$

identities; at one challenge per identity per window ($\lambda = 1/T$) this is $k \le FT/W$. Claiming $k > F/(\lambda W)$ makes the expected challenge load $k\lambda W$ exceed the service rate F, so the deadline-miss backlog diverges and the operator is detected with probability $\to 1$ over successive rounds. The bound holds against arbitrary software modification (answering requires performing the operations) and against time-sharing (it constrains the time-averaged rate, not instantaneous concurrency).

Proof. Each identity imposes expected load λW FLOP/s (rate λ challenges of W operations); k identities impose $k\lambda W$. The operator is a work-conserving server of rate F; by the stability condition of such a queue, if the arrival rate $k\lambda W$ exceeds F the backlog grows without bound, forcing missed deadlines with probability $\to 1$. Meeting all deadlines in steady state therefore requires $k\lambda W \le F$. Operations cannot be forged, only performed, and since the constraint is on the time-averaged rate, staggering or time-sharing challenges across identities does not relax it, which is why no synchronization of an operator's (unlinkable) identities is needed. □

Remark 6 (Duty cycle is a real cost, not a free win). The challenge load λW is genuine compute that competes with the operator's serving work; the F above is throughput *net* of it. Raising the duty cycle λW tightens the identity bound but taxes honest providers equally, so λ trades Sybil-resolution against overhead, a protocol parameter, not free. Each challenge is itself a verification job (§7), so part of the tax is recovered as useful audit work. Near the stability boundary $k\lambda W \approx F$ detection is slow rather than immediate; the guarantee is asymptotic in the number of rounds, not single-shot.

Corollary 3 (Verifier collusion reduces to honest-majority)**.** *An operator's identity fraction is bounded by its share of network throughput, so the probability it verifies its own request, k/N, is at most its compute share. Driving self-pairing to a $\frac{1}{2}$ majority therefore requires $\sim$50% of network throughput, the same threshold as Assumption 1, and no separate capital barrier.*

Earned-trust ramp. New identities receive only low-value traffic under heavy audit; the earnings forgone while reputation accrues is the cost of cycling an identity and makes eject-then-rejoin non-free, restoring the franchise Φ of Prop. 1 as a real, non-capital quantity. An optional proof-of-work [5] on registration throttles minting.

Unification. The timed challenge of Prop. 6 *is* a verifier audit (re-running a peer's request). The Sybil cost is thus productive verification work, not wasted computation: the "useful work" ideal of [46], realized honestly.

6 Machine-checked foundations and reference implementation

The formal claims were checked by a companion script (`verify.py`; symbolic algebra in SymPy, Monte-Carlo in NumPy). Symbolically, the closed forms for $U_{\mathsf{H}}, U_{\mathsf{C}}$, the identity $U_{\mathsf{H}} - U_{\mathsf{C}} = -\Delta + q(p+\Phi)$, the honesty boundary $q = \Delta/(p+\Phi)$, the bonded variant $b = \Delta/q - (p+\Phi)$, and the Φ=0 reduction $q = \Delta/p$ (Prop. 1, Remark, Cor. 1) are all confirmed exactly. Numerically: across 300 randomized games the simulated best response agrees with $q \ge \Delta/(p+\Phi)$ with no off-boundary mismatch; the SPRT (Prop. 3) shows a measured lifetime false-ejection rate of $4\times10^{-4} \le \beta = 10^{-3}$ (the Ville bound) and an expected stopping time under H_1 of 2.5 audits versus the $\log(1/\beta)/D(s\|\varepsilon) = 2.2$ prediction (here s=0.9, ε=0.02); and the queue-stability identity bound (Prop. 6) is confirmed with a near-zero deadline-miss rate for load $\rho < 1$ and miss-rate $\to 1$ for $\rho > 1$, the transition occurring at the predicted $k^\star = F/(\lambda W)$. Five further companion models support the economic design: `tokenomics_model.py` derives the supply trajectories of §4.6 (regime A burns to zero, A* asymptotes to the cap, with value tracking usage); `value_model.py` grounds the token value in the equation of exchange $MV = R$, checks its comparative statics (value up in throughput, down in velocity, up in the staking sink $1/(1-\sigma)$), and solves the endogenous staking equilibrium $\sigma^\star$ of Prop. 4; `token_schedule.py` composes the genesis vesting (TGE unlocks, cliffs, linear release) with the halving emission into the month-by-month circulating-supply schedule, asserting that the allocation sums to the 5B cap, that circulating supply opens near $\sim$2–3% and is monotone and

bounded by the cap, and that its asymptote is $S_{\text{genesis}} + E_0 H/\ln 2 = 5\text{B}$; `bootstrap_model.py` models the cold-start security-budget bridge (§10), finding the minimum halving subsidy that keeps the validator budget above a mis-verdict's attack value at every horizon while the mature fee budget self-sustains; and `payment_model.py` confirms the no-double-spend invariant of Prop. 5 over randomized concurrent attacks (admission holds stiffed providers at 0% versus up to 78% for a naive signature-only scheme).

Beyond simulation, the algebraic results are *formally proved*. Each is checked as a quantifier-free real-arithmetic validity in the SMT solver Z3 (`proofs.smt2`): the deterrence equivalence (Prop. 1), bond elimination (Cor. 1), the SPRT log-likelihood drift crossing zero exactly at the decision boundary $p^\star$ (Rem. 3) with $0 < p^\star < 1$, the Sybil queue bound (Prop. 6), coverage feasibility, the four comparative-statics monotonicities of the required coverage $\alpha^\star$ (Rem. 2: increasing in Δ and ε, decreasing in p and s), the endogenous staking equilibrium $\sigma^\star \in (0,1)$ and its statics (Prop. 4: increasing in ϕ and V, decreasing in r), and the escrow-admission invariant of Prop. 5 (reserved $\leq$ escrow is preserved by the admission rule, so every admitted payment is payable, with the over-commit a naive scheme permits exhibited as a genuine satisfiable failure the rule removes), all discharged as `unsat` of the negation, i.e. valid for all parameters in range. The same theorems are additionally mechanized in Lean 4 with mathlib (`OgongLean.lean`) as kernel-checked proofs. The probabilistic core of Prop. 3 is mechanized too (`Ogong/Ville.lean`): we prove `ville_bound`, the Ville maximal inequality for a nonnegative martingale ($\mu\{\sup_{k\leq n} f_k \geq c\} \leq \mathbb{E}[f_n]/c$, composed from mathlib's Doob `maximal_ineq`), the design-critical identity that the likelihood-ratio increment has mean one under H_0 ($\varepsilon\frac{s}{\varepsilon} + (1-\varepsilon)\frac{1-s}{1-\varepsilon} = 1$, the condition that makes the ratio a martingale), and `sprt_false_ejection`, the bound itself as a corollary ($\leq 1/c = \beta$ at $c = 1/\beta$). The link previously left to standard development is now mechanized as well: `product_martingale` proves that a process with $M_{i+1} = M_i\, L_{i+1}$, where each increment L_{i+1} is independent of the past $\mathcal{F}_i$ and has mean one, satisfies the martingale predicate, by pulling the $\mathcal{F}_i$-measurable factor M_i out of the conditional expectation (`condExp_mul_of_aestronglyMeasurable_left`) and reducing the independent factor to its mean (`condExp_indep_eq`). So the entire probabilistic core of Prop. 3, the maximal inequality, the mean-one identity, the martingale construction, and the false-ejection bound, is kernel-checked end to end with no appeal to an unproved step. The protocol's authentication property is verified symbolically under a Dolev-Yao adversary in ProVerif (applied-π, `ogong_audit.pv`): the correspondence $\mathsf{Release}(P, r, o) \Rightarrow \mathsf{Serve}(P, r, o)$ holds, i.e. a fee is released only against the provider's own signature over the committed output (*release-authentic*). The *injective* once-per-serve form fails in the stateless model, exactly the record replay that the validator's reply-id deduplication prevents; that stateful settle-once property is then *verified* in Tamarin (`ogong_audit.spthy`), whose mutable per-reply state captures what ProVerif's monotonic abstraction cannot: Tamarin discharges no_double_settle (each reply settles at most once) and reject_excludes_release (a rejected audit never also releases the fee), alongside its own proof of release_authentic. The split is itself evidence of the mechanism's structure: authentication is cryptographic, once-ness is stateful. We are explicit about what these symbolic models do *not* cover: they establish the authentication and settlement layer (fees release only against authentic, non-replayed records), not the *correctness* of the served computation. That `commit_root` reflects the work actually performed is discharged statistically by the audit (soundness s, §3), and the beacon's unbiasability and the confidentiality seal are likewise outside the ProVerif/Tamarin scope, so the machine-checked results should not be read as an end-to-end correctness proof.

