COMPUTENET PROTOCOL

An Open Protocol for Verified Useful Compute

Research Architecture, Consensus Concepts, Verification Pipeline, and Protocol Direction

Whitepaper v4 Open Public Testnet v0.11 Research Preview

compute-net.org

TABLE OF CONTENTS

0 Abstract
1 Introduction
2 Protocol Philosophy
3 Fair Launch Principles
4 What Makes ComputeNet Different
5 Truth Stack Verification Architecture
6 Proof of Useful Compute (PoUC)
7 Verification Pipeline
8 Validator Network
9 Consensus Model
10 Security Model
11 Telemetry & Observability
12 Genesis Candidate
13 Long-Term Vision
14 Open Research Questions
15 Indicative Roadmap
16 Public Research Endpoints
17 Disclaimer
18 Appendix

Abstract

ComputeNet is an experimental open protocol designed to transform compute into a verifiable, decentralized, utility-backed network resource.

Where Bitcoin introduced decentralized monetary consensus and Ethereum introduced decentralized execution, ComputeNet explores a third primitive:

> *Decentralized verification of useful compute.*

The protocol proposes a Proof of Useful Compute (PoUC) model in which computational work is only considered valid when:

1. the task is deterministic,
2. the output is reproducible,
3. the execution can be independently verified,
4. receipts can be cryptographically attested,
5. validators can reach consensus on execution validity.

ComputeNet is being developed under a research-first framework with no ICO, no premine, no founder allocation, no custodial promises, and no investment guarantees. The protocol is intended to follow a Bitcoin-style fair-launch philosophy where network participation emerges organically through open-source infrastructure and validator participation.

> *This document describes the early research architecture, consensus concepts, compute receipt system, validator topology, testnet structure, and long-term protocol direction.*

---

1 Introduction

The Compute Problem

Artificial intelligence, simulation systems, cryptographic verification, scientific modeling, rendering, and inference workloads are rapidly increasing global demand for compute.

Modern compute markets suffer from several structural problems:

- centralized cloud dependency
- opaque compute pricing
- unverifiable execution
- fragmented idle compute
- weak trust guarantees
- poor interoperability
- inefficient resource coordination

At the same time, existing proof-of-work systems consume energy while producing computational outputs with little reusable utility.

> *Computational work should produce reusable value.*

Core Thesis

The core thesis behind ComputeNet is simple:

> *Useful computation can become a native protocol primitive.*

Instead of hashing purely for difficulty competition, nodes may eventually compete by:

- executing deterministic workloads
- generating verifiable outputs
- producing cryptographic execution receipts
- participating in decentralized validation

The long-term objective is a decentralized network where compute itself becomes measurable, attestable, and exchangeable.

---

2 Protocol Philosophy

Research-First Development

ComputeNet is currently operating in Open Public Testnet / Research Preview mode.

The current network is:

- non-economic
- non-custodial
- non-commercial
- experimental

There is currently:

- no mainnet token
- no mining
- no mainnet rewards
- no economic issuance
- no investment mechanism

The purpose of the current network phase is:

- protocol experimentation
- validator architecture testing
- compute receipt verification
- consensus validation
- distributed systems research

---

3 Fair Launch Principles

ComputeNet intends to follow several foundational principles:

No Premine
No founder allocation or hidden supply.

No ICO
No public fundraising through token sales.

No Central Custody
The protocol is intended to operate without custodial control.

Open Participation
Validator participation should eventually emerge through open protocol rules.

Utility-First Orientation
The network should prioritize useful computation over speculative activity.

---

4 What Makes ComputeNet Different

ComputeNet does not position itself as a generic blockchain clone or conventional distributed compute marketplace. Its distinctive research direction is the Truth Stack: a layered verification architecture for useful computation.

The Truth Stack is designed as a layered verification architecture rather than a single-proof system. Instead, it combines deterministic workload manifests, reproducible execution, execution hashing, portable compute receipts, validator replay, witness observation, cross-validator attestations, and aggregate truth scoring.

ComputeNet additionally explores novel approaches to deterministic workload enforcement, probabilistic fraud detection, and cross-validator reproducibility scoring intended to improve decentralized verification reliability without relying solely on trusted hardware or centralized coordinators.

> *The guiding research objective is that verification should eventually become cheaper than execution.*

> *This approach allows ComputeNet to begin with narrow deterministic workloads while progressively expanding the research surface toward more complex useful compute categories. The protocol does not claim to have fully solved decentralized useful compute verification; it proposes an incremental path toward making useful computation independently reproducible and verifiable.*

---

5 Truth Stack Verification Architecture

At a high level, the Truth Stack attempts to turn compute execution into portable evidence. A workload is first expressed as a deterministic manifest. Execution produces an output hash. The output hash and execution metadata become a compute receipt. Other validators or witnesses can replay, inspect, or attest the receipt. Accepted receipts can then be aggregated into consensus proof bundles.

