Intercloud: Eventual Consistency for Decentralised Economies via Chilling-Effect Consensus

Anonymous author
Anonymous affiliation

Abstract

We present *Intercloud*, a decentralised economic network that achieves effectively unlimited transaction throughput by reducing global consensus to a single random seed per epoch. The FLP impossibility result [7] rules out deterministic consensus in fully asynchronous systems; Intercloud escapes this constraint by construction, providing the practical synchrony of Dwork, Lynch and Stockmeyer [6] without assuming it globally. Watcher swarms observe only cryptographic hashes — never plaintext — and reach finality via *chilling-effect consensus*: attesting to the *absence of conflicting evidence* rather than voting between alternatives. Any conflicting attestation automatically yields a self-certifying Proof of Corruption, making misbehaviour irrefutably detectable. We prove that ripples terminate in bounded time; that a swarm of approximately 35 Watchers assigned by a verifiable random function suffices for double-spending prevention matching Hoepman's lower bound [10]; that two correct clients can hold conflicting finality attestations only if the adversary both compromises a supermajority of the assigned swarm and eclipses both clients from all honest nodes [1]; and that Buridan's Principle [12] does not apply because the swarm is finite and the consensus question is absence of evidence. Security scales proportionally with value — not uniformly across all transactions. Stream states are coloured Green, Yellow, or Red; end users choose their own finality threshold. The single global random oracle costs $O(N)$ messages *per epoch*, amortising to zero per transaction at scale.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Cryptographic protocols; Networks → Protocol design and analysis

Keywords and phrases Distributed consensus, eventual consistency, decentralised ledgers, double-spending prevention, verifiable random functions, proof of corruption, privacy, Byzantine fault tolerance

Digital Object Identifier 10.4230/LIPIcs...

Related Version Anonymous related version(s)

Acknowledgements Anonymous acknowledgements

1 Introduction

1.1 The Blockchain Cost Problem

Consider transporting $1 by armoured truck. A rational security model assigns one truck to one dollar. A blockchain does the opposite: every validator in the network must verify every transaction, regardless of its value. The global security budget — billions of dollars of electricity in proof-of-work systems, billions of staked tokens in proof-of-stake systems — is deployed uniformly for every transaction. A one-dollar transfer and a billion-dollar transfer consume the same validation resources.

Beyond cost, the global ledger model has a privacy problem. Every transaction, every balance, every smart-contract invocation is visible to all participants. Transaction graph analysis can de-anonymise users even when addresses are pseudonymous.

Intercloud addresses both problems simultaneously. Its central architectural choice is that Watcher nodes — the nodes that provide security — see only *hashes* of stream states, never plaintext. Security is allocated *proportionally to the square root of economic value*: a stream holding one dollar uses approximately 35 Watchers; a stream holding one million dollars uses $\sqrt{10^6} \approx 1000$ times more Watchers — roughly 35,000. This is the armoured-convoy model made precise: the cost of security scales sub-linearly with the value transported.

1.2 The Two Core Mechanisms

1.2.1 Ripple deduplication

The Magarshak Machine [1] prevents cyclic reactive execution by tagging each ripple with a unique *ripple identifier* (rID). Nodes store rIDs in a hash table indexed by (rID, ep), retaining entries for two epoch durations before eviction. Because execution ripples carry fees at each hop, the table remains compact. This mechanism is proven correct for single-node Magarshak Machines; we extend it to distributed Intercloud in Section 4.

1.2.2 Chilling-effect consensus

A Watcher swarm for stream S_i reaches *finality* when a supermajority of its members has each independently verified via gossip that a supermajority of the swarm has attested to the same hash h and seen no conflicting hash during this epoch. Critically, this is not a vote between two candidate values. It is an attestation to the *absence of conflict*. If any two attestations conflict, both signers automatically produce a Proof of Corruption (Π) — a self-certifying evidence package requiring no trusted third party to verify. The chilling effect is the deterrence: a Watcher that produces conflicting attestations is immediately and irrevocably convicted.

1.3 Comparison with Existing Approaches

1.3.1 FLP and partial synchrony

Fischer, Lynch and Paterson [7] proved that deterministic consensus is impossible in a purely asynchronous system with even one faulty process. Dwork, Lynch and Stockmeyer [6] showed that eventual synchrony suffices to escape this bound. Intercloud reduces the globally synchronous surface to a single random seed per epoch, constructing practical synchrony economically rather than assuming it.

1.3.2 Global-consensus systems

Nakamoto [13] and proof-of-stake variants require every validator to process every transaction. The validation cost is $O(N)$ per transaction where N is the network size.

1.3.3 BFT protocols

PBFT [4] achieves safety with $f < n/3$ Byzantine nodes but requires $O(n^2)$ messages per consensus round. Tendermint [3] and HotStuff [15] reduce message complexity to $O(n)$ per round via pipelining and rotating leaders, but still require global participation per decision. Algorand [8] uses VRF-based committee selection to achieve scalable BFT with ≈ 1000 nodes per block, but commits to a global block ordering. Intercloud requires no global block at all:

finality is per-stream, committee size scales with stream value, and the only global agreement is one random seed per epoch.

1.3.4 Hoepman's result

Hoepman [10] proved that using coin identifiers to assign clerk sets, double-spending can be prevented with clerk sets whose size is independent of the total number of nodes N, depending only on security parameters. For standard parameters this yields sets of approximately 35 nodes. Intercloud's Watcher swarms are precisely these clerk sets, with stream identifiers replacing coin identifiers.

1.3.5 Buridan's Principle

Lamport [12] proved that a binary discrete decision on a continuous-valued input cannot be made in bounded time. We prove in Section 8 that chilling-effect consensus is not subject to this principle because the swarm is finite and the decision is not binary.

1.4 Paper Organisation

Section 2 reviews relevant prior work. Section 3 defines the Intercloud model. Sections 4–10 state and prove the main results. Section 13 formalises the junior-node corruption detection and PoC lottery reward mechanism. Section 14 develops the vesting and rational security argument. Sections 15–17 develop the local coin economy: emission, retirement, exchange rates, the dot-product security theorem, and the coin–content separation. Section 12 presents the zero-knowledge extension. Section 19 discusses limitations and future work. Section 20 concludes.

2 Background

2.0.1 Magarshak Machine

A Magarshak Machine [1] $\mathcal{MM} = (S, P, A, C, E, R)$ — the SPACER model — is a formal model for governed, append-only distributed state. Streams S_i are append-only message logs under a publisher's authority; R is a bidirectional relation index — two tables (*relatedTo*, *relatedFrom*) updated atomically when streams post `relateTo`/`unrelateTo` messages; actions A are policy-checked transformations declaring their read and write sets; capabilities C are the sole mechanism for side effects; policy P governs all transitions; and execution engine E is the push-only reactive scheduler driven by four reduction rules (CREATE, TRIGGER, EXECUTE, RETRY). The MM paper derives 22 structural theorems from these rules, including append-only safety, embarrassing parallelism scaling linearly with publishers, causal consistency, ripple termination via local deduplication, minimal cache invalidation, deterministic replay, and consensus freedom. The Magarshak Machine proves ripple termination for single nodes via the local ripple-log deduplication mechanism. We extend this to distributed networks in Theorem 13.

2.0.2 Lamport's causal ordering

Lamport [11] established that in a distributed system without synchronised clocks, the *happened-before* relation $\rightarrow$ defines a consistent partial order on events. Within each stream

in Intercloud, message ordering follows Lamport timestamps; competing transfer requests are resolved by this causal ordering at the Executor.

2.0.3 Lamport's Buridan's Principle

Written in 1984 and published in 2012 [12], this principle formalises the arbiter problem: any physical mechanism that must make a discrete choice based on a continuous input can be driven into an arbitrarily long period of indecision. Lamport proved that for any strategy an arbiter adopts, there exists a starting condition under which it fails to decide within any given bounded time. The principle applies to electronic arbiter circuits, traffic crossing decisions, and any consensus protocol that must choose between two nearly-equal alternatives. We revisit this in Section 8.