The economic *game*, not only its algebra, is model-checked. We render the per-request deterrence game (Prop. 1, Cor. 1) as a Markov decision process in which a forward-looking provider chooses Honest or Cheat on each of N requests to maximize its expected total payoff under coverage α, soundness s, false-positive rate ε, and ejection after K detected cheats, and synthesize the reward-

maximizing policy with the probabilistic model checker PRISM-games (`ogong_audit_game.prism`). The boundary emerges from exhaustive optimization rather than closed form: with ejection disabled the optimal policy flips from all-Cheat to all-Honest as α crosses $\alpha^\star = \Delta/(p(s-\varepsilon))$. At the calibration p=1, Δ=0.4, s=0.9, ε=0.02 ($\alpha^\star \approx 0.4545$), the synthesized optimum is all-cheat at α=0.45 but all-honest at α=0.455, bracketing the boundary to within the sweep step. A second model (`ogong_audit_sprt.prism`) replaces the stylized strike-count ejection with the *faithful* sequential test of Prop. 3: an integer-scaled log-likelihood ratio that moves by $+\ln(s/\varepsilon)$ on a reject and $-\ln((1-\varepsilon)/(1-s))$ on an accept and ejects at $\ln(1/\beta)$. Under it the cheating premium collapses well below the one-shot boundary, at α=0.4 a forward-looking cheat's advantage over honest play falls from ≈0.96 without ejection to ≈0.14 with the real test, and vanishes by $\alpha \approx 0.45$, which is exactly the dynamic, franchise (Φ) deterrent the SPRT supplies. This is the game-theoretic-model-checking analog of the protocol provers for the economic mechanism: the best response is verified over the whole strategy space, not argued only at the calculated boundary.

The decision logic these proofs cover is not only modelled but *implemented* and unit-tested as the validator core (`ogong-validator`): the VRF audit-coverage gate (Prop. 1), the per-provider SPRT with its decision boundary and ejection (Prop. 3), honeypot policing of lazy verifiers (the verifier's-dilemma defense), a hash-chained tamper-evident verdict log, and Ed25519-authenticated record intake whose canonical payload is the v2 record signed by the provider. The verdict itself is bound to the commitment: the validator never re-runs the model but adjudicates a verifier's per-window hidden-state and logprob scores only after recomputing each window's Merkle leaf and checking inclusion against the signed `commit_root`, so a score computed against material the provider never committed is rejected as tampered before any threshold is applied. Provider and validator share one definition of the commitment and the signed record (the `ogong-verification` crate), so the two sides cannot drift on what a root commits to; the sign-then-verify binding is exercised end to end as a cross-component test.

The full audited path is implemented and runs, not merely the decision core. A provider serves a reply, builds the commitment, signs the v2 record, and pushes it to a validator over QUIC (validators are attested backbone nodes, so the mesh uses raw QUIC rather than the home provider's Cloudflare-traversing fallback). The validator authenticates the record, draws the audit selection, dispatches a verifier, adjudicates the returned scores against `commit_root`, advances the SPRT, and appends to the log. The audit draw is an unbiasable threshold beacon. The committee establishes a single BLS key by a distributed key generation with no trusted dealer (Joint-Feldman / Pedersen [18, 47]): each validator deals a Feldman-verified secret sharing and the group key is the sum over the qualified dealers, so no party ever knows it. The beacon for an epoch is the *threshold signature* over that epoch; the per-reply audit seed is $\mathrm{H}(\text{beacon}(\text{epoch}) \,\|\, \texttt{reply\_id})$. We are precise about what this gives. For a *fixed* group key the threshold-BLS signature on an epoch is *unique* [over 7, 8]: any t-of-n valid shares combine to the same value, so no set of fewer than t corrupt parties can predict it and no honest-excluding withholder can move it: the remaining shares reconstruct the identical value, which is what eliminates the last-revealer residual of a bare VRF aggregate (a validator who, having seen its peers, eats a slash to suppress its single contribution). Two caveats we state rather than hide. First, plain Joint-Feldman does not yield a *uniformly* random key (a rushing adversary can bias its distribution [23]), but a beacon needs unpredictability, not uniformity, and Feldman-DKG suffices for threshold signatures (23, §4); the construction is the threshold-BLS beacon of Cachin et al. [10], as deployed by drand [15]. Second, an *equivocating* dealer (a valid share to some members, a bad one to others) must be excluded *consistently*, or honest validators finalize over different qualified sets and derive different keys. Soundly this needs (i) *verifiable* complaints (the accused dealer broadcasts the disputed share, checked by all against its public Feldman commitment, so one lying complainer cannot evict an honest dealer) and (ii) *agreement* on the qualified set, ratified by the validators'

BFT finality layer over a reliable-broadcast complaint round [9]. Our current implementation excludes pessimistically on complaint and is sound under the honest-majority assumption; the verifiable-complaint and agreed-QUAL rounds are the production hardening. The key is *preserved across committee rotation* by proactive resharing [25]: a quorum of the old committee re-shares its shares to the incoming one, Lagrange-weighted at $x{=}0$ so the sharing polynomial is re-randomized while its constant term (the group secret) is fixed, each reshare checked against the old member's public share g^{s_i}; the resharing math is implemented and unit-verified (the networked transport reuses the DKG layer and is not yet wired). The ceremony self-drives over QUIC under partial synchrony [16] (sealed deals (each share sealed to its recipient under a post-quantum X-Wing box [6]), a complaint round, then finalize), and the DKG is tested end to end on a local multi-validator cluster (the happy path; adversarial-network cases are future work), over the widely-used `blsttc` threshold-BLS library. The active beacon source is an epoch-indexed, committee-agreed value, so all honest validators derive the identical seed; until a committee's threshold key is established the audit path falls back to a per-validator Ristretto-VRF aggregate over (epoch, reply_id), verifiable and grind-proof (the input is fixed) and retained purely as a bootstrap, so the beacon is additive and the live path never regresses. The verifier itself is real: it re-executes the claimed tokens on a reference `llama-server` and computes the logprob and hidden-state distances (the reference path re-generates; the teacher-forced re-prefill of §3, validated to reproduce the hidden states, is the cost-optimized path). Measured live on a substitution pair (Gemma-2 2B serving, 9B verifying), an honest re-run reproduces the commitment to numerical zero (hidden relative-L2 and logprob total-variation both ≈ 0, same build) while the substitute is rejected at hidden relative-L2 ~ 1 and logprob TV 0.96, the hidden-state signal again the sharper discriminator. The diffusion/flow path is implemented in the same shape: the validator adjudicates a verifier's per-step single-step re-check against the trajectory root by the same inclusion-then-tolerance rule. Consensus over the verdict-log head is a BFT quorum certificate: each validator signs (height, head) with its attested Ed25519 key, a head is final once a stake-weighted $2f{+}1$ quorum (more than 2/3 of validator stake, $n{=}3f{+}1$ [11]) signs the same head, and a divergent minority cannot finalize; a local four-validator cluster reaches and verifies finality. This BFT log-finality threshold ($> 2/3$ honest stake) is stricter than the simple-majority median aggregation of the economic quantities (Assumption 1) and the $\frac{1}{2}$ verifier-collusion bound (Cor. 3); log safety is the binding one, and quorum intersection (any two $2f{+}1$ stake-quorums share $\geq f{+}1$, hence an honest signer) is what makes it safe. The ordering layer above finality is implemented too: a deterministic round-robin *leader* sequences each view, peers accept an ordering only from the view's leader (a non-leader's signed proposal is rejected), and a stalled leader is rotated past by a *view change* [11, 58], where a $2f{+}1$ quorum of signed view-change votes forms a certificate that advances the role to the next leader. So no validator is a permanent sequencer and a faulty one is rotated out. What remains is operational hardening rather than unbuilt mechanism, for example a production replay-fetch path. The diffusion engine's single-step service endpoint and the automatic trajectory-audit dispatch are now implemented and exercised end to end. The ordering layer is implemented and exercised on a local cluster end to end: a leader broadcasts its proposal and every node accepts it while rejecting a non-leader's, and a quorum of view-change votes rotates a stalled sequencer to the next leader. The on-chain settlement program is treated separately below.