> *The public whitepaper intentionally describes this process at protocol-architecture level. Low-level anti-gaming controls, replay sampling thresholds, challenge heuristics, and fraud-scoring details remain subject to change during public testnet research.*

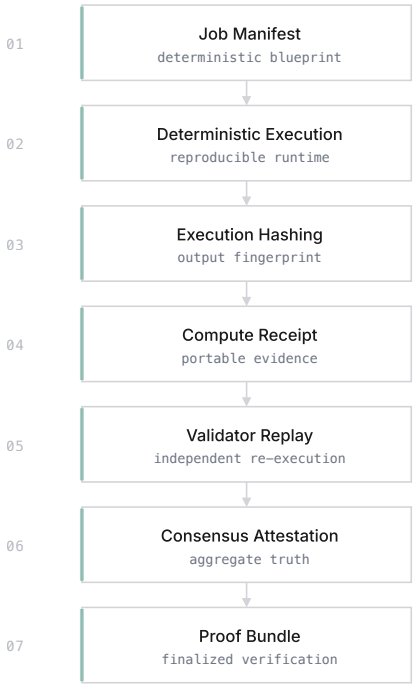

Fig. 1. Truth Stack Verification Pipeline

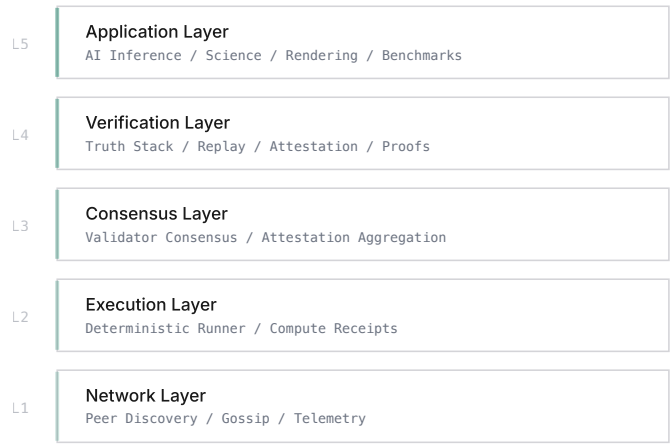

Fig. 2. ComputeNet Layer Model

6 Proof of Useful Compute (PoUC)

Definition

Proof of Useful Compute (PoUC) is the conceptual foundation of ComputeNet. In PoUC:

- computational work must produce reusable outputs
- outputs must be independently verifiable
- validators must be able to reproduce execution
- receipts must be consensus-verifiable

> *The protocol does not currently claim to have solved decentralized useful compute fully. Instead, ComputeNet should be viewed as an evolving research framework attempting to solve it incrementally.*

Desired Properties

An ideal PoUC system should provide:

Determinism Inputs should produce reproducible outputs.	**Verifiability** Independent validators should verify execution validity.
Replay Resistance Receipts should resist duplication or manipulation.	**Fraud Detection** Dishonest validators should be identifiable.
Cost Efficiency Verification should remain computationally cheaper than generation.	**Utility Production** The network should generate reusable computational value.

What Qualifies as Useful Compute

Potential workload categories may include:

- AI inference workloads
- deterministic model execution
- scientific simulations
- rendering workloads
- compression and optimization routines
- benchmarking tasks
- matrix and tensor operations
- cryptographic computation
- reproducible analytical workloads

7 Verification Pipeline

The central challenge ComputeNet attempts to address is not simply distributed computation. The deeper challenge is:

> *How can a decentralized network verify that useful computation was genuinely executed correctly?*

ComputeNet approaches this problem through a layered verification pipeline combining:

- deterministic execution
- reproducible workloads
- execution hashing
- portable compute receipts
- validator re-execution
- consensus attestations
- aggregate proof formation

Job Manifest

Every compute task begins as a deterministic job manifest. A manifest contains:

```
{
  job_id : string
  execution_target : string
  workload_def : WorkloadDefinition
  inputs : InputParams[]
  exec_params : ExecutionConfig
  verification_mode : VerificationMode
}
```

The manifest acts as the canonical execution blueprint. All validators should receive identical manifest definitions.

