Tyche

Draft Document v0.1

hello@tyche.network

1 Executive Summary 3
2. Motivation 3
3. Design Goals 5
4. System Overview 6
5. Rounds and the Tyche Attestation Model 8
6. Tyche Services 10
7. Roles in the Network 13
8. Security Properties 16
9. Governance & Trust Model 18
10. Use Cases 22
11. Call for Collaboration 25

1 Executive Summary

Tyche is a distributed witness network designed to provide real-time, cryptographically verifiable attestations of **timestamp**, **event ordering**, and **randomness** without the complexity of consensus or the overhead of a blockchain. The system operates on a global round clock, producing threshold-signed outputs every few hundred milliseconds. Each round functions as a neutral attestation checkpoint: operators sign the round output, gateways commit event trees, and entropy providers contribute randomness. The resulting artifact is a compact, verifiable proof that binds time, ordering, and entropy into a single, deterministic structure.

Modern distributed systems increasingly require trustworthy external witnesses. Applications need auditable timestamps to anchor logs, unbiased randomness for fairness protocols, and deterministic ordering to prove the sequence of events across organizational boundaries. Today, these guarantees require separate systems - Roughtime for time, drand or NIST for randomness, Certificate Transparency–style logs for auditability, and blockchains for ordering. Each solves a single problem, introducing duplicated infrastructure and inconsistent trust assumptions.

Tyche unifies these primitives. Instead of maintaining global state or achieving agreement on transactions, Tyche relies on a stateless, round-based protocol: every round commits what the network observed, attests the time, aggregates contributed entropy, and signs the result using a multiparty threshold signature. Gateways provide client access, letting applications submit digests and receive inclusion proofs, while entropy providers feed randomness commitments that strengthen the beacon against bias. Operators maintain synchronized clocks, exchange subtree roots, and produce the final signed output for each round.

Because Tyche is stateless and does not require consensus, it achieves sub-second finality with predictable performance and minimal operational burden. The security of the network derives from a diverse set of independent operators and entropy providers rather than economic incentives or token-based mechanisms. Applications verify round outputs offline, making Tyche suitable for compliance, audit logs, cross-organizational workflows, commit–reveal schemes, fair ordering, and emerging integrity needs in AI and automated decision systems.

Tyche's goal is not to compete with blockchains or replicate their economic models. Its purpose is simpler and more foundational: to provide a neutral, verifiable, globally consistent attestation layer that any system - centralized or decentralized - can rely on. By combining timestamping, ordering, and randomness in a single unified mechanism, Tyche offers a fast, composable trust primitive for the next generation of distributed computing.

2. Motivation

Digital systems increasingly depend on trustworthy attestations about what happened, when it happened, and in what order. These guarantees - time, ordering, and randomness - form the backbone of auditability, fairness, transparency, and cross-organizational coordination. Yet today's solutions are fragmented, inconsistent, and often inadequate for real-time applications.

Timestamping systems provide cryptographic proof that data existed at a certain time, but they lack a shared notion of *global sequence*. Logs can be timestamped independently, but comparing events across organizations remains ambiguous when clock drift and inconsistent time authorities

differ by milliseconds or seconds. Traditional timestamp authorities also create single points of failure, relying on centralized infrastructure and trust assumptions that do not scale to multi-party environments.

Randomness beacons, such as NIST's or drand's, offer public, verifiable randomness but do nothing to anchor events or prove ordering. A fairness system may rely on a beacon for unbiased randomness yet still lack a trustworthy mechanism to attest the sequence of bids, reveals, or commitments that surround it. The randomness is trustworthy; the context in which it is used is not.

Ordering, meanwhile, typically lives inside centralized systems or entire blockchains. Blockchains provide a global sequence of events, but at substantial cost: state replication, consensus overhead, probabilistic finality, unpredictable fees, and multi-second (or multi-minute) confirmation delays. They are too slow, too heavy, and too expensive for applications that only require attestation, not full economic consensus. Conversely, centralized logs - like those used in certificate transparency or cloud audit systems - provide strong ordering but depend on the trustworthiness of a single operator or small cluster.

These systems do not compose cleanly. Integrating time from one source, randomness from another, and ordering from a third introduces trust fragmentation. Applications must reconcile mismatched assumptions, inconsistent formats, different latency characteristics, and incompatible operational models.

Tyche addresses these gaps by treating time, randomness, and ordering as facets of a single attestation problem. Each Tyche round produces a compact, multiparty-signed record that:

- attests the current time;
- deterministically orders gateway-submitted digests;
- and incorporates entropy contributions from independent providers.

This unified structure removes the need to combine multiple external systems and eliminates the overhead of maintaining global consensus. Tyche offers a consistent, verifiable point of reference that any system can use to timestamp events, determine ordering relative to other parties, or consume unbiased randomness - all without relying on centralized issuers or blockchain economics.

