Orogen

Verifiable LLM inference, anchored to physical work.

Whitepaper v0.1 — Forge testnet edition
OROG · orogen.network
github.com/orogen-network

Contents

Abstract

Orogen is a decentralised network for verifiable Large-Language-Model inference. Customers consume an OpenAI-compatible HTTP API; independent GPU operators serve the requests; the operator who served each request publishes a signed receipt that binds the model, the prompt, the response, and the hardware that produced it; a stake-weighted random sample of those receipts is replayed by independent validators on independent hardware; and a Substrate-based settlement layer pays operators in a native token (OROG) that is itself burned, at oracle spot, when customers top up. The result is an inference API whose pricing, quality, and provenance are auditable end-to-end by anyone with a full node — and an income channel for the long tail of independent GPU operators that does not pass through a closed-API vendor.

This document is the technical whitepaper. A companion regulatory white paper, structured to Annex I of Regulation (EU) 2023/1114 (MiCA), is published separately at orogen.network/whitepaper-mica and treats the legal characterisation of OROG.

---

1 · The inference-trust problem

In 2026, almost every production application that calls a large language model does so through a closed HTTP API operated by one of a small number of US vendors. The customer cannot tell which weights actually served their request, which quantization, which kernel pack, which silicon revision, which jurisdiction. They cannot tell whether they were charged for the model they asked for or a cheaper substitute. They cannot tell whether the same prompt-and-seed will produce a comparable response next week. They cannot price-shop between operators of the same model without rebuilding their entire integration. And they cannot, in any practical sense, audit the bill.

Independent operators of GPU hardware face the mirror problem. There is no neutral rail on which a small datacenter, a sovereign cloud, or a power-cost-advantaged consumer rig can sell H100 hours to application developers without becoming a captive supplier of one of the same closed-API vendors. The few brokerage layers that exist are themselves closed APIs.

Orogen exists to provide the missing rail. The design rests on five claims:

1. **Trust in inference can be made verifiable.** A combination of multi-vendor hardware attestation, deterministic kernel pinning, validator replay on independent hardware, optimistic-ML challenge windows, and zero-knowledge proofs for selected sub-heads is enough to bound the rate at which an operator can cheat undetected to a level that an adversary cannot economically defeat against the available stake.
2. **Settlement can be made transparent.** Per-request receipts hashed into per-epoch Merkle roots, anchored on a public chain, give every customer, operator, and auditor the same view of who served what, when, on which hardware.
3. **Compute can be priced in fiat-stable units without a custodial middleman.** A burn-and-mint protocol decouples customer-facing pricing (USD-pegged compute credits) from the volatile asset (OROG) used to settle and to bond operators.

4. **The economic loop can be bounded without halvings or foundation discretion.** A pallet-enforced rule binds new mint per epoch to a rolling 90-day burn × an elasticity factor, with a hard floor and a hard ceiling. There is no discretion to print outside the rule.
5. **The whole stack can be open and permissionless from day one** to inspection, with permissionless operator registration phased in after a public testnet and a multi-firm audit.

The rest of this paper sets out how the system meets those claims.