Compute Receipts

Each completed workload produces a structured compute receipt:

```
{
  job_id : string
  manifest_hash : bytes32
  input_hash : bytes32
  output_hash : bytes32
  runtime : uint64
  validator_signature : Signature
  execution_timestamp : uint64
  consensus_metadata : ConsensusInfo
}
```

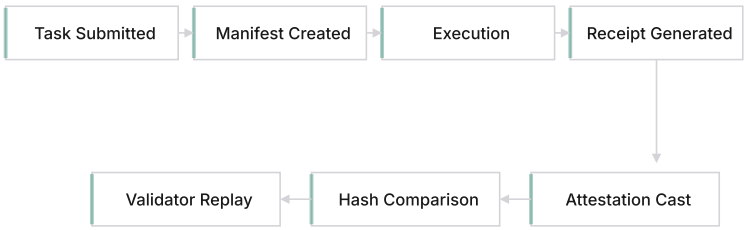

Fig. 3. Receipt Lifecycle

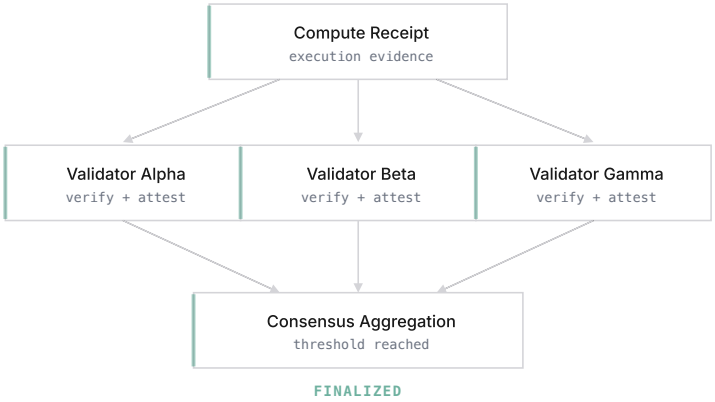

Fig. 4. Consensus Attestation Flow

Replay Sampling

The network may use probabilistic replay verification in which only a percentage of workloads are re-executed by validators. This is intended to make verification economically scalable while preserving fraud resistance. The long-term objective is: verification should remain cheaper than execution.

Challenge Windows

Future protocol versions may introduce challenge windows allowing validators to dispute suspicious receipts prior to finalization. Potential challenge triggers may include:

- inconsistent hashes
- abnormal runtime behavior
- duplicated outputs
- malformed manifests
- statistical anomalies

Future Zero-Knowledge Verification

Long-term research may explore:

- zkVM integration

- recursive proof systems
- SNARK/STARK compression
- hardware attestation
- trusted execution environments

> *ComputeNet does not currently claim production-grade zero-knowledge compute verification. This remains an active area of protocol research.*

---

8 Validator Network

Validator Roles

Validators serve as the trust infrastructure of ComputeNet. Their responsibilities include:

- receipt validation
- consensus participation
- peer propagation
- uptime reporting
- fraud detection

Validator Topology

The current architecture is evolving from single-machine simulated validators to multi-VPS distributed validators. Future topology goals include:

- geographically distributed validators
- independent operators
- peer discovery
- resilient gossip propagation
- autonomous recovery

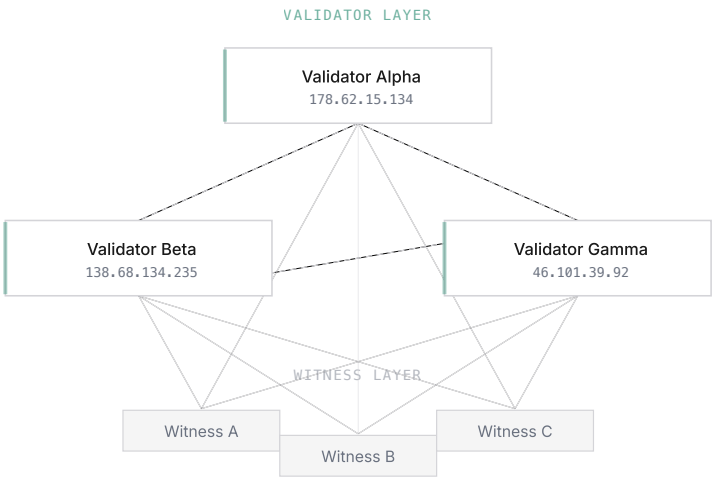

Fig. 5. Validator & Witness Mesh

Peer Discovery

Validators maintain peer registries. Future peer discovery may include:

- bootstrap peer lists
- gossip synchronization

- liveness scoring
- adaptive trust weighting

Reputation and Trust Weighting

Future research directions may include validator reputation systems based on:

- historical verification accuracy
- uptime reliability
- fraud detection history
- consensus alignment
- reproducibility consistency

> *Validator weighting should remain subordinate to decentralization and anti-capture principles.*

---

9 Consensus Model

Current Consensus

Current consensus is validator-attestation based. Validators:

1. verify receipts
2. cast attestations
3. aggregate acceptance ratios
4. finalize valid execution bundles

