Tangle Network

The Operating Layer for Autonomous Work

Protocol Specification v1.0

Tangle Foundation

January 2025

Abstract

Tangle Network is a decentralized protocol for coordinating computational services with cryptoeconomic accountability. This document describes the protocol's architecture, economic mechanisms, and security model. Tangle enables developers to define service templates (blueprints), operators to provide compute with staked collateral, and customers to consume services with cryptographic guarantees. The protocol uses O(1) algorithms for staking, slashing, and reward distribution, enabling unlimited scalability. Economic security emerges from aligned incentives: operators stake assets that can be destroyed if they misbehave, making honest behavior more profitable than cheating.

Contents

1 Introduction

The deployment of increasingly capable AI systems demands infrastructure that can scale with their requirements while maintaining accountability. Current infrastructure options—centralized cloud providers or fragmented decentralized alternatives—impose constraints that limit both innovation and access. Centralized systems concentrate power in few hands, creating single points of failure and extracting value from a position of control. Decentralized alternatives lack the coordination mechanisms necessary for production workloads.

Tangle addresses this gap by providing a protocol layer where computational services operate with cryptographic accountability and economic security. The protocol coordinates three participant types: developers who define service templates, operators who provide compute, and customers who consume services. Value flows through the system via service fees and inflation rewards, distributed according to contribution rather than position.

1.1 The Infrastructure Question

AI agents capable of autonomous action require infrastructure that matches their capabilities. An agent managing a portfolio needs compute that cannot be arbitrarily terminated. An agent processing sensitive data needs isolation guarantees beyond contractual promises. An agent making consequential decisions needs audit trails that cannot be falsified.

Traditional cloud infrastructure provides compute but not accountability. Terms of service change. Access can be revoked. Providers are few, and their market power grows as dependence deepens. For infrastructure underpinning significant economic activity, this concentration creates systemic risk.

Tangle provides an alternative where accountability is cryptographic rather than contractual. Operators stake assets that can be destroyed if they violate service terms. Customers pay for services with cryptographic proof of delivery. The protocol coordinates without controlling, enabling a competitive market of independent operators rather than dependence on a few large providers.

1.2 Design Philosophy

Seven principles guide Tangle's design.

Democratized Ownership. The value generated by AI infrastructure should accrue broadly, not concentrate in platform owners. Operators, developers, delegators, and users all capture value proportional to their contributions.

Permissionless Participation. Anyone can become an operator by staking the minimum bond. Anyone can deploy a blueprint. Anyone can use services. No committee approves applications. No foundation decides who may participate.

Developer Expressiveness. The protocol provides primitives that developers compose into applications. Blueprints define pricing models, membership rules, slashing conditions, and verification mechanisms through hooks.

Economic Security. Trust emerges from aligned incentives. Operators deposit assets that can be destroyed if they misbehave. The cost of cheating exceeds any benefit from cheating.

Scalable Efficiency. O(1) algorithms handle unlimited participants without iteration. Share-based accounting tracks delegation through exchange rates. Accumulated-per-share rewards distribute income through single accumulators.

Isolation by Default. AI agents execute in sandboxes with explicit permissions. Containers separate processes. Resource limits prevent exhaustion attacks.

Navigating Tensions. When principles conflict, mechanism design finds balance. Permissionless participation combined with market selection allows anyone to enter while quality emerges through competition.

2 Blueprints and Services

The blueprint system forms the foundation of Tangle's service architecture. Blueprints define what services do; services are running instances with assigned operators.

2.1 What Is a Blueprint?

A blueprint is a reusable template defining a service type. Developers create blueprints to specify computational tasks, pricing models, operator requirements, and optionally custom logic through hooks.

Formally, a blueprint comprises:

- Metadata for discovery (name, description, author)
- Configuration for pricing and membership
- Job definitions specifying tasks operators execute
- Validation schemas ensuring well-formed requests
- Optional custom service manager implementing hooks

Blueprints are identified by unique numeric IDs assigned at creation. Once an operator registers, the protocol locks blueprint metadata, preventing developers from modifying terms after operators have committed.

2.2 What Is a Service Instance?

A service is a running instance of a blueprint with assigned operators. Where blueprints define what a service type does, service instances represent actual deployments.