---

2 · Network model

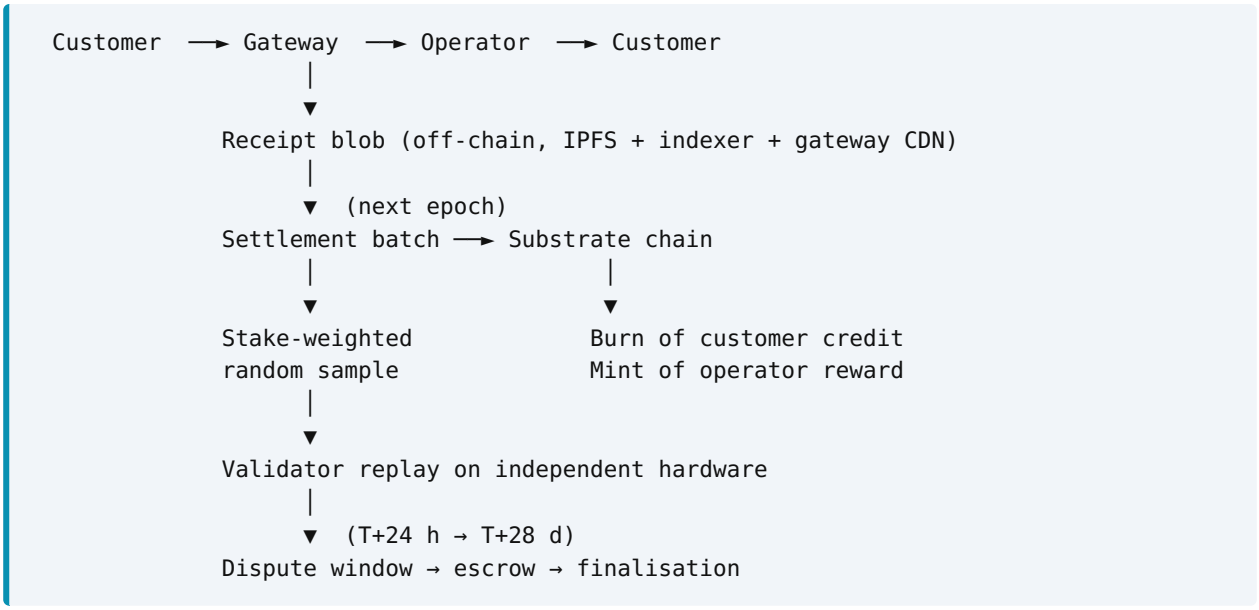
2.1 Participants

Role	What they do	What they stake / hold	What they earn
Customer	Calls the OpenAI-compatible API; pays in fiat or stablecoin	Compute credits (CUC)	Inference results
Gateway	Routes requests to operators; verifies receipts; burns OROG at TWAP spot when customers top up	Operator-class stake (gateways are themselves registered)	Routing margin out of per-job emission
Operator	Runs the inference engine on attested GPU hardware; signs the response receipt	OROG stake, slashing-eligible	75 % of the per-job emission
Validator	Replays a sampled subset of receipts on independent hardware; submits slashing evidence on mismatch	OROG stake, slashing-eligible	Share of 15 % verification pool
Optimistic-ML challenger	Opens a bisection challenge over disputed receipts	OROG bond	Share of 15 % verification pool, plus disputed operator's slash
zkML prover	Produces succinct proofs for selected sub-heads (moderation, routing, scoring)	OROG bond	Share of 15 % verification pool
Oracle member	Submits signed price samples for the OROG/USD feed	Separate oracle-pool stake, slashing-eligible	Share of treasury allocation
Governance staker	Votes on parameter changes within the timelock; serves on dispute panels by sortition	OROG stake	5 % of per-job emission, conditional on active participation

Each role is a distinct on-chain identity with its own stake bucket and its own slashing surface. An entity may hold more than one role; staking is not transferable across roles.

2.2 Inference request lifecycle

A single request walks through the network as follows:



In more detail:

1. The customer's SDK generates a 256-bit `customer_nonce`, hashes the request body, and signs both, then sends the request to a gateway endpoint that speaks `/v1/chat/completions` or `/v1/completions`.
2. The gateway selects an operator by tier, region, latency, and price. Routing also respects the customer's KV-cache prefix hint, where one is given.
3. The operator's worker runs the inference inside a Trusted Execution Environment whose freshness was attested no longer ago than seven days (validity window is per-tier). The worker produces the response, the first 64 token-position log-probability distributions (a Targon-style fingerprint), and a structured set of execution metadata: kernel-pack hash, weight tensor hash, driver and CUDA versions, GPU model, and the on-chain attestation report hash.
4. The operator signs a `Receipt` (RFC-0001) over BLAKE2-256 of all of the above. The receipt is returned to the gateway in-band with the response.
5. The gateway streams the response to the customer, pins the receipt blob to its CDN and to IPFS, and accumulates the receipt into the current epoch's settlement batch (RFC-0004).
6. At epoch close (≈ 72 minutes), the gateway publishes a single `SettlementBatch` extrinsic carrying the Merkle root over all receipts in that epoch, a per-operator summary, and the aggregate burn and aggregate mint figures.
7. The chain validates that the aggregate mint claim is consistent with the per-job emission, the BME elasticity, and the current TWAP from the oracle pool. It burns the gateway's CUC reserve and mints OROG to the operator summary accounts.
8. A commit-reveal randomness drawn from validator commitments in the prior epoch selects, by stake-weighted Fisher-Yates, which receipts each validator must replay. The sampling rate is at least 10 % network-wide, with a per-tier ceiling up to 25 % (edge) and 10 % (datacenter-premium and compliance).
9. Validators pull receipt blobs from the CDN, re-fetch model and adapter weights from content-addressed storage, replay the inference, and compare. A mismatch is filed as a slashing extrinsic (RFC-0005). A clean replay is silent.

10. The receipt enters its dispute window. From T+0 to T+24 h, operators may dispute batch-level errors. Validators may file slashing evidence at any time before the epoch closes. Filed slashings move stake into an escrow account, not into burn. At T+7 d, undisputed slashings move to a result account. At T+28 d, they burn. Disputes that overturn a slash return the escrowed stake to the operator and penalise the false claimant.

The end-state of step 10 is the moment at which a transaction is economically final: not when the response was returned, not when the batch was submitted, but when its dispute clock has run.

2.3 Hardware tiers

Operators register into one of seven hardware tiers. The tier determines the attestation matrix the operator must satisfy, the minimum USD-pegged stake they must post, and the per-tier sampling ceiling. The vendor matrix is set by RFC-0002.

Tier	Required hardware attestation	Indicative minimum stake	Sampling ceiling
`dc-premium`	NVIDIA Confidential Compute on H100 / H200 / B200 plus Intel TDX **or** AMD SEV-SNP	USD 5 000	10 %
`dc-standard`	NVIDIA CC plus Intel TDX or AMD SEV-SNP	USD 2 500	12 %
`cloud-rented`	NVIDIA CC plus Intel TDX	USD 2 500	15 %
`prosumer`	Stake-only sybil resistance	USD 2 000	20 %
`edge`	Stake-only	USD 1 500	25 %
`embed-only`	Stake-only	USD 1 000	(per epoch policy)
`compliance`	NVIDIA CC plus Intel TDX plus AMD SEV-SNP plus SOC 2 attestation hash	USD 25 000	10 %

Stake amounts are denominated in OROG and ratcheted to the USD figure every thirty days by governance vote against the on-chain oracle TWAP. Tiers without hardware attestation cannot serve workloads that require any guarantee on weights or environment; they are intended for best-effort small-model serving where the customer accepts the looser model.