Future Consensus Research

Future consensus research areas may include:

- asynchronous BFT
- validator reputation systems
- adaptive verification thresholds
- probabilistic sampling
- challenge-response dispute windows

> *Decentralized compute validity without centralized trust.*

---

10 Security Model

Current Security Position

ComputeNet is currently experimental. The network has not undergone:

- formal audits
- adversarial stress testing
- production-grade security certification

> *The current public research testnet exists specifically to discover weaknesses, test assumptions, validate architecture, and identify attack vectors.*

Threat Areas

- validator collusion
- replay attacks
- receipt forgery
- fake compute outputs
- Sybil attacks
- peer poisoning
- denial-of-service attacks
- consensus manipulation

Future Security Research

- cryptographic proof compression
- zero-knowledge verification
- secure enclaves
- hardware attestation
- post-quantum upgradeability
- distributed fraud scoring

---

11 Telemetry & Observability

Protocol Telemetry

ComputeNet includes telemetry systems for:

- validator uptime
- peer liveness
- receipt activity
- consensus reports
- runtime compute jobs

Public Observer Mode

The current public-facing infrastructure operates in Public Observer Mode. This mode allows:

- public-readable telemetry
- bootstrap peer visibility
- genesis candidate visibility
- explorer access

Without:

- public mining
- public validator onboarding
- economic participation

---

12 Genesis Candidate

Genesis Philosophy

ComputeNet intends to treat genesis as a protocol freeze event. Genesis should only occur after:

- multi-node validation
- adversarial testing
- consensus stability
- deterministic compute verification
- public documentation
- reproducible snapshots

Genesis Principles

Open-Source Launch
Fully transparent and publicly reproducible.

No Premine
Zero founder allocation or hidden supply.

No ICO
No public fundraising through token sales.

Transparent Issuance
Candidate maximum supply: 21,000,000 COMPUTE.

---

13 Long-Term Vision

The long-term objective of ComputeNet is:

> *A decentralized protocol for verifiable useful compute.*

Potential future use cases may include:

- AI inference verification
- distributed scientific workloads
- rendering networks
- simulation markets
- decentralized benchmarking
- verifiable agent execution
- cryptographic proof marketplaces

---

14 Open Research Questions

Several major research questions remain unresolved:

- How should useful compute be measured?
- How should fraud be penalized?
- How can deterministic execution scale?
- What workloads qualify as useful?
- How should validator incentives function?
- Can useful compute remain decentralized?
- Can verification remain cheaper than execution?

> *ComputeNet does not claim to have fully solved these questions. Instead, the protocol exists to explore them experimentally.*

---

15 Indicative Roadmap

Fig. 6. Genesis Roadmap

V0.11
Open Public Testnet
● LIVE

V0.12
Controlled External Validator Onboarding
PLANNED

V0.13
Reproducible Public Releases & Installer Hardening
PLANNED

V0.14
Extended Stress Testing & Uptime Window
PLANNED

V0.15
Security Review, Economic Review, Final Genesis Prep
PLANNED

V1.0
Mainnet Genesis (after public testnet stability)
PLANNED

16 Public Research Endpoints

VALIDATOR ALPHA
`http://178.62.15.134:8787` — ● LIVE

VALIDATOR BETA
`http://138.68.134.235:8787` — ● LIVE

VALIDATOR GAMMA
`http://46.101.39.92:8787` — ● LIVE

PUBLIC EXPLORER
`https://www.compute-net.org/explorer` — ● LIVE

MANIFEST EXAMPLE
`/v011/manifest` — RESEARCH

GENESIS CANDIDATE
`/v010/genesis_candidate` — RESEARCH

17 Disclaimer

ComputeNet is experimental software.
The protocol is currently in public research testnet mode. Nothing in this document constitutes:

- investment advice
- an offer of securities
- financial guarantees
- economic promises
- mining guarantees
- token issuance commitments

Participation in future testnets may involve technical, operational, and security risks. The protocol may change substantially prior to any potential public launch.
No real COMPUTE mainnet token is live. TEST-COMPUTE is non-economic and exists only for public testnet experimentation. Nothing in this document should be interpreted as investment advice, a token sale, a securities offering, mining guarantee, or economic promise.

18 Appendix

Current Research Components

Current research modules include:

- validator APIs
- proof engines
- compute receipt engines
- deterministic runners
- consensus coordinators
- peer registries
- telemetry systems
- bootstrap manifests
- public observer infrastructure

Protocol Ethos

ComputeNet is being developed around several foundational ideas:

- openness over exclusivity
- utility over speculation
- experimentation over marketing
- infrastructure over hype
- decentralization over control

> *The project should be understood primarily as a protocol research initiative exploring verifiable useful compute.*