The router layer, the request hot-path match that pairs a consumer with a provider (§2.5), is implemented and tested as the same pure-core-plus-QUIC-service split. The match core (`ogong-router`) selects a provider by filtering on model, price, attestation tier, and free capacity, then drawing proportionally to *stake*$\times$*reputation*, the routing weight of §4, seeded by a per-request value so the draw is deterministic and reproducible for a dispute; the realized share tracks stake and audit quality, checked over a large-sample test. The validator-to-router seam is wired: a provider the SPRT

ejects is removed from routing, and a verdict-derived quality score reweights it, so the audit layer's findings shape future routing. The §4.5 split is implemented in the same crate, each released fee divided among provider, router, validator settlement, verification reserve, vig, and the 2 bp license, in OGONG end to end with the provider taking the rounding residual so the split conserves the fee exactly, and a withheld fee refunded whole to the consumer rather than split. The QUIC router service (`ogong-routerd`) carries this onto the mesh: a consumer matches over the wire, providers register, and the validator drives the ejection and quality seam over the same transport, exercised by a live client/server test. The relay data-path forwards a matched request to the selected provider's endpoint and returns the response, either as a one-shot body or as a streamed sequence of chunks for a streaming (SSE) completion; the provider, serving under commitment, pushes its own signed record into the audit path, so the router relays bytes without a second push. Consumer-to-provider confidentiality is implemented as a *post-quantum hybrid* sealed box: the key exchange is X-Wing [6], whose KDF combiner of X25519 and ML-KEM-768 is IND-CCA secure if *either* component is (so an attacker must break both the classical and the post-quantum part), feeding a ChaCha20-Poly1305 AEAD [43]. A fresh KEM encapsulation per seal yields a fresh AEAD key, so the fixed nonce is safe by construction: each key encrypts exactly one unit, and a streamed reply uses a fresh encapsulation per delta, so keystream reuse is structurally impossible rather than an invariant the code must police. The consumer seals the request to the provider's published key, the router relays the opaque ciphertext, and only the provider opens it, so content confidentiality is a property of the cryptography rather than the relay operator and resists a harvest-now-decrypt-later adversary. (This network tier is post-quantum; the confidential browser-to-enclave tier of §8 rides classical secp256k1 ECDH, so the harvest-now-decrypt-later guarantee is the network tier's, not yet the confidential tier's.) It is tested by relaying a sealed payload through the router and confirming the plaintext never appears in what it forwards. The signature side has the matching hybrid: a forgery would have to break both Ed25519 and ML-DSA-44, since a signature carries both and verification requires both. The served record is signed and verified this way today, the provider signing with its hybrid key and the validator checking both parts on intake; the record's 32-byte provider identity is the hash of the hybrid public key, so the verdict log, reputation, and consensus keep their constant-size identifiers unchanged while the authentication is post-quantum. The consensus votes carry the same hybrid: the head vote, view-change vote, and sequencing proposal now sign with the validator's Ed25519 and ML-DSA-44 key, and the constant-size 32-byte validator id those certificates count is the hash of that hybrid key, so finality and ordering are post-quantum-unforgeable while the set and quorum logic keep their small identifiers. The provider's hybrid record-signing key is persisted with its identity, so its on-chain identity and the reputation keyed on it survive a restart. What remains on this layer is the in-enclave (SGX/TDX) realization of §2.5 and the optional onion-routing layer for relay unlinkability.

The on-chain program (Solana/Anchor) that settles value is implemented and tested as well, developed local-first against an ephemeral validator with on-chain state confined to balances and hashes. Three economic primitives are in place. *Staking* locks OGONG as either a provider's priority stake or a validator's bond, and encodes the zero-bond result structurally: a slash succeeds only against a validator bond and is rejected outright against a provider stake, so the priority stake of §4 can never become a correctness bond on-chain. *Escrow* realizes the payment-channel settlement of §4.9: a consumer deposits OGONG, the settlement authority admits a spending allowance, and settlement debits that allowance and disburses the full six-way §4.5 split on-chain (provider residual, router, validator settlement, verification reserve, vig, and the capped license), the shares floored and the provider taking the residual so the division conserves the gross to the unit, the same invariants the off-chain schedule enforces and checked on-chain against the license cap and over-allocation. A withheld request is simply never settled, so its funds remain the consumer's, the

refund of §4.5 realized as non-settlement. *Emission* issues the work-earned tranche on a Bitcoin-style halving schedule: the per-epoch reward halves every period, minting is gated to the consensus authority and to the worker that earned it, and the running total can never exceed the cap, the 80% earned allocation enforced as a hard on-chain invariant. Sixteen tests exercise these against a live local validator, asserting the balance movements and every gate (over-cap, over-allowance, non-authority, provider-not-slashable). One of them runs the whole economic loop in sequence, emission of the earned tranche, a provider's priority stake, a consumer's deposit and the authority's admitted allowance, the audit-released settlement with its six-way fee split, the consumer's refund of the unreserved balance, and the provider's unstake, and reconciles every resulting balance, so the primitives are shown to compose into the full lifecycle and not only to work in isolation.

Three further on-chain primitives make participation permissionless and the coverage parameter a protocol constant rather than an operator's setting. A *validator registry*: a holder of a validator bond publishes its endpoint, QUIC certificate, and consensus id on-chain, in an account anyone enumerates, so the validator set is a decentralized directory with no central server, and registration is gated on the stake so the bond is also the Sybil cost. A *network-parameters* singleton holds the audit coverage α, which every validator reads from chain rather than from a local flag, so the whole set runs one agreed coverage and no operator can quietly lower it. And *quorum-gated settlement*: when an escrow requires it, a release settles only if a quorum of distinct registered validators co-signs the transaction, each checked on-chain against its canonical registration, so no single node, in particular one running α=0, can settle a release alone. Together with the aggregate beacon and the BFT finality, this makes the coverage a value the set enforces and the payout a quorum authorizes, the trustless complement to the statistical deterrent. The set the quorum is evaluated against is itself read from the on-chain registry, so it is defined by no single node. The provider discovers this set without configuration: it bootstraps from a seed and learns the live mesh from any node's discovery response, which a validator populates from the on-chain registry, so a newly registered, staked validator appears in everyone's discovery automatically, weighted by stake. The whole loop runs end to end against a local validator: a provider serving a real model commits and pushes a metered record, the validator audits under the protocol coverage, dispatches a verifier that re-runs the reference model (an honest reply accepted, a substitute rejected), and on release submits the on-chain settlement whose six-way split and escrow debit reconcile to the unit. Embeddings are verified the same way as generation, the output vector sketched and committed and the verifier's re-run scored by the same relative-L2 metric, so the non-text modalities reuse the commitment and adjudication unchanged, with in-engine trajectory emission now wired for audio, image, and video. The mesh gathering of the settlement co-signatures and the assembly of those remote signatures into the on-chain quorum-settle transaction are implemented and validated locally (a settling node collects distinct verdict-gated signatures from peers over the mesh, which assemble into a transaction that verifies). What remains is the settlement sink's use of that path on release and the live multi-validator on-chain submit, together with per-epoch proportional emission accounting, before any move off the local validator toward an audited deployment.

7 Protocol instantiation

The constructions above are stated abstractly; we now instantiate them on the deployed system, whose provider binary already emits a per-reply signature that a browser verifies as a provenance "chip." The protocol extends that path rather than replacing it, so the verification layer is additive and the existing provenance feature is preserved verbatim.

7.1 Deployed baseline

Each provider holds a long-lived key triple: an Ed25519 signing key and X25519 and ML-KEM-768 keys for hybrid post-quantum session encryption, published at `GET /v1/attestation/report` together with a short fingerprint. The router, an attested enclave that terminates the provider tunnel, relays every request; on the verified tier it observes the request body, while on the confidential tier (§7.6) the request is sealed end-to-end to the provider enclave and the router relays only ciphertext. The provider streams the response and hashes it on the fly. On end-of-stream the provider computes, for a fresh 16-byte `reply_id`,

```
DOMAIN  = "ogong-provider-sig-v1"
payload = DOMAIN || "\n" || reply_id || "\n"
        || hex(sha256(request))  || "\n" || hex(sha256(response))
sig     = Ed25519(sk_provider, payload)
```

and caches `(reply_id, sha256(request), sha256(response), sig)` under a bounded TTL, served at `GET /v1/signature/:reply_id`. This binds a reply to an attested key but says nothing about *which model* produced it; that is the gap the audit layer closes.