---

3 · The verification stack

No single verification technique is sufficient against an adversary willing to spend. The network composes several. Each is independently insufficient; together they bound the rate at which an operator can cheat undetected to a level below the slashing economics.

3.1 Multi-vendor hardware attestation

The Orogen network treats single-vendor PKI compromise as a foreseeable threat. The attestation report for any tier above `edge` therefore combines at least two independent vendor quotes:

- An NVIDIA Confidential Compute device-identity quote, signed by the silicon and validated against the NVIDIA NVTrust RIM service and the NVIDIA Device Identity CA.
- An Intel TDX quote signed by the platform and validated against Intel Trust Authority.
- An AMD SEV-SNP attestation report validated against the AMD Key Distribution Service.

The attestation service aggregates these into a `CombinedAttestationReport`, hashes it, and submits the hash on-chain via `pallet-attestation-registry`. The on-chain record is small (operator ID, report hash, GPU UUID for cross-account collision detection, vendor-set bitflags, expiry block). The full report blob lives in the indexer plus IPFS.

The chain maintains a Certificate Revocation List addressable by firmware hash, GPU UUID, model hash, or vendor PKI chain hash. CRL writes are gated by a 5-of-7 multisig with a 14-day timelock under normal conditions and a 3-of-7 fast-track for sanctions hits and known-CVE firmware. Operators query the CRL every ten minutes and on every job start; an operator running past a CRL expiry is auto-slashed via the `AttestationStale` fault code.

The network does not claim defence against silicon-undisclosed side channels. Several known ones (unencrypted Hopper NVLink, BAR0 register leakage, bimodal-timing batch-size leakage, GPUBreach) are acknowledged in operator terms of service and customer documentation; routing filters expose features such as `nvlink_encryption: bool` so customers can require Blackwell-class operators for their workload.

3.2 Signed response receipts

Every inference produces a `Receipt` (RFC-0001) carrying the fields necessary to reconstruct and replay it. The structure is canonical, SCALE-encoded on chain, JSON off chain, and includes:

- The job identifier and timestamp.
- The operator hotkey and the gateway that routed the job.
- The model identifier (content hash), the exact weight-tensor hash served (this catches a quantization swap), and the LoRA adapter hash if one was applied.
- The customer's nonce, the SHA-256 of the canonical-encoded request, and the SHA-256 of the canonical-encoded response.
- A 64-token log-probability fingerprint, sufficient for the Targon-style statistical check.
- The kernel-pack hash, which pins the serving engine, the kernel version, and the chat-template revision.
- The GPU model string, driver version, and CUDA version.
- The on-chain attestation report hash (resolvable in `pallet-attestation-registry`).
- An optional batch-invariant proof (used by deterministic-mode SGLang).
- KV-cache metadata: prefix hint, cache-hit flag, blocks used.
- An ed25519 signature by the operator hotkey over BLAKE2-256 of the canonical encoding of every field above.

Receipts are roughly 1.5 KiB worst case. They are not stored on chain individually — only as leaves of a Merkle root in the per-epoch settlement batch.

3.3 Validator replay on independent hardware

A validator that draws a receipt in the sample for an epoch:

1. Pulls the receipt blob from the indexer or the gateway's pinned CDN.
2. Pulls the model and adapter weights, by content hash, from the model registry.
3. Replays the inference on independent hardware using the same kernel pack, weight hashes, GPU model and driver-and-CUDA pair declared in the receipt — within a per-tier floating-point ε.
4. Recomputes `response_hash` and the 64-token log-prob fingerprint.
5. Compares.

A match is silent. A mismatch is filed as evidence (`WrongModel` , `WrongResponse` , `LogProbDrift` , `CacheReplay` , `QuantizationSwap` , or `KernelPackMismatch`) into a slashing extrinsic. Severities run from 0.5 % (a non-deterministic kernel used in a deterministic-tier slot) to 10 % (a model substitution).

Validators are themselves staked and slashable for false claims. Watcher false-positive penalties (RFC-0005) apply ×10 on second offence within ninety days, plus a permanent ban.

3.4 Optimistic-ML challenge windows

For the tier of workloads where replay is too expensive (long-context calls, large mixture-of-experts batches), the network supports an optimistic-ML model. The operator's claim is posted, and stands unless a challenger opens a bisection game over the disputed inference within the challenge window. Bisection converges on a minimal disputed sub-trace which is then replayed by independent hardware, with the loser paying the bond. The bisection game state machine is implemented by `opml-challenger` ; participation is open to any staked challenger.

The combination of validator replay (covering all tiers) and opML (covering workloads where naive replay is uneconomic) gives the network broad coverage with bounded verifier overhead.

