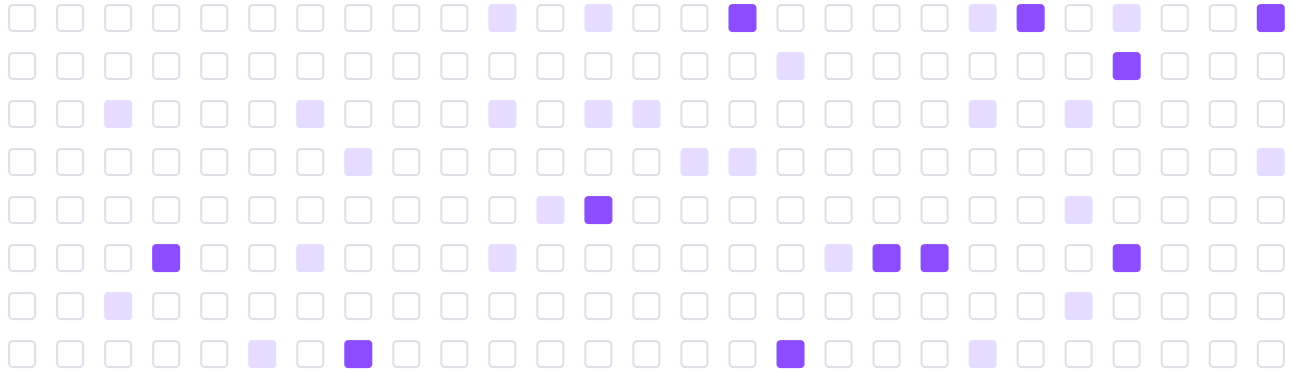




TECHNICAL YELLOW PAPER

Private and Verifiable AI Compute on Untrusted Hardware

The design of TasQ v1.0: assurance modes, ledger, economics, and the benchmarking, all mentioned here.

AUTHORS

The TasQ Project
Lead author: Newman
Wang

VERSION

1.0, revision 2
Design specification

DATE

March 2025
Revised October
2026

LINKS

tasqnetwork.io
github.com/tasqProject

ABSTRACT

Running AI workloads on rented hardware requires trusting the operator with model weights, prompts and training data, and trusting that the job was executed as requested. TasQ is a compute marketplace that replaces part of this trust with mechanisms a client can check. Jobs are split into execution units and run on contributor GPUs and CPUs under one of three assurance modes, chosen per job and recorded with the result: hardware-attested enclaves for confidentiality, redundant execution with quorum agreement and hidden audits for integrity, and zero-knowledge proofs where proving is affordable. Only commitments and attestations are written to a ledger, and long-lived records rely on hash-based and symmetric primitives. We specify the design and its trust assumptions, and report reproducible simulations and microbenchmarks. Simulation shows that because seat selection is public, colluding contributors can attack exactly when they hold a quorum, so slashing alone does not deter them; hidden audits restore deterrence, and the simulated break-even matches our admission rule to within five per cent. GPU and enclave costs are modelled rather than measured, and the network has not yet been evaluated on a testnet.

Keywords: confidential computing · verifiable inference · GPU marketplaces · zero-knowledge proofs · redundant execution · post-quantum cryptography

1 Introduction

AI workloads handle some of the most sensitive data an organisation holds. Model weights embody substantial training investment, prompts and documents carry personal and commercial information, and fine-tuning sets are often proprietary. The standard way to run these workloads is to rent GPUs from an operator and trust it twice: not to read the data, and to run the job correctly. Neither is checkable by the client.

Two families of tools address these problems separately. Confidential computing isolates a workload inside hardware the operator cannot inspect, and proves this to the client through remote attestation [1]. Verifiable computation lets a client check a result without re-running it, either by replication across independent parties [2] or by succinct proofs [3][4]. Separately, open GPU marketplaces have shown that idle capacity can be pooled, but generally without either guarantee.

TasQ combines these tools in a single marketplace. Each job declares an assurance mode, the protocol enforces it, and the ledger

records which mode produced each result, so a client always knows what a result actually proves. This paper makes four contributions:

- A threat model for AI computation on untrusted, voluntarily contributed hardware, with explicit goals and non-goals (§3).
- An execution design with three assurance modes, attested, redundant and proven, recorded per job (§4–§6).
- An admission rule for redundant execution that remains valid against contributors who coordinate, based on hidden audits (§9).
- Reproducible simulations and microbenchmarks, including a negative result that motivated the audit mechanism (§12).

Remark 1 (Scope) *TasQ uses no quantum hardware; post-quantum refers only to the choice of cryptographic primitives (§6.3). In v1.0 the coordination layer is operated by the TasQ Project, so the network is not yet decentralised. Results in §12 come from simulations, models and*

microbenchmarks of individual primitives, not from a deployed network.

2 Background and Related Work

2.1 Confidential computing

Trusted execution environments such as Intel SGX [1] run code in memory that the host cannot read and produce a signed attestation of what is running. Recent data-centre GPUs, starting with NVIDIA’s Hopper generation, add a confidential mode that extends this protection to GPU memory. The guarantee depends on the hardware vendor and has been weakened in the past by side-channel attacks [5].

2.2 Verifiable computation

Pairing-based zk-SNARKs such as Groth16 [3] give constant-size proofs but need a trusted setup and rely on elliptic-curve assumptions. zk-STARKs [4] need no trusted setup and rely only on hash functions, at the cost of larger proofs. Proving neural-network inference (zkML) is possible for small models [6], but proving cost still grows far faster than native execution, which rules it out for large models today.

2.3 Post-quantum cryptography

Shor’s algorithm [7] would break RSA and elliptic-curve schemes, including Ed25519, X25519 and pairing-based SNARKs. Symmetric ciphers and hash functions are only modestly affected [8][9]. NIST standardised ML-KEM [10] and ML-DSA [11] in 2024. The practical risk today is that data or signatures recorded now are attacked later [12].

2.4 Decentralised compute

Truebit [2] verifies off-chain computation through an interactive game backed by deposits. Golem, iExec and Akash operate marketplaces for CPU, GPU or container capacity, and iExec uses trusted execution for confidentiality.

Replication-based verification, attestation and staking are therefore all established techniques. TasQ’s contribution is to combine them under a per-job assurance label and to bound, economically, what replication can be trusted to secure.

3 System and Threat Model

3.1 Participants and notation

- **Clients**, typically AI developers and enterprises, submit inference, fine-tuning or data-processing jobs with an execution policy.
- **Contributors** run the TasQ runtime and lend GPU, CPU, memory and bandwidth. They lend processing only, never their own storage.
- **Verifiers** check proofs and attestation reports and compare output commitments.
- **Coordinators** admit jobs, run the scheduler and operate part of the ledger. In v1.0 they are run by the TasQ Project.

A job T is split into execution units $u_1, \dots, u_n$ forming a directed acyclic graph. Each unit has a resource requirement $\text{req}(u)$ and an input commitment $c_u = \text{BLAKE3}(\text{Enc}(x_u))$. Contributor j has a key pair, a capability vector v_j , a reputation $R_j \in [0, 1]$ and a stake S_j . Under redundant execution a unit runs on r contributors and a result is accepted when q of them agree.

3.2 Adversary

The adversary controls a fraction p of eligible contributor seats, may return arbitrary outputs, attempts to read job data, registers many identities, and records all traffic for later cryptanalysis. Up to f of n ledger validators may be Byzantine, with $n \geq 3f + 1$ [13]. The adversary cannot break AES-256, BLAKE3 or proof soundness, cannot bias seat selection, and cannot break enclave hardware beyond known side-channel classes.

3.3 Goals and non-goals

- **Integrity.** An accepted result is correct unless the adversary holds a full quorum (mode R), breaks proof soundness (mode P) or breaks the enclave (mode A).
- **Confidentiality.** Verifiers and ledger readers never see job data. Executors see it only inside attested enclaves, and only when the client selects mode A.
- **Accountability.** Assignments, results and penalties are attributable to pseudonymous identities on an append-only ledger.
- **Non-goals.** v1.0 does not hide network metadata, does not compute on encrypted data [14], and does not offer trustless coordination.

3.4 Trust in the v1.0 operator

Because the TasQ Project runs coordination in v1.0, clients must know what that operator can and cannot do.

- **It can** delay, refuse or reorder jobs, and it controls when the validator set changes.
- **It cannot** undetectably bias seat or audit selection: seats are derived from a public seed, and audits from a seed committed in advance and revealed after outputs are committed (§5.2, §9.4).
- **It cannot** forge results in modes A or P: attestation reports and proofs are checked against keys and commitments it does not control.
- **It cannot** read mode-A data: decryption keys are released by the client's own key service directly to an attested enclave.

