Omerta: A Trust-Based Alternative to Blockchain Consensus for Decentralized Compute Markets

Participation Verification Through Subjective Trust and Automated Monetary Policy

Technical Whitepaper

January 2026

Contents

Abstract

Decentralized compute sharing faces a fundamental challenge: how can strangers cooperate without a trusted central authority? Blockchain-based systems address this through proof-of-work or proof-of-stake consensus, achieving Byzantine fault tolerance at the cost of significant resource overhead and limited throughput. We present Omerta, a practical implementation that synthesizes decades of research on trust, reputation, and mechanism design into a working system for decentralized compute markets.

Omerta computes trust *locally* relative to each observer, rather than maintaining global scores. Trust is derived from verified transactions—actual compute sessions with measurable outcomes—rather than subjective ratings. This design is *machine-native*: unlike prior reputation systems that assumed humans would rate each other, Omerta expects trust signals to be generated automatically from transaction outcomes, enabling measurement at scales human rating could never achieve. This enables natural scaling without global consensus overhead, at the cost of accepting that different observers may have different views of the same identity's trustworthiness. Crucially, local computation does not mean isolated computation: transactions are witnessed by third parties, records propagate through gossip, and cryptographic signatures ensure authenticity. We retain the benefits of distributed verification and immutable records—we simply don't require the entire network to agree on a single global ordering before transactions can proceed.

This paper presents: (1) a trust model derived from verified transactions rather than subjective ratings; (2) local trust computation with path decay; (3) automated monetary policy that adjusts parameters in response to detected threats; (4) economic analysis demonstrating when unreliable home compute creates genuine value; and (5) double-spend resolution where currency "weight" scales with network performance, providing a practical alternative to global consensus for bilateral transactions.

We draw on the observation that human societies have always traded privacy for trust—villages had high trust precisely because everyone knew everyone's business. Omerta recreates this visibility at global scale through on-chain transparency. Unlike villages with their arbitrary social punishment, we aim to maximize freedom within the trust constraint: only verifiable anti-social behavior—failed deliveries, double-spends, verification failures—affects trust scores, where "verifiable" means the determination is reproducible from on-chain data by any observer.

We argue that implementing fair trust systems at scale was computationally intractable until machine intelligence provided the reasoning capacity to model behavior, tune parameters, and explain decisions. This paper itself was developed through human-machine collaboration, demonstrating the thesis: machine intelligence both demands the compute that systems like Omerta could provide and enables the trust mechanisms that make such systems work.

Prior trust systems like EigenTrust and FIRE were never widely deployed—they remained academic exercises, computing trust scores that connected to nothing. Omerta brings these ideas into implementation with real economic consequences: trust scores that affect payments, transfer costs, and access. The contribution is both theoretical and practical—new mechanisms where prior work was insufficient, adoption of proven approaches where they exist, and integration into a working system. The software is given away for free with no preallocation of tokens.

Unlike prior decentralized compute platforms that struggled with adoption, Omerta targets a specific opportunity: billions of home computers sit idle most of the time, representing low marginal cost compute for owners who have already paid for hardware and internet. The software will pro-

vide transparency about actual operating costs—electricity, bandwidth, wear—with user controls for participation thresholds. The software is open source with no platform fees. And machine intelligence dramatically increases the utility of distributed compute—enabling humans to orchestrate complex parallel workloads across unreliable infrastructure in ways they couldn't manage manually. This combination of low-cost supply, zero-rent platform, and machine-intelligence-amplified demand may succeed where blockchain-based alternatives with mining overhead, token economics, and human-centric design have struggled.

1. Introduction

The vision of decentralized computing—where anyone can contribute resources and anyone can consume them, without intermediaries extracting rents—has motivated decades of research. The core challenge is threefold: protecting against the rare bad actors who spoil cooperation for everyone, reducing the technical barriers that have limited participation to experts, and making the experience simple enough to be worthwhile. The practical reality is that most participants in even "trustless" systems don't write their own code—they trust wallet software, exchange interfaces, and protocol implementations written by others. The question is not whether to trust, but whom, how much, and at what cost.

1.0 A Brief History of Trust Systems

The question of computational trust saw intensive research in the early 2000s, driven by the rise of peer-to-peer file sharing networks and online marketplaces. Researchers developed algorithms to compute reputation scores, integrate multiple trust sources, reason under uncertainty, and detect manipulation. This body of work established core insights that remain valid: trust propagates through networks with decay; local computation can substitute for global consensus; time and history provide unforgeable credentials; and detection of manipulation patterns enables defensive responses. (See Section 2.1 for detailed treatment of specific systems.)

Yet these systems were never widely deployed. They remained academic exercises—published, cited, and largely forgotten in practice. Why?

First, they were purely reputational. EigenTrust computed trust scores, but those scores didn't connect to payments, incentives, or economic flows. Trust was a number with no consequences. Without economic integration, there was no compelling reason to deploy them.

Second, they lacked a killer application. P2P file sharing—the motivating use case—didn't have the economics to justify sophisticated trust systems. Free music downloads didn't require trustworthy cooperation; users would simply try another node. The theory outpaced the need.

Third, and perhaps most importantly, blockchain arrived. Bitcoin (2008) appeared to solve the trust problem through cryptographic consensus. Research attention shifted. Why model trust computationally when proof-of-work could enforce cooperation mathematically? The reputation systems literature quieted.

A decade later, we understand blockchain's limitations more clearly. Budish [38] demonstrated that blockchain security has inherent economic limits—the recurring costs of running a secure blockchain must be large relative to the value at stake, making it expensive for high-value applications. Proof-of-stake faces similar constraints [39]. Existing decentralized compute networks built on blockchain (Golem [15], iExec [16]) have struggled with adoption despite years of operation. The costs of global consensus may exceed what many applications require.