3.5 Zero-knowledge proofs for selected sub-heads

Some inference work — moderation classifiers, routing decisions, credit-scoring sub-models — produces small outputs from small models and is well-suited to a succinct proof. For those, an operator may attach a zero-knowledge proof of correct execution (via the EZKL adapter) instead of relying on replay. The proof is verified on chain or by the gateway, and consumes no validator resources.

zkML is **not** offered for full transformer inference in the current scope. The cost is not yet economic. Where it becomes economic — likely for sub-heads of mixture-of-experts routers, and for moderation pipelines — it will be added on a per-workload basis.

3.6 Customer nonces and commit-reveal sampling

Two independent mechanisms keep the operator honest about which receipts will be sampled.

The first is the **customer nonce** (RFC-0007). The customer's SDK generates a 256-bit random nonce per request, signs it together with the request, and the operator includes it in the receipt. The operator cannot anticipate which exact prompts the customer will send, and cannot pre-compute a forged response without committing to a specific nonce. The gateway also maintains a Bloom filter over the last 24 h of nonces as a second-line replay defence; the chain burns a short-hash form of each nonce at settlement.

The second is **commit-reveal sampling randomness** (RFC-0006). Each epoch, validators commit to a 256-bit pre-image; the next epoch, they reveal it. The per-epoch sampling seed is derived deterministically from the revealed values. The operator therefore cannot know in advance which of its receipts will be sampled, even after the receipts have been published. Defection from the commit-reveal protocol — failure to reveal, or revealing the wrong pre-image — costs the validator that epoch's emissions and, on second defection within thirty epochs, its permit.

3.7 Watermarking

Where a model author supports model-side watermarking (recent open-weight families increasingly do), the watermark is preserved end-to-end through the serving pipeline and exposed in the receipt. Watermarks are a defence-in-depth signal: a sampled receipt whose response carries the wrong watermark for the declared weights is a `WrongModel` slash. The network does not require a watermark; it uses one where it exists.

3.8 What the stack does not claim

The verification stack does **not** claim:

- That an operator running entirely correct hardware and software cannot return a particular output for a particular input. It claims that they cannot do so while substituting cheaper weights, or while running a non-deterministic kernel in a deterministic tier, without a non-negligible probability of being caught and slashed at the relevant severity.
- Defence against vendor silicon-undisclosed side channels.
- That zkML covers full transformer inference; it does not, in the current scope.
- That validator collusion at scale is impossible. The pool-disjoint validator-watcher pattern (separate verification by validators and watchers, drawn from non-overlapping stake pools) and the multi-signature requirement on high-severity slashings (3 corroborators for 10 %, 5 for 100 %) raise the cost of collusion, but do not eliminate it.

The honest framing is that the network bounds the rate of undetected cheating to a level below the slashing economics for the available stake — not that cheating is impossible.

---

4 · Chain and runtime

4.1 Substrate plus Frontier EVM

The Orogen chain is built on Substrate, with AURA block production, GRANDPA finality, and the Frontier EVM compatibility layer enabled for Ethereum-tooling interoperability. OROG is a **native pallet asset**, not an ERC-20: balances live in `pallet-balances`, not in a deployed contract. The EVM layer exists so that wallets, indexers, and DeFi tooling that speak Ethereum JSON-RPC continue to work; for native settlement and slashing extrinsics, the chain's own pallets are the canonical surface.

A planned Snowbridge fork (deferred to early 2028) provides a future trust-minimised bridge to Ethereum L1; until then, OROG lives only on the Substrate chain.

4.2 Pallets

The runtime composes eleven Orogen-specific pallets in addition to the Substrate FRAME defaults:

Pallet	What it does
`pallet-operator-registry`	On-chain registry of operators with attestation, tier, stake, geo-region, IP-hash, sanctions-check proof. Implements RFC-0009.
`pallet-attestation-registry`	Stores attestation report hashes, GPU UUIDs, expiry windows, the CRL. Implements RFC-0002.
`pallet-receipt-anchor`	Anchors per-batch Merkle roots; resolves Merkle proofs for dispute.
`pallet-batch-settlement`	Receives `SettlementBatch` extrinsics, validates aggregates, dispatches burn and mint. Implements RFC-0004.
`pallet-bme`	The burn-and-mint engine: enforces the emission rule, computes per-epoch mint headroom, exposes the BME state.
`pallet-yuma-consensus`	Per-epoch incentive aggregation: stake-weighted sampling, validator scoring, emission distribution. Inspired by the Yuma family but adapted to per-job receipts rather than per-block validator votes.
`pallet-oracle-twap`	Aggregates price samples from the oracle pool into a `CurrentTwap`. Implements RFC-0008.
`pallet-slashing`	The slashing surface: fault codes, severity table, evidence submission, escrow, dispute and arbitration. Implements RFC-0005.
`pallet-vesting`	Vests the founder, team, investor, and reserve allocations on the schedules disclosed in the regulatory white paper.
`pallet-governance`	On-chain governance: proposals, voting, timelocks, multisig overlay for sensitive parameters.
`pallet-treasury`	Holds the protocol treasury; spends only by governance vote within a runway-bounded rule.

The runtime exposes the ten standard Substrate runtime APIs in addition to four Orogen-specific ones for receipt resolution, attestation lookup, emission read-out, and oracle TWAP read-out.

4.3 Batch settlement

A naive system that wrote one extrinsic per inference would write on the order of 100 million extrinsics per day at modest customer volume — orders of magnitude beyond any general-purpose chain. Orogen instead aggregates receipts into batches: one extrinsic per epoch per gateway, carrying the Merkle root over that epoch's receipts plus a per-operator summary (RFC-0004).