The motivation for Tyche is not to replace existing systems but to fill the gap between heavy-weight blockchain consensus and single-function trust primitives. Systems that need reliable, low-latency attestations today must either deploy complex infrastructure or depend on operators they cannot fully trust. Tyche provides a neutral, multi-party witness layer built for real-time applications, offering stronger guarantees than centralized services and dramatically better performance than consensus-driven networks.

As AI systems, multi-agent workflows, and cross-organization digital interactions become increasingly prominent, the need for a unified, real-time attestation layer will only grow. Tyche exists to meet that need.

3. Design Goals

Tyche is built around a small set of explicit, non-negotiable design goals. These principles shape every aspect of the architecture, ensuring that the system remains both practical and robust as it evolves.

3.1 Real-Time Attestation

Tyche must provide cryptographic attestations with predictable, sub-second finality. The system is designed to operate on a global round clock, producing signed outputs at fixed intervals. This enables applications to anchor events, determine ordering, and consume randomness within tight latency budgets. Real-time performance is a first-class requirement, not an optimization.

3.2 Statelessness

Tyche maintains no global state and no notion of transactions or ledgers. Each round is independent: operators sign what they observe, but do not carry forward state from previous rounds. Statelessness simplifies implementation, eliminates consensus complexity, and ensures that Tyche can scale without accumulating permanent data or maintaining historic logs. Verification happens client-side using compact proofs.

3.3 No Consensus

Tyche deliberately avoids consensus protocols such as Proof-of-Work, Proof-of-Stake, or BFT-based agreement. Consensus introduces latency, coordination overhead, and dependency on economic incentives. Instead, Tyche uses threshold signatures over round outputs to ensure authenticity and correctness. Operators do not need to agree on a full state - only on the round output structure and signature scheme.

3.4 Multi-Party Trust

The security model requires that multiple independent operators and entropy providers participate in each round. Tyche assumes that some participants may be malicious or offline but guarantees correctness as long as a threshold of operators behaves honestly. Randomness is secure as long as at least one entropy provider contributes honest entropy. This model avoids reliance on a single authority, providing strong bias-resistance and fault tolerance without consensus.

3.5 Deterministic Structure

Every Tyche round follows a deterministic protocol: fixed message formats, a canonical ordering of gateway-submitted digests, and a predefined method for aggregating entropy commitments. Determinism ensures that all honest operators derive the same round output, and that proofs remain verifiable long after the round has passed. Deterministic behavior is essential to maintain simplicity and avoid emergent inconsistencies that would require consensus to resolve.

3.6 Composability

Tyche is designed as a trust primitive, not an all-encompassing platform. Its outputs - timestamp, ordering digest, randomness - are intentionally compact and easy to embed inside other systems. Applications can incorporate Tyche proofs into workflows, logs, commit–reveal schemes, smart

contracts, identity systems, or regulatory processes. Composability allows Tyche to augment existing architectures rather than replace them.

3.7 Operational Practicality

Operators should be able to run Tyche nodes using standard infrastructure: cloud VMs, on-prem servers, or well-managed hosting. The protocol avoids exotic dependencies, consensus-heavy communication patterns, or complex state synchronization. Gateways and entropy providers have even lighter requirements, allowing broad participation. Operational simplicity encourages adoption and keeps the network reliable.

3.8 Extensibility Without Complexity

Tyche's core protocol is intentionally minimal. Optional extensions - such as VDF-delayed randomness, advanced fairness mechanisms, or enhanced ordering rules - can be layered on without modifying the core round structure. This modularity ensures that Tyche can evolve to meet emerging needs without sacrificing stability or simplicity.

Together, these design goals establish a foundation for a trust network that is fast, neutral, composable, and secure - without relying on heavyweight consensus or blockchain economics. They ensure that Tyche remains focused on delivering real-time attestations while minimizing operational burden and maximizing practical usefulness.

4. System Overview

Tyche is organized as a distributed witness network composed of three independent participant types - **operators**, **gateways**, and **entropy providers** - coordinated through a global, stateless round protocol. Each round produces a threshold-signed attestation that encodes: (1) the current time, (2) a deterministic ordering digest summarizing all client submissions, and (3) an aggregate entropy commitment that forms the basis of the randomness beacon. This attestation, along with its compact proofs, forms the core output of the system.

Tyche makes no attempt to maintain a global ledger, transaction history, or persistent state. Instead, each round acts as a neutral checkpoint that captures what the network observed during a fixed window of time. These checkpoints are independent of one another and do not accumulate into a chain or sequence beyond their monotonically increasing round numbers. This stateless design enables Tyche to operate with predictable sub-second latency and without the scaling overhead that typically accompanies consensus-driven systems.

4.1 Rounds and the Global Round Clock

Every Tyche operator maintains a synchronized clock and participates in rounds at the same fixed interval (e.g., 500 ms). The round boundary is a simple timer, not a consensus decision. At the start of each round, gateways stop accepting submissions for the previous interval and begin constructing a Merkle tree over all digests received during that interval. Operators then exchange the Merkle subtree roots provided by their gateways and aggregate them into a global tree. Once aggregated, operators threshold-sign a canonical round output that includes:

- the round number

- the wall-clock timestamp
- the global Merkle root representing ordered events
- the aggregated entropy commitment root
- the optional previous VDF output (if enabled)
- the threshold signature.