4 Architecture

TasQ separates three planes (Figure 1). The client plane encrypts inputs, commits to them and submits a job with a policy naming its assurance mode. The execution plane schedules units onto contributors and returns outputs together with

proofs or attestation reports. The verification and settlement plane checks these, resolves redundant results, records the outcome and releases payment.

Code and data travel separately. Execution logic goes to the compute network; payloads go to the storage layer, from which contributors fetch only the encrypted segments their unit needs (§10).

Remark 2 (Ledger contents) *The ledger records commitments, proof references, attestation digests and payment events. It never records prompts, weights or data.*

5 Execution

5.1 Decomposition and scheduling

A submitted job is split into units annotated with GPU and CPU requirements, memory footprint, latency tolerance and workload type. Data-parallel work follows the MapReduce pattern [15], and dependencies are ordered by DAG scheduling [16]. For each ready unit the scheduler ranks eligible contributors by

$$\kappa(u, j) = \mathbf{1}[v_j \geq \text{req}(u)] \cdot (\alpha R_j + \beta \hat{s}_j - \gamma \ell_{u,j}) \quad (1)$$

where $\hat{s}_j$ is the normalised benchmark score and $\ell_{u,j}$ an estimated latency. The weights α, β, γ are protocol parameters; v1.0 tunes them from historical assignments within fixed bounds. Units that require mode R are placed on physically distinct machines.

5.2 Redundant execution and resolution

In mode R a unit runs on r contributors in parallel (Figure 2). Seats are drawn by weighted sampling over κ with a seed $H(b \parallel \text{id}(u))$, where b is the hash of the latest finalised ledger block; the seed is stored with the assignment, so anyone can recompute the draw. Results are accepted when q output hashes agree, and the job result is the

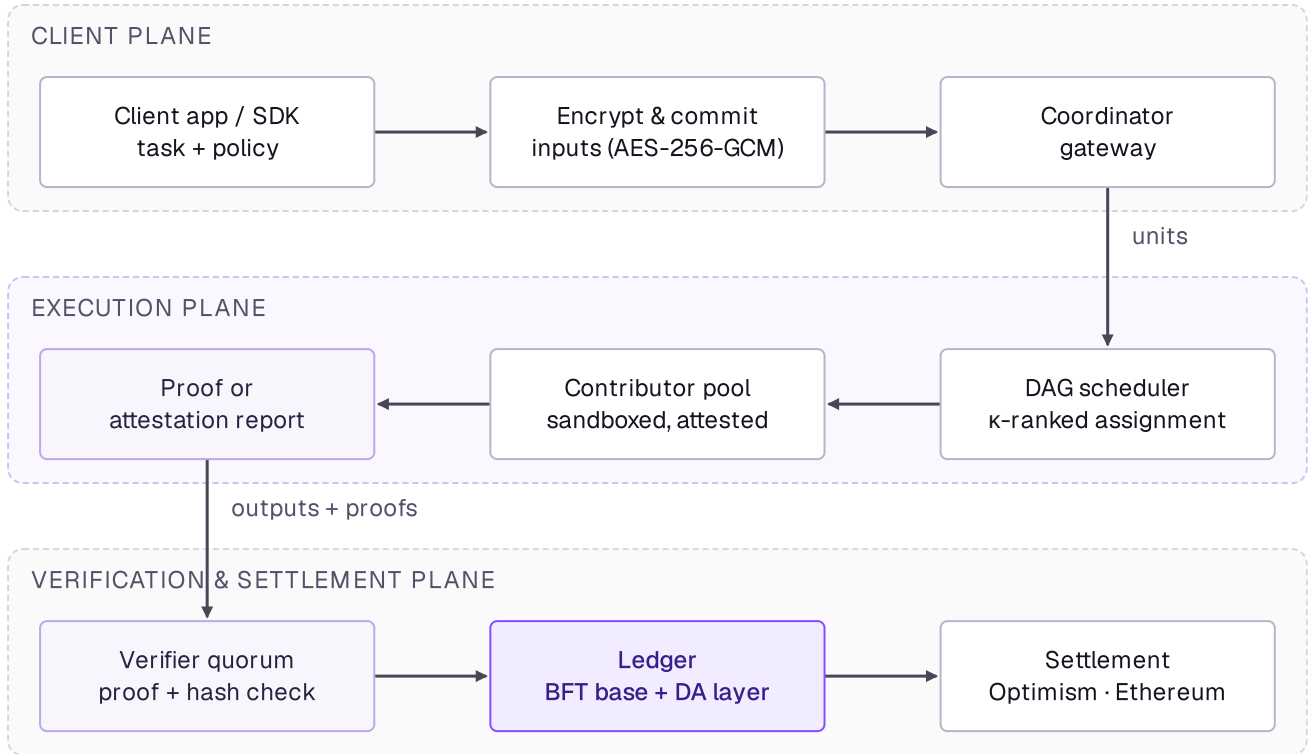


Figure 1. Architecture. Jobs move from the client plane through execution to verification and settlement. Only commitments and evidence reach the ledger.

Merkle root [17] over accepted unit outputs (Algorithm 1). Disagreement leads to slashing of the outvoted seats and reassignment of the unit (Figure 3).

Inference is not bit-reproducible when the order of floating-point additions changes, as it does across GPU models, drivers and kernel choices. Our CPU experiment (§12.6) found no bit-identical outputs under reordered accumulation, while the selected tokens agreed in every case. Mode R therefore compares decoded outputs, such as tokens, labels or rounded values, rather than raw tensors, pins the runtime image, and records the comparison rule in the ledger.

5.3 Assurance modes

Every job declares one of three modes (Table 1). For AI workloads the practical defaults in v1.0 are mode A for confidential inference and fine-tuning, and mode R for non-confidential batch work. Mode P is reserved for small models and

deterministic data jobs where proving cost is acceptable.

6 Privacy and Verification

6.1 Proofs and attestation reports

In mode P, a contributor proves the unit against a constraint system that models the expected computation and submits the proof with the output commitment (Figure 4). Verifiers check it against c_u without seeing any data. In mode A, the contributor instead submits the enclave’s attestation report, which binds the measured runtime image to the output commitment.

6.2 What each mechanism hides

Zero-knowledge proofs hide inputs from verifiers and ledger readers, but not from the machine that executes the computation, which must operate on the values. Confidentiality from executors therefore comes only from mode A. Decryption

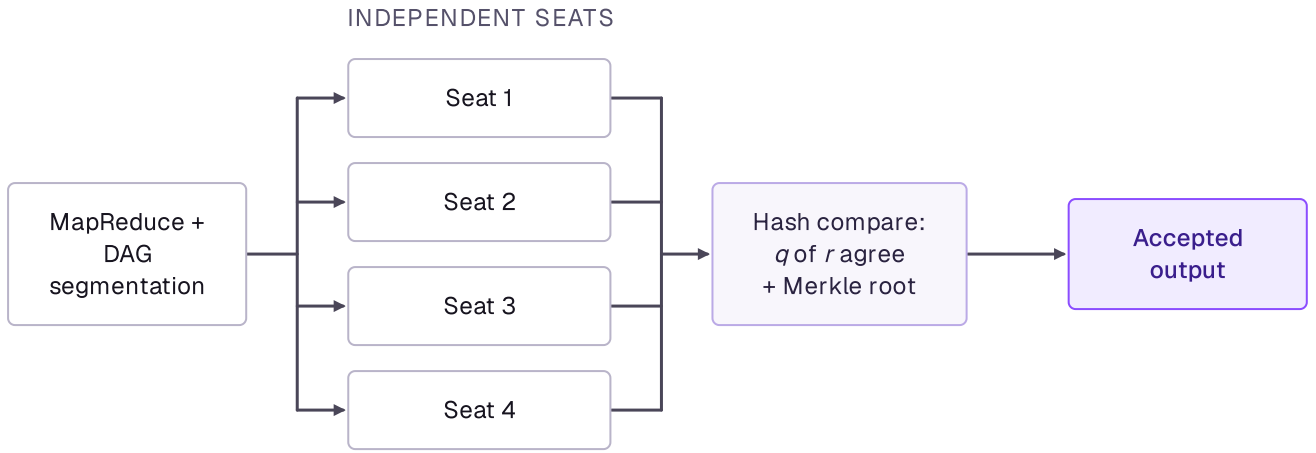


Figure 2. Redundant execution in mode R. Independent seats run the same unit; resolution accepts the output that reaches quorum.