At target volume this reduces the chain load to roughly 280 000 extrinsics per day across the entire network. Batch headers occupy about 10 KiB each; at 280 000 batches per day the chain accrues approximately 2.8 GB per day of header data on archive nodes. Full receipt blobs live off-chain (indexer plus IPFS pin plus gateway CDN); the Merkle root is the only on-chain anchor.

Per-batch validation enforces three invariants:

1. The receipt-count and operator-summary aggregates are internally consistent.
2. The aggregate mint claim does not exceed the epoch mint headroom from `pallet-bme`.
3. The aggregate burn satisfies the burn-to-mint ratio at the current TWAP, modulo the elasticity factor.

A batch that fails any of these is rejected at extrinsic-validation time. A batch that passes opens its dispute window (24 hours for batch-level disputes; epoch-end for validator slashings).

4.4 Operator registration

Registration into the network is an extrinsic carrying:

- A coldkey-and-hotkey pair (long-term stake-holding identity and short-term operational signer).
- The hash of the operator's combined attestation report.
- The declared tier.
- The amount of OROG staked, denominated to satisfy the USD-pegged minimum for the tier.
- ISO-3166 region code.
- A BLAKE2 hash of the operator's public IP /24 (for geographic and network diversity diagnostics, not for routing).
- A sanctions-check proof: a signed message from the network's sanctions screener confirming that the coldkey passed screening within the last 24 hours, naming the upstream provider (Chainalysis, TRM, or Elliptic).
- Separate coldkey and hotkey signatures, proving control of both.

Validation checks the signatures, looks up the attestation in the registry, confirms the vendor set matches the tier, and runs a device-cert collision check across the registry. A GPU UUID that appears under two coldkeys triggers `DeviceCertCollision` — a 100 % slash with no dispute (the only such case besides `SanctionsHit`). This is the network's primary defence against the "same machine, two stake pools" sybil pattern.

Operators may re-register a hotkey to a new device by deregistering first; staking is preserved across re-binding.

4.5 Oracle pool

OROG/USD pricing for settlement is produced by a price-oracle pool that submits signed samples from four sources:

- Binance OROG/USDT (post-listing).
- Coinbase OROG/USDC (post-listing).
- A Uniswap V3 OROG/USDC pool on Ethereum, observed via the future Snowbridge adapter.
- A foundation-bonded Uniswap V4 hook-AMM running on the Orogen chain itself from day one.

Per epoch, each source's TWAP is computed; the median across the four is taken; any source whose TWAP deviates by more than 5 % from the median is dropped; the remaining sources are stake-

weighted into a single `CurrentTwap`. If fewer than two sources survive the cut, the pool falls back to the last accepted TWAP and emits a degraded-mode event.

The lookback window is intentionally long at launch: 4–12 hours for the first 18 months post-TGE, tunable by the oracle operators within bounds; 30 minutes thereafter, by governance proposal. The long window blunts flash-loan and thin-market attacks at fresh listings — at the cost of latency in incorporating genuine market moves into settlement. Window changes require 5-of-7 multisig plus a 14-day timelock; emergency reductions below 30 minutes are not permitted without a runtime upgrade.

An emergency-pause multisig is wired into `pallet-oracle-twap`. Oracle pool members are themselves staked and slashable for provable manipulation under `pallet-slashing`'s `OracleManipulation` fault code.

4.6 RPC endpoint contract

At TGE the network commits to running, or contracting partners to run, **three independent JSON-RPC endpoints** (RFC-0010): one foundation-operated, two partner-operated. Each provides Substrate JSON-RPC, Frontier EVM RPC, WebSocket subscriptions, and an archival service, with a published 99.9 % uptime SLA, a public status page, and a documented incident-response contract. Operator daemons auto-failover across the three within two minutes; if all three are down, the operator's local validator fallback continues to produce signed receipts whose final settlement is reconciled when an RPC returns.

This is a deliberate trade between decentralisation and operational reality: a brand-new chain has no organic full-node population yet, and a single foundation-operated RPC would be a single point of failure. The three-provider rule is structural until organic full-node density is sufficient.

---

5 · Burn-and-mint economics

5.1 Customer surface and compute credits

The customer surface is dollar-denominated. A customer tops up a gateway with fiat or stablecoin and receives a non-transferable USD-pegged credit balance, denominated in **Compute Units of Credit (CUC)**. One CUC is intended to clear at one USD of compute at the prevailing operator pricing for the requested tier. The credit is not a stablecoin; it cannot leave the network; it is consumed only against inference work.

Behind the scenes, when the customer tops up:

1. The gateway accepts the fiat or stablecoin payment through its KYC- and AML-compliant payment processor.
2. The gateway purchases OROG on the open market or from its own reserve at the prevailing TWAP.
3. The gateway burns that OROG against `pallet-bme`, which credits the customer's CUC balance at the on-chain TWAP rate.

When the customer's request is served and the batch settles, the chain mints fresh OROG to the operator, drawn from the per-epoch emission headroom. The customer's CUC balance is debited by the operator's price for the work.