Meanwhile, machine intelligence has advanced dramatically. Tasks that seemed intractable—modeling complex behavior, tuning parameters across high-dimensional spaces, generating explanations for decisions—are now feasible. This creates an opportunity: **the trust systems research of 2000-2010 may be ready for practical implementation, enabled by machine intelligence capabilities that didn't exist when the theory was developed.**

Omerta represents an attempt at this synthesis. We return to the trust-based approaches developed before blockchain's dominance, informed by what we've learned since, and enabled by machine

intelligence to handle the complexity that made pure implementation difficult.

1.1 Three Paths

Traditional approaches fall into two categories. **Trusted intermediary** models, exemplified by cloud computing providers, centralize authority in organizations that can enforce contracts and punish misbehavior. This works but introduces single points of failure, censorship risk, and rent extraction. **Blockchain-based** models, exemplified by Ethereum and its descendants, replace trusted intermediaries with cryptographic consensus protocols that theoretically eliminate the need for trust. This also works but imposes significant costs: massive energy expenditure (proof-of-work), capital lockup requirements (proof-of-stake), limited transaction throughput, and delayed finality.

Between these extremes lies a spectrum of approaches trading trust for cost. Fully homomorphic encryption (FHE) and multi-party computation (MPC) enable trustless computation but with 1,000-1,000,000× overhead. Trusted execution environments (TEEs) like Intel SGX reduce overhead but trust hardware manufacturers. Layer-2 solutions (rollups, sidechains) inherit security from a base chain while improving throughput, but still pay consensus costs. Smart contracts execute deterministically but are limited to on-chain data and simple computations.

We propose a third path—or rather, we return to one that was overshadowed. Omerta is a trust-based distributed compute network that neither centralizes authority nor attempts to eliminate trust. Instead, it makes trust *subjective*, *local*, and *earned*—computed by each participant from their own position in the network, based on verifiable on-chain data accumulated over time. This approach builds directly on EigenTrust, FIRE, and related work, while making specific adaptations for compute markets and leveraging machine intelligence for implementation.

1.2 The Trust Spectrum

Trustlessness is not binary—it exists on a spectrum. Proof-of-work and proof-of-stake mechanisms genuinely increase trustlessness compared to centralized alternatives. They represent real achievements in distributed systems research. However, historical episodes reveal that a social layer always remains:

The DAO Hack (2016): An attacker exploited a smart contract vulnerability to drain $60 million from The DAO. The Ethereum community responded with a hard fork that reversed the theft—creating Ethereum (rolled back) and Ethereum Classic (preserved the "immutable" history). The community chose social consensus over mechanical execution.

Bitcoin Value Overflow (2010): A bug created 184 billion bitcoins out of thin air. Developers and node operators coordinated to deploy a fix and roll back the chain. Human judgment overrode the protocol when stakes were high enough.

These interventions were controversial—but the social layer can also work as intended:

Exchange Coordination: When exchanges collectively delist contentious tokens or coordinate responses to theft, they demonstrate genuine community action in support of shared values. This is humans exercising collective judgment when protocol alone is insufficient, and it represents the system working, not failing.

These episodes do not invalidate blockchain achievements. Rather, they reveal that we operate on a spectrum from full trust (centralized authority) to reduced trust (cryptographic consensus) to some irreducible social layer. Sometimes that social layer intervenes controversially; sometimes it acts

in clear support of community values. No practical system reaches the zero-trust endpoint—nor should it.

The question becomes: given that we cannot achieve absolute trustlessness anyway, what are we paying for the trustlessness we do achieve? And could we relax our requirements slightly to capture most of the practical benefit at dramatically lower cost?

This is the same reasoning that motivates ephemeral compute over fully homomorphic encryption (FHE). FHE provides the ultimate guarantee: compute on encrypted data without ever decrypting it. No trust in the compute provider required. But FHE imposes 1,000-1,000,000x computational overhead [21], making it impractical for most workloads. Ephemeral compute—where data exists briefly on untrusted hardware with verification and economic penalties—provides weaker guarantees but serves far more use cases at practical cost.

Omerta applies this spectrum thinking to consensus itself. Blockchain consensus mechanisms genuinely reduce trust requirements, but at significant cost: energy expenditure (PoW), capital lockup (PoS), limited throughput, and delayed finality. We ask: for compute markets specifically, can we relax the global consensus requirement while preserving the practical security properties that matter?

Our hypothesis is yes. Compute markets do not require global agreement—they require pairwise trust between specific buyers and sellers. By computing trust locally rather than achieving global consensus, Omerta aims to capture most of the practical benefit of decentralization at dramatically lower cost, making trustworthy compute sharing accessible to more people.

Compute is also uniquely suited to trust-based systems because of the nature of the goods being traded:

- **Revocable**: Providers can reclaim their machines at any time. Unlike transferring money or physical goods, access can be terminated instantly.
- **Low stakes per transaction**: Each session consumes only a bit of time, energy, and wear. No single transaction is catastrophic.
- **Already sunk costs**: Most home computers sit idle—owners have already paid for hardware, electricity, and internet. The marginal loss from a bad transaction is minimal.
- **Verifiable during execution**: Compute can be checked while running through heartbeats, random audits, and result validation. Fraud is detectable, not just punishable after the fact.

These properties make compute an ideal domain for experimenting with trust-based systems. The downside risk is bounded, the verification is tractable, and the resources were often going unused anyway.

1.3 Our Contribution

This paper's primary contribution is a **practical synthesis**—bringing established trust system research into implementation for compute markets. We distinguish between mechanisms adapted from prior work and novel contributions:

Adapted from prior work (with modifications):

1. **Local trust computation** extending EigenTrust [3], TidalTrust [42], and path-based approaches [4, 37]. Where EigenTrust computes global scores, Omerta computes trust relative

to each observer—a design approach with substantial prior art [42] that we apply to compute markets.

2. **Trust propagation with decay**, a standard technique in graph-based trust systems [3, 35, 37], applied here to transaction histories rather than explicit ratings.