2.0.4 Hoepman's distributed double-spending bounds

Hoepman [10] studied the problem of preventing double-spending in a distributed payment system without a central bank. The main result relevant here: if the coin (or stream) identifier is used to assign clerks, a clerk set of size

$$n_{\min} = \frac{\beta}{r} \log_e(s + 1 + \log(r + 2)) \tag{1}$$

suffices to detect any double-spending with probability $\geq 1 - e^{-s}$, where r is the maximum tolerated number of double-spendings, s is the security parameter, and β depends on n and f (the fraction of dishonest nodes) but is *independent of the total network size* N. For $r = 1$, $s = 30$, $f/n = 1/3$, this gives $n_{\min} \approx 35$. This is the exponentially diminishing returns result: adding nodes beyond $\approx$35 provides negligible additional security for any fixed f/n ratio.

2.0.5 FLP impossibility

Fischer, Lynch and Paterson [7] proved that no deterministic algorithm can solve consensus in an asynchronous system if even one process may fail. This rules out global consensus as a foundation for Intercloud. The epoch mechanism sidesteps FLP by requiring global agreement only on a single random scalar per epoch — a much weaker problem than transaction ordering.

2.0.6 Partial synchrony and DLS

Dwork, Lynch and Stockmeyer [6] showed that consensus is achievable if the system is *eventually synchronous*: messages are delivered within some unknown bound Δ after a global stabilisation time. Intercloud's epoch construction instantiates this: within each epoch, the network is assumed to deliver messages within T_{ep}, and the VRF seed commits to the swarm assignment before the epoch begins.

2.0.7 HotStuff

Abraham et al. [15] achieve linear message complexity BFT consensus via a three-phase pipeline with a rotating leader. HotStuff requires global participation per consensus round and $O(n)$ messages per decision. Intercloud's Watcher swarms are local to each stream and require no global participation; the per-transaction message cost is $O(1)$ amortised.

2.0.8 Algorand

Gilad et al. [8] use a VRF-based committee selection mechanism to scale BFT consensus, drawing a random committee of ≈ 1000 nodes per block. Intercloud's VRF assignment is structurally similar but applied at the stream level rather than the block level, with swarm sizes scaling as $\sqrt{value}$ rather than being fixed globally. Algorand achieves global block finality; Intercloud achieves per-stream finality without any global block.

Shapiro et al. [14] formalise data structures that can be updated concurrently without coordination and merged deterministically. Intercloud streams are more constrained: they have a single authoritative publisher and append-only semantics. This enables stronger consistency guarantees (a single head per stream) at the cost of requiring Executor coordination for writes.

3 The Intercloud Model

3.1 Streams and the Hash-Data Separation

▶ **Definition 1** (Stream). *A* stream S_i *is a tuple* $(\mathcal{M}_i, k_i, INTER_i, Rules_i)$ *where:*

- $\mathcal{M}_i = (m_0, m_1, \ldots)$ *is an append-only sequence, each message satisfying* $m_j.prev = H(m_{j-1})$.
- k_i *is the stream owner's public key.*
- $INTER_i \in \mathbb{R}_{\geq 0}$ *is the Intercoin weight staked.*
- $Rules_i$ *is the stream's deterministic Rules program.*

The state hash *is* $h_i = H(m_k)$ *where* m_k *is the latest appended message.*

▶ **Definition 2** (Integrity layer and data layer). *For each stream* S_i*, the* integrity layer *stores only* $(h_i, INTER_i, H(Rules_i))$*. This is all that Watcher nodes hold. The* data layer *stores plaintext* $\mathcal{M}_i$ *and* $Rules_i$*, held only by the stream owner and authorised Executors. A Watcher never receives data-layer content.*

▶ Remark 3 (Privacy guarantee). Because Watchers see only hashes, a fully compromised Watcher reveals no amounts, balances, counterparties, or rule logic. There is no global transaction ledger to analyse. This is strictly stronger privacy than Monero (which reveals a transaction graph structure) and Zcash (which maintains a shielded pool with publicly visible entry and exit events).

3.2 Epochs and Verifiable Random Swarm Assignment

▶ **Definition 4** (Epoch and shuffle). *Time is divided into epochs of duration* T_{ep}*. At the start of each epoch, a verifiable random function (VRF) seeded by a shared random oracle assigns each stream* S_i *a Watcher swarm:*

$$\mathcal{W}_i = \mathsf{VRF}(seed_{ep}, i, n_i), \tag{2}$$

where $seed_{ep}$ *is unpredictable before the epoch begins. No node chooses its own swarm assignment.*

▶ **Definition 5** (Swarm size and Intercoin weight). *The swarm size* $n_i = \min(N, \lceil c\sqrt{INTER_i} \rceil)$ *for a system constant* c *calibrated so that streams at the base security level* ($s = 30$, $r = 1$) *use approximately 35 Watchers. Higher stake attracts larger swarms proportionally.*

3.3 Ripple Identifiers

▶ **Definition 6** (Ripple identifier). *A* ripple identifier *is* $rID = (originId, msgHash, ep)$. *Each node* v *maintains a hash table* $seen_v$ *of rID values. Before processing a message with identifier rID: if* $rID \in seen_v$, *discard; otherwise insert and process. Entries with* $ep < ep_{current} - 1$ *are evicted (retained for two epochs).*

3.4 Economic Relations and Transfers

▶ **Definition 7** (Intercoin backing invariant). *For every stream* S_i *and every coin type* c, *the* exchange rate $exRate(c) \in \mathbb{R}_{>0}$ *is a publicly computable function converting units of* c *to units of Intercoin (the default formula is given in Definition 45; pluggable alternatives are discussed in §15). The* backing invariant *requires:*

$$INTER_i \geq \sum_c S_i.balance[c] \cdot exRate(c). \tag{3}$$

This invariant is established at stream creation (when initial stake is deposited) and maintained as an invariant of every valid transfer (Lemma 8).

▶ **Lemma 8** (Backing invariant preservation). *Every valid transfer preserves the backing invariant (Definition 7) at both the sender stream* S_i *and the recipient stream* S_j.

Proof. Let $\Delta = a \cdot exRate(c)$. After a valid transfer of coin type c: $S_i.balance[c]' = S_i.balance[c] - a$ and $INTER_i' = INTER_i - \Delta$. All other balances are unchanged. We verify the backing invariant for S_i after the transfer:

$$\begin{aligned} INTER_i' &= INTER_i - \Delta \\ &\geq \sum_{c''} balance[c''] \cdot exRate(c'') - a \cdot exRate(c) \\ &= \sum_{c'' \neq c} balance[c''] \cdot exRate(c'') \\ &\quad + (balance[c] - a) \cdot exRate(c) \\ &= \sum_{c''} balance[c'']' \cdot exRate(c''), \end{aligned}$$

which is exactly the backing invariant for the updated balances. For S_j: $INTER_j' = INTER_j + \Delta$ and $balance[c]_j' = balance[c]_j + a$; since both sides of the invariant increase by Δ, the invariant is preserved. ◀

▶ **Definition 9** (Economic relation). *An* economic relation R_{ij} *from* S_i *to* S_j *is a pre-established channel specifying a rate-limit function* $r_{ij}(\Delta t)$ *and a set* $\mathcal{T}_{ij}$ *of permitted coin types. Transfers may flow only through pre-existing relations. A single transfer uses exactly one relation: a transfer of amount* a *in coin* c *from* S_i *to* S_j *debits* a *from* $S_i.balance[c]$ *exactly once.*

▶ **Definition 10** (Transfer message). *A* transfer *from* S_i *to* S_j *of amount* a *in coin type* c *is valid if: (i)* R_{ij} *pre-exists; (ii)* $c \in \mathcal{T}_{ij}$; *(iii)* $a \leq r_{ij}([t_{\text{last}}, t])$; *(iv)* $S_i.balance[c] \geq a$.