7.2 The signed commitment record

We extend the signed payload with a model identity and a verification commitment, under a new domain tag (so the two coexist and old verifiers ignore the new fields):

```
DOMAIN  = "ogong-provider-sig-v2"
payload = DOMAIN || "\n" || reply_id || "\n"
        || hex(req_hash)   || "\n" || hex(resp_hash) || "\n"
        || hex(model_root) || "\n" || hex(commit_root) || "\n"
        || dec(n_tokens)   || "\n" || dec(t0_ns) || "\n" || dec(t1_ns)
```

Every field is fixed-width (lowercase-hex digests and decimal integers) and newline-delimited, so the encoding is *injective*: distinct tuples map to distinct byte strings, and the signature is over the tuple itself, not merely a string that happens to parse back to it. The record is signed with a hybrid post-quantum scheme (Ed25519 ‖ ML-DSA-44) and carries the provider's public key and a derived 32-byte identity `provider_pk`, so the commitment survives a future break of either signature primitive and binds to a stable on-chain payee. `model_root` is the SHA-256 over the ordered GGUF shard content hashes of the loaded weights (quantization is bound *implicitly*, as it is part of the hashed GGUF content rather than a separate tag); it equals the catalog's canonical identifier for the model, so a verdict can be checked against a published value. We stress that on the verified tier `model_root` is a *claim*: nothing binds the signed value to the weights actually resident in GPU memory except the audit that follows: it is the audit, not the signature, that establishes model identity. The confidential tier instead anchors model identity in the enclave attestation and a per-inference receipt (§7.6). `commit_root` is the root of the commitment tree of §7.3. `t0_ns,t1_ns` are the provider's self-reported generation timestamps; we stress they are *provider-signed and therefore untrusted for detection*: a smaller-model cheat would simply report a slower time to match its declared model. The trustworthy timing comes instead from the *attested router*, which is on the hot path and *observes* the real reply stream: it measures the wall-clock token-arrival rate with code the provider does not control, so unlike the signed `t0/t1` it cannot be falsified. From that observed throughput (`n_tokens`/ observed-duration) the system derives a *marketplace and audit-prioritization* signal, not a hard latency gate (different hardware makes absolute latency incomparable): the

router compares a provider's observed throughput to the peer distribution *for the same* `model_root`, and a provider serving implausibly faster than its peers on that model (the signature of a smaller substituted model) both raises its audit probability (§7) and feeds the reputation the router already routes on (§4). The signed `t0/t1` then serve only as a cross-checkable *claim*: a provider whose self-reported timing contradicts the router's observation is itself flagged. Honest hardware variance merely shifts rank; a cheaper-model cheat is a visible outlier *and* draws harder auditing, where the commitment check binds correctness. On the confidential tier the enclave's signed receipt timestamps anchor the times. Two trust caveats make this safe to rely on *only* as a heuristic. The router measuring throughput is itself an attested enclave (§2), so its measurement is made by code the operator does not control: it cannot fabricate the number the way the provider can fabricate `t0/t1`. More fundamentally, throughput is *not load-bearing*: a colluding router can at most under-prioritize its favored provider's *extra* audits, but that provider is still sampled at the base coverage α and caught by the commitment re-execution, and its reputation still sheds on real verdict history; symmetrically a smaller-model provider can *sandbag* (delay to fake a slow time) and evade the timing signal entirely. The signal is therefore a cheap prioritization-and-ranking layer over the verification, never a substitute for it.

The signed record, `commit_root` included, is *pushed* to the handling validator at end-of-stream (a few hundred bytes), so the commitment is anchored independently of the provider's later availability. The per-window openings of §7.3 are fetched on audit; a provider that cannot produce openings consistent with its anchored root counts as a *failed* audit. A provider therefore cannot evade verification by going offline or withholding data after serving, the standard data-availability gap in pull-based commitment schemes.

7.3 Commitment construction

The provider's engine, generating in windows of $W = 32$ tokens, emits per window w a leaf

$$c_w \;=\; \mathrm{H}\big(\,\mathsf{toploc}(h_w) \,\|\, \mathsf{lp}(w)\,\big),$$

where $\mathsf{toploc}(h_w)$ is the locality-sensitive commitment to the window's last-hidden states, a *sign-random-projection* sketch (a fixed bank of dense random directions, in the spirit of [45]), so the committed object is not a provider-chosen coordinate subset (hundreds of bytes), and $\mathsf{lp}(w)$ is the set of top-k log-probability digests at every decode position in w [27] (committing all positions is strictly stronger than a sampled subset). The leaves form a Merkle tree [41] whose root is `commit_root`; the per-window blobs are served, with inclusion proofs, at a new `GET /v1/commitment/:reply_id` (same TTL discipline as the signature cache). Total commitment overhead is $O(|\pi|)$ per window, hundreds of bytes, not megabytes, so it streams alongside the response without buffering. This construction is implemented in our engine: both $\mathsf{toploc}(h_w)$ and $\mathsf{lp}(w)$ ship in the per-token response and the leaf binds them jointly, with the verifier checks of §3 measured on real models (the hidden-state proximity is the single-shot separation reported there).

7.4 Request lifecycle and audit

A served request traverses the state machine

$$\mathsf{ROUTED} \to \mathsf{SERVED} \to \mathsf{ESCROW\text{-}HELD} \xrightarrow{\text{audit?}} \{\mathsf{VERIFIED}\} \to \mathsf{RELEASED} \;\big|\; \mathsf{WITHHELD},$$

with the fee escrowed during ESCROW-HELD for an audit window τ (Assumption 3). The router that served the request pushes the signed record to the validators at end-of-stream; a validator then runs:

1. **Selection.** Derive the audit seed from the committee beacon (§6): the threshold-BLS draw over the epoch when a committee key is established, else the per-validator VRF aggregate over (epoch $\|$ `reply_id`), bound to the reply as H($\cdot \,\|$ `reply_id`); audit iff it falls below the coverage threshold α (Assumption 2). Because the seed is unpredictable and the audit may run any time within τ on a logged reply, the served provider cannot distinguish audited from unaudited requests (Assumption 4).
2. **Assignment.** VRF-select a verifier $v \neq$ server from the providers serving `model_root`. The validator holds the logged request and the pushed `commit_root`; it dispatches to v an *audit job* (the prompt, the claimed output tokens, and `commit_root`) wire-shaped as an ordinary request, with the control fields readable only inside the validator enclave.
3. **Scoring.** v re-executes the window by a *teacher-forced re-prefill* of the prompt and claimed output: a single prefill pass, which is the source of the $\rho \approx 10^{-2}$ cost of §3 rather than full autoregressive re-generation. Both fingerprints come from prefills: the per-token hidden state from the engine's embedding output, and the per-position logprobs from a *score mode* that marks the claimed-output tokens for output in the single prefill (an off-by-default engine flag; for $n = 0$ scored tokens it is a normal completion). We confirmed this is sound, not merely cheaper: on our engine the teacher-forced re-prefill reproduces the generation-time hidden states to relative-$L_2 \approx 0.005$ *and* the generation-time logprobs to total-variation ≈ 0.005 (the argmax matching at every position), both far below the reject thresholds. It returns, per window, the hidden-state proximity (relative-L_2 over the committed *sign-random-projection* sketch, dense projections of the hidden state, so a substitute cannot match a provider-chosen subspace) and the LOGIC distribution distance (top-k total variation plus the decode-position KS test), committing its own scoring pass. The *verifier's dilemma*, a rational v rubber-stamping `accept` to save the recompute, is defeated not by overlap alone (two lazy verifiers agree on `accept`) but by *honeypot* audits: the validator injects jobs carrying a planted, known-bad output, indistinguishable from real ones (Assumption 4); a verifier that passes a honeypot is itself slashed and ejected. (The text-output honeypot runs on every audit; the diffusion-trajectory honeypot is active wherever the validator is configured with the engine's latent shape.) Honest scoring is then v's dominant strategy (the standard forced-error resolution of the verifier's dilemma [40, 49]), and overlapping VRF-assigned verifiers catch residual disagreement.
4. **Verdict.** The validator checks Merkle inclusion of the scored windows against the server's `commit_root`, applies the verifier thresholds, and appends (`reply_id`, server, v, verdict, seed) to the tamper-evident log, updating the server's SPRT statistic (Prop. 3).
5. **Settlement.** `accept` $\rightarrow$ release escrow; `reject` $\rightarrow$ withhold the fee (Cor. 1, refunded to the consumer per §4.5) and advance the SPRT; threshold crossing $\rightarrow$ ejection and reputation update under Yuma consensus (Assumption 1).