3. **Age-based Sybil resistance**, building on temporal defense mechanisms discussed since Douceur's original Sybil attack paper [6] and whitewashing analysis [44]. We note that this defense has known limitations (Section 8.9).

4. **Cluster detection for Sybil defense**, applying standard anomaly detection techniques [9, 10] to transaction graph analysis.

Novel contributions:

5. **Economic mechanisms integrated with trust**. Prior reputation systems (EigenTrust, TidalTrust, FIRE, PowerTrust) are purely reputational—they compute scores but don't connect them to economic flows. Even TrustChain [51], which uses bandwidth tokens, doesn't tie token mechanics to trust scores. Omerta introduces:

   - *Trust-based payment splits*: Provider earnings scale with trust score via an asymptotic formula
   - *Transfer burns*: Coin transfers taxed based on minimum trust, preventing reputation laundering
   - *Trust-proportional distribution*: Daily coin minting distributed proportionally to trust scores
   - *Negative bids*: Providers can burn coins to accelerate trust building through verified work

6. **Structured accusation mechanism**. Prior systems use binary ratings or simple scores. Omerta's trust assertions include evidence hashes, derive credibility from the asserter's own trust (recursive), and apply impact/context multipliers for appropriate gray areas. This enables nuanced handling of infractions.

7. **Double-spend resolution via currency weight**. Since we detect double-spends rather than prevent them, transaction finality depends on how quickly conflicting transactions would be discovered. Omerta introduces "currency weight"—the confidence level required before a transaction is considered final—that scales with network connectivity. Well-connected networks detect double-spends quickly, so transactions reach finality faster with fewer confirmations. Poorly-connected networks have detection gaps, requiring longer wait times or more confirmations. This allows graceful degradation: the system adapts its trust requirements to actual network conditions rather than assuming uniform connectivity.

8. **On-chain verification logs**. Prior systems record transactions or ratings, but not third-party verification results. Omerta stores verification outcomes on-chain, enabling trust computation to incorporate objective performance data rather than only self-reported transactions.

9. **Application to compute markets**. The specific integration of trust, payment, verification, order book, and session lifecycle mechanisms for decentralized compute rental is novel, though individual components draw on established work.

10. **Machine-native trust measurement**. Prior reputation systems (eBay, EigenTrust, FIRE) assumed humans would rate each other—clicking stars, leaving feedback, issuing certifica-

tions. Omerta assumes trust signals are generated automatically from verified transaction outcomes. Human input comes through engineering the automation and reviewing edge cases, not through direct rating. This enables trust measurement at scales and frequencies that human rating could never achieve.

2. Related Work

Omerta draws on multiple research traditions that have evolved largely independently. Reputation systems from e-commerce and P2P networks established how to aggregate trust signals. Sybil resistance research addressed identity manipulation. Blockchain and consensus work explored the trust-cost spectrum. Secure computation approaches (FHE, MPC, TEEs) pushed toward trustless execution at high cost. Volunteer and commercial distributed computing projects demonstrated both the potential and the pitfalls of shared compute. Finally, computational economics provided tools for modeling incentives and validating mechanism designs.

A key distinction runs through all comparisons: prior reputation systems assumed humans would rate each other. Omerta assumes machine-generated trust signals from verified transactions. This difference is fundamental—it changes what scales are achievable and what attack surfaces exist.

This section surveys each tradition, identifies what Omerta borrows, and clarifies where we diverge.

2.1 Reputation Systems

The challenge of establishing trust among strangers online has motivated extensive research on reputation systems [1, 47]. Resnick et al. [1] identified the core requirements: long-lived identities, captured feedback, and feedback-guided decisions. The eBay feedback mechanism demonstrated these principles at scale, though its binary ratings created opportunities for manipulation [2]. Tadelis [47] provides a comprehensive review of feedback systems in online platforms, documenting issues like grade inflation, retaliation concerns, and fraudulent reputation building.

More sophisticated approaches emerged from peer-to-peer networks in the early 2000s:

EigenTrust [3] computed global trust through iterative aggregation similar to PageRank. Its key insight—that trust can be aggregated through matrix operations—remains foundational. However, it produces global scores rather than observer-relative trust, and relies on explicit transaction ratings.

PeerTrust [4] incorporated transaction context, feedback scope, and community context factors. It recognized that trust depends on more than simple rating counts. Omerta adopts this insight but derives context from transaction records rather than explicit metadata.

FIRE [35] integrated four trust sources: interaction trust (direct experience), role-based trust (position in organization), witness reputation (third-party reports), and certified reputation (references from trustees). Omerta shares the recognition that trust has multiple components, though we deliberately exclude witness and certified reputation to eliminate subjective input vectors—relying only on verifiable transaction outcomes.

Subjective Logic [36] provided mathematical foundations for reasoning under trust uncertainty, modeling opinions as probability distributions with explicit uncertainty parameters. Jøsang's framework for trust transitivity and fusion operations could strengthen Omerta's formal foundations; we note this as future work.

PowerTrust [37] discovered power-law distributions in user feedback patterns and leveraged this for faster convergence through "power nodes." Omerta doesn't explicitly designate power nodes, but we expect trust scores to follow a similar power-law distribution—early participants, reliable providers, and high-volume traders will accumulate disproportionate trust. This is arguably a feature (natural meritocracy where proven participants gain influence) and a risk (concentration

that could enable collusion or create single points of failure). We acknowledge this dynamic rather than pretending it won't occur.

Similarities with prior work: Trust propagation with decay, local computation principles, transaction context sensitivity, and the recognition that different trust components require different handling.