5.2 Per-job emission split

The OROG minted on each finalised job is allocated as follows:

Recipient	Share	What for
Operator	75 %	The work itself
Verification work	15 %	Validators, opML challengers, zkML provers
Treasury	5 %	Protocol expenses, audits, grants
Active-security governance stakers	5 %	Stakers who participate in dispute panels, oracle attestation, governance votes within the rolling window

The "active-security" share is not a passive yield. It is paid only to governance stakers whose participation across panels, oracle attestations, and parameter votes exceeds a rolling threshold over the prior thirty days. The intent is to align governance compensation with governance work.

5.3 Emission policy

The per-epoch mint is bounded by a pallet-enforced rule:

```
target_epoch_mint = MIN(
    BOOTSTRAP_CAP_THIS_EPOCH,
    RollingBurn90d × elasticity_factor / epochs_per_90d,
    HARD_CEILING_PER_EPOCH                  # 5 % of supply per year
)
target_epoch_mint = MAX(
    target_epoch_mint,
    FLOOR_PER_EPOCH                         # 0.5 % of supply per year
)
```

`BOOTSTRAP_CAP` is set at 8 % of supply per year in Year 1, ramping linearly to 4 % over Year 2, and the rule is then governed by `RollingBurn90d × elasticity_factor` thereafter. `elasticity_factor` is governance-set within `[0.8, 1.5]`.

Three properties follow:

- **There is no halving schedule.** Emission is bounded by burn (i.e. by customer demand), not by a calendar.
- **There is no foundation discretion to mint.** The Foundation cannot mint outside the rule. The rule's structure is itself protected: changes to floor, ceiling, or bootstrap shape require a runtime upgrade on a 90-day timelock; changes to `elasticity_factor` require a governance vote on a 14-day timelock.

- **The system can sustain itself in either direction.** A demand floor of 0.5 %/yr keeps validators and operators paid even in lean periods; a ceiling of 5 %/yr prevents runaway inflation in boom periods. The rolling-90-day window is short enough to track real demand and long enough to prevent reactive spikes.

5.4 Token allocation at TGE

The initial supply at the Token Generation Event is allocated, in percentages of initial supply:

Bucket	Share	Notes
Mining / emission pool	40 %	Released over ≈10–15 years per the emission rule above
Pre-launch investors (seed + Series A)	12 %	48-month cliff + 48-month linear vesting
Team & advisors	12 %	48-month cliff + 48-month linear vesting
Treasury (Foundation runway)	10 %	DAO-controlled, runway-bounded
Ecosystem grants & RFPs	8 %	DAO-controlled, multi-year
Provider bootstrap allocation	8 %	Released as part of operator emission over a 24-month ramp
Public / community	5 %	Fully unlocked at TGE
Strategic reserve	5 %	DAO-controlled, 12-month cliff

The exact integer total supply at TGE is set out in the regulatory white paper. Vesting and lock-up are enforced by on-chain contracts in `pallet-vesting`. Foundation entity arrangements and the regulatory classification of OROG (a crypto-asset other than an asset-referenced token or e-money token, under MiCA Art. 3(1)(5)) are detailed in the companion MiCA white paper.

---

6 · Slashing and dispute

6.1 Fault codes

The slashing surface is intentionally enumerated. Every fault code corresponds to a single class of provable misbehaviour, with a fixed base severity and explicit evidence requirements:

Fault code	Severity	Triggered by
`WrongModel`	10 %	The replayed weight-tensor hash does not match the receipt
`QuantizationSwap`	10 %	The actual quantization differs from the declared quantization
`WrongResponse`	5 %	The replayed response hash does not match the receipt
`CacheReplay`	5 %	`cache_hit=true` claimed without a fresh-compute signal
`LogProbDrift`	2 %	The log-prob fingerprint diverges beyond the per-tier ε
`AttestationStale`	2 %	Operating past a CRL grace period
`KernelPackMismatch`	0.5 %	A non-deterministic kernel was used in a deterministic-tier slot
`ValidatorCollusion`	10 %	A cross-validator outlier is independently confirmed
`BatchOvercommit`	10 %	A gateway claimed mint that the burn does not support
`FakeBurn`	50 %	Gateway-side fraud against the burn engine
`DeviceCertCollision`	100 %	The same GPU UUID appears under two coldkeys (no dispute)
`SanctionsHit`	100 %	Stake is frozen, not burned, on confirmed sanctions hit (no dispute)
`OracleManipulation`	(severity per evidence class)	Oracle pool member submits manipulated samples
`HeartbeatMiss`	0 % (soft)	Sustained missed heartbeats; no slash, emission decay only

6.2 Submission and escrow

Slashing evidence is submitted by validators or watchers, drawn from pool-disjoint stake pools. The submission extrinsic carries the operator identifier, the fault code, the hash of the off-chain evidence, the related receipt and batch identifiers, and a vector of corroborating signatures. The required corroboration count scales with severity: one signer for 0.5 %, two for 2–5 %, three for 10 %, three plus an independent burn-engine signature for 50 %, and five (or a multisig fast-track) for 100 %.

A valid submission moves the slashed stake into an escrow account, **not into burn**. The operator is notified off-chain and has seven days to dispute. If no dispute is filed, at T+7 d the stake moves to a slashing-result account; at T+28 d it is burned. A dispute that overturns the slash returns the escrow to the operator and penalises the false claimant.