The validator never performs the forward pass. Its enclave is a CPU TEE with no GPU, so, rather than recomputing them, it *adjudicates* the committed results returned by GPU verifiers (steps 3–4): signature, Merkle-inclusion, and threshold checks plus the verdict log, all kilobytes of work. A small enclave therefore suffices, and the validator set can be permissionless with *no* GPU or model-weight requirement: a validator needs only an attested CPU enclave, stake, and bandwidth. Verification *throughput* instead scales with the provider GPUs that double as verifiers, not with the number of validators.

7.5 Tiers and participation

Every provider runs the same instrumented engine, so every provider can emit `commit_root`; there is no uninstrumented pass-through path to exclude. Two orthogonal distinctions remain. *Tier*: on the confidential tier the provider's enclave attestation binds the serving identity to the hardware and a per-inference receipt binds each reply to that attested workload (§7.6), so an attestation check together with the receipt subsumes the per-request audit of steps 1–5; on the verified tier the audit above applies. The two ogong tiers are thus *confidential* (attested) and *verified* (re-executed): there is no unverified pass-through tier. *Mode*: an operator may serve the open network and earn, or serve only its own account (the latter requiring no network verification at all), and may switch between the two simply by turning public serving on. The v1 provenance chip is unchanged (a browser that ignores the v2 fields still verifies reply authenticity), so the consumer-facing provenance feature and the supplier-facing audit compose without interference.

7.6 The confidential tier: attestation and end-to-end encryption

The confidential tier replaces per-request re-execution with hardware attestation: the provider runs inside a TEE (Intel TDX under a dstack-style runtime, optionally with NVIDIA confidential-computing GPUs), and trust derives from a remote-attestation chain rather than from the verifier recomputing the forward pass. Three mechanisms compose, all implemented and verified live against a deployed enclave.

Remote attestation. The provider serves, at `GET /v1/attestation/report`, a TDX quote the validator checks in three layers: (i) the DCAP signature chain to the Intel provisioning roots (genuine TDX hardware in production mode), including the quoting-enclave identity and the TCB-recovery status against Intel's collateral, so a cryptographically valid quote on revoked or down-level microcode is rejected; (ii) a replay of the runtime's measured-boot event log to reconstruct the RTMRs and recover the *application identity* (the compose-hash of the deployed workload), checked against an approved set; and (iii) the `report_data` binding, which carries the enclave's serving public key together with the validator's fresh challenge nonce, so the quote is fresh and bound to the key that will sign for this workload. The validator runs this as a periodic challenge–verify–eject audit: a provider whose quote fails the chain, whose application identity is not approved, or whose freshness binding is wrong is ejected from routing. Because end-to-end encryption is *mandatory* on this tier (the request is sealed to the attested encryption key before it leaves the client, see below, and a confidential-tier request that arrives unsealed is refused), the serving path moves only ciphertext: it cannot read the request or substitute the served model. The per-request guarantee therefore rests on two things the audit establishes: the attested *workload identity* (which model image is running) and the client's seal to the attested key, with no trusted intermediary in between.

Per-inference receipt. Attestation establishes the *workload*; a per-reply receipt binds each *response* to it. The enclave signs, with the serving key bound into the current quote's `report_data` (so the signing key is itself attested and fresh within the challenge period, the maximum staleness of a receipt is exactly that challenge interval, which we state as the replay window), a receipt over the request digest, the reply identifier, and a transcript of the served reply; the validator (and any client) verifies the receipt signature against the attested key, that its request digest matches the request actually sent, and that the quote backing the key is within the freshness window. Settlement on the confidential tier is gated on this check: a reply whose receipt does not verify is not paid. The resulting per-request guarantee is *hardware-attested* rather than statistical (the attested enclave

certifies, per reply, that it served the claimed workload), but we are precise that this is not a bare cryptographic proof: it is rooted in the TEE vendor's signing infrastructure and an unbroken enclave (§10.1), a different and stronger-but-trust-anchored basis than the verified tier's statistical re-execution, not an unconditional one.

End-to-end encryption. Confidentiality is a property of the client, not a promise by any relay. For a confidential-tier request the client seals each sensitive message field to the enclave's attested encryption key with the dstack attestation seal (secp256k1 ECDH → HKDF-SHA256 under a fixed `info` domain string → AES-256-GCM): the scheme is the one the attested workload decrypts, so the construction is fixed by the enclave rather than chosen at the relay. The seal derives a *fresh ephemeral ECDH key for every field and every streamed delta*, carried in the sealed blob, so each AES-256-GCM key encrypts exactly *one* unit: nonce reuse is structurally impossible rather than an invariant the implementation must police, and per-field key separation comes from the distinct ephemerals (not from per-field key derivation). The router, the settlement relay, and the operator's own front end all move *ciphertext* and only the enclave opens it; the reply is sealed back the same way, each streamed delta as its own sealed unit, and opened on the client. This confidential-tier channel (browser ↔ enclave, scheme fixed by the enclave) is distinct from the *network-tier* sealed box of §6 (X-Wing → ChaCha20-Poly1305, where both endpoints are ours and we therefore choose a post-quantum hybrid): they are two different channels with different counterparties, not two specifications of one channel. Because the seal targets a key the client read out of the attestation it just verified, confidentiality reduces to the soundness of the attestation rather than to trusting the routing layer, strengthening the earlier posture ("the router runs attested code that cannot read the plaintext") to one where the router need not be trusted for confidentiality at all. We implement this seal both natively and in the browser and verify that it round-trips through the full routing path against a live enclave.

Boundary. Confidentiality on this tier is only as strong as the underlying TEE (§10.1): a working enclave break on the host serving a request defeats it, and we do not claim security against such an adversary; correctness, by contrast, survives a compromised enclave minority through the committee beacon and the on-chain log. The load-bearing residual is that the client must *verify* the attestation before sealing to its key: a client that takes the encryption key from an unverified report trusts whoever relayed it, so the attestation check is precisely what turns "private" into "verifiably private."

7.7 Verified split inference

The zero-bond result is what makes the network's most demanding capability viable: serving a model too large for any single node by *sharding* it across a cohort of untrusted commodity GPUs. Bonded verified-inference designs cannot reach this at casual scale, because each shard would carry its own slashable bond and multiply the capital barrier by the number of segments; at zero bond a cohort of unbonded nodes serves a frontier model, each segment independently verified and paid for the layers it ran. The deterrence bound applies per segment: a shard's compute saving Δ and its fee share p both scale with the layers it runs, so the whole-model honesty condition implies the per-segment one (Cor. 2, machine-checked in Z3): full-coverage verification that deters the monolith deters every shard, and the per-segment audit drives $q \to 1$ exactly as the whole-model check does. A single faked boundary breaks the commitment chain and is localized to its signer, so the deterrence applies independently to every node in the cohort.

A model too large for any single provider can be served by a *cohort* that shards its layers across nodes, with the same per-request audit applied per segment: each shard commits the hidden state

at its layer boundary, the commitments chain (one segment's output is the next segment's input), and a validator re-executes a sampled segment and checks the boundary reproduces, localizing any cheat to a single node. This is the per-segment re-execution of Wang et al. [56] composed with our commitment format and zero-bond settlement: each shard is paid for its slice, and a caught segment withholds the whole request and refunds the consumer, so an honest shard risks no bond. Each shard additionally signs its boundary commitment with its hybrid Ed25519 and ML-DSA provider key, so a boundary is non-repudiably attributable to the one node that produced it: a forged or reattributed segment is rejected before any re-execution, and a caught one ejects and withholds from exactly that provider, with no trusted lead.

We have built and tested this path end to end. The engine emits the residual-stream boundary for all token positions and runs an arbitrary layer range either from the prompt (the first shard) or from an injected upstream boundary (interior shards); a resident segment server answers successive forwards without reloading the model; and a network transport drives remote shards across the same interface that local ones use, so a cohort spans machines. On a 2B model the residual-stream boundary at an interior layer reproduces *exactly* on an honest same-engine re-run, drifts $\sim 0.6\%$ across backends (Metal versus CPU), and lands $\sim 30\%$ away for a substituted sub-computation: a roughly $50\times$ separation between the honest cross-backend band and a cheat, on the same calibration the whole-model check uses. Composing these, a two-shard cohort in which each shard loads only its own layers and signs its slice reproduces the monolithic model's output end to end to a relative ℓ_2 of $\sim 10^{-5}$. The mechanism is architecture-agnostic because it rides the residual stream that every decoder transformer exposes; the only per-model detail is the input embedding transform an interior shard must skip.