What Omerta changes: Trust derives primarily from verified on-chain transactions rather than subjective ratings. This dramatically reduces the fake feedback attack surface—some surface remains (colluding parties can generate fake transactions), but statistical analysis makes this harder than simply posting fake reviews. Humans can still influence trust scores directly through assertions, but at a cost: making an accusation stakes the asserter's own credibility. If you assert something others can't verify, you pay for it in trust. This mirrors how human trust networks actually work—when someone makes an accusation in a social group, their own reputation is on the line. Baseless accusations damage the accuser. The protocol codifies this natural dynamic, creating economic pressure toward honest, explainable feedback while preserving the ability to flag genuinely problematic behavior when the cost is worth it. Meanwhile, we collect objective metrics from every transaction (latency, uptime, resource delivery, verification outcomes)—more data than occasional human ratings could ever provide.

The fundamental shift: All systems above assumed humans would generate trust signals—clicking stars, writing reviews, issuing certifications. This creates inherent scale limits: humans won't rate every file download, every API call, every 30-second compute session. It also creates attack surfaces: fake reviews, rating manipulation, retaliation. Omerta assumes trust signals are generated primarily by machines observing verified transaction outcomes. Human input enters the system through multiple channels: engineering the automation that generates ratings, setting policy parameters, reviewing edge cases that automated systems flag, and—when warranted—direct assertions that stake the asserter's own credibility. This same mechanism allows machine intelligences—including future superhuman systems—to participate in the trust network: they can make assertions, stake credibility, and have their judgments weighted by their accumulated trust like any other participant. The protocol is agent-agnostic. Compute markets may be an almost ideal testing ground for such intelligences: they operate on resources society has already deemed marginally beneficial (idle compute that would otherwise go unused), security mechanisms on personal hardware bound potential damage, and users retain the ability to reclaim their machines at any time. If something goes wrong, the system can be shut down—unlike financial markets or critical infrastructure where superhuman participants could cause irreversible harm. This is not a minor implementation detail—it fundamentally changes what scales are achievable, what attacks are possible, and what kinds of intelligence can safely participate.

TrustChain [51] from the Tribler project deserves special attention as the most architecturally similar prior work. Like Omerta, TrustChain uses bilateral ledgers without global consensus, detecting double-spending rather than preventing it. Both scale by avoiding network-wide agreement. However, TrustChain focuses on bandwidth accounting for file sharing, while Omerta integrates trust with economic mechanisms for compute markets.

Mechanism Comparison

The following table compares Omerta's mechanisms against prior work. Checkmarks (Yes) indicate the mechanism is present; dashes (—) indicate absence.

Mechanism	EigenTrust	TidalTrust	FIRE	TrustChain	Omerta
Trust Computation					
Global trust scores	Yes	—	Yes	—	—
Observer-relative (local) trust	—	Yes	—	—	Yes
Path-based decay	Yes	Yes	—	—	Yes
Data Source					
Subjective ratings	Yes	Yes	Yes	—	—
Verified transactions only	—	—	—	Yes	Yes
On-chain immutable records	—	—	—	Yes	Yes
Stored verification results	—	—	—	—	Yes
Economic Integration					
Trust-based payment splits	—	—	—	—	Yes
Transfer burns (reputation laundering defense)	—	—	—	—	Yes
Trust-proportional coin distribution	—	—	—	—	Yes
Negative bids for trust acceleration	—	—	—	—	Yes
Sybil Defenses					
Age-based derate	—	—	—	—	Yes
Cluster/graph analysis	—	—	—	Yes	Yes
Economic penalties	—	—	—	—	Yes
Double-Spend Handling					
Prevention (global consensus)	—	—	—	—	—
Detection + penalties	—	—	—	Yes	Yes
Currency weight scaling	—	—	—	—	Yes
Accusation Mechanism					
Signed assertions with evidence	—	—	—	—	Yes
Credibility from asserter's trust	—	—	—	—	Yes
Parameterized infraction severity	—	—	—	—	Yes

Key observations:

1. **Economic integration is novel**: No prior trust system integrates trust scores directly with payment splits, transfer taxation, or coin distribution. EigenTrust, TidalTrust, and FIRE are purely reputational; TrustChain has bandwidth tokens but not trust-weighted economic splits.

2. **On-chain verification logs are novel**: Prior systems record transactions or ratings, but not verification results from third-party audits. This enables trust to incorporate objective performance data.

3. **Structured accusations are novel**: Prior systems use binary ratings or simple scores. Omerta's assertions include evidence hashes, asserter credibility weighting, and impact/context multipliers that create appropriate gray areas.

4. **Currency weight for double-spend resolution is novel**: TrustChain detects double-spends but has no concept of scaling finality requirements to network connectivity. Omerta's "currency weight" allows graceful degradation across network conditions.

5. **Age as derate (not bonus) is a design choice**: While temporal defenses exist in prior work, Omerta specifically ensures age never *adds* trust—only removes a penalty. This prevents

dormant identity accumulation attacks.

Why this matters: The theory developed in these prior systems is sound—the algorithms work, the math is correct, the insights are real. What was missing was economic integration and a compelling application domain. Omerta builds on this foundation by connecting trust to payments, making reputation have economic consequences, and targeting compute markets where machine-native trust measurement and verifiable delivery align naturally. The contribution is both theoretical (novel mechanisms listed above) and practical (a working system with real economic stakes).

2.2 Sybil Resistance

Douceur [6] proved that without a trusted central authority, a single adversary can present arbitrarily many identities indistinguishable from honest participants. This "Sybil attack" undermines any reputation system where influence scales with identity count.

This is a fundamental impossibility result that Omerta does not overcome. We can make Sybil attacks expensive, but not impossible—an approach shared by recent work like MeritRank [46], which explicitly "bounds" rather than prevents Sybil attacks. Honesty requires acknowledging this limitation.

Defenses fall into three categories:

Resource-based: Require each identity to demonstrate control of scarce resources—computational power [7], financial stake [8], or hardware attestation. Effective but expensive, and doesn't prevent well-resourced attackers.

Social-based: Leverage trust graph structure, noting that Sybil identities have sparse connections to honest nodes [9, 10, 45]. SybilGuard [9] and its improved successor SybilLimit [45] showed that social graph analysis can bound the number of accepted Sybils to O(log n) per attack edge. SybilInfer [10] further refined detection through statistical inference. Effective when the social graph reflects real relationships.