▶ **Definition 11** (Simple coin). *A* simple coin *is a stream* S_i *whose Rules program maintains a single encrypted* `owner` *field, representing indivisible ownership of the stream's balance. Simple coins are a special case of streams; all results proved for streams apply to simple coins. The more general* currency stream *(Definition 43) supports divisible supply with emission and retirement.*

3.5 The Three-Colour State Model

▶ **Definition 12** (Stream colour). *A stream S_i is:*

- ***Green**: The swarm $\mathcal{W}_i$ has reached finality (Definition 17).*
- ***Yellow**: Transactions are in progress but the swarm has not yet achieved the coherent supermajority-of-supermajority attestation.*
- ***Red**: Valid Πs have been propagated by junior observer nodes such that more than $\frac{2}{3}|\mathcal{W}_i|$ active (non-junior) swarm members are implicated — no honest supermajority can form, finality is unachievable, and the stream cannot be trusted. The stream remains Red until the next epoch shuffle assigns a fresh, independently drawn swarm.*

4 Ripple Termination in Distributed Intercloud

▶ **Theorem 13** (Distributed Ripple Termination). *Let $\mathcal{N}$ be the finite set of nodes in the Intercloud network ($|\mathcal{N}| = N < \infty$), and assume every message hop incurs a positive fee $\delta_{fee} > 0$. In an Intercloud network with the ripple-ID mechanism of Definition 6, every reactive execution ripple terminates within at most N message hops, and no node processes the same ripple more than once per epoch.*

Proof. *No duplicate processing within an epoch.* Let *rID* carry epoch *ep*. Before processing, node v checks $rID \in seen_v$. If present, v discards. Otherwise v inserts *rID* into $seen_v$ and processes. Since $seen_v$ is a set, the insertion occurs at most once per epoch. Therefore v processes any message bearing *rID* at most once per epoch.

Termination. Suppose a ripple with *rID* does not terminate. Then there is an infinite sequence of nodes $v_1, v_2, \ldots$ each processing a message with *rID*. By the previous step, each v_k is distinct. Since $|\mathcal{N}| = N < \infty$, no such infinite sequence exists. Contradiction.

Retention window sufficiency. Let D be the diameter of the network graph, finite since $|\mathcal{N}| < \infty$. Each hop takes wall-clock time at least $\delta_{\min} > 0$. Therefore any ripple initiated in epoch *ep* completes propagation to all reachable nodes within $D \cdot \delta_{\min}$ time. The epoch duration T_{ep} is chosen so that $T_{ep} > D \cdot \delta_{\min}$, ensuring every ripple completes within one epoch. Retaining *rID*s for two epochs therefore guarantees that any duplicate message carrying *rID* from epoch *ep* that arrives during epoch *ep* or *ep*+1 is detected and discarded. ◀

▶ Remark 14. The single-node version is proved in [1]. The distributed version requires two conditions: finiteness of the node set (ensuring the infinite-path argument reaches a contradiction) and a positive per-hop fee (ensuring a relay node cannot forward a ripple gratuitously to an unbounded chain of zero-cost intermediaries). No global coordination is required.

5 Chilling-Effect Consensus and Finality

5.1 Attestations and Proofs of Corruption

▶ **Definition 15** (Watcher attestation). *A* Watcher attestation *from node v for stream S_i is*

$$attest(v, i, h, ep) = Sign(k_v,\ v \| i \| h \| ep), \tag{4}$$

asserting that at epoch ep, node v has seen no hash other than h for stream S_i, and no Proof of Corruption has been propagated to v for this stream during this epoch.

▶ **Definition 16** (Conflicting attestations and Proof of Corruption). *Two attestations* $attest(v, i, h_1, ep)$ *and* $attest(v, i, h_2, ep)$ *from the same node* v *in the same epoch are* conflicting *if* $h_1 \neq h_2$. *A* Proof of Corruption $\Pi_v = (a_1, a_2)$ *is any pair of conflicting attestations from* v. *It is* self-certifying*: verification requires only* v*'s public key* k_v.

▶ **Definition 17** (Finality). *Swarm* $\mathcal{W}_i$ *has reached* finality *on hash* h *when:*

(i) A supermajority ($M_1 \geq \frac{2}{3}|\mathcal{W}_i|$) *of members have each signed* $attest(v, i, h, ep)$.
(ii) Each attesting member v *has verified via swarm gossip that at least* M_1 *other members have also attested to* h *— not just reported hearing about it, but signed their own attestation.*
(iii) No attesting member has seen any Π *involving* S_i *this epoch.*

Condition (ii) is a supermajority-of-supermajority *requirement that prevents split swarms from each locally declaring finality on different values.*

▶ Remark 18 (Why absence-of-evidence, not voting). Each Watcher attests that it has *not seen* a conflict. This is fundamentally different from voting on which of two competing hashes is correct. The distinction is critical for Buridan's Principle (Section 8): there is no "continuous signal" distinguishing two competing values, because the question has a definite default answer (no conflict seen) and deviates from the default only when concrete contradictory evidence exists.

▶ Remark 19 (Dynamic swarm membership and liveness). The formal model assumes the swarm $\mathcal{W}_i$ is fixed for the duration of the epoch by the VRF assignment. Nodes that go offline mid-epoch simply fail to contribute attestations, causing the stream to remain Yellow until the next epoch shuffle. In practice, an implementation may improve liveness by updating the effective swarm via DHT: unreachable nodes are removed from the active set and replaced, allowing finality to proceed with fewer nodes. This weakens the formal guarantees of Theorem 27 slightly but, with $n \geq 35$ and moderate churn rates, the probability of two clients computing conflicting supermajorities from genuinely divergent views remains negligibly small. High-value streams should use epoch-fixed membership as specified in the formal model.

5.2 Junior Observer Nodes and List of Liars

▶ **Definition 20** (Junior observer nodes). Junior observer nodes *are not current swarm members. They monitor swarm gossip and maintain: (a) a* List of Liars *containing all* Π*s seen this epoch, and (b) a read-only replica of swarm-signed local consensus for observed streams. Junior nodes with a clean record (no LoL entries across recent epochs) are eligible for promotion to swarm membership in the next shuffle. Earning Intercoin for discovering and broadcasting valid* Π*s creates a competitive incentive for honest monitoring.*

5.3 Swarm Size and Security Theorem

▶ **Theorem 21** (Swarm Security). *Using stream-identifier-based swarm assignment, a swarm of size*

$$n = \frac{\beta}{r} \log_e(s + 1 + \log(r + 2)) \tag{5}$$

detects any stream double-spent more than r *times with probability* $\geq 1 - e^{-s}$, *where* β *depends on* n *and* f *but is independent of total network size* N. *For* $r = 1$, $s = 30$, $f/n = 1/3$*:* $n \approx 35$.

Proof. We verify that Intercloud's Watcher swarms satisfy the three conditions required for Hoepman's Theorem 5.3 [10] to apply with stream identifiers substituted for coin identifiers.

(a) Unique, adversarially unpredictable stream identifiers. Each stream has a globally unique identifier i assigned at creation. The VRF-based swarm assignment $\mathcal{W}_i = \mathsf{VRF}(seed_{ep}, i, n)$ maps stream i to a specific clerk space from which n Watchers are drawn. Since $seed_{ep}$ is unpredictable before epoch start (Definition 4), the adversary cannot determine the clerk space for stream i before the epoch begins, matching Hoepman's requirement that coin-specific clerk spaces are not foreseeable.

(b) Adversary model compatibility. Hoepman assumes f total dishonest nodes and $d \leq f$ nodes corruptible *after* joining the system. Intercloud's VRF shuffle reassigns nodes each epoch, so any node corruptible after assignment corresponds to Hoepman's d. The parameter $d \approx 0$ for short epoch durations; for longer epochs, d is an explicit system parameter.