6.3 Dispute and arbitration

A disputing operator posts a 10 % dispute bond and a counter-evidence hash. The dispute is then arbitrated by a panel selected by on-chain sortition from the top 50 stake-holders, excluding the slashed operator, the slashing validator, anyone in the same operator's coldkey group, and operators registered within the last 14 days (anti-flash-panel-stack). Each panelist posts a 1 % stake bond; panel votes are `Uphold`, `Overturn`, or `Insufficient`. A final ratification by the Foundation multisig closes the dispute.

6.4 Caps, rate limits, and circuit breakers

- A single incident is capped at 10 % of stake.
- Cumulative slashing against a single operator is capped at 50 % over a rolling 30 days, 30 % over rolling 24 hours.
- A network-wide circuit breaker kicks in if total slashing across the network exceeds three times the rolling 24-hour baseline: `pallet-slashing` enters a paused state requiring a 5-of-7 multisig and a 2-day public delay to resume.

6.5 Watcher economics

Watchers are an independent, staked detection layer. They register by posting a bond (≥ ~1 ETH-equivalent in OROG) and earn a share of any slashing they correctly trigger. A watcher whose evidence is rejected loses the full bond on first offence, the bond × 10 on second offence within 90 days, and is permanently banned on the second offence. Forged evidence is referred to law enforcement under the applicable jurisdiction.

The asymmetry is deliberate: the network rewards honest detection generously and punishes false detection at a multiple, so that the equilibrium pulls toward truthful reporting even when reporting is profitable.

---

7 · Governance

Governance is on-chain, in `pallet-governance`, with two timelock tiers and a multisig overlay:

- **Soft parameters** (sample rate within bounds, oracle window within bounds, elasticity factor, per-tier stake floors, treasury spends) are governance-mutable behind a 5-of-7 multisig and a 14-day timelock.
- **Hard structure** (the emission rule itself, the slashing severity table, the fault-code enum, the constitutional bound on the elasticity factor) requires a runtime upgrade with a 90-day timelock. Holders affected by such a change have sufficient time to transfer or otherwise act before it takes effect.
- **Sortition arbitration** (dispute panels under §6.3) is on-chain, panel-selected from stake, and ratified by the Foundation multisig.

Two specific overrides exist:

- **CRL fast-track.** For sanctions hits and known-CVE firmware hashes, the CRL multisig is a 3-of-7 fast-track without the 14-day timelock. The intent is to revoke a compromised key set, or to stop an operator running known-vulnerable firmware, within hours rather than weeks.
- **Emergency pause.** The oracle pool and the slashing pallet expose an emergency-pause multisig (5-of-7) so that confirmed manipulation or runaway slash cascades can be halted while a governance proposal is drafted.

OROG holders vote in proportion to stake. Voting confers no claim against the Foundation or any affiliated legal entity; it is a protocol-level governance right over the parameters listed above. The full set of rights and obligations attached to OROG is set out in the regulatory white paper.

---

8 · Customer interface

8.1 Gateway API

The customer-facing surface is an OpenAI-compatible HTTP API exposed by the `gateway-router` service. Supported endpoints at v0.1 include `/v1/chat/completions`, `/v1/completions`, `/v1/embeddings`, and the streaming variants of each. Authentication is by API key with the `orog_` prefix; routing parameters are passed in the request body for tier, region, and per-million-token price ceilings.

Customers can either run their own gateway (against any of the three RPC providers) or use a foundation-listed gateway. Gateways are themselves registered, attested, and slashable — see RFC-0004 § `FakeBurn` and § `BatchOvercommit`.

8.2 SDKs

Three first-party SDKs ship at v0.1:

Package	Language	Use case
`orogen-sdk`	Python	Direct application integration
`@orogen/sdk`	TypeScript	Browser and Node application integration
`litellm-orogen-provider`	Python	LiteLLM backend driver, for any application already speaking LiteLLM

All three speak the gateway's OpenAI-compatible surface, generate and sign customer nonces, and verify operator response receipts on the customer side using `@noble/ed25519` (TypeScript) or the `cryptography` package (Python).

8.3 Pricing model

Operator pricing is published per tier, per region, per million tokens, for both input and output. The customer pays the operator's price out of CUC balance; the operator is minted the per-job emission share. The two flows are independent: the customer's payment is a debit against their CUC balance at the operator's posted price; the operator's mint is governed by the network's emission rule, not by the customer's payment. The gap between the two is, in equilibrium, the network's BME subsidy — which scales down to zero as customer revenue catches up with emission, and the chain becomes self-funding.

---

9 · Risks and honest limitations

The Orogen network claims to bound, not eliminate, the rate of undetected misbehaviour. The following risks are material to that claim.

Trusted Execution Environment compromise. A CVE in NVIDIA Confidential Compute, Intel TDX, or AMD SEV-SNP affecting the attestation chain could, in principle, allow an operator to forge attestation reports for a window of time. The network responds with CRL fast-track entries, mandatory re-attestation, and revocation of all affected `vendor_pki_chain_hashes`. The window between disclosure and CRL response is the practical risk; the system targets hours, not days.