Temporal: Require identities to exist over time before gaining influence. This defense, explored in various systems including Freenet's Web of Trust, cannot prevent patient attackers who pre-create identities years in advance.

Omerta employs a hybrid approach: economic penalties (transfer burns), social detection (cluster analysis), temporal constraints, and computational investment. Crucially, effective aging only starts when identities begin contributing meaningfully to the network—simply creating an identity and waiting provides no benefit. An attacker cannot pre-create a pool of dormant identities; they must actually run compute sessions, which costs real resources.

Limitations we acknowledge: A well-resourced attacker who creates thousands of identities and actively matures them through real computational contribution over years will have thousands of mature identities. This attack requires substantial capital (to pay for compute during maturation) and patience. Omerta's defenses make this expensive in time and money, but do not make it impossible. We discuss residual attack surfaces in Section 8.9.

2.3 Blockchain Consensus and Its Limits

Bitcoin [7] introduced proof-of-work consensus, achieving Byzantine fault tolerance through computational cost. Subsequent systems explored alternatives: proof-of-stake [8], delegated proof-of-stake

[11], practical Byzantine fault tolerance [12], and various hybrid approaches.

All these mechanisms solve the Byzantine Generals Problem: achieving agreement among distributed parties despite malicious actors. This requires $n \geq 3f + 1$ nodes to tolerate f failures, with significant coordination overhead.

Budish [38] demonstrated fundamental economic limits of blockchain security: the recurring flow payments to miners must be large relative to the one-time stock benefits of attacking the system. This makes high-value applications expensive to secure. Gans and Gandal [39] extended this analysis to proof-of-stake, showing similar cost structures manifest as illiquid capital requirements. These analyses suggest blockchain may be over-engineered for applications that don't require global consensus.

Federated approaches occupy a middle ground. Stellar's Federated Byzantine Agreement and Ripple's trust lines allow nodes to choose which other nodes they trust for consensus, rather than trusting the entire network. However, recent analysis [48] reveals that FBA's "open membership" is limited in practice—"membership in the top tier is conditional on approval by current top tier nodes if maintaining safety is a core requirement." Despite this limitation, these systems influenced Omerta's design—the local trust computation is conceptually similar to choosing trusted validators, though Omerta applies this to reputation rather than consensus.

Layer-2 solutions (optimistic rollups, zk-rollups, sidechains) address blockchain scalability by moving computation off the main chain while inheriting security from L1 through various mechanisms (fraud proofs, validity proofs, or periodic checkpoints). Rollups achieve significantly higher throughput at lower cost, making them attractive for high-frequency applications. However, L2s introduce their own trust assumptions: most rely on centralized sequencers operated by single entities, creating censorship risk and single points of failure. Operators can withhold transaction data, preventing users from independently verifying state. The fragmented ecosystem of 140+ L2s requires bridges that have suffered billions in losses (Ronin $615M, Wormhole $320M). These are real trade-offs, not free scaling. For compute markets specifically, L2 solutions face an additional limitation: smart contracts cannot verify that off-chain computation was performed correctly, requiring oracles or optimistic schemes that reintroduce trust. L2s trade increased user trust for decreased transaction cost, but without managing that trust—the assumptions are implicit, scattered across sequencer operators, bridge security, and data availability guarantees. Omerta makes a similar trade-off but explicitly: by acknowledging our trust assumptions and managing them as first-class protocol concepts with reputation scores, decay mechanisms, and economic penalties, we aim to require a much smaller increase in user trust relative to the main chains.

Omerta sidesteps global consensus entirely. While individual transactions are bilateral (buyer-seller), trust has ripple effects: each interaction updates trust scores that propagate through the network. For the economy to function, the majority of interactions must be honest—which is why trust measurement exists in the first place. But this is different from requiring every node to agree on every transaction before it can proceed. By computing trust locally, Omerta eliminates the coordination overhead of global agreement while providing the security properties actually needed for compute rental. This is not superior to blockchain for applications requiring global consensus; it is a different trade-off appropriate for different applications.

Blockchain solved the Byzantine Generals Problem. Compute markets don't have Byzantine generals—they have landlords and tenants.

2.4 Secure Computation Approaches

The ultimate solution to untrusted compute would be **fully homomorphic encryption (FHE)** [21], which enables computation on encrypted data without decryption. FHE provides mathematical guarantees: the compute provider learns nothing about the data. However, current FHE implementations impose 1,000-1,000,000x overhead compared to plaintext computation [22], restricting practical use to narrow applications like encrypted database queries or private set intersection where the security requirement justifies the cost.

Trusted execution environments (TEEs) like Intel SGX [23] provide hardware-based isolation with much lower overhead (typically 5-30%), but require trusting the hardware manufacturer and have been vulnerable to side-channel attacks [24] including Spectre, Meltdown, and Foreshadow. TEEs are practical for applications where you trust Intel/AMD/ARM but not the cloud operator—a meaningful threat model, but not trustless.

Secure multi-party computation (MPC) [25] distributes computation across parties such that no single party learns the inputs. MPC overhead depends heavily on circuit complexity and number of parties—simple operations may run at 100-1000x slowdown, while complex operations can be millions of times slower. MPC is practical for specific high-value operations: private auctions, secure voting, threshold cryptography. It is not practical for general-purpose compute.

Where these approaches make sense: When the data is genuinely sensitive (medical records, financial data, trade secrets) and the computation is well-defined and bounded, the overhead may be justified. A hospital running private analytics on patient data, or banks computing fraud scores across institutions without sharing raw data—these are legitimate MPC/FHE use cases.

Where Omerta fits: Most compute workloads don't require cryptographic privacy guarantees. Running a build pipeline, training a model on public data, rendering video, processing batch jobs—these benefit more from cheap, available compute than from mathematical privacy proofs. Omerta targets this larger space, accepting trust requirements in exchange for practical performance.