Algorithm 1 Redundant execution with hidden audits (mode R)

```

1: Input: job  $T$ , replication  $r$ , quorum  $q$ , value  $V$ , audit rate  $\pi$ 
2: require  $V \leq \pi q \sigma S / (1 - \pi)$   $\triangleright$  Eq. 7
3:  $\{u_1 \dots u_n\} \leftarrow \text{Decompose}(T)$   $\triangleright$  DAG of units
4: for each ready unit  $u$  in topological order do
5:    $N_u \leftarrow \text{SelectSeats}(u, r, H(b \parallel \text{id}(u)))$   $\triangleright$  public, by  $\kappa$ 
6:   Commit  $H(a_u)$  for a secret audit seed  $a_u$   $\triangleright$  hidden audit
7:   parallel for  $j \in N_u$ : commit  $\text{Dec}(y_j)$   $\triangleright$  decoded output
8:    $h^* \leftarrow$  most frequent  $H(\text{Dec}(y_j))$ 
9:   Reveal  $a_u$ ; if audited( $a_u, \pi$ ) then  $h^* \leftarrow \text{Execute}_A(u)$   $\triangleright$  mode A
10:  if  $|\{j : H(\text{Dec}(y_j)) = h^*\}| \geq q$  then Commit( $u, h^*$ ) else Reassign( $u$ )
11:  Slash every seat that disagrees with  $h^*$   $\triangleright$  verified fault
12: return MerkleRoot( $\{h^*_u\}$ )  $\triangleright$  job commitment

```

keys for a mode-A job are released to an enclave after its attestation has been verified, and contributors without attestable hardware never receive such jobs. All units, in every mode, run in ephemeral sandboxes that cannot read the host's files and are erased on completion.

Remark 3 (Privacy guarantee) *Verifiers and the public never see prompts, weights or data. Executors see plaintext only inside an attested enclave, for the lifetime of one unit.*

6.3 Cryptographic choices

Table 2 lists each primitive and its status against quantum attacks. The rule is that anything relied on after settlement must rest on hash-based or symmetric assumptions, so zk-STARKs are used for proofs anchored on the ledger. Payloads use AES-256-GCM [18], and transport uses TLS 1.3 [19] with hybrid X25519 + ML-KEM-768 key exchange where both endpoints support it. These choices protect stored records and recorded

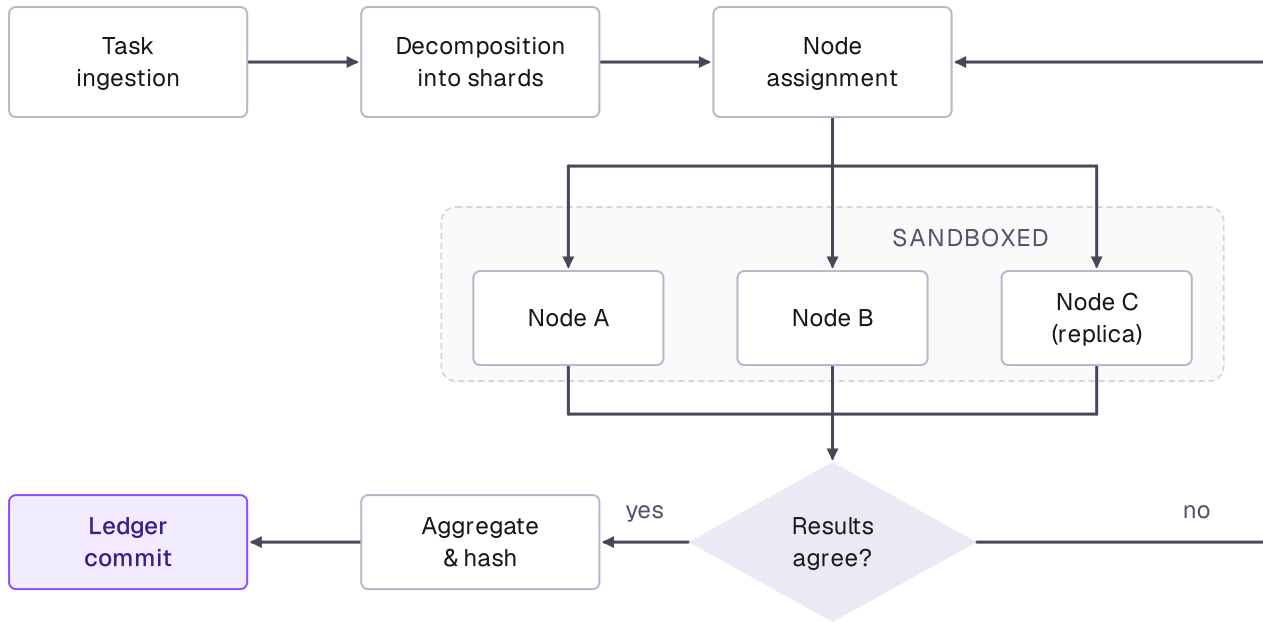


Figure 3. Operational cycle. Units are assigned, executed redundantly in sandboxes and compared. Disagreement triggers reassignment; agreement is aggregated and committed.

Table 1. Assurance modes. The mode is fixed at submission and recorded with the result.

MODE	INTEGRITY RESTS ON	DATA HIDDEN FROM EXECUTOR	INTENDED USE
A • Attested	Enclave attestation of the runtime image	Yes	Confidential inference and fine-tuning
R • Redundant	q of r independent seats agreeing, within the bound of Eq. 7	No	Non-confidential batch inference and data jobs
P • Proven	zk-STARK proof against the input commitment	Only with A	Small models, deterministic data jobs

traffic; they do not change the AI computation itself.

7 Ledger

The ledger records assignments, assurance modes, verification outcomes, penalties and payments. Each entry carries the fields in Table 3, and entries are linked by hash chains and Merkle trees.

The ledger has two layers (Figure 5). The base layer runs Byzantine fault-tolerant consensus

over state transitions and settles a result only after its proof or attestation has been checked. An auxiliary layer stores extended metadata and metrics for parallel reads. In v1.0 the validator set includes nodes operated by the TasQ Project; attestations are replicated across TasQ infrastructure and Microsoft Azure geo-redundant storage.

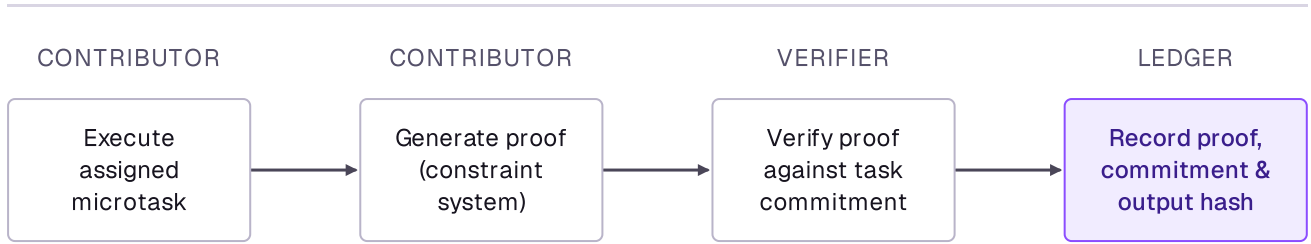


Figure 4. Proof flow in mode P. Proofs are generated by the contributor and checked against the input commitment, so verifiers never see the data.

Table 2. Primitives and their status against known quantum attacks.

PRIMITIVE	USE	QUANTUM STATUS	V1.0
AES-256-GCM	Payload encryption	About 128-bit security under Grover	Used
BLAKE3, Merkle trees	Commitments, ledger links	No efficient attack known	Used
HKDF	Identifier derivation	Hash-based	Used
zk-STARK	Proofs anchored on the ledger	Hash-based, no trusted setup	Used
zk-SNARK (pairing)	Not used for anchored proofs	Broken by Shor’s algorithm	Excluded
X25519 + ML-KEM-768	TLS 1.3 key exchange	Secure if either component holds	Where supported
Ed25519	Node identity keys	Broken by Shor’s algorithm	Migration planned

8 Benchmarking and Identity

Scheduling needs capability claims that cannot simply be self-reported. On joining, each node runs the deterministic suite in Table 4. Each run is seeded with a fresh nonce from the coordinator to prevent replay, results are cross-checked against device telemetry and cross-run variance, and scores are normalised by z-score with percentile caps into v_j .