(c) Detection event mapping. Hoepman defines double-spending detection as two or more spending requests for the same coin reaching the clerk set. In Intercloud, two transfer requests tx_1, tx_2 for the same balance reach the Watcher swarm. The Watchers sign attestations for the resulting hashes; if these differ, a Π is produced (Definition 16). Π production is equivalent to Hoepman's detection event: it is certain (not probabilistic) given any two conflicting requests reaching any two swarm members.

With these three conditions verified, Hoepman's Theorem 5.3 gives swarm size $n = (\beta/r)\log_e(s + 1 + \log(r + 2))$ with β independent of N. For $d = 0$, $\beta = s/\log_e(n/f)$. Since $f = n/3$, we have $n/f = 3$ and $\beta = s/\log_e(3)$. Substituting into the formula for n with $r = 1$:

$$n = \frac{s}{\log_e 3} \cdot \log_e(s + 1 + \log_e 3). \tag{6}$$

For $s = 30$: $n \approx 27.3 \cdot \log_e(32.1) \approx 95$. This bound holds for $d = 0$. Hoepman's full formula for $d > 0$ yields $n \approx 35$ for the same $s = 30$, $r = 1$, $f/n = 1/3$ parameters. We adopt $n \approx 35$ as the *operational parameter* following Hoepman's calibration. The key structural result that this theorem proves — that n is *independent of the total network size* N — holds for any calibration. ◀

▶ **Corollary 22** (Proportional security). *The probability of successful double-spending against stream S_i is at most e^{-s} where $s \propto \sqrt{INTER_i}$. Doubling the security level requires quadrupling the Intercoin stake.*

Proof. From Definition 5, $n_i = \lceil c\sqrt{INTER_i} \rceil$. From Theorem 21, the detection failure probability is e^{-s} where $n = (\beta/r)\log_e(s + 1 + \log(r + 2))$. Inverting for fixed r and β: $s \approx nr/\beta - 1 - \log(r + 2) + O(\log s)$, which is linear in n *to leading order*. Therefore $s \propto n_i \propto \sqrt{INTER_i}$. To double n_i we need $INTER_i' \approx 4INTER_i$; quadrupling the stake doubles the swarm, which doubles s, reducing the failure probability from e^{-s} to $e^{-2s} = (e^{-s})^2$. ◀

6 Transfer Theorems

▶ **Theorem 23** (Zero-Sum Transfer). *A valid transfer (Definition 10) preserves total Intercoin weight: $INTER_i' + INTER_j' = INTER_i + INTER_j$, and no valid transfer reduces any stream's weight below zero.*

Proof. Let $\Delta = a \cdot exRate(c)$. After transfer: $INTER_i' = INTER_i - \Delta$ and $INTER_j' = INTER_j + \Delta$. The sum $INTER_i' + INTER_j' = INTER_i + INTER_j$ is preserved. Non-negativity: by the backing invariant (Definition 7), $INTER_i \geq balance[c] \cdot exRate(c) \geq a \cdot exRate(c) = \Delta$,

where the last inequality uses validity condition (iv). Therefore $INTER'_i \geq 0$. By Lemma 8, the backing invariant is maintained after the transfer. ◀

▶ **Theorem 24** (Double-Spend Prevention). *Assume the stream S_i has a single correct (non-Byzantine) Executor. Let $\mathcal{W}_i$ have reached finality on h_i (Definition 17). No valid transfer can spend the same balance twice under state h_i.*

Proof. The Executor processes transfer messages in the order they arrive, using Lamport's happens-before order [11] to break ties between concurrent submissions. This serialisation is total.

Sequential case. If tx_1 is appended first, reducing $balance[c]$ to $balance[c] - a_1$, and $a_1 + a_2 > balance[c]$, then $a_2 > balance[c] - a_1$, so tx_2 fails validity condition (iv) and is rejected.

Concurrent case. If tx_1 and tx_2 are submitted concurrently, both may initially appear pending to different Watchers. Once the Executor serialises them, honest Watchers observe the canonical post-serialisation hash and attest to it. If an adversarial party presents a different claimed hash to some Watchers before the Executor's result is confirmed, those Watchers sign conflicting attestations, immediately producing a Π. By Lemma 26, this Π reaches all correct clients with probability $\geq 1 - f_J^{\kappa r}$, and upon receipt, finality condition (iii) fails for both conflicting hashes. Neither transfer achieves Green finality.

Executor crash. If the Executor crashes mid-serialisation, the stream is temporarily unavailable but not corrupted: the append-only log retains the last consistent state. A Byzantine Executor is excluded by assumption; full Byzantine Executor resistance is left as future work. ◀

▶ **Theorem 25** (Capital-Flight Prevention). *The total value that can flow out of S_i in interval $[t, t + \Delta t]$ is bounded by $\sum_j r_{ij}(\Delta t)$.*

Proof. A valid transfer of amount a in coin c from S_i uses exactly one relation R_{ij} and deducts a from $S_i.balance[c]$ exactly once. By the backing invariant of Lemma 8, the same balance unit cannot be deducted twice without violating validity condition (iv) of a subsequent transfer. Therefore each unit of value leaving S_i flows through exactly one relation and is counted in exactly one $r_{ij}(\Delta t)$ term. The total outflow in $[t, t+\Delta t]$ is at most $\sum_j r_{ij}(\Delta t)$. ◀

7 Conditions for Client Disagreement

▶ **Lemma 26** (PoC Propagation). *Let f_J be the fraction of junior observer nodes controlled by the adversary, and κ the gossip fan-out (each node forwards to κ randomly chosen peers per round). Assume honest nodes remain available for all r rounds of gossip within the epoch. If a Π_v is held by at least one honest junior observer node, then after r gossip rounds the probability that it has* not *reached a specific client C_ℓ is at most $(f_J^\kappa)^r = f_J^{\kappa r}$.*

Proof. Consider one round of gossip. Every honest node that holds the Π *always* forwards it. Each such node forwards to κ peers chosen independently and uniformly at random from the full node set. For none of these κ forwarding steps to deliver the Π (even transitively) to C_ℓ, every one of the κ chosen peers must be adversarially controlled, since any honest peer would in turn forward onward. The probability that all κ chosen peers are adversarial is f_J^κ. After r rounds, the event "the Π has still not reached C_ℓ" requires that at every round, all κ peer-selection choices made by each honest holder landed on adversarial nodes. The peer sets chosen in distinct rounds are independent. Therefore the probability that all r rounds

of peer selection fail to place the Π within reach of C_ℓ is at most $(f_J^\kappa)^r = f_J^{\kappa r}$. For $\kappa = 5$, $f_J = 1/3$, $r = 3$: $f_J^{\kappa r} = (1/3)^{15} \approx 7 \times 10^{-8}$. By standard gossip analysis [5], the expected fraction of honest nodes holding the Π approaches 1 exponentially in r for any $f_J < 1$. ◀

▶ **Theorem 27** (Disagreement Characterisation). *Let C_1, C_2 be correct clients querying stream S_i with assigned swarm $\mathcal{W}_i$ of size n. Define:*

(I) ***Swarm compromise.*** *The adversary controls a set $A \subseteq \mathcal{W}_i$ with $|A| > \frac{2}{3}n$.*

(II) ***Client isolation.*** *The adversary can eclipse both C_1 and C_2 from all honest nodes: no message from any honest swarm member or honest junior observer reaches either client.*

Then: (a) Necessity*: (I)∨(II) is necessary for conflicting finality; (b)* Sufficiency*: (I)∧(II) is sufficient for the adversary to cause conflicting finality. By the union bound,* $\Pr[(I) \vee (II)] \leq e^{-s} + f_J^{\kappa r}$*, negligible for standard parameters.*

Proof. *Part (a): Necessity.* We prove the contrapositive: ¬(I)∧¬(II) implies no conflicting finality. Assume the adversary controls $|A| \leq \frac{2}{3}n$ of $\mathcal{W}_i$, so honest nodes form a set H with $|H| \geq n/3$. Suppose for contradiction both clients hold valid finality for conflicting hashes. There exist $F_1, F_2 \subseteq \mathcal{W}_i$ with $|F_1|, |F_2| \geq \frac{2}{3}n$, each satisfying Definition 17. By inclusion-exclusion:

$$|F_1 \cap F_2| \geq |F_1| + |F_2| - n \geq \tfrac{4n}{3} - n = \tfrac{n}{3} > 0. \tag{7}$$

Let $v \in F_1 \cap F_2$. Node v has signed both $attest(v, i, h_1, ep)$ and $attest(v, i, h_2, ep)$ with $h_1 \neq h_2$, so Π_v exists. For finality condition (ii) to hold for any $v \in F_1$, node v must have received gossip attestations from the full set F_1, and similarly for F_2. These messages transit through swarm gossip, which by assumption includes honest nodes $u \in H$. Each $u \in H$ that receives both attestations for the same v constructs Π_v immediately and begins forwarding it.