2.5 Decentralized Computing

The altruistic distributed computing projects—BOINC [13], Folding@home [14], SETI@home—are among Omerta's closest ancestors. They demonstrated that volunteers would contribute compute resources for causes they believed in, without direct compensation. One purpose of Omerta is to make such projects easier: lowering barriers so that researchers without large budgets can access distributed compute for scientific work, citizen science, or public-benefit computation. We built awareness of this heritage into the protocol's design.

These systems face common challenges: verifying that claimed work was actually performed, preventing providers from delivering inferior resources, and detecting collusion. Omerta addresses these through continuous verification, trust-based payment splits, and statistical detection of manipulation patterns.

Why prior commercial systems struggled: Golem, iExec, and similar platforms have operated for years with limited adoption. We identify several contributing factors:

- *High barriers to entry*: Complex setup, specialized knowledge required, blockchain transaction fees for every operation
- *Wrong customer base*: Human developers demand reliability, low latency, and predictable pricing—exactly what unreliable home compute struggles to provide

- *Extractive economics*: VC funding requires returns; token economics require appreciation; platforms extract fees. These create friction that erodes the cost advantage of distributed compute
- *Mining overhead*: Proof-of-work and proof-of-stake consensus impose costs unrelated to compute delivery

Omerta's approach differs: the software is free and open source, requiring only a download to participate. There are no platform fees—providers keep what they earn (minus trust-based burns that fund network security). There is no preallocation of tokens, no founder stake, no investors requiring returns. Machine intelligence workloads are naturally fault-tolerant and retry-friendly, well-suited to unreliable infrastructure. And the "mining" is the useful compute itself, not a separate consensus mechanism.

What we expect to gain: We are transparent about benefits to Omerta's creators. By participating early, we gain access to cheaper compute for our own research and the ability to study trust systems on real networks with real people—something prior academic projects lacked. Early participants naturally accumulate higher trust scores through participation history, which may translate to economic advantages (lower transfer costs, priority matching). These benefits are available to all motivated early participants, not reserved for founders.

The gap in prior trust research: Decentralized compute is not just a blockchain problem—it's also a trust problem. Prior trust systems (EigenTrust, TidalTrust, FIRE) could have addressed the reputation challenges in compute markets, but they remained confined to academia. Why? They lacked economic integration and a compelling application. Omerta attempts to bridge this gap: taking the trust theory developed in the 2000s, connecting it to real economic mechanisms, and applying it to a market with genuinely unbounded demand.

2.6 Computational Economics and Mechanism Design

The design and validation of economic mechanisms increasingly relies on computational methods. Agent-based computational economics [26, 27] provides tools for studying emergent phenomena in complex markets where analytical solutions are intractable [28]. This approach has proven particularly valuable for mechanism design [29], where simulating agent behavior under proposed rules reveals edge cases and failure modes before deployment.

These computational methods have produced substantial real-world successes. Auction theory [31] informed the FCC spectrum auctions that raised over $200 billion while efficiently allocating scarce radio frequencies—a problem intractable without computational mechanism design. Matching market algorithms [32] now assign medical residents to hospitals (the NRMP match), students to schools, and kidneys to recipients, handling constraints and preferences that would overwhelm manual processes. Google's AdWords auction, designed using algorithmic game theory principles from the same literature we draw upon, processes billions of transactions daily. These are not laboratory curiosities but deployed systems handling high-stakes allocation problems.

Validation of agent-based models follows established practices [30]: parameter sensitivity analysis, comparison against theoretical predictions where available, and testing under adversarial conditions. These methods inform our simulation methodology in Section 7.

The trust propagation model relates to work on reputation mechanism design [33], particularly the challenge of eliciting honest feedback in the presence of moral hazard. The concept that trust

mechanism overhead should scale with network uncertainty echoes transaction cost economics [34], which analyzes how institutions emerge to reduce uncertainty in exchange.

Omerta's economic simulations build on these foundations while extending them to a novel domain: trust-based compute markets where machine intelligence both creates demand and enables the trust mechanisms that make supply possible.

3. System Architecture

Having established the theoretical foundations Omerta draws upon, we now describe the system's concrete architecture. The design translates the trust and economic principles discussed above into specific data structures, protocols, and market mechanisms.

3.1 Design Principles

Omerta is built on several core principles:

Free Software, No Rent Extraction: The Omerta software is open source and free to use. There are no platform fees, no token preallocation, no founder stake. Participants earn their position through contribution, not investment. This removes the extractive economics that plagued prior decentralized compute platforms.

Verifiable Facts, Subjective Trust: The blockchain stores facts (transactions, verification logs, assertions). Trust scores are computed locally by each participant from these facts according to their own criteria.

Identity Age as Credential: Of all possible credentials, only time-on-chain cannot be manufactured. New identities start with nothing and must earn trust through participation.

Uniform Pricing, Trust-Based Splits: All consumers pay the same market rate. Trust determines how payments split between provider and burn. High trust = more to provider. Low trust = more burned.

Earned Write Permission: Only identities above trust thresholds can write to the chain, with conflict resolution through trust-weighted voting. Trust score IS the stake.

3.2 On-Chain Data

The blockchain records five types of data:

Identity Records: Public key, creation timestamp, and signature capabilities. All derived data (age, transaction count, trust scores) is computed from other records.

Transactions: Consumer and provider identities, amounts paid/received/burned, resource specifications, duration, and signatures from both parties.

Verification Logs: Results of resource checks, including verifier identity, claimed vs. measured resources, pass/fail result, and verifier signature.

Trust Assertions: Signed claims by one identity about another, including score, classification (positive or negative), evidence hashes, and reasoning.

Order Book: Bids and asks for compute resources, enabling price discovery through market mechanisms.

3.3 Market Structure

Omerta implements an on-chain order book for price discovery:

Order Types: Bids (consumer wants to buy) and asks (provider wants to sell), with resource specifications, price, duration constraints, and expiration.