The round output is final once a threshold of operator signatures has been collected and verified.

4.2 Operators

Operators are the backbone of the network. Each operator runs a Tyche core node responsible for:

- maintaining accurate local time
- participating in subtree exchange with other operators
- verifying entropy commitments
- producing partial threshold signatures
- aggregating signed outputs

Operators do not see client data directly; they only receive Merkle subtree roots from their associated gateways. Their duties are purely structural: they validate commitments, sign the round output, and cross-check the contributions made by other operators. Because operators hold shares of the threshold key, they are intentionally limited in number and subject to governance oversight.

4.3 Gateways

Gateways interface directly with clients. A gateway receives digests (typically hashes of events or data), batches them during the current round window, and constructs a Merkle subtree. At the end of each round, the gateway hands the subtree root to its paired operator. After the round output is finalized, the gateway returns inclusion proofs to clients, enabling them to verify their own participation in the round. Gateways do not influence the global ordering beyond the contents of their own subtrees, and they hold no signing authority. Multiple gateways may operate under various pricing models or service guarantees without affecting the core protocol.

4.4 Entropy Providers

Entropy providers contribute randomness to each round through a simple commit–reveal scheme. During the commit phase, each provider submits a cryptographic hash of its entropy value. Operators aggregate these commitments into an ordered vector based on a statically signed registry. After the round is complete or after a VDF delay (if enabled), the providers reveal their entropy values, allowing clients or operators to verify correctness. Tyche requires only one honest entropy provider to ensure unbiased randomness.

4.5 Clients

Clients are any external systems or applications that rely on Tyche to attest time, ordering, or randomness. Clients do not interact with operators directly; instead, they communicate with gateways. After submitting data to a gateway, clients receive a compact inclusion proof once the round completes. Clients verify round outputs using the public threshold verification key and can

independently check entropy reveals, ordering commitments, and timestamps. No trust is placed in gateways beyond timely submission.

4.6 Final Attestations

The result of each round is a signed attestation that unifies three trust primitives:

1. Timestamp: an operator-signed time value with tight drift bounds
2. Deterministic ordering digest: the global Merkle root
3. Randomness commitment: derived from multi-party entropy

This attestation acts as a cryptographic witness for events observed during the round and can be verified indefinitely with no dependency on the live network.

5. Rounds and the Tyche Attestation Model

Tyche's core function is the production of periodic, cryptographically authenticated "round outputs" that serve as unbiased, externally verifiable attestations of what occurred during a specific interval of real time. Rounds provide the temporal rhythm of the network, anchoring client submissions, operator signatures, and entropy contributions into a unified structure. The round protocol is stateless: it does not record history or maintain a ledger, and each round is processed independently of all others.

5.1 Round Interval and Timing

All operators adhere to a global round interval - typically on the order of a few hundred milliseconds. Operators synchronize their clocks using standard time protocols (NTP or PTP) and enforce strict drift tolerances. A round begins when the local node's timer reaches the interval boundary; no consensus is required to coordinate start times. The absence of consensus enables Tyche to maintain consistent, low-latency performance without complex coordination.

5.2 Round Structure

Each round consists of three phases:

1. Collection Phase - Gateways collect digests from clients and prepare a Merkle subtree representing all submissions received during the interval.

2. Aggregation Phase - Operators exchange subtree roots with one another, assemble them into a global Merkle tree, verify entropy commitments, and compute the round output.

3. Signing Phase - Operators generate partial threshold signatures over the round output and exchange shares until a threshold is reached. The combined signature finalizes the round.

The entire process completes within a deterministic window, producing sub-second finality.

5.3 The Round Output

The round output is a canonical structure that includes:

- **round_number:** A monotonically increasing identifier
- **timestamp:** An operator-signed wall-clock time for the round

- **global_merkle_root:** A hash committing to all gateway-submitted digests in deterministic order
- **entropy_commitment_root:** An ordered vector-root of entropy commitments
- **vdf_output (optional):** The result of a verifiable delay function seeded from the previous round's randomness
- **threshold_signature:** A multi-operator signature proving that the round output is authentic

This signed output is the "witness artifact" for the round. All proofs and attestations derive from it.

5.4 Deterministic Ordering

Tyche does not attempt to define intra-round ordering across gateways in real time. Instead, ordering is defined deterministically:

- Digests within a gateway are ordered by local arrival sequence
- Gateways themselves occupy fixed positions in the global structure.

This produces a single deterministic Merkle root that can be computed by any honest operator. The ordering digest is sufficient for audit logs, compliance workflows, and multi-party sequencing applications.