Under ¬(II), at least one honest node can reach C_1 or C_2. The honest node u holding Π_v delivers it to both clients with probability $\geq 1 - f_J^{\kappa r}$ via Lemma 26. A correct client receiving Π_v rejects finality (condition (iii) fails), contradicting the assumption.

Part (b): Sufficiency. We construct an adversary strategy. Let the adversary control $A \subseteq \mathcal{W}_i$ with $|A| = \lceil \frac{2}{3}n \rceil + 1$. Define $h_1 \neq h_2$.

Phase 1. Each $v \in A$ signs both $attest(v, i, h_1, ep)$ and $attest(v, i, h_2, ep)$, generating Π_v (which the adversary suppresses).

Phase 2. The adversary eclipses both clients (condition (II)). To C_1, the adversary delivers only $\{attest(v, i, h_1, ep) : v \in A\}$; to C_2, only $\{attest(v, i, h_2, ep) : v \in A\}$.

Phase 3. The adversary arranges for each $v \in A$ to receive (from other members of A) gossip messages attesting to h_1 when interacting with C_1's network view and h_2 when interacting with C_2's. Each $v \in A$ thus satisfies condition (ii) for the hash it presents to each client.

Phase 4. The adversary instructs all $v \in A$ to suppress any Π they receive. Under condition (II), no honest node can deliver a Π. Thus condition (iii) holds vacuously for all $v \in A$.

Each client C_ℓ receives a set of attestations satisfying all three conditions of Definition 17. Both clients therefore hold valid finality attestations — C_1 for h_1 and C_2 for h_2 — simultaneously. ◀

8 Buridan's Principle Does Not Apply

Lamport's Buridan's Principle [12]: *a discrete decision based upon an input having a continuous range of values cannot be made within a bounded length of time.* Applied to consensus: if two competing transaction proposals are "arbitrarily close" — differing by an infinitesimally small timing difference — any arbiter may be driven into indecision of arbitrary length.

▶ **Theorem 28** (Buridan Inapplicability). *Chilling-effect consensus (Definition 17) is not subject to Buridan's Principle.*

Proof. Lamport [12] formalises the principle as follows. Let $A_t(x)$ be the state of an arbiter at time t given initial condition $x \in [0, 1]$ (a continuous parameter). If A_t is a continuous function of x and must converge to one of two discrete states, then for every $T > 0$ there exists an x such that $A_T(x)$ is neither 0 nor 1. Chilling-effect consensus is not subject to this principle for two reasons.

(a) There is no continuous input parameter. In Lamport's model, x represents a physical quantity with a continuous range. In chilling-effect consensus, the "input" is the count of Watcher attestations for hash h: an integer $k \in \{0, 1, \ldots, n\}$. There is no $x \in (0, 1)$ parameterising a "nearly indifferent" input. The threshold is a crisp integer: either $k \geq \lceil \frac{2}{3}n \rceil$ or $k < \lceil \frac{2}{3}n \rceil$. The integer gap between k and $k{+}1$ is exactly 1 attestation.

(b) The decision is not between two competing values. Chilling-effect consensus does not choose between two competing hash values. It asks: *has a supermajority attested to some single hash with no* Π *observed?* This has a ground state (Yellow: not yet) and transitions to Green when the integer count crosses the threshold. If a Π is produced, the stream transitions to Red — a discrete event triggered by concrete evidence. At no point does the protocol need to choose between h_1 and h_2.

Residual Yellow state. The Yellow state is not indefinite indecision but bounded waiting. The maximum duration is T_{ep}, after which the epoch shuffle provides a fresh swarm. ◀

▶ Remark 29 (Practical Yellow duration). In practice, T_{ep} is chosen so that a correctly functioning swarm reaches Green well within a single epoch for any stream with legitimate traffic. The Yellow state is overwhelmingly the transient state between message submission and swarm attestation, not a sign of conflict.

9 End Users as the Court of Final Resort

A blockchain aggregates all ambiguity into a single chain and resolves it by expending resources proportional to total network stake. Every disagreement — including those involving transactions of negligible value — is resolved by the full network. This is the Ministry of Truth model: global arbitration for every dispute.

Theorem 27 shows that genuine ambiguity (two correct clients disagreeing) requires doubly negligible probability events. When a stream is Yellow, it does not indicate fraud — it indicates that finality has not yet been reached. The recipient decides:

▶ **Proposition 30** (User-determined finality). *A recipient who requires certainty waits for Green (at most T_{ep}). A recipient who tolerates small risk accepts Yellow immediately. A recipient who sees Red waits for the next epoch shuffle. No global process adjudicates between these choices.*

This faithfully models economic reality. Merchants already make risk-adjusted acceptance decisions (cash versus cheque versus credit card). Intercloud makes the risk explicit and cryptographically bounded.

10 The Single Global Agreement

▶ **Theorem 31** (Minimal Global Consensus). *The only global agreement required by Intercloud is the sequence of random seeds* $\{seed_{ep}\}$ *for VRF-based swarm assignment. All other consensus is local to each stream's swarm.*

Proof. Finality (Definition 17) requires only swarm-internal gossip. Transfer validity requires only the stream's current state. Ripple deduplication (Theorem 13) requires only the local $seen_v$ table. Capital-flight prevention requires only pre-existing relations. None of these requires global agreement. The VRF seed $seed_{ep}$ requires global agreement to prevent adversarial swarm selection. A sparse RANDAO achieves this in $O(N)$ messages *per epoch*; amortised over all transactions in an epoch, the per-transaction cost approaches zero as volume grows. ◀

▶ Remark 32 (Source of the random oracle). The RANDAO may be drawn from Intercloud's own nodes or from an external source (Ethereum beacon chain, drand, trusted hardware). The security of swarm assignment requires only unpredictability before epoch start.

11 Network Privacy and DDOS Resistance

▶ **Definition 33** (Attestation-gated requests). *All cross-node requests must include a recent OCP-signed node attestation proving the sender is a genuine, AMI-verified instance. Requests without a valid attestation are silently dropped. Attestations are epoch-bounded and cannot be replayed.*

▶ **Proposition 34** (DDOS resistance). *Attestation-gating prevents anonymous DDOS. Any attacker must operate a genuine instance, incurring verifiable infrastructure costs and creating an on-chain identity eligible for slashing. Anonymous volume attacks are eliminated; economically motivated attacks face stake penalties.*

Operators may configure nodes to strip IP addresses after the first forwarding hop, passing only signed OCP envelopes. This provides origin anonymity at the cost of routing efficiency, and is an optional per-deployment configuration.

12 Zero-Knowledge Extension

The base model requires Executors to hold plaintext Rules and balances. A theoretical extension replaces these with ZK commitments established at rail-creation time.