Resource Classes: Standardized categories (small_cpu, medium_cpu, gpu_consumer, gpu_datacenter) enabling liquidity aggregation.

Matching Engine: Price-time priority matching. When orders cross, sessions initiate and escrow triggers.

Spot Rate: Volume-weighted average of recent trades, providing simple consumer-facing pricing while preserving price discovery.

Consumers see simple pricing ("8 cores = 0.09 OMC/hr") without needing to understand the underlying market mechanics.

3.4 Session Lifecycle

A compute session proceeds as follows:

1. **Order Placement**: Consumer places bid or provider places ask
2. **Matching**: Orders cross, session initiates
3. **Escrow Lock**: Consumer's payment locked in escrow
4. **VM Configuration**: Provider configures VM with consumer's identity key for access
5. **Compute Execution**: Consumer uses resources
6. **Verification**: Random audits check resource claims
7. **Settlement**: Escrow released based on outcome and trust scores

Either party can terminate at any time—the market handles quality through consumer exit and provider reliability signals.

4. Trust Model

Trust in Omerta derives from verified transactions, not subjective ratings. This dramatically reduces the fake feedback attack surface while actually increasing data richness through continuous objective measurement rather than occasional subjective ratings. For compute markets, where delivery can be objectively verified, this approach is well-suited.

4.1 Trust Accumulation

Each identity accumulates a **base trust score** (T_{base}) representing their track record in the network. This score has two components: trust earned from completing transactions ($T_{transactions}$) and adjustments from explicit assertions by other participants ($T_{assertions}$).

$$T_{base} = T_{transactions} + T_{assertions}$$

Why this formulation? Transactions are objective and machine-verifiable—either the compute was delivered or it wasn't. Assertions handle everything else: exceptional performance, suspected manipulation, off-chain behavior. By separating these, we keep the core trust signal clean while allowing for human judgment where needed.

$$T_{transactions} = \sum_i (CREDIT \times resource\_weight_i \times duration_i \times verification\_score_i \times cluster\_weight_i)$$

Each completed transaction contributes to trust. The terms normalize and weight this contribution: resource weights account for different compute types (a GPU hour is worth more than a CPU hour), duration captures commitment length, verification scores reflect audit outcomes (did resources match claims?), and cluster weights downweight transactions suspected of being within Sybil clusters.

Assertion-based trust adjusts for reported incidents:

$$T_{assertions} = \sum_i (score_i \times credibility_i \times decay_i)$$

Assertions are signed reports of specific incidents with scores in [-1, 1]. Positive scores (commendations) add trust; negative scores (violations) subtract. Credibility derives from the asserter's own trust, creating recursive dependency resolved through iterative computation.

4.2 Age as Derate Factor

A critical design choice: age should **never add trust**, only remove a penalty from young identities:

$$T_{effective} = T_{base} \times age\_derate$$

$$age\_derate = \min\left(1.0, \frac{identity\_age}{AGE\_MATURITY\_DAYS}\right)$$

New identities start at zero effective trust regardless of transaction volume. This prevents attackers from pre-creating dormant identities that accumulate trust through mere existence. You can only earn trust by participating over time.

Why linear? We considered alternatives: exponential growth (fast early gains, slow later) favors new users but makes age easily purchased; logarithmic (slow early, faster later) is harsh on newcomers and may discourage participation. Linear provides predictable progress: a 30-day-old identity at 33% has exactly one-third the age credit of a 90-day identity. The simplicity also makes the system easier to reason about—both for participants and attackers calculating costs.

4.3 Local Trust Computation

Trust is not global. Each observer computes trust relative to their position in the network:

$$T(subject, observer) = T_{direct} + T_{transitive}$$

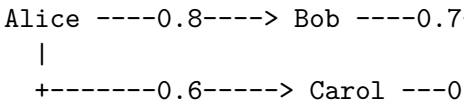
$$T_{transitive} = \sum_{intermediary} T(intermediary, observer) \times T(subject, intermediary) \times DECAY^{path\_length}$$

Direct trust comes from personal transaction history. Transitive trust propagates through trusted intermediaries with exponential decay per hop.

Why this matters: An attacker cannot build trust in Community A and exploit it in Community B. Observers in B see the attacker's trust discounted by lack of network path. The attacker must build trust directly with each community they wish to exploit—exactly how human trust works.

Trust Propagation Example

Consider a small network where Alice wants to assess trust in Dave:



Alice has direct trust 0.8 in Bob and 0.6 in Carol. Bob has 0.7 trust in Dave; Carol has 0.9 trust in Dave. Alice has no direct transactions with Dave.

With DECAY = 0.5 per hop, Alice computes trust in Dave:

Path	Calculation	Contribution
Alice → Bob → Dave	$0.8 \times 0.7 \times 0.5^1$	0.280
Alice → Carol → Dave	$0.6 \times 0.9 \times 0.5^1$	0.270
Total transitive trust		**0.550**

Now consider Eve, who has built 0.95 trust with a separate community but has no path to Alice's network:

```
Alice ----0.8----> Bob          Eve (0.95 in other community)
  |                              |
  +-------0.6-----> Carol        (no path to Alice's network)
```

Alice's trust in Eve = 0.0 (no path). Eve's high reputation elsewhere is invisible to Alice. This is the core Sybil defense: trust must be earned locally, not imported.

4.4 Parameterized Infractions

Not all violations are equal. Infraction severity scales with potential network impact:

$$effective\_score = base\_score \times impact\_multiplier \times context\_multiplier$$

Impact scales with transaction value, resources affected, and duration. Context includes repeat offense history. This creates appropriate gray areas—small mistakes don't destroy trust, while large attacks risk proportional penalties.