5.5 Timestamp Semantics

The timestamp included in the round output represents the end of the round's collection window. It reflects the operators' view of real time within drift tolerances defined by the protocol. Because the timestamp is bound inside a threshold-signed structure, clients can rely on it as an independent attestation of the moment their digest was observed.

5.6 Entropy Commit–Reveal

Entropy providers submit commitments (hashes of entropy) during the collection phase, which are placed into a deterministic index based on a static provider registry. After the round, providers reveal preimages. Clients verify that:

H(e_i) == commitment_i

Because commitments are fixed before reveals occur, no provider can bias the final randomness result after seeing others' contributions. If a VDF is enabled, the entropy commitment root is also fed into a delay function to remove predictability.

5.7 Statelessness

Critically, Tyche does not store any accumulated state:

- operators discard per-round data once the round is finalized
- gateways retain only the proofs they choose to store
- clients keep the proofs relevant to them.

Verification requires only the round output, the public key for signature verification, and a Merkle proof for the relevant digest.

5.8 Finality and Verification

Once the threshold signature is produced, the round output is final and immutable. A client can independently verify:

1. the threshold signature
2. its inclusion in the Merkle tree;
3. the correctness of randomness reveals
4. the validity of the timestamp
5. the deterministic ordering relative to other digests

Tyche's verification requires no external connectivity and no interaction with any gateway or operator.

6. Tyche Services

Tyche provides three primary trust primitives - **timestamping**, **deterministic ordering**, and **randomness generation** - each derived from the same stateless, round-based attestation process. Rather than operating as separate systems with different trust assumptions, Tyche unifies these primitives into a single structure produced every round. This consolidation reduces operational overhead for implementers while providing stronger guarantees to applications that rely on a combination of these features.

6.1 Timestamping

Timestamping is the most immediately intuitive service Tyche provides: an attestation that a particular digest existed prior to a specific moment in time. Unlike centralized timestamp authorities, Tyche issues timestamps through multiparty threshold signatures, eliminating reliance on any single operator's clock.

6.1.1 Semantics

The timestamp included in a round reflects the end of the round's collection interval. Any digest included in that round's Merkle tree is therefore provably no newer than the corresponding timestamp.

6.1.2 Multiple Independent Clocks

Because all operators sign the same timestamp, Tyche's time attestations are:

- independent of any one operator's system clock
- bounded by protocol-defined drift tolerances
- authenticated by threshold signature
- globally consistent across the network

This design eliminates weak points common in centralized timestamp systems, such as clock drift, misconfiguration, or state compromise.

6.1.3 Verification

Clients verify timestamps by:

1. checking the threshold signature
2. verifying their own Merkle inclusion proof
3. confirming that the round timestamp is monotonic
4. validating operator clock drift rules

No interaction with the network is required after a round completes.

6.2 Deterministic Ordering

Ordering refers to Tyche's ability to prove the relative sequence of events submitted during the same round, across different gateways, in a deterministic and verifiable manner. Tyche does not attempt to serialize events in real time; instead, it defines a **canonical ordering function** that produces the same deterministic Merkle root across all honest operators.

6.2.1 Intra-Gateway Ordering

Within a gateway:

- Each gateway builds one Merkle subtree per round over **an application-defined sequence** of digests.
- Tyche **does not define or guarantee** the semantics of that sequence (arrival order, queue order, etc.)
- Tyche only guarantees that:
    - the subtree root is included in the global tree in the gateway's fixed slot
    - the client's proof matches the subtree the gateway actually committed

6.2.2 Inter-Gateway Ordering

Gateways themselves occupy predetermined positions in the global Merkle tree. The ordering is derived from a canonical, static configuration known to all operators. This prevents gateways from influencing their relative position or gaining ordering advantage.

6.2.3 Ordering Digest

The **global_merkle_root** in the round output commits to:

gateway_0 subtree

gateway_1 subtree

...

gateway_N subtree

Every operator independently reconstructs the same tree. Any ordering misbehavior by a gateway is detectable by clients who examine their own subtree.

6.2.4 Use Cases

Deterministic ordering is critical for:

- fair queueing
- commit–reveal protocols
- audit logs
- multi-party workflows
- sequencing events across organizations
- prevention of replay or insertion attacks

Tyche provides ordering without consensus, state, or economic incentives.

6.3 Randomness Beacon

Tyche's randomness beacon produces unpredictable, bias-resistant randomness in each round via contributions from independent entropy providers. This randomness is embedded into the signed round output, making it cryptographically linked to time and ordering.

6.3.1 Commit–Reveal Model

Entropy providers submit commitments (hashes of random values) during the round. These commitments are aggregated in deterministic index order based on a static registry. After the round, providers reveal the preimages. Any mismatch invalidates the provider's contribution for that round.

Because commitments are fixed before reveals occur, no provider can adaptively bias randomness.

6.3.2 Bias Resistance

The randomness is secure under a simple assumption:

> **If at least one entropy provider is honest, the randomness is unbiased.**

Tyche does not require all entropy providers to reveal; it requires only that one honest provider participates fully.