▶ **Definition 35** (ZK-committed stream). *Each valid state transition produces a proof*

$$\pi = \mathsf{ZKP}\{(\sigma, \sigma', m) : C(\sigma, m) = \sigma' \wedge Com(\sigma) = h \wedge Com(\sigma') = h'\}$$

where C *is the Rules circuit compiled when the stream's economic relations were established. Watchers verify* π *using the public circuit commitment; no plaintext is revealed to any node.*

zk-SNARKs [9] provide compact proofs ($\approx$ 200 bytes, Groth16) but require trusted per-circuit setup at rail creation; appropriate for high-volume streams. zk-STARKs [2] have no trusted setup and are post-quantum secure, but with larger proofs ($\approx$ 40 KB); appropriate for regulated streams. Universal ZK deployment makes the network opaque to

all external parties, including regulators. We recommend selective deployment: personal and communication streams use ZK commitments; regulated financial streams use selective disclosure proofs that can reveal specific fields to authorised recipients on lawful demand.

13 Junior Nodes, Corruption Detection, and PoC Rewards

13.1 Junior Nodes as the Corruption Detection Layer

Active swarm members earn Intercoin for securing transactions. They have a conflict of interest: a corrupt swarm member might prefer to suppress a Π against a fellow member rather than lose the stream's fees by triggering a Red transition. The design therefore delegates corruption *detection* to agents who have the opposite incentive: junior observer nodes, who are not in the current swarm and actively want corrupt members expelled so they can take their place.

▶ **Definition 36** (Corruption detection by junior nodes). *A junior observer node watching stream S_i monitors the signed attestations of all active swarm members $v \in \mathcal{W}_i$. If it observes two signed claims from the same v that constitute a Π_v, it broadcasts Π_v to all peers. If valid Πs accumulate against more than $\frac{2}{3}|\mathcal{W}_i|$ distinct active members, the stream transitions to Red and those members' Intercoin stakes are eligible for slashing in the next epoch.*

The stream goes Red not because a single node misbehaved, but because the swarm as a whole can no longer form an honest supermajority. Junior nodes holding the List of Liars provide read-only availability of the last known-good consensus state during the Red period.

13.2 PoC Reward Mechanism: Lottery over Forwarders

A naive reward rule — "pay the node that first submits a valid Π" — is vulnerable to front-running: a dishonest node with fast network access could observe an incoming Π, strip the discoverer's identity, re-sign it as its own, and race to submit first.

▶ **Definition 37** (PoC lottery reward). *When a Π is accepted, every node that* forwarded *the Π during the current epoch receives one* lottery ticket. *The winning ticket is drawn at the start of the next epoch using the new VRF seed:*

$$winner = \mathsf{VRF}(seed_{ep+1},\ PoC\_id,\ ticket\_list). \tag{8}$$

The slashed stake of the corrupt nodes is distributed to the winner.

This mechanism has three properties. *Front-running resistance*: replacing the discoverer's identity does not increase one's lottery odds. *Propagation incentive*: suppressing a Π yields zero tickets and zero reward; forwarding yields a chance at the entire slashed stake. *Unpredictability*: the winner is determined by the future VRF seed, unknown at forwarding time.

▶ **Proposition 38** (PoC forwarding dominates suppression). *For any junior observer node with positive belief that a Π is valid, the expected reward from forwarding strictly exceeds the expected reward from suppressing, regardless of network position or timing.*

Proof. Suppression yields expected reward 0. Forwarding yields expected reward V_{slash}/T, where T is the total number of forwarders in the epoch and V_{slash} is the slashed stake.

$V_{slash} > 0$ and $T < \infty$, so the expected reward is strictly positive. The cost of forwarding is a single signed broadcast message — negligible. Forwarding strictly dominates suppression in expected net value. The timing of forwarding does not affect T, so there is no advantage to racing. ◀

14 Intercoin Stake, Vesting, and Rational Security

14.1 Vesting as a Security Primitive

Watcher and junior nodes earn Intercoin for securing streams. If earned Intercoin could be withdrawn immediately, a rational node might accept a bribe to corrupt a single high-value stream and immediately exit. Intercloud prevents this with a vesting constraint enforced by the network's own rate-limiting mechanism.

▶ **Definition 39** (Intercoin vesting). *Earned Intercoin is credited to a node's* vesting stream. *Withdrawal is subject to a rate-limit function $r_{vest}(\Delta t)$, e.g., $r_{vest}(1\,\text{day}) = 0.10 \cdot INTER_{vested}$. The* staked balance *is total vested-but-not-withdrawn Intercoin, which determines how many streams it is eligible to watch and how large a slash it would suffer upon corruption.*

▶ **Definition 40** (Per-node staking requirement). *To be eligible to watch stream S_i, a node must maintain a staked balance of at least*

$$INTER^{\min}_{stake}(i) = INTER_i/n_i, \tag{9}$$

i.e., each Watcher's stake must cover its pro-rata share of the stream's Intercoin weight. This requirement is enforced by the network policy layer and verified at swarm assignment time.

▶ **Theorem 41** (Rational incorruptibility). *Under Definition 40, no rational node — acting alone or as part of a coalition — will attempt to corrupt stream S_i: for every coalition $\mathcal{C} \subseteq \mathcal{W}_i$, the expected gain is strictly less than the expected cost.*

Proof. *Single-node case.* A single corrupt node v generates a Π by producing conflicting attestations. By Lemma 26 and Proposition 38, detection is near-certain. Detection triggers slashing of v's entire stake $INTER_{stake}(v) \geq INTER_i/n_i$. A single node cannot complete a double-spend: achieving Green finality requires a supermajority to attest. Expected gain $= 0 < \epsilon =$ expected honest income.

Coalition case. Consider a coalition $\mathcal{C}$ of $|\mathcal{C}|$ nodes. Their combined extractable value is at most $INTER_i \cdot |\mathcal{C}|/n_i$. Their combined slashing cost, upon detection, is $\sum_{v \in \mathcal{C}} INTER_{stake}(v) \geq |\mathcal{C}| \cdot INTER_i/n_i$. So even for a coalition, expected gain $\leq$ expected loss.

Game-theoretic stability. Let $g = INTER_i/n_i$ and $\ell = INTER_{stake}(v) \geq g$. The payoff matrix is:

$$\text{Payoff}(v) = \begin{cases} +\epsilon & v \text{ honest}, \\ g - \ell \leq 0 & v \text{ corrupts \& detected}, \\ g & v \text{ corrupts \& evades}. \end{cases}$$

Expected payoff from corruption:

$$\begin{aligned} \mathbb{E}[\text{Corrupt}] &\leq f_J^{\kappa_T} g + (1 - f_J^{\kappa_T})(g - \ell) \\ &= g - (1 - f_J^{\kappa_T})\ell \leq g - \ell + f_J^{\kappa_T} \ell. \end{aligned}$$

Since $\ell \geq g$ and $f_J^{\kappa r}$ is negligible, $\mathbb{E}[\text{Corrupt}] \approx 0$. Honest operation yields $+\epsilon > 0$. Honest strictly dominates Corrupt for every node regardless of what others do. Honest for all nodes is the unique Nash equilibrium. ◀

▶ Remark 42 (Gradual vesting as a circuit breaker). The vesting rate also functions as a capital-flight circuit breaker at the security layer: even if many nodes simultaneously tried to withdraw, the rate limit caps the reduction in total staked security per unit time, giving the network time to detect anomalous behaviour and respond.

15 Local Coins, Emission, Retirement, and Exchange Rates

15.1 Streams as Smart-Contract Analogues

Streams in Intercloud can hold and transfer balances of *local coins* — community-issued currencies or application tokens — in addition to the global Intercoin reserve. This makes a stream the embarrassingly parallel analogue of a smart contract: each stream is an independent, append-only state machine that can hold assets and execute rules, but streams execute in parallel with no global ordering requirement. Unlike Ethereum smart contracts, all state transitions are private by default: Watchers see only hashes. Regulators and auditors can follow aggregate coin flows between streams without observing the content of any individual stream.