The network path runs on the same control plane as whole-model serving: a shard advertises its layer range, the router assembles the cheapest cohort that tiles the model, and the zero-bond settlement is realized on-chain as a single cohort-settle that pays each shard its slice under a conservation invariant, the per-shard amounts must sum to the provider's share, so the release can never exceed the request's fee. On a local test validator a two-shard cohort is paid its slices (47,990 each of a 95,980 provider residual) and a non-conserving split is rejected with no funds moved, beside the whole-model settlement tests. Split inference is therefore a serving *topology* orthogonal to the trust tier rather than a tier of its own: it composes with the verified and TEE tiers alike, and a cohort's guarantee follows the tier of its shards. What remains is the live multi-machine deployment: running the segment servers across physical nodes, emitting each request's signed cohort record into the audit and settlement paths, and binding cohort formation to the supply-side coverage interface that keeps enough providers serving each segment.

8 Security analysis

Attack	Defense
Model substitution	commitment check [45]; timing
Decode-time substitution	log-probability test [27]
Quantization fraud	precision detection [45]; declared + priced; timing
Distillation mimicry	hidden-state commitment (activations, not just logprobs) [45]
Stale / cached output	commitment bound to this request's activations
Partial execution	downstream-activation divergence [45]
Verifier's dilemma (lazy scoring)	honeypot traps + overlapping VRF (§7)
Selection / randomness steering	threshold-BLS committee beacon (Assump. 2)
Commitment withholding	serve-time root anchoring; missing openings = reject
Sybil / coalition	throughput bound (Prop. 6) + ramp; $< \frac{1}{2}$ share

Sybil resistance and per-request verification are complementary: the former bounds identity fraction (protecting sampling and consensus), the latter catches per-request fraud (protecting correctness). An adversary who *rents* a genuine fleet defeats the Sybil bound but pays market rate, constitutes real capacity, and is still caught per request by Cor. 1. We claim no defense against an adversary willing to honestly run the correct model: there is, by construction, nothing to detect.

Coalitions and concentration. The relevant centralization metric is *throughput share*, not identity count: a pool of many real GPUs holds many legitimate identities (Prop. 6), but to control audit sampling or consensus it must reach a $\frac{1}{2}$ share of network throughput (Cor. 3), the same honest-majority threshold (Assumption 1) as any such system, and the same place pools could in principle concentrate. Bribing a single verifier yields little: refunds flow to the consumer, not the catcher (§4.5), so there is no catch-bounty to capture, and a briber must corrupt a quorum of independently beacon-assigned verifiers *and* evade honeypots, which again reduces to a throughput majority.

Graceful degradation under TEE compromise. The TEE is doing real work, and SGX-class enclaves have a documented history of side-channel and speculative-execution breaks; we therefore design so that a single compromised enclave does not break the system. Randomness is a committee beacon, not one enclave's VRF (Assumption 2), so a broken router cannot steer audit selection or verifier assignment; verdicts are on-chain-anchored under honest-majority consensus, so it cannot forge settlement alone. What a compromised *routing* enclave can violate is the *confidentiality* of requests passing through it; privacy, not correctness, is what degrades under TEE compromise, and only for traffic that chose a plaintext-relaying path rather than end-to-end encryption to the provider. Correctness rests on the audit and the economic majority, which survive a minority of broken enclaves.

9 Positioning

The verification primitives and the no-slash economics each exist in prior systems; the *join* does not. Below, "crypto verify" denotes a per-request commitment (not subjective scoring) and "no correctness bond" denotes that honest providers risk no capital for correctness.

System	Crypto verify	No correctness bond
Hyperbolic / PoSP [59]	yes	no (verifiers slashed)
Gensyn / Verde [4]	yes	no (collateral posted)
Inference Labs / DSperse [26]	yes (zk)	no (slashed on failed proof)
EigenAI [3]	yes (deterministic)	**no (reduces, not eliminates)**
Bittensor [46]	no (subjective rank)	yes (dilution)
Venice / VVV [52]	n/a (demand-side)	n/a
STAKESURE [14]	n/a (general PoS)	no (insures slashing)
Fortytwo [35]	no (peer rank)	undocumented
This work	**yes [27, 45]**	**yes**

Alves et al. [3] is the closest prior art and the clearest evidence of the gap: from the same premise (cheap cryptographic verification is achievable), it retained the bond. Our contribution is a systems result rather than a cryptographic one; its defensibility is the argument and being first to deploy it on a production engine, not exclusivity over primitives that others may freely reuse.

Protocol, not product (the "survives the issuer" line). A second axis separates a *protocol with a token* from a *product with a token*: does the system keep functioning if its originator disappears? Several "decentralized-AI" tokens are in substance access passes to one company's hosted product, the company runs a closed proxy, rents GPUs from third-party compute markets, and the token buys credits against *its* continued operation; if the company stops, the token stops. That the GPUs are rented from a permissionless market makes the *supplier* decentralized, not the product. Ogong is on the other side of this line by construction: the verification, routing, audit, and settlement are an open permissionless protocol that no single operator runs, and the originating company is the network's *first consumer*, not the network. This is both the competitive distinction and, as §4.7 notes, a structural input to the token's regulatory characterization, value that does not depend on one operator's efforts is a different instrument from a credit redeemable against one company.

10 Limitations and future work

10.1 The assumption ledger

The formal results of this paper are *conditional*: each theorem establishes that *if* its premises hold *then* its conclusion follows, and the machine-checked artifacts (the SMT proofs, the Lean development, the ProVerif and Tamarin models, the PRISM-games model) discharge the *reasoning*, not the premises. We therefore gather the premises in one place, classify each by *how* it is discharged, and say where. The ledger is candor with a direction: the proofs make the consequences airtight, so the genuine residual risk lives entirely in the rows below, and most of those rows are *measurable or already measured* rather than matters of faith. We mark each proven (follows from a checked artifact or a standard result), measured (an empirical quantity instrumented on our own engine), behavioral (a rationality premise standard in mechanism design), trust (an external root we depend on but do not prove), or forecast (a deployment-time adoption premise).

Assumption	Type	How it is discharged or bounded
Validator honest majority (Ass. 1)	proven/std	Standard BFT premise; finality is a $2f+1$ quorum certificate (§6); a stake minority can neither frame nor shield.
Unbiasable beacon (Ass. 2)	built	Threshold-BLS signature over a no-dealer DKG key (any t-of-n shares give the same value; equivocation-excluded; key-preserving resharing); VRF aggregate as bootstrap fallback; the enclave is defense-in-depth, not the sole root.
Escrow and audit window (Ass. 3)	construction	Enforced by the settlement contract; a fee is released only after the audit window.
Audit indistinguishability (Ass. 4)	measured	Retroactive scoring of logged requests plus wire-indistinguishability; degrades gracefully ($\alpha \to \alpha(1-\delta)$), co-batch drift measured small (§2.3).
Separation ρ adequate on the production engine (Def. 1)	measured	Hidden-state commitment separates honest reruns ($\sim$0) from cheats (0.016 at near-lossless Q8 up to ~ 1.1 for a substituted model; threshold 0.10) (§3); per-architecture calibration is the open work and the central deployment gate.
Statistical (not cryptographic) soundness, verified tier	measured	Per-request total-variation margins measured (below); objective soundness awaits practical ZK [24, 26, 34].
Rational, risk-neutral attacker ($q \geq \Delta/(p+\Phi)$)	behavioral	Standard mechanism-design premise; per-query $\Delta < p$ and the forfeited fee bound even a non-economic griefer's edge.
Staker rationality (the $\sigma^\star$ equilibrium)	behavioral	Lock-or-hold yield equalization; comparative statics machine-checked (`value_model.py`).
Demand elasticity $e > 1$ (value *appreciation*)	forecast	Jevons precedent [2, 28] plus the inference cost/volume record; needed only for appreciation, not function; breadth is a second channel (§4.6).
Adoption reaches fee-sustainability ($\phi_v R_{\max} \geq B$)	forecast	`bootstrap_model.py` sizes the bridging subsidy keeping $\min_t S(t) \geq B$ at every horizon; satisfiable, a deployment parameter.
TEE soundness (gold-tier confidentiality)	trust	External root; correctness survives a compromised minority (beacon plus on-chain log), but confidentiality is only as strong as the TEE (below).
Sybil cost via attested throughput bound	construction	Identity fraction bounded by physical throughput (§5); collusion reduces to the honest-majority threshold.
Progressive decentralization occurs	forecast	On-chain, governance-gated treasuries and a milestone-gated handover (§2.5, §4.6); disclosed, not assumed complete.