The node then generates an Ed25519 key pair [20], commits to its public key, capability vector and attestation token a_j with BLAKE3 [21], and derives its identifier with HKDF [22]:

$$C_j = \text{MerkleRoot}(\text{BLAKE3}(pk_j), \text{BLAKE3}(v_j), \text{BLAKE3}(a_j)) \quad (2)$$

$$\text{ID}_j = \text{HKDF}(pk_j \parallel C_j) \quad (3)$$

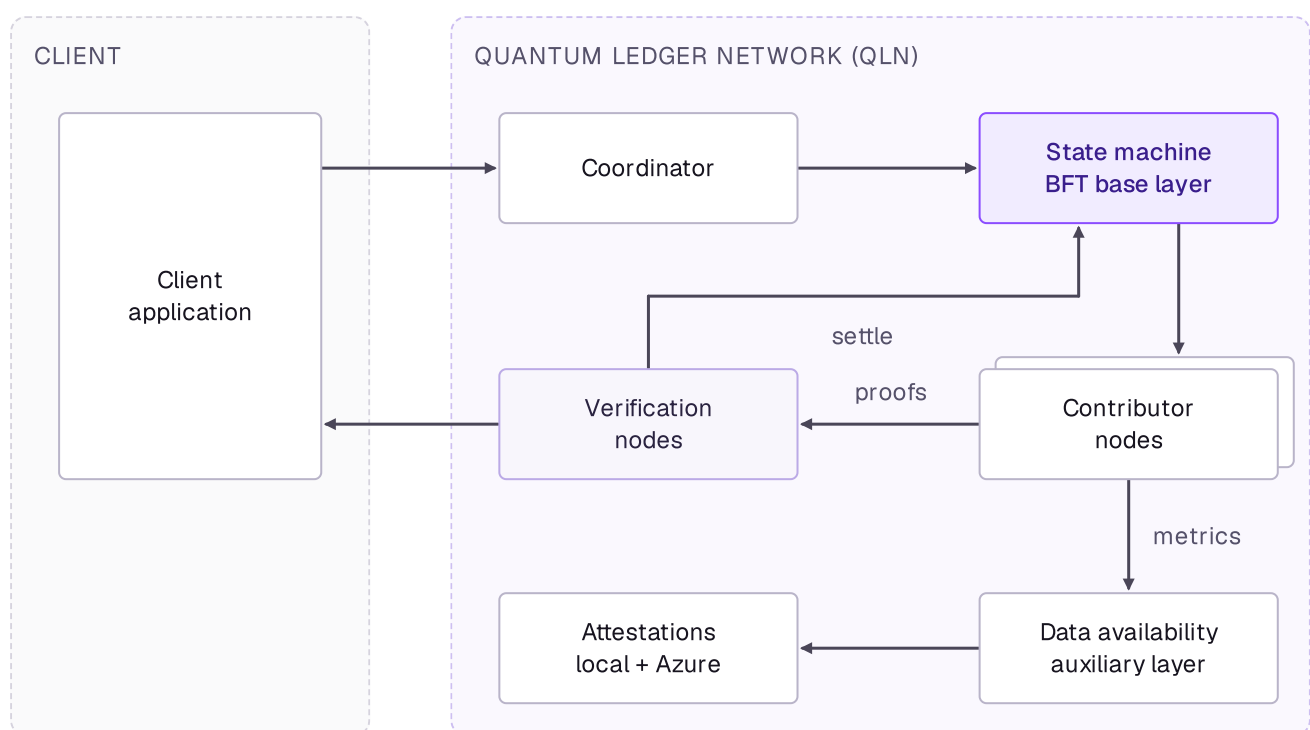
Capabilities are refreshed as an exponentially weighted moving average of new measurements:

$$\hat{v}_j(t) = \lambda m_j(t) + (1 - \lambda) \hat{v}_j(t - 1) \quad (4)$$

Nodes are re-tested on a rolling schedule and whenever task latencies drift from their scores. Nodes below threshold are paused until they pass again.

Table 3. Fields of a ledger entry.

FIELD	CONTENT
Contributor ID	Identifier from Eq. 3
Job hash	Binds the entry to its job and input commitment
Mode	A, R or P, with tolerance if used
Evidence	Proof reference or attestation digest
Timestamps	Order of state changes
Metrics	Inputs to reputation and payment


Figure 5. Ledger. Clients interact through a coordinator; results settle on the BFT base layer only after their evidence is checked.

Remark 4 (Identity migration) *Ed25519 is not post-quantum. Because an identifier is derived from a commitment rather than from the signature scheme, nodes can rotate to hybrid Ed25519 + ML-DSA-65 keys without losing reputation or history.*

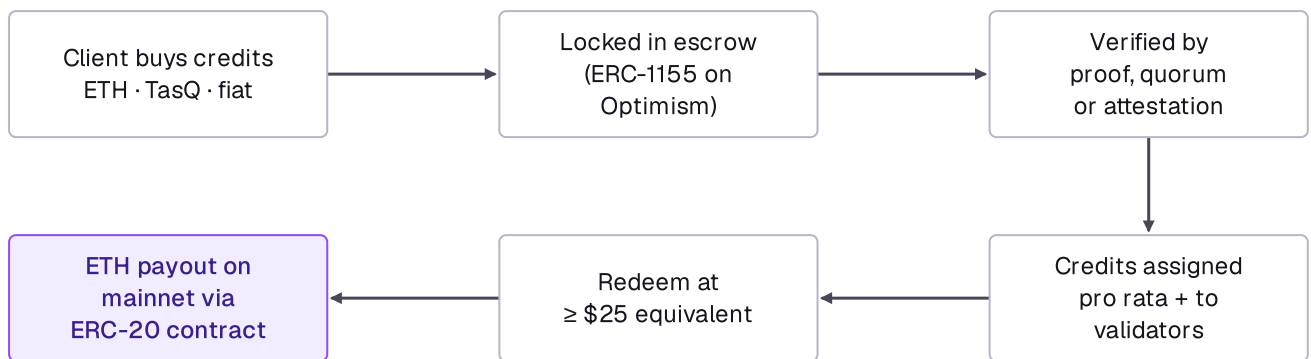
9 Economic Design

9.1 Units of account

The protocol uses two units. *Credits* are non-transferable ERC-1155 balances [23] on the Optimism rollup, denominated in a fixed US-dollar reference so that job value and stake are measured in the same unit. *ETH* is the settlement asset for contributor payouts. Credits are used for

Table 4. Benchmark suite run at onboarding and on revalidation.

RESOURCE	SUITE	MEASURES
CPU	CoreMark-Pro and LINPACK kernels	Integer and floating-point throughput
GPU	GEMM and FFT microbenchmarks via cuBLAS or cBLAS	Matrix and transform throughput
Memory	STREAM	Bandwidth and latency
Storage I/O	fio, sequential and random profiles	Read and write throughput
Network	iperf3 and QUIC round-trip sampling	Throughput and latency

**Figure 6.** Credit flow. Credits are escrowed until verification, then released to contributors and verifiers and redeemable for ETH.

pricing, escrow and stake; they cannot be traded, which keeps prices and security deposits free of speculative swings.

9.2 Payment flow

Clients buy credits with ETH or fiat. A job's credits are locked in escrow at submission and released after verification, in proportion to each contributor's share of the work, with a separate allocation to redundant seats and verifiers (Figure 6). Contributors redeem credits for ETH on Ethereum mainnet through an audited payout contract once their balance reaches USD 25 equivalent.

9.3 Reputation, stake and slashing

Reputation is updated from task accuracy, verification success, uptime and latency, and decays without activity:

$$R_j(t) = \mu R_j(t-1) + (1 - \mu) s_j(t) \quad (5)$$

where $s_j(t) \in [0, 1]$ scores the latest period. Contributors post stake S in credits, mirrored to a staking contract on Ethereum mainnet. A failed proof, a rejected attestation or an outvoted output is a verified fault, and the contract slashes a fraction σ of the stake (Figure 7).