6.3.3 Optional VDF Strengthening

If applications require unpredictability in addition to bias resistance, Tyche can feed the entropy commitment root into a Verifiable Delay Function:

- commitments seed the VDF input
- VDF output is unpredictable until delay completes
- final randomness incorporates both

This optional layer allows Tyche to serve use cases that require stronger cryptographic guarantees, such as lotteries, challenge beacons, and adversarial coordination.

6.3.4 Use Cases

Randomness from Tyche is suitable for:

- fair draws
- sealed-bid selection
- game fairness
- randomized assignment
- MPC protocols
- AI sampling reproducibility
- scientific experiments
- regulatory audits.

6.3.5 Integration with Other Tyche Services

Randomness, timestamping, and ordering are all embedded in the same proof. This eliminates cross-system reconciliation issues and allows applications to anchor their entire fairness context to a single attestation.

Tyche's three services - timestamping, ordering, and randomness - are not independent modules. They are three simultaneous outputs of the same round, produced deterministically, signed collectively, and verifiable in isolation or combination. This unification is one of Tyche's core strengths and a primary differentiator from existing trust primitives.

7. Roles in the Network

Tyche divides responsibilities across three participant types - operators, gateways, and entropy providers - with clients consuming the resulting attestations. This separation of roles is intentional: it minimizes trust in any single entity, reduces operational load on the core nodes, and creates a clear boundary between the network's cryptographic responsibilities and its application-facing services.

7.1 Operators

Operators are the core of the Tyche network. Their role is to collectively produce threshold-signed round outputs that attest to time, ordering, and randomness. Operators are intentionally limited in number, and their onboarding is governed by a static or governance-controlled configuration.

Responsibilities

- Maintain accurate system clocks within drift bounds
- Participate in every round by exchanging Merkle subtree roots
- Verify the integrity of entropy commitments
- Construct the canonical round output
- Generate partial threshold signatures
- Validate signatures from other operators
- Detect missing, malformed, or inconsistent contributions

Trust Model

An operator is trusted only to contribute correctly to threshold signing and to maintain reasonable clock accuracy. The network tolerates malicious or offline operators as long as a sufficient threshold behaves honestly.

Operational Requirements

Operators require:

- stable network connectivity
- a reliable time source (NTP, PTP, GPS-disciplined clock)
- a secure environment to store their threshold key share
- robust monitoring and alerting
- the ability to exchange messages within round intervals.

Operators do **not** see client data or raw digests; they only handle aggregated subtree roots and entropy commitments.

7.2 Gateways

Gateways are service nodes that interact directly with clients. They allow applications to submit digests to be included in the next Tyche round and deliver inclusion proofs once the round completes. Gateways are the "API layer" of the network and can be operated by anyone, including operators, businesses, or independent developers.

Responsibilities

- Receive client digests and metadata
- Sequence digests in arrival order
- Build a Merkle subtree each round
- Deliver the subtree root to an operator
- Provide Merkle inclusion proofs to clients after finalization.

Gateways **do not** participate in threshold signing and hold **no** cryptographic authority.

Trust Model

Clients place minimal trust in gateways:

- Gateways cannot forge inclusion in a round - proofs are verified independently
- Gateways cannot alter time or ordering outside their own subtree
- Misbehavior by a gateway is provable due to the global Merkle structure.

If a gateway refuses to submit a client's digest, the client can switch providers without affecting the protocol.

Operational Requirements

Gateways require:

- sufficient bandwidth for client load
- ability to construct Merkle trees quickly

- a persistent connection to an operator
- storage of proofs if they provide archival services

Gateways may differentiate themselves through uptime, pricing, SLAs, compliance features, or additional verification services.

7.3 Entropy Providers

Entropy providers contribute randomness to every round. They strengthen Tyche's randomness beacon by offering independent entropy values that are aggregated into the round's entropy commitment root.

Responsibilities

- Generate high-quality random values locally
- Submit commitments (hashes) before or during the round
- Reveal preimages after the round
- Maintain availability long enough for reveal windows

Trust Model

Entropy providers are not trusted individually. The security assumption is:

- At least one entropy provider must be honest for the randomness to be unbiased
- Colluding providers cannot bias randomness unless *all* of them collude

Operational Requirements

Entropy providers require:

- a reproducible random generation method
- ability to commit and reveal within timing windows
- registration in the static provider list
- optional integration with hardware RNGs or QRNGs.

Entropy providers do not handle client data, do not influence ordering, and do not interact with gateways.

7.4 Clients

Clients are systems or applications that consume Tyche's services. They interact only with gateways and independently verify all results. Clients operate with zero trust beyond the correctness of the threshold signature and the public parameters of the protocol.

Typical Client Workflows

- **Timestamping:** submit a digest → receive proof → verify offline
- **Ordering:** compare Merkle proof positions or subtree paths
- **Randomness:** retrieve the round output → verify entropy reveals
- **Audit Logs:** anchor log digests each round for verifiable integrity
- **Workflows:** use ordering and timestamps to demonstrate process correctness

Trust Model

Clients trust:

- the threshold signature verification key
- the deterministic round structure
- the static gateway ordering
- the entropy provider registry.