The rows that carry real exposure are the measured separation ρ (the deployment gate), the forecast adoption and elasticity premises, and the trust in the TEE for gold-tier confidentiality. Each is instrumented on our engine today, sized by a published model, or scoped as defense-in-depth, and none is load-bearing for the core result that a sufficiently audited network needs no correctness bond. The remaining rows are standard or constructive. We restate the individual boundaries in detail below.

10.2 Detailed boundaries

We state the boundaries of the result plainly.

Statistical, not cryptographic, soundness (verified tier). Definition 1's soundness is empirical over a substitute class, not a worst-case cryptographic bound; a crafted adversarial collision against the top-k commitment is outside its guarantee. Fine-grained quantization fraud is the subtlest detectable case. By single-request argmax a model at a different quantization agrees 94% with the reference (versus 51% for a materially cheaper model), too close to flag; the full distribution check of Inference.net [27] separates it. Total- variation distance between the committed and recomputed top-k distributions gives a cross-backend quant cheat 0.054 against 0.014 for honest cross-backend drift, a clean per-request margin that the crude argmax discards (measured on a near-lossless quant pair, Q5 variants; coarser gaps separate further). The threshold needs per-backend-pair calibration and the margin is tighter than for gross substitution, and although co-batch honest drift is now measured small (§2.3), its threshold still needs per-backend-pair calibration, so declaration and pricing remain a backstop. Quantization fraud is thus detectable rather than near-floor. The hidden-state commitment, which we implemented in our engine as a sign-random-projection sketch and measured (§3), scores that Q5 pair at relative-L_2 0.065 against 0.0 for the honest re-run (threshold 0.10), with the gross size substitution at ~ 1.1, wide enough to flag the quant cheat single-shot. More important than the quant margin, the hidden-state check is the one a distillation-mimicry adversary cannot pass by matching the output distribution alone: a cheap model fine-tuned to reproduce the reference's logprobs would still have to reproduce its internal activations, a strictly harder target, so committing the activations and not only the softmax removes the most plausible route around the statistical bound. We note that off-the-shelf distilled models are not effective logprob mimics in the first place: a knowledge-distilled 2B model, scored against the 9B it was distilled toward, agrees only 75% by argmax (versus 99% honest) and scores total variation 0.22, caught by the logprob check alone; distillation for capability is not token-level distribution matching, so the mimicry attack requires dedicated adversarial training and, against the hidden-state check, an architecture that also matches the target's activations. We make the economic structure of this explicit, since it is what carries the no-bond design through the soundness gap rather than a collateral cushion. The hidden-state commitment makes the *verified quantity be the computation itself*: the activations are expensive to produce and cheap to check, so forging them is as hard as producing them. A successful cheat must therefore furnish, for a *cheaper* model, activations that match the reference's on audited requests the provider cannot predict in advance, and matching the activations across the input distribution is matching the work. Such a model is either the reference itself, in which case no compute was saved and the motive to cheat is gone, or a faithful white-box distillation that reproduces a frontier model's internal trajectory at lower cost, which is a model-compression result worth far more sold as a model than spent evading a per-request fee. The attack's success condition, *cheap and activation-faithful*, contradicts its premise, that the work was too expensive to do honestly. In the full-coverage regime the design targets ($\alpha = 1$), the cheat must hold on *every* request, so it collapses to reproducing the reference everywhere, which is being the reference. This is an economic argument against a rational attacker rather than a cryptographic impossibility, and a method that synthesized faithful activations cheaply would weaken it, which is why the TEE and zero-knowledge tiers remain the backstop for assurance that does not rest on the cost of cheating. Objective per-request soundness without a TEE awaits practical zero-knowledge inference [24, 26, 34]; that is the endgame, and until then the verified tier offers probabilistic, not cryptographic, correctness.

TEE fragility. The TEE tier's confidentiality and the validator enclaves rest on TEE security (SGX/TDX/CC), which has a history of side-channel and speculative-execution breaks. We treat the enclave as defense-in-depth (randomness is a committee beacon and verdicts are on-chain-anchored, §8), so correctness survives a minority of compromised enclaves; but gold-tier *confidentiality* is only as strong as the underlying TEE, and we do not claim security against an adversary with a working enclave break on the host serving its own traffic.

Cold start and security budget. The economic guarantees (audit coverage near full, the throughput-majority bound) are asymptotic in network size. At bootstrap, with few providers, they are weak, and security rests instead on attestation (the TEE tier) and a small, partly governed operator set; the validator set decentralizes as independent attested nodes stake in (§2.5). Tapering emissions for verified work (§4.6) subsidize early supply and bridge the cold-start security budget, decaying as fees grow; sizing the emission curve so fee revenue overtakes it before it ends, keeping the validator security budget above a mis-verdict's attack value throughout, is the central deployment parameter. We do not hand-wave it: `bootstrap_model.py` models the budget $S(t) = \phi_v R(t) + E_0\, 2^{-t/H}$ (fee share plus a halving subsidy) against an attack value B, confirms the gap is *real* (without a subsidy the budget dips to a small fraction of B at launch), finds the minimum subsidy E_0 that keeps $\min_t S(t) \geq B$ at every horizon, and confirms the mature fee budget $\phi_v R_{\max}$ clears B on its own once adoption saturates. It also tabulates the decentralization trajectory: the $\geq \frac{1}{2}$-throughput collusion threshold (Cor. 3) costs half the network's physical throughput, which rises as operators join and the largest single share falls as $1/n$. So the cold-start trust is bounded and the bridge condition is satisfiable, not merely hoped, though it remains a parameter a deployment must set and an external audit should check.

Privacy is content, not metadata. We claim content confidentiality (end-to-end encryption to the provider) and optional unlinkability (onion routing), not full metadata privacy: the routing layer and the chain learn coarse metadata: which consumer paid for which model, when, and at what price. Traffic-analysis correlation and TEE side-channel leakage are out of scope.

Scope. The validator agreement layer (BFT/PoS consensus over the verdict log and reputation) is standard, and our contribution is the verification and economic mechanism atop it, not a new consensus; we nonetheless implement its finality core as a quorum certificate over the log head (a head is final once $2f+1$ of n attested validators sign it, and a divergent minority cannot finalize), verified on a local cluster, with round-robin leader sequencing, leader-authorized ordering proposals, and view-change rotation implemented above it and exercised on the same cluster (§6). The diffusion/codec verification of §3.1 (trajectory commitment + single-step re-check, $\rho \approx 1/N$) is implemented and measured for audio, image, and video, with the engine's single-step re-check exposed as a stateless service endpoint and the validator's automatic dispatch of trajectory audits running on the shared commitment primitives end to end; its tolerance bands still need per-architecture calibration. The guarantees are bounded by Assumptions 1–4 and the rented-fleet limit of §8.

11 Conclusion

The slashable correctness bond has been treated as a fixture of verified inference. It is instead a consequence of expensive verification. Once verification is cheap enough to cover nearly every request, the honesty condition $q \geq \Delta/(p + \Phi)$ is met with a zero bond and, under escrow, a zero franchise: forfeiting the fee on the cheated request suffices. Stake is then free to serve priority and availability,

and Sybil resistance is discharged by a non-capital cost that bounds identity fraction by physical throughput, reducing verifier-collusion to the standard honest-majority threshold. The result is a network an operator joins by opening an application, with no capital at risk. We contribute no new primitive; we contribute the mechanical design that turns known primitives into the system.

A Empirical validation details

All figures below are measured on our `llama.cpp`/ggml fork via the `verify_commit` harness and the validation scripts in the reference implementation. Runs are single-host on Apple Silicon (Metal) unless noted; the *cross-hardware* tail is the open calibration gate (§10.1), not a settled result. The raw per-position records are reproducible from the harness; we report the per-window summaries the adjudication uses.

A.1 Honest drift vs. the logprob cheat signal

Honest co-batch drift (four concurrent slots, continuous batching) and the substitution signal on the log-probability checks:

	logprob TV	decode-position KS
Honest cross-backend re-run (Metal vs. CPU)	0.014	$\sim 10^{-4}$
Quant cheat (declared f16, served cheaper)	0.054	n/a
Clean f16 quant ladder (KS, argmax agreement):		
honest f16	n/a	$\sim 10^{-4}$ (100%)
Q8 (near-lossless)	n/a	0.14 (82%)
Q6 / Q5	n/a	0.25 / 0.42
Q4	n/a	0.45 (45%)
gross size substitution (2B→9B)	n/a	0.71

A.2 Hidden-state sign-random-projection sketch

The activation check (§3) under the dense-projection sketch, $k = 256$ directions. Size substitution (Gemma-2 2B vs. 9B): honest re-run 0.0, substitute 1.15 per output position. Quant ladder (Qwen2.5-0.5B, provider declaring f16), token-aligned relative-L_2:

served	f16	Q8	Q6	Q5	Q4_K_M	Q3/Q4_0	Q2
SRP rel-L_2	0.0	0.016	0.024	0.065	0.093	0.145/0.173	0.302

A same-width but different-method Q4 (declared Q4_K_M, served Q4_0) scores 0.22. The activation check rejects the gross-quant band (Q4 and below, threshold 0.10); the logprob-KS check above covers the near-lossless band (Q8–Q5); together they span the spectrum.