A service comprises:

- Reference to its blueprint
- Set of participating operators with committed exposures
- Configuration parameters validated against the schema
- Time-to-live (TTL) defining service duration
- Payment configuration specifying token and amount

Services move through defined states: pending (awaiting operator approvals), active (operational), and terminated (completed lifecycle).

2.3 Service Lifecycle

The lifecycle proceeds through request, approval, activation, operation, and termination.

A customer initiates by submitting a request specifying the blueprint, desired operators, configuration, payment, TTL, permitted callers, and security requirements. The protocol validates parameters and notifies operators.

Each operator must respond. Approval requires committing an exposure (fraction of stake backing the service). If any operator rejects, the service fails. When all approve, the service activates.

Upon activation, operators provide the service. Customers submit jobs; operators execute and submit results. The payment system distributes funds according to the configured model.

Services terminate through customer request, TTL expiration, or failure to maintain minimum operator count. Remaining escrow refunds to the customer.

2.4 Pricing Models

Three pricing models accommodate different service types:

Pay-once collects a single upfront payment, distributed when operators approve. This suits one-time computations.

Subscription collects recurring payments from customer-funded escrow. This suits ongoing services.

Event-driven collects payment per job execution. This suits variable workloads.

2.5 Membership Models

Fixed membership locks the operator set at service creation. Operators cannot join or leave once activated.

Dynamic membership permits operators to join after activation and leave subject to exit queue constraints. This enables long-running services to adapt.

3 Operators and Staking

Operators are staked entities that run services and earn rewards. The staking system provides economic security backing service guarantees.

3.1 Operator Registration

An entity becomes an operator by depositing a bond meeting minimum stake requirements. Registration creates metadata tracking self-stake, delegation count, and status.

Operators then register for specific blueprints, providing an ECDSA public key for gossip identity and an RPC endpoint for communication. The protocol validates that the blueprint is active, minimum stakes are met, and schema validation passes.

3.2 Delegation Modes

Operators configure delegation mode:

- **Disabled**: Only self-stake accepted
- **Whitelist**: Only approved addresses may delegate
- **Open**: Any address may delegate

3.3 Exposure Selection

When operators approve service requests, they commit an exposure in basis points (0-10000). This determines what fraction of stake backs the service.

Exposure has two consequences: it determines reward share (higher exposure = more rewards) and bounds slashing (operators lose at most their committed percentage).

This design enables participation in multiple services with bounded total risk.

3.4 O(1) Share-Based Accounting

The staking system uses share-based accounting for O(1) operations. Each operator maintains a reward pool tracking total shares and total assets.

When delegating amount x to a pool with state (T_s, T_a), the delegator receives shares:

$$h = x \cdot \frac{T_s + V}{T_a + V}$$

where V is a virtual offset preventing first-depositor attacks.

The delegation's value is $v = h \cdot (T_a/T_s)$. When assets change through rewards or slashing, this value changes proportionally without per-delegator updates.

When slashing occurs, only total assets decrease. Shares remain constant. The exchange rate decreases, affecting all delegators proportionally. No iteration is required.

4 The Service Marketplace

The service marketplace bridges supply (operators) with demand (customers) through request-for-quote pricing.

4.1 Why Request-for-Quote

Service pricing mechanisms face design choices. Fixed pricing cannot adapt to changing costs. Order books work poorly for heterogeneous services. AMMs struggle with operator differentiation.

Request-for-quote asks operators to provide binding prices for specific requests. Customers collect quotes, evaluate holistically, and select. RFQ accommodates operator heterogeneity naturally, requires no on-chain state, and provides instant finality.

4.2 The RFQ Flow

Operators register with ECDSA keys and RPC endpoints. They run software that listens for quote requests and returns signed quotes containing:

- Blueprint ID and service duration (TTL)
- Total cost and payment token
- Timestamp and expiry
- Security commitments (exposure per asset)

Customers collect quotes, verify terms, and submit all quotes in a single transaction. The protocol verifies signatures, checks expiry and replay protection, validates registrations, and confirms payment.

4.3 Quote Security

EIP-712 signatures bind quotes to specific chains and contracts. Each quote includes a unique timestamp; the protocol tracks used quotes. Quotes expire after their specified time. Maximum quote age (default one hour) ensures quotes reflect current conditions.