9.4 Admission rule for redundant execution

If seats are drawn independently and the adversary holds a fraction ρ of eligible seats, the probability that it controls at least q of r seats is

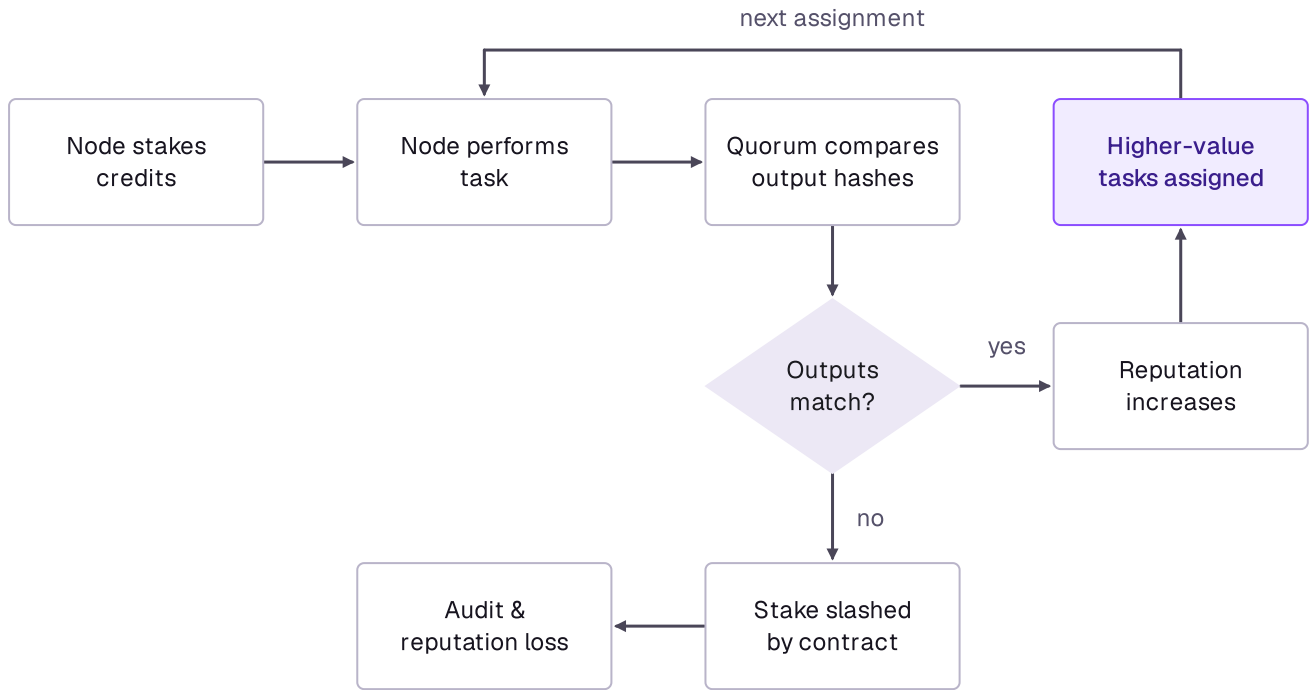


Figure 7. Reputation and stake. Verified work raises reputation and access to larger jobs; verified faults are slashed by contract.

$$p = \sum_{k=q}^r C(r, k) p^k (1-p)^{r-k} \quad (6)$$

An earlier version of this design admitted a unit when $pV < (1-p)\sigma S$, on the assumption that a failed attack costs at least one slashed stake. Simulation showed this to be wrong (§12.3). Because seat selection is public, colluding contributors know when they jointly hold a quorum and deviate only then, so no attack ever fails and slashing never applies.

TasQ therefore adds hidden audits. With probability π , a unit is re-executed in mode A by an attested auditor. The audit decision is derived from a seed that the coordinator commits to at assignment and reveals only after every seat has committed its output, so seats cannot know in advance whether they will be audited. An audited deviation slashes every deviating seat. A rational coalition holding $k \geq q$ seats deviates only if $(1-\pi)V > \pi k \sigma S$, so the protocol admits a unit to mode R only if

$$V \leq \pi q \sigma S / (1-\pi) \quad (7)$$

Equivalently, a unit of value V requires an audit rate $\pi \geq V / (V + q \sigma S)$. Table 5 shows that useful job values need either stakes well above job value or substantial audit rates. Eq. 6 still matters: it sets how often a coalition gets the chance to attack, and therefore how often audits catch one.

Table 5. Largest admissible $V / \sigma S$ under Eq. 7.

AUDIT RATE π	$Q = 2$	$Q = 3$
0.05	0.11	0.16
0.1	0.22	0.33
0.2	0.50	0.75
0.3	0.86	1.29
0.5	2.00	3.00

Remark 5 (What the rule does not establish) *Eq. 7 is a necessary condition for rational security, not a*

Table 6. Positioning against existing options. Entries describe typical offerings, not every provider.

	CENTRALISED CLOUD GPU	OPEN GPU MARKETPLACE	TASQ V1.0
Data hidden from operator	Contractual; confidential VMs on some offerings	Typically not	Mode A, by attestation
Result verifiable by client	No	Typically not	Modes R and P, recorded per job
Capacity	Provider data centres	Independent hosts	Contributors plus backbone
Price	Published list prices	Market-set	Posted; competitiveness unmeasured
Maturity	Production	Production	Design specification

proof of security. It ignores bribery and off-ledger side payments, adversaries who accept losses, correlated failures on shared infrastructure, and the future earnings a node forfeits when removed, which strengthen deterrence. High-value jobs should use mode A or P.

9.5 Pricing, capacity and positioning

In v1.0 prices are posted per resource class and adjusted with utilisation. Mode R costs roughly r times the base price, and modes A and P add enclave and proving surcharges. When demand exceeds contributor supply, operator capacity absorbs the overflow, and remaining jobs queue by priority. TasQ does not assume it will be cheaper than established providers. Its case rests on the properties summarised in Table 6; whether its prices are competitive is an open question for the testnet.

9.6 The TasQ token and project financing

The TasQ token is not part of the protocol. Pricing, escrow, stake and settlement use credits and ETH, and none of the security arguments in this paper depend on the token's price or liquidity. The token exists to finance the project: a tax on token transfers funds the dedicated compute facility (§10). Clients who hold the token

may convert it to credits at the posted rate, exactly as with ETH. In v1.0 the token confers no governance or protocol rights.

Remark 6 (Disclosure) *The TasQ Project operates the v1.0 coordinators, part of the validator set and the backbone, and receives the token-tax revenue that funds the facility. If that revenue falls, expansion of the facility slows; the contributor network and provisioned cloud capacity are unaffected.*

10 Infrastructure and Scaling

To keep capacity available while the contributor network grows, TasQ operates backbone capacity in two forms. The first is a dedicated facility under development, a secure site of high-performance nodes funded by token-tax allocations and confirmed technology grants. The second is capacity provisioned on Microsoft Azure and configured with TasQ's orchestration software. The backbone also runs coordination and part of the validator set, which is the main source of centralisation in v1.0.

Storage is separated from compute (Figure 8). Payloads are held on TasQ infrastructure of 16 TB and above, extended by Azure for overflow and redundancy. Contributors read only

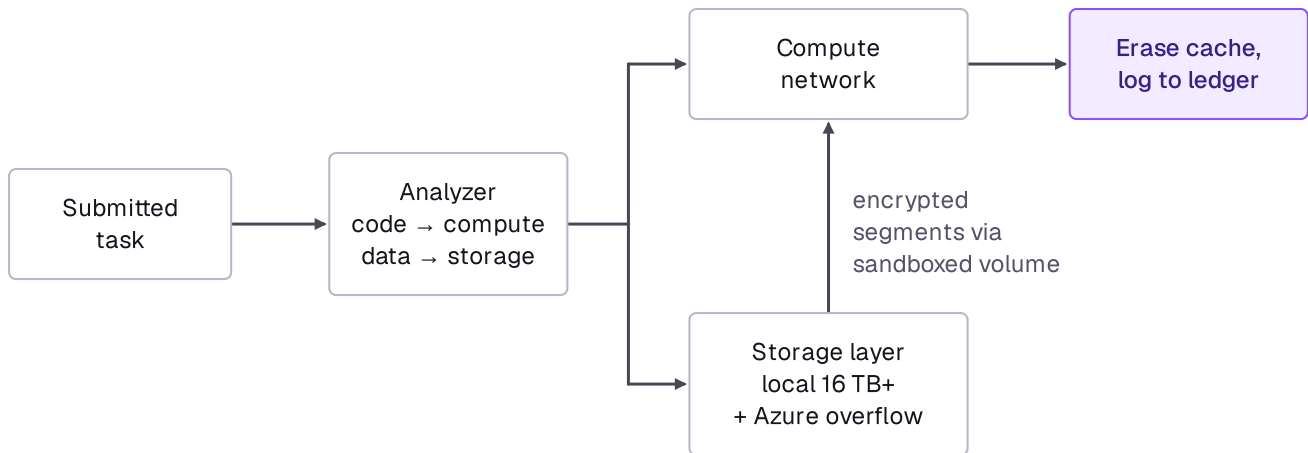


Figure 8. Separation of compute and storage. Contributors handle only encrypted segments through temporary volumes.

encrypted segments through temporary sandboxed volumes, which are erased when the unit completes. Capacity grows by adding contributors; ledger indexing and parallel queries keep validation fast as history grows, and routing by reputation, hardware class and latency spreads load.