Vendor PKI failure. The same logic applies one layer up: if a vendor's root of trust is compromised, the multi-vendor rule means that workload routed through that vendor's attestation alone is at risk until governance revokes the chain. The defence is that operators above `edge` must satisfy at least two independent vendors; a single-vendor PKI compromise cannot, by itself, collapse the network.

Oracle manipulation. A coordinated attack on the OROG/USD price feed could move settlement burn-to-mint ratios. The network's defences are a long TWAP window at launch (4–12 hours), a 5 % cross-source outlier clip, a stake-weighted aggregation, a static-fallback rule when fewer than two sources survive, an emergency-pause multisig, and slashable oracle pool members. None of these is sufficient against an adversary that controls 3-of-4 sources simultaneously.

Verifier collusion. Validators and watchers are drawn from pool-disjoint stake pools, and high-severity slashings require 3–5 corroborating signatures. None of this prevents a coordinated coalition. The economic defence is that watcher false-positive penalties scale ×10 on second offence and a permanent ban; the structural defence is that watchers are an independent, stake-bonded layer alongside validators rather than the same set of actors.

Smart-contract and runtime-pallet bugs. All on-chain logic — pallet bodies, vesting contracts, the slashing escrow, the oracle aggregation — will undergo multi-firm audit prior to mainnet TGE. Discovered issues are gated by the 90-day runtime-upgrade timelock; in-flight stake is protected by the escrow-not-burn rule.

Regulatory change. Crypto-asset regulation in the European Union and elsewhere is actively evolving. MiCA itself may evolve through delegated acts and implementing standards. The network's structural responses are (i) a regulatory white paper aligned to MiCA Annex I, (ii) a Foundation-issuer

model intended to maintain offeror-issuer separation, and (iii) integration of `sanctions-screener` at primary distribution. Specific regulatory open items are tracked publicly.

Environmental footprint. The Orogen blockchain itself uses Substrate's AURA/GRANDPA consensus and has a footprint comparable to other small PoS L1s. The *application* — LLM inference performed on GPU hardware — is energy-intensive, and the network's growth means total emissions grow with usage. Per-job environmental indicators in the format required by the MiCA delegated regulation on sustainability disclosures are reported in the companion MiCA white paper.

Operator and oracle concentration. A network whose top ten operators serve more than half of traffic, or whose top three oracle members weight more than half of TWAP, is structurally fragile. The /24 IP-hash diversity tag, the per-tier sampling ceilings, the device-cert collision rule, and a published concentration dashboard exist so that concentration is observable; the social and economic responses to it are governance-level.

Total loss of value. OROG may lose its value in part or in full, may not always be transferable, and may not be liquid. The full enumeration of risks is in the regulatory white paper, which is the governing risk disclosure for any acquisition decision.

---

10 · Roadmap

The network's milestones, as planned at the date of writing, are:

Phase	Window	Description
Stealth / research	2025 H2 → 2026 Q1	Research dossier, design, initial team. (Complete.)
Centralised stack	2026 Q1 → 2026 Q2	Inference gateway operational; first paying customers in pilot. (Complete.)
Forge testnet	2026 Q2 → 2027 Q1	Public testnet "Forge" live. Operators register, validators replay, customers route OpenAI-compatible traffic. Testnet OROG only — no economic value.
Multi-firm audit	2027 Q1 → 2027 Q2	Chain and protocol audits.
Mainnet TGE	2027 Q2	Token Generation Event, mainnet block production, BME loop active from day one.
Permissionless transition	2027 Q3	Operator registration becomes fully permissionless; multi-vendor attestation required by tier.
Subsidy < 1×	2028	Annualised mint USD ≤ annualised real customer revenue USD on a rolling 90-day basis.

The following items are explicitly **out of scope** for the current phase, with the windows at which they are scheduled or deferred:

- Snowbridge fork to Ethereum: deferred to Q1 2028.
- cuPOW pool (optional Hopper-only proof-of-useful-work emission lane): deferred to Q4 2028.
- Mobile wallet (uniffi + BIP-32): deferred to Q4 2027.
- Full transformer zkML: out of scope until economically viable.
- Cross-operator KV-cache sharing: out of scope under current trust assumptions.

These items will, where retained, become subject to their own design RFCs and audit cycles before they are integrated.

---

11 · Further reading

- **Documentation site** — docs.orogen.network
- **Public chain RFCs (canonical specs for the integration contracts summarised above)** — github.com/orogen-network/chain-tooling-rust/specs/
- **Block explorer** — explorer.orogen.network
- **Attestation explorer** — attestation.orogen.network
- **Network status** — status.orogen.network
- **Subsidy dashboard** — subsidy.orogen.network
- **Operator onboarding** — onboarding.orogen.network
- **Regulatory white paper (MiCA Annex I)** — orogen.network/downloads/orogen-mica-whitepaper.pdf

The source for every component named in this paper is published under the `orogen-network` GitHub organisation. The Forge testnet is live at the date of publication; the public RPC endpoint, the on-chain genesis, and the validator and operator clients are linked from the documentation site.

This whitepaper is a technical description of the Orogen network as currently designed and partially implemented. It is not an offer to sell OROG, an investment prospectus, or legal or financial advice. The MiCA-aligned regulatory white paper, published separately, is the governing document for any acquisition decision regarding OROG.