A.3 Teacher-forced re-prefill reproduces generation

The cost win (§3): both fingerprints from prefills rather than re-generation. Measured on Gemma-2 2B, per output position, teacher-forced vs. generation:

quantity	teacher-forced vs. generation
hidden state (prefill vs. generated prefix), rel-L_2	0.003–0.008 (7/7 tokens agree)
hidden sketch via embeddings endpoint, rel-L_2	0.007–0.015
logprobs via score mode, total variation	0.004–0.016 (mean 0.005; argmax matches)

These sit far below the reject thresholds for gross substitution ($\sim$1.1) and the coarse-quant band ($\geq$0.15), so a single prefill yields both fingerprints at $\rho \approx 10^{-2}$ with no soundness loss *there*. One honest tension we surface: the deployed hidden path is the embeddings endpoint (0.007–0.015 rel-L_2), and that reproduction noise floor is the *same magnitude* as the near-lossless Q8 activation signal (0.016), so under teacher-forcing the activation sketch cannot by itself catch near-lossless quant fraud; that band leans on the logprob KS/TV check (whose own reproduction noise, TV 0.004–0.016, likewise overlaps the Q8 region), and ultimately on the cross-hardware calibration of §10.1. The cheap path is sound for the gross and coarse bands; near-lossless detection is exactly the regime the calibration gate governs.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

[2] Blake Alcott. Jevons' paradox. *Ecological Economics*, 54(1):9–21, 2005.

[3] David Ribeiro Alves, Vishnu Patankar, Matheus Pereira, Jamie Stephens, Nima Vaziri, and Sreeram Kannan. EigenAI: Deterministic inference, verifiable results, 2026.

[4] Arasu Arun, Adam St. Arnaud, Alexey Titov, Brian Wilcox, Viktor Kolobaric, Marc Brinkmann, Oguzhan Ersoy, Ben Fielding, and Joseph Bonneau. Verde: Verification via refereed delegation for machine learning programs, 2025. Gensyn.

[5] Adam Back. Hashcash — a denial of service counter-measure. Technical report, Independent, 2002.

[6] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karolin Varner, and Bas Westerbaan. X-Wing: The hybrid KEM you've been looking for. IACR Cryptology ePrint Archive, Paper 2024/039; draft-connolly-cfrg-xwing-kem, 2024. `https://eprint.iacr.org/2024/039`.

[7] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

[8] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[9] Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2): 130–143, 1987.

[10] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 123–132. ACM, 2000.

[11] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186. USENIX Association, 1999.

[12] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388. ACM, 2002.

[13] KD Conway et al. opML: Optimistic machine learning on blockchain, 2024. ORA / Hyper Oracle.

[14] Soubhik Deb, Robert Raynor, and Sreeram Kannan. STAKESURE: Proof of stake mechanisms with strong cryptoeconomic safety, 2024.

[15] drand / League of Entropy. drand: A distributed randomness beacon daemon. Threshold-BLS randomness beacon, `https://drand.love`, 2023.

[16] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

[17] EigenLayer. EigenLayer: The restaking collective. Technical report, Eigen Labs, 2024.

[18] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 427–438. IEEE, 1987.

[19] Irving Fisher. *The Purchasing Power of Money: Its Determination and Relation to Credit, Interest and Crises*. Macmillan, New York, 1911.

[20] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[21] Rūsiņš Freivalds. Probabilistic machines can use less running time. *Information Processing (IFIP Congress)*, pages 839–842, 1977.

[22] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991.

[23] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007. Preliminary version in EUROCRYPT '99.

[24] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *ACM Symposium on Theory of Computing (STOC)*, 2008.

[25] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology — CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer, 1995.

[26] Inference Labs. DSperse: Model slicing for scalable verifiable inference in decentralized AI markets. `https://decentralizedinference.org`, 2026.

[27] Inference.net. LOGIC: Trustless inference through log-probability verification. Technical report, `https://inference.net/blog/logic`, 2025.

[28] William Stanley Jevons. *The Coal Question: An Inquiry Concerning the Progress of the Nation, and the Probable Exhaustion of Our Coal-Mines*. Macmillan and Co., London, 1865.

[29] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[30] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up trustless DNN inference with zero-knowledge proofs, 2022.

[31] Adam Karvonen, Daniel Reuter, Roy Rinberg, Luke Marks, Adrià Garriga-Alonso, and Keri Warr. DiFR: Inference verification despite nondeterminism, 2025.

[32] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning (ICML)*, 2023.

[33] Andrey Nikolaevich Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4:83–91, 1933.

[34] Lagrange Labs. DeepProve: Zero-knowledge machine learning inference. Technical report, 2025.

[35] Vladyslav Larin, Ihor Naumenko, Aleksei Ivashov, Ivan Nikitin, and Alexander Firsov. Fortytwo: Swarm inference with peer-ranked consensus, 2025.

[36] Andrew Lee. Trajectory commitments: Cheap verifiable inference for diffusion and codec models, 2026. Companion to this work, in preparation.

[37] Andrew Lee. Zero-bond deterrence: When verification is cheap, the correctness bond is redundant, 2026. Companion to this work, in preparation.

[38] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

[39] Elizabeth Lui and Jiahao Sun. Bittensor protocol: The Bitcoin in decentralized artificial intelligence? a critical and empirical analysis, 2025.

[40] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 706–719. ACM, 2015.

[41] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology — CRYPTO '87*. Springer, 1988.

[42] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[43] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF protocols. RFC 8439, Internet Engineering Task Force (IETF), 2018. `https://www.rfc-editor.org/rfc/rfc8439`.

[44] NVIDIA. Confidential computing on NVIDIA H100 tensor core GPUs. Technical report, NVIDIA Corporation, 2023.

[45] Jack Min Ong, Matthew Di Ferrante, Aaron Pazdera, Ryan Garner, Sami Jaghouar, Manveer Basra, Max Ryabinin, and Johannes Hagemann. TOPLOC: A locality sensitive hashing scheme for trustless verifiable inference, 2025. Prime Intellect.

[46] Opentensor Foundation. Yuma consensus. Bittensor documentation, `https://docs.learnbittensor.org/learn/yuma-consensus`, 2024.

[47] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1992.

[48] Kyle Samani. Understanding token velocity. Multicoin Capital, `https://multicoin.capital/2017/12/08/understanding-token-velocity/`, 2017.

[49] Jason Teutsch and Christian Reitwießner. A scalable verification solution for blockchains. Technical report, TrueBit, 2017.

[50] Florian Tramèr and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations (ICLR)*, 2019.

[51] U.S. SEC, Division of Corporation Finance. DoubleZero Foundation — no-action letter. September 29, 2025.

[52] Venice AI. Understanding Venice compute units and the VVV token. `https://venice.ai/blog/understanding-venice-compute-units-vcu`, 2025.

[53] Jean Ville. *Étude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939.

[54] Abraham Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2): 117–186, 1945.

[55] Abraham Wald and Jacob Wolfowitz. Optimum character of the sequential probability ratio test. *Annals of Mathematical Statistics*, 19(3):326–339, 1948.

[56] Ke Wang, Zishuo Zhao, Xinyuan Song, Zelin Li, Libin Xia, Chris Tong, Bill Shi, Wenjie Qu, Eric Yang, and Lynn Ai. VeriLLM: A lightweight framework for publicly verifiable decentralized inference, 2025.

[57] Kai Yao and Marc Juarez. AuthPrint: Fingerprinting generative models against malicious model providers, 2025.

[58] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347–356. ACM, 2019.

[59] Yue Zhang, Shouqiao Wang, Sijun Tan, Xiaoyuan Liu, Ciamac C. Moallemi, and Raluca Ada Popa. Proof of sampling: A Nash equilibrium-based verification protocol for decentralized systems, 2024. Hyperbolic.