11 Security Analysis

Table 7 lists the principal threats from §3, the mechanism that addresses each and the risk that remains. Mode R is only as strong as the admission rule in §9.4 and its audit rate. Mode A moves trust from the operator to the hardware vendor; it does not remove trust.

12 Simulation and Measurement Results

This section reports what can be established before a testnet exists. Microbenchmarks measure the cryptographic primitives on a cloud CPU. Monte Carlo and discrete-event simulations test the protocol rules. GPU and enclave costs are modelled from published measurements, and are labelled as models. All scripts are seeded, and the scripts and raw data are published [24].

12.1 Method

Microbenchmarks run single-threaded through the Python bindings of each library, and report the median of seven runs. Simulations draw seats, faults and workloads from seeded generators, with the parameters in Appendix B. None of the results measures a deployed TasQ network, contributor hardware or real AI workloads.

12.2 Quorum capture

Figure 9 compares Eq. 6 with simulated draws from a finite pool of 1 000 seats. Across all 27 points the largest absolute difference is 0.0012, within Monte Carlo error, and stake-weighted selection gives the same results. Requiring seats on distinct machines changed capture probability very little (from 0.028 to 0.026 with eight adversarial identities per machine), so the rule is retained for fault isolation rather than as a security measure [25].

12.3 Attack economics and hidden audits

The simulation in Figure 10 is the negative result behind §9.4. Without audits, a coalition that deviates only when it holds a quorum makes a profit at every unit value and is never slashed, while a naive adversary that deviates whenever it holds any seat always loses. With hidden audits

Table 7. Threats, mitigations and residual risk.

THREAT	MITIGATION	RESIDUAL RISK
Incorrect result	Quorum agreement plus hidden audits within Eq. 7 (R); proof (P); attestation (A)	Coalitions where V exceeds the bound
Colluding quorum	Audit seed committed before, revealed after outputs; slashing of all deviating seats	Audit rate set too low for job value
Data exfiltration	Key release only after attestation (A); ephemeral sandboxes	Enclave side channels
Sybil identities	Benchmark cost, stake, optional hardware attestation	Minimum stake set too low
Benchmark spoofing	Nonce-seeded tests, telemetry checks, revalidation	Partial spoofing on virtualised hosts
Recorded traffic	AES-256 payloads; hybrid ML-KEM transport	Endpoints without hybrid support
Ledger equivocation	BFT consensus with $n \geq 3f + 1$	Validator set partly operated by TasQ
Coordinator failure or abuse	Assignments and results logged on the ledger	Single operator in v1.0

the coalition's profit crosses zero where Eq. 7 predicts; Table 8 shows simulated break-even values within five per cent of the rule [26].

Table 8. Break-even unit value for a coordinating coalition ($p = 0.1$, $r = 3$, $q = 2$): Eq. 7 against simulation. Source: [26].

Π	EQ. 7	SIMULATED
0.05	0.105	0.110
0.1	0.222	0.226
0.2	0.500	0.504
0.3	0.857	0.862

12.4 Scheduling, audits and removal

Table 9 reports a discrete-event simulation of mode R under increasing load. Seat selection weighted by κ lowered mean latency by about 12% at load 0.5, but slightly raised it near saturation, where the fastest nodes are already busy. Each audit catch removes the coalition seats

involved, so the wrong-acceptance rate fell from 1–2% in the first quarter of each run to about 0.1% in the last, and 67–87% of adversarial nodes were removed. No honest node was removed in any run [27].

Table 10 shows how the reputation rule separates faulty nodes from honest ones. With $\mu = 0.9$ and a threshold of 0.8, nodes faulting on up to 5% of units are never excluded, while nodes at 20% are excluded after a median of 20 periods. The threshold therefore tolerates occasional faults, which is intended, but slow-faulting nodes near 10% can persist [28].

12.5 Cryptographic cost

Table 11 lists the measured costs. Hashing, encryption and identity derivation are negligible relative to AI workloads: committing to 65 536 unit outputs takes 60 ms, and a hybrid post-quantum key exchange about 0.2 ms. The exception is ML-DSA-65 signing at about 10 ms in the library tested. Optimised implementations

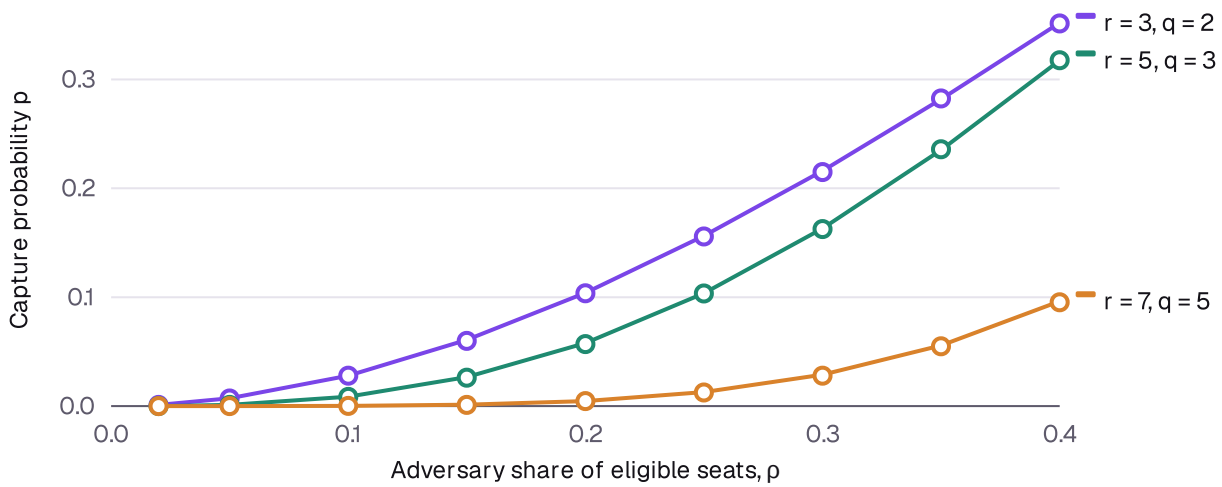


Figure 9. Quorum capture probability. Lines show Eq. 6; circles show simulated draws from a pool of 1 000 seats (400 000 draws per point).

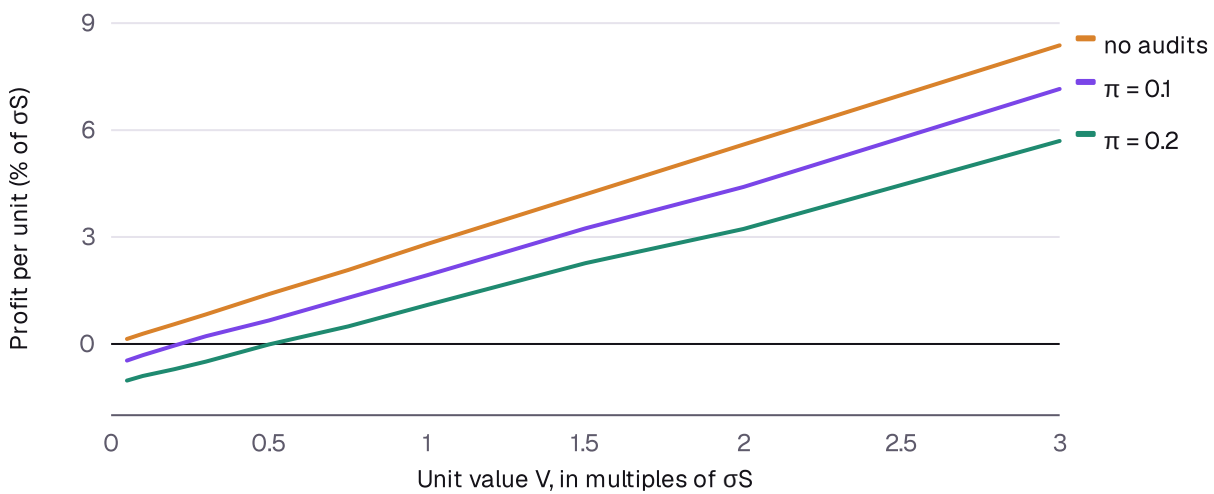


Figure 10. Profit per unit of a coalition that deviates only when it holds a quorum ($\rho = 0.1$, $r = 3$, $q = 2$), with and without hidden audits. A naive adversary, not shown, loses 16–24% of σS per unit across the whole range, even without audits.

exist, and since node identities sign rarely, this does not affect throughput [29].