4.4 MEV Considerations

Quote sniping fails because quotes specify security commitments only registered operators can provide. Price manipulation is mitigated by competition. Sandwich attacks have limited applicability since RFQ transactions do not interact with liquidity pools.

5 Fee Distribution and Rewards

Value flows through Tangle via service fees and inflation rewards.

5.1 Service Fee Splits

When customers pay, the payment splits across recipients. Default allocation:

Recipient	Share
Developer	20%
Protocol Treasury	20%
Operators	40%
Delegators	20%

These percentages are governance-controlled. The delegator share captures rounding dust.

5.2 Operator Revenue

The operator allocation distributes weighted by committed exposure:

$$\text{operatorShare}_i = \frac{\text{operatorAmount} \times e_i}{\sum_j e_j}$$

Operators who commit more stake earn more from that service.

5.3 Delegator Rewards

Lock duration multipliers reward longer commitments:

Lock Duration	Multiplier
None	1.0x
One Month	1.1x
Two Months	1.2x
Three Months	1.3x
Six Months	1.6x

A delegator's score is amount × multiplier. Rewards distribute proportionally using accumulated-per-share accounting for O(1) efficiency.

5.4 Developer Rewards

Developers earn from service fees and inflation rewards through scoring:

$$\begin{aligned}
\text{blueprintScore} &= \text{blueprintCount} \times 500 \\
\text{serviceScore} &= \text{serviceCount} \times 1000 \\
\text{jobScore} &= \text{jobCount} \times 100 \\
\text{feeScore} &= \sqrt{\text{totalFees}/10^{18}} \times 10^9
\end{aligned}$$

The square root on fees prevents whale dominance.

6 Inflation and Epoch Distribution

Inflation rewards provide income while service demand develops, encouraging early participation.

6.1 The Pre-Funded Model

The InflationPool uses a pre-funded model. The pool cannot mint tokens; it distributes tokens it holds. Governance funds the pool with yearly allocations.

Each epoch's budget:

$$\text{epochBudget} = \frac{\text{poolBalance}}{\text{epochsRemaining}}$$

This smooths distribution over the funding period.

6.2 Distribution Weights

Default allocation:

Category	Weight
Stakers (delegators)	40%
Operators (performance)	25%
Developers (merit)	25%
Customers (usage)	10%

Governance can adjust weights to steer network growth.

6.3 Operator Scoring

Operator inflation rewards use performance metrics:

$$\text{successRate} = \frac{\text{successfulJobs} \times 10000}{\text{totalJobs}}$$

$$\text{stakeWeight} = \frac{\text{totalStake}}{10^9}$$

$$\text{jobScore} = \frac{\text{jobs} \times \text{successRate} \times \text{stakeWeight}}{10000}$$

Stake weight is linear to defeat Sybil attacks.

7 Slashing and Accountability

Slashing provides consequences for misbehavior, making economic security meaningful.

7.1 Developer-Defined Conditions

The protocol provides slashing mechanism; developers define when to use it. Each service type has different requirements the protocol cannot anticipate.

Developers implement conditions through hooks: `querySlashingOrigin` specifies who may propose slashes; `onSlash` executes when slashing finalizes.

7.2 The Dispute Window

Slash proposals do not execute immediately. A dispute window (default 7 days) provides operators opportunity to contest.

The lifecycle:

1. **Proposal**: Evidence submitted, pending proposal created
2. **Dispute**: Operator may submit counter-evidence; administrators may cancel
3. **Execution**: After window expires, any address may execute undisputed slashes

7.3 Proportional Slashing

Slashing is proportional to committed exposure:

$$\text{effectiveSlashBps} = \frac{\text{slashBps} \times \text{exposureBps}}{10000}$$

An operator with 10% exposure faces at most 10% of stake at risk.

7.4 Withdrawal Protection

Pending slashes block withdrawals. Delegators cannot withdraw while slashes are pending against their operator.

8 Governance

Token holders control protocol evolution through on-chain governance.

8.1 Governance Philosophy

Tangle prioritizes **safety over liveness**. Proposals that fail do less damage than malicious proposals that pass. Thresholds are high, quorum requirements ensure participation, and timelocks provide escape hatches.