They do not trust:

- any individual operator
- any individual gateway
- any entropy provider
- any external system clock.

Tyche's division of responsibilities ensures that no single role can influence the core security properties. Operators produce signatures, gateways batch digests, entropy providers contribute randomness, and clients verify everything independently. This structure keeps Tyche lean, resilient, and easy to integrate into existing systems.

8. Security Properties

Tyche's security model is intentionally conservative and built on minimal assumptions. By avoiding consensus, global state, and economic incentive mechanisms, Tyche's guarantees are derived from cryptographic structure rather than game theory or probabilistic finality. This section defines the core security properties Tyche provides for timestamping, ordering, and randomness - along with the threat assumptions under which they remain valid.

8.1 Correctness

Correctness means that an honest client can verify that

- the round output was produced by a threshold of authorized operators
- the timestamp corresponds to a legitimate round boundary
- all valid gateway contributions are included in the Merkle tree
- entropy commitments match their reveals
- the deterministic ordering function was applied faithfully

Correctness is guaranteed as long as the threshold of honest operators is met and the canonical round format is adhered to.

8.2 Authenticity

Authenticity ensures that a round output:

- cannot be forged by a single operator
- cannot be forged by any minority coalition below the signing threshold
- is cryptographically bound to the public verification key
- includes only data that was part of the round

Threshold signatures provide a single, compact attestation that is valid indefinitely.

8.3 Bias Resistance

Bias resistance applies to the randomness beacon. Tyche guarantees that randomness cannot be biased after the commit phase begins.

Security assumption:

At least one entropy provider contributes an honest, unpredictable entropy value.

Even if all other providers collude, the final combined randomness output remains unpredictable and unbiased.

Optional use of a VDF further removes predictability by delaying the ability of any party to compute the final beacon value ahead of time.

8.3 Ordering Consistency

Ordering consistency guarantees that:

- all honest operators compute the same global Merkle root
- the tree cannot be reordered by any operator or gateway
- each client can independently verify its subtree position
- gateway misbehavior (e.g., reordering or omission) is detectable

Operators' deterministic combination of gateway subtrees ensures that the ordering digest is globally consistent.

8.4 Censorship Detection

Tyche cannot prevent gateways from refusing to include a client's digest, but censorship is *detectable* because:

- a missing inclusion proof indicates non-submission
- clients can instantly switch to another gateway
- gateway position in the Merkle tree is fixed and known
- operators cannot retroactively insert omitted events.

As with Certificate Transparency, the protocol provides transparency, not censorship resistance.

8.5 Timestamp Integrity

Timestamp integrity ensures that the time value in each round

- is signed by a threshold of operators
- reflects the actual round boundary within bounded drift
- is not modifiable by a single operator
- is consistent across the network

Tyche provides cryptographic assurance that "this event existed at or before this time," forming the basis for verifiable logs, compliance proofs, and audit chains.

8.6 Fault Tolerance and Redundancy

Tyche's threshold signing scheme tolerates:

- offline operators (up to the threshold limit)
- malicious operators attempting to produce incorrect outputs
- faulty or malicious entropy providers
- misbehaving or overloaded gateways.

Even if a gateway is compromised, its actions are confined to its own subtree. Even if an entropy provider withholds reveals, randomness remains secure as long as at least one honest provider participates. Operators continually validate and cross-check each other's contributions.

8.8 Non-Repudiation

Because round outputs are threshold-signed, they provide strong non-repudiation: operators cannot later deny having participated in a specific round. Clients can store round outputs indefinitely, providing immutable evidence of attested events.

8.9 Minimal Trust Surface

Tyche's trust model is intentionally narrow. Clients trust:

- the threshold signature verification key
- the static registry of gateways and entropy providers
- the deterministic rules of the round format.

They do **not** trust:

- gateway uptime or honesty
- operator clocks individually
- entropy providers individually
- the network itself.

Verification is entirely client-side, with no need to query the network or rely on any operator after round completion.

Tyche's security properties aim to strike a balance between the rigidity of blockchains and the fragility of centralized timestamping services. By combining threshold signatures, deterministic ordering, and multi-party randomness, Tyche provides strong guarantees while avoiding the performance and complexity tradeoffs associated with consensus systems.

9. Governance & Trust Model

Tyche is designed as a neutral, cryptographic infrastructure layer rather than a permissionless or economically-incentivized ecosystem. As such, its governance model is deliberately simple and rooted in transparent membership, static configuration, and operational accountability. Governance exists to ensure diversity, independence, and reliability of operators and entropy providers - not to mediate state, allocate rewards, or manage a consensus process.

9.1 Governance Philosophy

Tyche operates under three principles:

1. **Minimalism:**
   Only decisions affecting operator lists, entropy provider registries, and protocol upgrades require governance. The system avoids governance over daily operation, economic incentives, or application-layer rules.