12.6 Modelled GPU and enclave costs

Published measurements of confidential computing on NVIDIA Hopper GPUs report overheads below about 7% for most LLM inference queries, dominated by encrypted transfers between CPU and GPU [30]. We model mode A as that steady-state overhead plus a

per-session cost of key release and attestation, in which the measured cryptography is under 1 ms and assumed network round trips dominate. Figure 11 shows that sessions shorter than a few seconds pay most of their overhead in setup, so mode A keys should be released per session, not per job [31].

Comparing the modes on the same model gives the result in Table 12: with three seats, mode R is cheaper than mode A only when attested GPU

Table 9. Discrete-event simulation of mode R with audits ($\pi = 0.1$), 300 contributors, 10% coordinating adversaries, 5% faulty nodes, 20 000 units per run. Latency in units of mean work time. Source: [27].

LOAD	POLICY	MEAN LATENCY	P95 LATENCY	WRONG, FIRST QUARTER	WRONG, LAST QUARTER	ADVERSARIES REMOVED
0.5	uniform	0.91	2.72	1.30%	0.12%	77%
0.5	κ -weighted	0.80	2.48	1.32%	0.10%	67%
0.7	uniform	0.87	2.65	1.16%	0.08%	83%
0.7	κ -weighted	0.82	2.53	1.96%	0.02%	87%
0.85	uniform	1.28	3.11	2.04%	0.06%	77%
0.85	κ -weighted	1.42	3.35	2.08%	0.10%	73%
0.95	uniform	6.11	12.33	1.86%	0.12%	67%
0.95	κ -weighted	6.34	12.77	2.02%	0.16%	77%

Table 10. Exclusion under Eq. 5 with $\mu = 0.9$ and threshold 0.8, 20 000 nodes over 300 periods. Source: [28].

FAULT RATE	EXCLUDED	MEDIAN PERIODS
0%	0.0%	–
2%	0.0%	–
5%	0.0%	–
10%	0.1%	187
20%	100.0%	20
40%	100.0%	7

time costs about three times more than commodity time. Mode R is therefore a fallback for scarce attested hardware, not the default for valuable work.

Table 12. Modelled price premium of attested over commodity GPU time at which modes A and R cost the same (5% enclave overhead). Source: [31].

SEATS R	$\pi = 0$	$\pi = 0.1$	$\pi = 0.2$	$\pi = 0.3$
2	1.9×	2.1×	2.4×	2.7×
3	2.9×	3.2×	3.6×	4.1×
5	4.8×	5.3×	6.0×	6.8×

As a CPU analogue of GPU non-determinism, we reordered the float32 accumulation in a two-layer network (Table 13). No output vector was bit-identical, but the top-1 token and the top-5 token set agreed for every input. This is the basis for comparing decoded outputs in mode R (§5.2); confirming it on GPUs requires the testnet.

12.7 What remains unmeasured

Table 14 summarises the evidence after this revision. The measurements in Table 15 still require the testnet, and we will publish them with methodology and raw data alongside the source code at github.com/tasqProject. The design

Table 11. Cryptographic primitives, median of seven runs, one thread on a 2-vCPU Intel Xeon (2.1 GHz) cloud instance. Source: [29].

PRIMITIVE	ROLE IN TASQ	MEDIAN TIME	THROUGHPUT
BLAKE3, 1 MiB	Commitments	182.1 μ s	5.4 GiB/s
SHA-256, 1 MiB	Comparison only	772.8 μ s	1.3 GiB/s
AES-256-GCM encrypt, 1 MiB	Payload encryption	118.0 μ s	8.3 GiB/s
Ed25519 sign / verify	Node signatures today	38.1 μ s / 122.2 μ s	–
ML-DSA-65 sign / verify	Post-quantum identity	10.20 ms / 217.0 μ s	–
X25519 key exchange	Transport	38.8 μ s	–
ML-KEM-768 encaps / decaps	Transport (post-quantum)	79.7 μ s / 98.8 μ s	–
Hybrid X25519 + ML-KEM-768	Key release to enclaves	215.4 μ s	–
Merkle root, 65 536 leaves	Job result commitment	60.18 ms	–
Ledger ID derivation (Eq. 2–3)	Node registration	9.18 μ s	–

should be considered to have failed its purpose if any of the following holds:

- enclave overhead on TasQ workloads makes mode A uncompetitive with confidential offerings from established clouds;
- decoded outputs disagree across common GPU classes often enough to make mode R comparisons unreliable;
- the stakes and audit rates required by Eq. 7 for typical job values exceed what contributors will post or pay for;
- contributor supply cannot serve demand without relying mainly on operator capacity.
- **Hardware trust.** Mode A depends on the enclave vendor and inherits known side-channel risks [5]. Attestable GPUs are a small share of contributor hardware.
- **Cost of redundancy.** Mode R multiplies cost by r and adds audit cost; the model in §12.6 makes it the cheaper mode only when attested hardware is scarce.
- **Non-determinism.** Comparing decoded outputs is supported only by a CPU experiment so far; GPU agreement rates are unmeasured.
- **Simulation scope.** The results in §12 come from simulations and models with stated parameters; real adversaries, workloads and hardware may behave differently.
- **Centralisation.** Coordinators, part of the validator set and the backbone are operated by the TasQ Project in v1.0 (§3.4).

13 Limitations

- **Proving cost.** zkML is impractical for large models, so mode P covers only small models and deterministic data jobs.

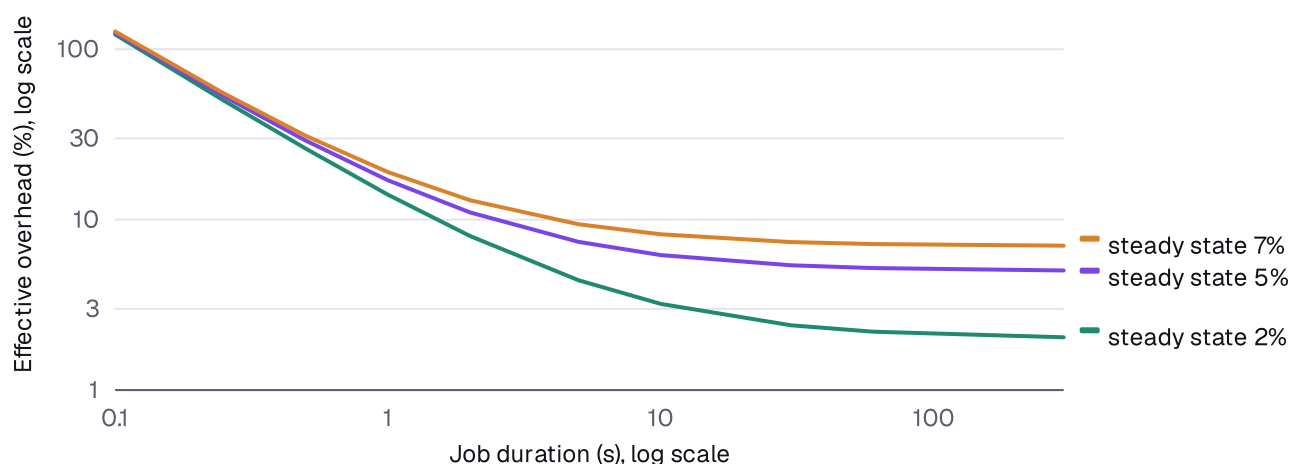


Figure 11. Modelled effective overhead of mode A as a function of session length, for three steady-state overheads. Model, not measurement.

Table 13. Reordered float32 accumulation in a two-layer network (CPU, 400 inputs, 8 000 outputs). Source: [31].

ACCUMULATION ORDER	BIT-IDENTICAL OUTPUTS	TOP-1 TOKEN AGREEMENT	TOP-5 SET AGREEMENT	MAX RELATIVE DIFFERENCE
Reversed blocks	0%	100%	100%	6.1e-07
Interleaved	0%	100%	100%	6.3e-07

- **Financing dependence.** The dedicated facility is funded by token-tax revenue, which depends on token activity.
- **Open parameters.** α , β , γ , λ , μ , σ , minimum stake and prices will be set from testnet data.
- **Post-quantum coverage.** Node identities still use Ed25519 until the migration in §8 is complete.

14 Roadmap

Table 16 lists the planned work, grouped by track. We will describe the network as decentralised only once independent operators run coordination and the TasQ Project holds less than one third of validator voting power. Timelines and funding allocations are maintained in the TasQ GitBook.