Stake-weighted voting gives those with more at stake more say. This reflects that economic exposure confers governance rights.

8.2 Governance Mechanism

On-chain governance uses OpenZeppelin Governor with ERC20Votes:

Parameter	Default
Proposal threshold	1% of supply
Voting delay	2 days
Voting period	7 days
Quorum	10% of supply
Timelock	3 days (7 for upgrades)

8.3 Attack Resistance

Governance capture: High thresholds, long voting periods, and timelocks enable defensive mobilization.

Flash loan attacks: Vote checkpointing determines power at snapshot, not current holdings.

Voter apathy: Quorum requirements and delegation provide defense.

8.4 Vote Delegation

Token holders may delegate voting power. Delegation enables passive holders to participate through trusted representatives.

8.5 Emergency Mechanisms

Pause functionality allows PAUSER_ROLE to halt operations.

Emergency withdrawal enables DEFAULT_ADMIN_ROLE to withdraw funds to safety.

Guardian multisig may hold emergency roles during early protocol period.

8.6 Upgrade Mechanism

Contract upgrades use UUPS pattern. Upgrades follow standard governance with the longest timelock (7 days).

9 Security Model

9.1 Threat Model

Three adversary classes:

Rational adversaries attack if expected value exceeds expected cost. Defense is economic: ensure slashing cost exceeds potential gain.

Byzantine adversaries may behave irrationally. Defense bounds damage through stake requirements, exposure limits, and dispute windows.

Colluding adversaries coordinate to overcome individual defenses. Defense requires structural mechanisms: randomized selection, stake distribution requirements.

9.2 Economic Security Analysis

The **cost of corruption** (CoC) is the minimum stake an adversary must control to corrupt a service.

The **profit from corruption** (PfC) is the maximum value extractable.

The **security ratio** is CoC/PfC. Ratios above 1 make honest behavior more profitable under perfect detection. We recommend minimum ratios of 1.5x, with 2x+ for high-value services.

9.3 Formal Security Properties

Withdrawal Safety: Operators with pending slashes cannot reduce stake below slash amount.

Proportional Slashing: When $x\%$ slash executes, all delegators lose exactly $x\%$ of delegation value.

Dust Conservation: Payment splits distribute exactly the input amount.

Temporal Separation: Stake cannot participate in services until locked for `minStakeEpochs`.

Bounded Governance: Proposals cannot execute until timelock elapses.

9.4 Security Recommendations

1. Size stake requirements to exceed maximum corruption profit
2. Implement detection mechanisms appropriate to threat model
3. Use multi-operator services for high-value applications
4. Monitor economic conditions and pause if security margins decline
5. Participate actively in governance

10 Comparison to Existing Protocols

10.1 The Restaking Landscape

Restaking protocols share a premise: staked assets can secure additional systems beyond their primary use.

EigenLayer enables ETH stakers to opt into additional slashing for yield.

Symbiotic accepts diverse collateral with modular design.

Karak uses risk-adjusted collateral matching.

10.2 Differentiating Tangle

Compute focus: Tangle focuses on compute services (AI inference, data processing), not just validation tasks.

Developer-defined services: Blueprints enable new service types through configuration, not protocol modification.

Integrated products: Tangle combines infrastructure with products that demonstrate and utilize it.

O(1) algorithms: Fundamental design choice enabling unlimited scalability.

RFQ marketplace: Accommodates heterogeneous services naturally.

11 Conclusion

Tangle provides the coordination layer for a decentralized compute economy. The protocol enables developers to define computational services, operators to provide those services with economic accountability, and customers to consume services with cryptographic guarantees.

The mechanisms described—blueprints and services, staking and delegation, the service marketplace, fee distribution, slashing, and governance—compose into a coherent system where honest participation is more profitable than cheating, where value distributes according to contribution, and where no single party controls the infrastructure on which others depend.

This is not merely an ideological position but a practical one. Decentralized infrastructure is more resilient, competitive markets produce better pricing, and cryptographic verification provides stronger guarantees than legal agreements. The same properties that made decentralized finance compelling make decentralized AI infrastructure compelling.

The operating layer for autonomous work must be built now. Tangle is that layer.