2. **Neutrality:**
   Participation should be open to organizations with strong operational history, but no single type of entity (e.g., cloud providers, universities, independent operators) should dominate.

3. **Cryptographic Authority Over Political Authority:**
   All trust derives from threshold keys, static registries, and cryptographic proofs. Governance is used only to maintain these configurations.

9.2 Operator Selection and Responsibilities

Operators are the most sensitive role because they hold shares of the threshold signing key. Their membership must be curated carefully.

9.2.1 Eligibility Criteria

Operators may include:

- Universities
- independent research labs
- nonprofit infrastructure organizations
- commercial infrastructure providers
- security-focused engineering teams.

The ideal operator set is diverse across:

- geography
- jurisdiction
- organizational type
- funding model
- operational environment.

9.2.2 Admission

New operators are admitted through a governance process that updates:

- the operator registry
- the threshold key shares
- the round verification public key.

This typically involves a key ceremony requiring participation from existing operators.

9.2.3 Removal

Operators may be removed for:

- repeated downtime
- malicious contributions
- refusal to participate in rounds
- security breaches
- key compromise.

Removal also requires a threshold key resharing event.

9.2.4 Transparency

Operator lists, public keys, and uptime metrics must be public for accountability.

9.3 Gateway Ecosystem

Gateways do **not** require governance approval. Anyone may run a gateway as long as they follow the protocol and register their identity with the gateway ordering registry.

9.3.1 Freedom to Compete

Gateways may differentiate on:

- Performance
- Availability
- compliance features
- pricing
- SLAs
- client libraries
- API support

This creates a competitive ecosystem without affecting protocol security.

9.3.2 Static Ordering Registry

Although gateways are permissionless, their global ordering position in Tyche's Merkle tree is determined by a static registry. This registry must be publicly visible and cryptographically signed so that operators can maintain deterministic ordering.

9.4 Entropy Provider Governance

Entropy providers are essential to Tyche's randomness beacon. Their membership must be transparent and verifiable.

9.4.1 Registration

An entropy provider must:

- publish a public identity key
- register a deterministic index position

- commit to providing entropy every round
- meet uptime requirements.

9.4.2 Removal

A provider may be removed for

- failing to reveal commitments consistently
- submitting malformed commitments
- participating in observable collusion
- key compromise.

9.4.3 Independence

Providers should be selected to minimize correlated risk. Ideal participants include:

- academic institutions
- cloud operators
- hardware RNG manufacturers
- quantum RNG labs
- nonprofit infrastructure groups.

Entropy diversity is a core security requirement: only one honest provider is needed to ensure unbiased randomness.

9.5 Static Registries and Signed Configuration

Tyche relies on static, signed registries that define:

- operator public keys
- gateway ordering
- entropy provider ordering
- protocol constants (round interval, drift tolerance, etc.).

These registries are controlled through governance and updated infrequently. Clients download the latest signed configuration to verify round outputs.

9.6 Protocol Upgrades

Tyche supports upgrades through versioned round formats and threshold-key migration.

9.6.1 Backward Compatibility

Whenever possible, upgrades:

- introduce optional fields
- avoid breaking existing proofs
- allow old and new nodes to coexist for a transition period

9.6.2 Threshold-Governed Activation

Protocol upgrades activate only when:

- a threshold of operators has updated
- new signing keys or registry files have been published
- clients can verify compatibility.

There is no need for soft forks, hard forks, or consensus votes.

9.7 No Token, No Economic Governance

Tyche intentionally excludes:

- Tokens
- Staking
- Slashing
- economic penalties
- on-chain governance
- market-driven voting.

Trust derives from threshold cryptography and institutional diversity - not financial mechanisms.

Tyche's governance model is intentionally boring. This is a strength. By reducing governance to the maintenance of static registries and threshold keys, Tyche avoids the instability, politics, and attack surfaces common in token-governed systems. The result is a stable, predictable, and secure foundation suitable for long-term public infrastructure.

10. Use Cases

Tyche provides three fundamental trust primitives - timestamping, deterministic ordering, and unbiased randomness - that can be embedded into a wide range of applications. These primitives are composable, verifiable offline, and do not require interaction with the live network after a round completes. The following use cases represent practical areas where Tyche provides immediate value without the overhead of consensus systems or blockchains.

10.1 Audit and Compliance Anchoring

Modern audit logs are vulnerable to tampering, backdating, and deletion. Tyche provides a cryptographic witness that an event existed at a specific time and in a specific position relative to others.

Applications include:

- regulatory audit trails (finance, healthcare, energy)
- chain-of-custody systems
- IT security logs
- database write ahead logs (WALs)
- incident response and forensics.

Tyche's proofs can be embedded directly into logs or SIEM systems, giving auditors offline verifiability without needing periodic notarization.