15 Conclusion

TasQ does not introduce a new cryptographic primitive or a new form of computation. It applies established mechanisms, namely attested enclaves, redundant execution and zero-knowledge proofs, to a marketplace for AI compute, and makes their guarantees explicit: every result carries the mode that produced it, and redundant execution is admitted only within an economic bound.

The open questions are empirical: proving and enclave overheads, agreement rates across GPUs, the stake contributors will actually post, and prices against established providers. Those measurements, not this paper, will determine whether the design is useful, and we have stated in advance which results would show that it is not.

Table 14. Claims and the evidence behind them after this revision.

CLAIM	EVIDENCE IN THIS PAPER	STILL REQUIRES
Eq. 6 predicts quorum capture	Simulation agrees to within 0.002 (§12.2)	Measured stake concentration
Hidden audits deter coalitions	Simulated break-even within 5% of Eq. 7 (§12.3)	Implementation of commit-reveal audits
Audits remove adversaries over time	Wrong acceptance falls about tenfold within one run (§12.4)	Testnet adversarial trials
Cryptographic costs are small	Microbenchmarks (§12.5)	Measurement on contributor hardware
Mode A overhead is acceptable	Model from published GPU measurements (§12.6)	Measurement on TasQ workloads
Mode R needs decoded comparison	CPU reordering experiment (§12.6)	GPU agreement measurement
Prices are competitive	No evidence	Testnet price data
The network is decentralised	Not claimed for v1.0	Criteria in §14

References

- [1] V. Costan and S. Devadas. Intel SGX explained. IACR ePrint 2016/086.
- [2] J. Teutsch and C. Reitwießner. A scalable verification solution for blockchains. Truebit whitepaper, 2017.
- [3] J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*.
- [4] E. Ben-Sasson, I. Bentov, Y. Horesh and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. IACR ePrint 2018/046.
- [5] J. Van Bulck et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *USENIX Security*, 2018.
- [6] D. Kang, T. Hashimoto, I. Stoica and Y. Sun. Scaling up trustless DNN inference with zero-knowledge proofs. arXiv:2210.08674, 2022.
- [7] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5), 1997.
- [8] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th ACM STOC*, 1996.
- [9] G. Brassard, P. Høyer and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN '98*, 1998.
- [10] NIST. Module-Lattice-Based Key-Encapsulation Mechanism Standard. FIPS 203, 2024.
- [11] NIST. Module-Lattice-Based Digital Signature Standard. FIPS 204, 2024.
- [12] D. J. Bernstein and T. Lange. Post-quantum cryptography. *Nature*, 549:188–194, 2017.
- [13] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *Proc. OSDI*, 1999.
- [14] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. 41st ACM STOC*, 2009.

Table 15. Measurements to be published with the testnet.

MEASUREMENT	METHOD	QUESTION ANSWERED
Proving time per unit	Mode P on reference models of increasing size and fixed hardware	Where mode P stops being viable
Replication overhead	End-to-end cost and latency of mode R against single execution	Real cost of r
Enclave overhead	Throughput in mode A against the same GPU without attestation	Price of confidentiality
Output agreement	Agreement rate across GPU models and drivers under pinned runtimes	Viability of exact comparison
Scheduling	Utilisation, queueing delay and reassignment rate under load	Behaviour at capacity
Benchmark integrity	Red-team spoofing attempts on virtualised hosts	Reliability of capability scores
Price	Cost per inference against list prices for equal hardware classes	Competitiveness

- [15] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. OSDI*, 2004.
- [16] Y.-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4), 1999.
- [17] R. C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO '87*, 1987.
- [18] NIST. Advanced Encryption Standard (AES). FIPS 197, 2001.
- [19] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.
- [20] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe and B.-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2), 2012.
- [21] J. O'Connor, J.-P. Aumasson, S. Neves and Z. Wilcox-O'Hearn. BLAKE3: One function, fast everywhere. Specification, 2020.
- [22] H. Krawczyk and P. Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, 2010.
- [23] W. Radomski et al. ERC-1155: Multi Token Standard. Ethereum Improvement Proposal 1155, 2018.
- [24] TasQ Project. TasQ v1.0 simulation and microbenchmark data. tasqnetwork.io/benchmark, 2026.
- [25] TasQ Project. Quorum capture and distinct-machine simulations. tasqnetwork.io/benchmark/quorum, 2026.
- [26] TasQ Project. Attack economics and audit break-even simulations. tasqnetwork.io/benchmark/quorum, 2026.
- [27] TasQ Project. Mode R scheduler simulation. tasqnetwork.io/benchmark/scheduler, 2026.
- [28] TasQ Project. Reputation threshold simulation. tasqnetwork.io/benchmark/reputation, 2026.
- [29] TasQ Project. Cryptographic microbenchmarks. tasqnetwork.io/benchmark/crypto, 2026.
- [30] J. Zhu et al. Confidential computing on NVIDIA Hopper GPUs: A performance benchmark study. arXiv:2409.03992, 2024.
- [31] TasQ Project. GPU and enclave cost models and reordering experiment. tasqnetwork.io/benchmark/gpu-model, 2026.

Table 16. Planned work.

TRACK	ITEM
Evaluation	Public testnet with the measurements listed in §12
Decentralisation	Independent coordinators; TasQ Project below one third of validator voting power
Verification	Extend mode P to larger models as proving costs fall
Confidentiality	Support additional GPU and CPU attestation platforms
Post-quantum	Hybrid Ed25519 + ML-DSA-65 identities; hybrid key exchange on every link
Open source	Runtime, scheduler and contracts published at github.com/tasqProject
Developers	Contribution SDK across platforms; client SDKs and enterprise APIs
Integrations	Ethereum payout bridge, Kubernetes orchestration, training frameworks
Privacy	Secure multi-party computation for selected workloads

A Notation

Table 17. Notation.

SYMBOL	MEANING
T, u	Job and execution unit
c_u	Input commitment of unit u
v_j, R_j, S_j	Capability vector, reputation and stake of contributor j
r, q	Seats per unit and agreement quorum in mode R
ρ	Adversary share of eligible seats
p	Probability the adversary holds at least q of r seats (Eq. 6)
π	Probability that a unit is audited in mode A
σ	Fraction of stake slashed for a verified fault
V	Value declared for a unit by the client
κ	Compatibility index used for seat selection (Eq. 1)
λ, μ	Smoothing factors for capabilities (Eq. 4) and reputation (Eq. 5)

B Simulation Parameters and Reproducibility

Table 18 lists the parameters of every simulation, and Table 19 the scripts that produce each result. Running the scripts with the published seeds reproduces every number in §12 exactly; microbenchmark timings vary with hardware.

Table 18. Simulation parameters. All runs are seeded and reproducible.

SIMULATION	PARAMETERS
Quorum capture (S1)	Pool of 1 000 seats; 400 000 uniform draws and 8 000 stake-weighted draws per point; lognormal stakes
Distinct machines (S2)	1 000 identities; adversary packs 1–8 identities per machine; 30 000 draws per point
Attack economics (S3)	$\rho = 0.1$, $r = 3$, $q = 2$; 400 000 units per point; coalition deviates only when holding q seats
Reputation (S4)	20 000 nodes, 300 periods, 20 units per period, $\mu = 0.9$, threshold 0.8
Scheduler	300 nodes in three GPU classes (speeds 1, 2, 4); $\rho = 0.1$; 5% faulty nodes at 15%; $r = 3$, $q = 2$, $\pi = 0.1$; 20000 units
Mode A model	Steady-state overhead 2–7% from published confidential-GPU measurements; two 50 ms round trips and 20 ms attestation check per session; key-exchange cost measured

Table 19. Scripts, outputs and where the data are published.

SCRIPT	PRODUCES	PUBLISHED AT
micro_crypto.py	Cryptographic costs (§12.5)	tasqnetwork.io/benchmark/crypto
sim_protocol.py	Quorum capture and attack economics (§12.2–12.3)	tasqnetwork.io/benchmark/quorum
sim_protocol.py	Reputation thresholds (§12.4)	tasqnetwork.io/benchmark/reputation
sim_scheduler.py	Scheduling and removal (§12.4)	tasqnetwork.io/benchmark/scheduler
model_gpu.py	Modelled GPU and enclave costs (§12.6)	tasqnetwork.io/benchmark/gpu-model



TasQ

Your AI, your data. Private, verifiable compute.

WEBSITE	tasqnetwork.io
CODE	github.com/tasqProject
ROADMAP	TasQ GitBook

This paper describes the technical design of TasQ v1.0. It reports simulations, models and microbenchmarks, not measurements of a deployed network, and is not an offer of securities or tokens. Parameters described as open may change before release.