15.2 Currency Streams: Emission and Retirement

▶ **Definition 43** (Currency stream). *A stream of type* `Intercloud/currency` *is a* currency stream S_c *that maintains: supply*$_c$ *(cumulative emissions minus cumulative retirements),* $INTER_c^{reserve}$ *(the Intercoin reserve backing coin c), and an emission and retirement policy encoded in Rules*$_c$.

▶ **Definition 44** (Emission and retirement). *An* emission event *appends a message to* S_c *minting* $\delta > 0$ *new coins, increasing supply*$_c$ *by* δ *and depositing them into a specified recipient stream. A* retirement event *burns* δ *coins, decreasing supply*$_c$ *by* δ. *Both are valid only if permitted by Rules*$_c$. *The Intercoin reserve may be increased by depositing Intercoin, or decreased (up to the rate limit) by withdrawing.*

15.3 Exchange Rate Formula

▶ **Definition 45** (Default exchange rate). *The* default exchange rate *of coin type c to Intercoin at any moment is:*

$$exRate(c) = \frac{INTER_c^{reserve}}{supply_c} = \frac{INTER_c^{reserve}}{emitted_c - retired_c}, \tag{10}$$

provided $supply_c > 0$. *If* $supply_c = 0$, *the rate is undefined.* $exRate(c)$ *is the Intercoin value per unit of c, equal to the reserve per circulating unit. It rises when Intercoin is deposited or coins are retired; it falls when coins are emitted without a proportional reserve increase.*

▶ Remark 46 (Pluggable exchange rates). The default formula is not mandatory. A currency stream may specify an alternative function in its Rules program — a fixed peg, an algorithmic curve, or an oracle-fed rate. Whatever formula is used, $exRate(c)$ must be publicly computable from the integrity layer.

▶ Remark 47 (Reserve manipulation and rate limits). Because the currency stream operator controls the Intercoin reserve, they could in principle inflate $exRate(c)$ by depositing Intercoin just before a transfer and withdrawing afterward. This is mitigated by the rate-limit mechanism of Definition 9: reserve deposits and withdrawals are subject to pre-subscribed economic relations. A sudden large reserve change is therefore bounded in magnitude and takes time proportional to the rate limit.

▶ Remark 48 (Exchange rate update timing). $exRate(c)$ changes only when emission or retirement events occur on S_c, or when the reserve is adjusted via a rate-limited relation. Between such events the rate is constant. The security weight of a coin transfer, $\Delta = a \cdot exRate(c)$, is therefore deterministic and auditable from the integrity layer.

16 The Dot-Product Security Theorem

16.1 Security Follows Value

The core economic property of Intercloud is that the security allocated to any stream is automatically proportional to the economic value it holds.

▶ **Definition 49** (Stream economic value). *The* economic value *of stream S_i, measured in Intercoin, is the dot product of its local coin balances with their exchange rates:*

$$V_i = \sum_c S_i.balance[c] \cdot exRate(c). \tag{11}$$

By the backing invariant (Definition 7), $INTER_i \geq V_i$ at all times.

▶ **Theorem 50** (Security tracks value). *Let S_i hold economic value V_i. Then:*

(i) $INTER_i \geq V_i$.
(ii) $n_i \geq \lceil c\sqrt{V_i} \rceil$.
(iii) Double-spending probability is at most e^{-s} where $s \propto \sqrt{V_i}$.
(iv) When value $a \cdot exRate(c)$ moves from S_i to S_j, exactly that amount of Intercoin weight moves with it: $INTER_j$ increases and $INTER_i$ decreases by $\Delta = a \cdot exRate(c)$, preserving $INTER \geq V$ at both streams.

Proof. (i) Directly from the backing invariant and Definition 49. (ii) From Definition 5: $n_i = \lceil c\sqrt{INTER_i} \rceil \geq \lceil c\sqrt{V_i} \rceil$ since $INTER_i \geq V_i$. (iii) From Corollary 22: $s \propto n_i \propto \sqrt{INTER_i} \geq \sqrt{V_i}$. (iv) From Theorem 23: Δ is deducted from $INTER_i$ and added to $INTER_j$. By Lemma 8, the backing invariant is preserved. The exchange rate is determined by the state of S_c. By Remark 48, it changes only on emission/retirement events or reserve adjustments, which are themselves serialised by S_c's Executor. The transfer on S_i is serialised by S_i's Executor. Δ is computed and committed atomically with the transfer message appended to $\mathcal{M}_i$ — the message records both a and the $exRate(c)$ value used. Any subsequent change to $exRate(c)$ does not retroactively alter Δ. The backing invariant holds at each stream at the logical time of the append to that stream. ◀

▶ **Corollary 51** (Self-calibrating security). *No operator needs to manually configure how much security a stream receives. As value flows into a stream, Intercoin weight flows with it, the backing invariant ensures $INTER_i \geq V_i$, and the next epoch shuffle assigns a larger swarm proportional to $\sqrt{INTER_i}$. Security is a local property of each stream, automatically calibrated to the value it holds.*

16.2 Compartmentalisation

▶ **Proposition 52** (Security compartmentalisation). *A successful attack on stream S_i requires bribing more than $\frac{2}{3}n_i$ swarm members. The minimum cost of such a bribe exceeds $\frac{2}{3}V_i$.*

Proof. Each corrupt member's stake is at least $INTER_i/n_i$ (Definition 40). By Theorem 41, a rational member accepts a bribe only if it exceeds their staked balance. The minimum bribe per member is $> INTER_i/n_i$. Corrupting a supermajority requires bribing more than $\frac{2}{3}n_i$ members:

$$\text{total bribe} > \tfrac{2}{3}n_i \cdot \frac{INTER_i}{n_i} = \tfrac{2}{3}INTER_i \geq \tfrac{2}{3}V_i, \tag{12}$$

where the last inequality uses $INTER_i \geq V_i$. ◀

An attacker must spend more than two-thirds of a stream's value to corrupt it. This is the formal expression of the armoured-convoy principle: the cost of attack scales with the value of the target, not with the size of the global network.

17 Privacy, Transparency, and the Coin–Content Separation

17.1 Two Orthogonal Layers

Every stream in Intercloud has two logically distinct layers. The **coin layer** comprises balances, transfers, exchange rates, emission, and retirement events. These are secured by Watchers who see only hashes. The aggregate flow of coins between streams is observable to anyone watching the integrity layer, enabling regulatory oversight of value movement without revealing individual transaction contents. The **content layer** comprises the actual messages, documents, agreements, chat histories, ownership records, or disbursement rules encoded in $\mathcal{M}_i$. These are end-to-end encrypted and visible only to parties holding the appropriate decryption keys.

▶ **Definition 53** (Coin–content separation). *The* coin layer *of stream S_i consists of the subset of messages in $\mathcal{M}_i$ that modify balances, trigger emission/retirement events, or adjust the Intercoin reserve. These must produce integrity-layer effects (changes to h_i, $INTER_i$, and exchange rates) visible to Watchers. The* content layer *consists of all remaining messages — private communications, signed agreements, governance votes, or any other application data, which may be fully encrypted.*

▶ Remark 54 (Pre-signed coin movements). A stream's content layer can contain pre-signed authorisations for future coin movements. For example, a stream holding an escrow agreement can contain a cryptographically signed disbursement instruction that, when revealed, triggers a transfer on the coin layer. The *existence* of an agreement is private, but its *execution* on the coin layer is observable.