10.2 Multi-Party Workflow Integrity

Coordination across organizations often requires trusted sequencing

- supply chain events
- contract-signing workflows
- cross-organizational APIs
- federated transactions
- automated dispute resolution
- provenance attestations.

Tyche enables each step to be:

- timestamped
- ordered
- verifiably anchored to a neutral witness network.

This eliminates arguments about who acted first and prevents fabricated histories.

10.3Fair Ordering for Competitive Systems

Applications where ordering matters benefit from Tyche's deterministic ordering digest:

- sealed-bid auctions (ordering commitments before reveals)
- RFP submission windows
- queue fairness and rate limiting
- fair access allocation
- identity minting
- priority determination for public services.

Unlike blockchains, Tyche provides deterministic ordering without congestion fees or competition for block space.

10.4Public Randomness for Fairness and Selection

Tyche's randomness beacon can be used in

- lotteries and raffles
- randomized audits
- scientific experiment randomization
- tournament seeding
- challenge generation
- sampling in AI systems
- cryptographic protocols (commit–reveal, MPC, zk systems).

Bias-resistance is ensured as long as one entropy provider is honest.

10.5 AI and ML Integrity

As AI systems automate decisions, third-party attestation becomes essential.

Tyche can anchor:

- inference logs
- model outputs
- ranked-choice results
- LLM decision paths
- agent actions in multi-agent environments.

This enables reproducibility, auditability, and dispute resolution in high-stakes AI workflows.

10.5 Digital Identity and Verifiable Credentials

Tyche strengthens DID/VC environments by offering:

- timestamps for credential issuance or revocation
- ordering for revocation logs
- randomness for cryptographic protocols involving selective disclosure or BBS+ signatures.

VC ecosystems benefit from a neutral attestation layer independent of issuers or verifiers.

10.6 Proof-of-Existence and Intellectual Property

Applications requiring evidence of prior existence of:

- Documents
- code repositories
- research data
- digital art
- legal agreements
- notarized records

Tyche provides the benefits of OpenTimestamps but with lower latency and unified randomness/ordering context.

10.7 Gaming and Interactive Protocols

Real-time fairness is essential in competitive or multiplayer systems:

- Matchmaking
- loot generation
- procedural content
- verifiable game state transitions
- turn-order fairness
- anti-cheating randomness

Tyche offers millisecond-scale attestations usable by both on-chain and off-chain games.

10.8 Cross-Chain or Inter-System Coordination

Blockchains rely on:

- randomness (e.g., leader selection, validator rotation)
- timestamps
- ordering proofs (for bridges and external logs).

Tyche can serve as a neutral, verifiable external oracle for these purposes without requiring on-chain state replication.

10.9 Scientific Integrity and Research Reproducibility

Many scientific processes require:

- randomization sequences
- experiment ordering
- timestamped lab events
- reproducible sampling.

Tyche provides these without reliance on a single institution or lab.

10.10 Public Policy, Procurement, and Government Integrity

Government workflows benefit from neutral witnesses:

- RFP submissions
- grant selection
- voting sequence verification
- publication timestamping
- transparent contract awarding

Tyche helps eliminate “quiet manipulation” of ordering or retroactive edits.

These use cases illustrate Tyche’s practical relevance. Tyche is not a speculative blockchain platform or a niche security tool - it is general-purpose infrastructure for systems that need trustworthy time, ordering, and randomness without the burden of consensus.

11. Call for Collaboration

Tyche is at an early stage. What exists today is a coherent architecture, a clear problem statement, and a path toward a practical, multiparty witness network that unifies timestamping, deterministic ordering, and randomness. What does *not* exist yet is an implementation, a working group, or a diverse community of operators. For Tyche to become a meaningful public-good infrastructure, it requires collaboration from organizations and individuals who care about cryptographic integrity, transparency, fairness, and verifiable systems.

We are looking for contributors who can help refine the specification, validate the security model, and build the initial reference implementation. Researchers with experience in distributed systems, threshold cryptography, VDFs, or verifiable logging can provide critical feedback. Infrastructure operators, including universities, non-profits, and engineering teams, can help run

early testnets and shape operational best practices. Developers interested in building gateways, client tools, or integrations can push the ecosystem forward.

Tyche is not a company, a token, or a commercial product. It is an attempt to design a simple, durable, and neutral trust primitive that can be used by anyone. The project needs collaboration from people and institutions who believe that public infrastructure should be verifiable, fast, and free from central points of control. If the problems described in this document resonate with you - if you believe that the internet needs a reliable way to attest time, ordering, and randomness - your participation is welcome.

Plans for a working group, community discussions, and a reference implementation will be published alongside the specification draft. Feedback, critique, and contributions are encouraged. No single organization, including the authors, should ultimately control Tyche. The goal is a network that is collectively operated, openly specified, and broadly useful.

If you are interested in participating in the next steps - reviewing the spec, contributing code, operating early nodes, running a gateway, or serving as an entropy provider - we invite you to reach out and join the conversation. The future of digital trust depends on collaboration, and Tyche is designed to be built together.