17.2 Regulatory Observability

▶ **Proposition 55** (Regulatory coin-weight observability). *A regulator with access to the integrity layer can observe, for every transfer between streams S_i and S_j, the Intercoin weight transferred ($\Delta = a \cdot exRate(c)$), the direction, and the timestamp — without learning the plaintext amount a, the coin type c, the identities of stream owners, or any message content.*

Proof. Every valid transfer updates h_i and h_j and moves Intercoin weight Δ from $INTER_i$ to $INTER_j$ (Theorem 23). The integrity layer stores $(h_i, INTER_i, H(Rules_i))$, so the change $\Delta INTER_i = -\Delta$ and $\Delta INTER_j = +\Delta$ are observable at the integrity layer, as is the timestamp. The pre-established relation R_{ij} is also integrity-layer metadata. The plaintext amount a, coin type c, stream owner identities, and message content are in the data layer and are not accessible to Watcher nodes. ◀

▶ Remark 56 (Flow graph vs. transaction graph). A regulator sees a directed weighted graph of Intercoin weight flows between stream identifiers, with timestamps. This is weaker than a full transaction graph but stronger than nothing: the topology of economic relationships and the magnitude of value flows are visible. This matches the regulatory need to detect systemic risk and large-scale money flows without individual transaction surveillance.

This is a strictly weaker transparency requirement than any public blockchain: instead of revealing all transaction data globally, Intercloud reveals only the aggregate flow structure. Communities can publish their own coins and participate in the global economy while their internal transactions remain private.

17.3 The Global Intercoin Reserve

Intercoin is the single global reserve currency of the Intercloud network. Its role is to represent *security weight*. When a community issues local coins backed by Intercoin, the Intercoin reserve is locked in the currency stream until coins are retired. The exchange rate formula (Definition 45) ensures that the security weight of a unit of local coin is always exactly proportional to the Intercoin reserve per circulating unit.

▶ **Corollary 57** (Conservation of security weight). *The total Intercoin weight across all streams is conserved: $\sum_i INTER_i = INTER_{total}$ (a fixed global supply). Security is not created or destroyed — it moves between streams as coins move.*

Proof. *Base case.* When a new stream is created with initial Intercoin weight $INTER_i^{(0)}$, that weight is transferred from an existing funding stream. By Theorem 23, this transfer preserves the global sum.

Inductive step. Assume $\sum_i INTER_i = INTER_{total}$ after k transfers. A valid $(k{+}1)$-th transfer preserves $INTER_i + INTER_j$ (Theorem 23) and leaves all other $INTER_\ell$ unchanged. By induction, the sum is conserved for all sequences of valid transfers. ◀

18 Comparison and Related Work

Table 1 summarises the key differences between Intercloud and other major decentralised consensus approaches. Intercloud is the only system among these in which per-transaction validation cost is independent of network size and Watcher nodes do not see plaintext.

19 Limitations and Future Work

19.0.1 Byzantine Executor (primary open problem)

Theorems 24 and 25 rest on a bounded assumption: each stream has a single *correct* (non-Byzantine) Executor. This is the primary limitation of the current model. A Byzantine Executor could produce conflicting hashes, generating Πs against itself and driving the stream to Red — but crucially, it cannot forge a valid transfer without the stream owner's

Table 1 Comparison of decentralised consensus systems.

System	Global consensus	Per-tx validation	Ledger visible	Double-spend	Max ambiguity
Bitcoin	PoW, global	$O(N)$	Yes	Probabilistic	Unbounded
Ethereum	PoS, global	$O(N)$ + gas	Yes	Probabilistic	$\leq$ epoch
PBFT	BFT voting	$O(n^2)$	Config.	Deterministic	None[†]
HotStuff [15]	BFT pipeline	$O(n)$	Config.	Deterministic	None[†]
Algorand [8]	VRF committee	$O(n)$	Yes	Probabilistic	$\leq$ round
Zcash	PoW, global	$O(N)$	Pool only	Probabilistic	Unbounded
Intercloud	VRF seed	$O(1)$ amort.	Hashes only[‡]	$\geq 1 - e^{-s}$	$\leq T_{ep}$

[†]PBFT guarantees termination only under partial synchrony [4]; under full asynchrony it may not terminate.
[‡]Watchers hold only hashes; coin-weight flows between streams are observable at the integrity layer (§17).

private key, so funds cannot be stolen outright. Only service availability is at risk. The Watcher swarm thus provides *integrity* unconditionally and *liveness* conditional on Executor correctness. Full Byzantine Executor resistance — providing both under $f < n/3$ Executor faults — requires a threshold-signature Executor cluster or a small BFT sub-committee per stream, and is the principal direction for future work.

19.0.2 Executor availability

A crashed Executor makes the stream temporarily unavailable. The append-only log ensures crash recovery to the last consistent state. High-availability Executor deployments (active-passive replication) are straightforward but not formalised here.

19.0.3 Exchange rate oracle

Definition 7 uses a publicly known exchange rate. In practice this rate changes over time and must come from an oracle. If the oracle is compromised, the backing invariant could be violated. A multi-oracle range-agreement mechanism mitigates this risk but is not formalised in the current model.

19.0.4 Epoch duration

The bound $T_{ep} > D \cdot \delta_{\min}$ depends on the network diameter D and minimum hop time $\delta_{\min}$. Choosing T_{ep} too short risks ripple ID collisions across epochs; choosing it too long increases the maximum Yellow duration. Adaptive epoch-duration protocols are future work.

19.0.5 Network-level privacy

Even with IP stripping after the first hop, the first-hop IP is visible to the first relay. Full network-level anonymity requires a Mixnet or onion-routing layer, deferred to future work.

20 Conclusion

Intercloud achieves decentralised economic consensus with properties that no prior system simultaneously exhibits: sub-linear per-transaction validation cost, hash-only privacy for all Watcher nodes, chilling-effect deterrence replacing Byzantine voting, explicit user-controlled

finality via the three-colour state model, and a self-calibrating security economy in which protection is automatically proportional to the value being protected.

The main theorems build on the Magarshak Machine [1] and Hoepman's clerk-set bounds [10]: Ripple Termination, Swarm Security, Disagreement Characterisation, Buridan Inapplicability, Rational Incorruptibility, Security Tracks Value, and Conservation of Security Weight form a coherent stack from the network layer to the economic layer. The junior-node corruption detection mechanism and PoC lottery reward system ensure that exposing dishonest nodes is the dominant strategy for every rational participant. Intercoin vesting ensures that a node's stake always exceeds the value it could extract from any single stream it watches.

Local coins, backed by a pluggable exchange-rate formula tied to the ratio of Intercoin reserve to circulating supply, flow across pre-subscribed economic rails. The dot-product of balances and exchange rates determines each stream's security weight automatically. The coin layer and content layer are strictly separated: regulators can follow the flow of value between streams without observing identities, messages, or rules.

The blockchain model deploys an armoured convoy for every dollar. Intercloud deploys a convoy proportional to the square root of the value transported — and proves that the convoy self-assembles, self-calibrates, and self-polices without any global coordinator. The single global agreement required is an epoch random seed, costing $O(N)$ messages per epoch and amortising to zero per transaction at scale. The Turing Machine defines what can be computed. The Magarshak Machine defines how distributed state evolves safely. Intercloud defines how economies of Magarshak Machines coordinate without a global ledger.

AI Disclosure

Claude (Anthropic) was used to assist with LaTeX formatting and conversion to LIPIcs format, and with prose editing of the abstract and introduction. The author verified the correctness and originality of all content including references.

References

1 Anonymous. The Magarshak Machine: A stream-partitioned model for governed state evolution in distributed systems, 2026. Companion paper, preprint.

2 E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.

3 E. Buchman. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*. PhD thesis, University of Guelph, 2016.

4 M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, pages 173–186, 1999.

5 A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12, 1987.

6 C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

7 M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

8 Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, pages 51–68, 2017.

9 J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, 2016.

10 J.-H. Hoepman. Distributed double spending prevention. arXiv preprint arXiv:0802.0832, 2008.

11 L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

12 L. Lamport. Buridan's principle. *Foundations of Physics*, 42(8):1056–1066, 2012. Written 1984, published 2012.

13 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

14 M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In *Symposium on Self-Stabilizing Systems*, pages 386–400, 2011.

15 M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347–356, 2019.