5. Economic Mechanisms

Prior trust systems—EigenTrust, TidalTrust, FIRE, PowerTrust—computed reputation scores that had no direct consequences. A user with high trust and a user with low trust experienced the same platform. This may explain why these systems remained academic: reputation without economic consequence is reputation without purpose.

Omerta integrates trust directly into economic flows. Trust affects what you earn, what you can transfer, and what you receive from daily distribution. This gives trust a reason to exist and gives participants a reason to care about it.

5.1 Payment Splits

Consumer payments split between provider and cryptographic burn based on provider trust:

$$provider\_share = 1 - \frac{1}{1 + K_{PAYMENT} \times T}$$

As trust increases, provider share approaches 100% asymptotically but never reaches it. New providers with low trust see most of their payment burned; established providers keep most of it.

Payment Split by Trust Level

The following table shows provider share for different trust scores and K_PAYMENT values:

Trust Score	K=1.0	K=2.0	K=5.0	K=10.0
0.0 (new)	0%	0%	0%	0%
0.1	9%	17%	33%	50%
0.2	17%	29%	50%	67%
0.5	33%	50%	71%	83%
1.0	50%	67%	83%	91%
2.0	67%	80%	91%	95%
5.0	83%	91%	96%	98%
10.0	91%	95%	98%	99%

Reading the table: With K_PAYMENT=5.0, a provider with trust score 0.5 keeps 71% of payments; the remaining 29% is burned. A highly trusted provider (score 5.0) keeps 96%.

Design insight: Higher K values reward trust more steeply. K=5.0 means a provider must reach trust ~1.0 to keep most of their earnings, creating strong incentive for sustained good behavior.

This creates natural incentives: build trust to keep more of your earnings. No external enforcement needed—the economics handle it.

5.2 Transfer Burns

Transfers between identities are taxed based on the minimum trust of sender and receiver:

$$transfer\_burn\_rate = \frac{1}{1 + K_{TRANSFER} \times \min(T_{sender}, T_{receiver})}$$

Low-trust identities cannot easily transfer coins. This prevents reputation laundering (build trust, exploit, transfer coins to fresh identity) and creates strong incentive to donate compute rather than buy coins.

5.3 Daily Distribution

New coins are minted daily and distributed proportionally to trust scores:

$$daily\_share(i) = \frac{effective\_trust(i)}{\sum_j effective\_trust(j)} \times DAILY\_MINT$$

This creates the core incentive: misbehave and your trust drops, reducing tomorrow's share. The motivation to be honest is simply: keep getting your handout.

5.4 Donation and Negative Bids

Providers can accept negative-price bids from research organizations, burning their own coins for accelerated trust:

Bid Type	Price	Trust Multiplier
Commercial	Positive	1x
Zero donation	Zero	1x
Negative donation	Negative	Up to 4x

This enables bootstrapping: new providers can burn coins to accelerate trust building while providing verified compute to research projects. Trust cannot be purchased without actual work—the burn is additional, not replacement.

6. Attack Analysis and Defenses

The trust and economic mechanisms described above create a system with specific attack surfaces. This section analyzes the major attack vectors and Omerta's defenses against them. We aim for honest assessment: some attacks are prevented, others are merely made expensive, and some remain as acknowledged limitations.

6.1 Sybil Attacks

Attack: Create many fake identities to manipulate trust or distribution.

Defenses: - **Age derate**: New identities earn nothing initially - **Cluster detection**: Tightly-connected subgraphs with few external edges are flagged - **Behavioral similarity**: Identities behaving too similarly are downweighted - **Activity requirements**: Must maintain ongoing participation

6.2 Collusion and Trust Inflation

Attack: Colluding parties mutually vouch for each other to inflate trust.

Defenses: - **Graph analysis**: Detect circular trust flows and isolated cliques - **Verification requirements**: Trust requires verified compute, not just assertions - **Statistical anomaly detection**: Burst activity, coordinated timing flagged

6.3 Trust Arbitrage

Attack: Build trust in Community A, exploit in Community B.

Defense: Local trust computation means trust doesn't transfer across network distance. Attackers must build trust directly with each target community.

6.4 Multi-Identity Exploitation

Attack: Maintain multiple identities to hedge risk or enable sacrificial attacks.

Critical Distinction: Some multi-identity strategies must be absolutely prevented; others may be tolerated.

Absolute Protections: - UBI distribution: Malicious behavior in any linked identity reduces combined distribution - Trust from activity: Same work split across N identities yields at most single-identity trust - Accusation credibility: N low-credibility accusations don't sum to high credibility

Tolerated Advantages: - Risk diversification: Legitimate businesses may operate multiple identities - Community separation: Operating in isolated communities without cross-contamination - Recovery via new identity: Starting fresh with appropriate penalties

6.5 Identity-Bound Access

Problem: Traditional credential theft enables exploiting stolen identity's reputation.

Solution: VM access is bound to the consumer's on-chain private key. No key, no access. *If you have the key, you ARE the identity—there is no "stealing," only "being."*

6.6 Double-Spend Attacks

Problem: Unlike blockchain where global consensus mathematically prevents spending the same currency twice, Omerta's local trust model allows an attacker to sign conflicting transactions and broadcast them to different parts of the network before detection.

Why blockchain prevents it: Proof-of-work and proof-of-stake achieve global consensus on transaction ordering. Once a transaction is confirmed, the entire network agrees it happened, making conflicting transactions impossible to confirm.

Why Omerta can only detect it: We deliberately avoid global consensus (and its costs). Without a single authoritative ordering, conflicting transactions can temporarily coexist until nodes compare notes.

Our defenses:

1. **Detection via gossip**: Nodes share transaction records. When a node receives conflicting transactions from the same sender, the double-spend is detected. Well-connected networks detect nearly all double-spends; poorly-connected networks have detection gaps.
2. **Economic penalties**: Detected double-spends trigger a 5× penalty on the attacker's stake. At 100% detection, expected ROI is -400%. Even at 50% detection, ROI remains negative (-150%).
3. **Currency weight scaling**: Trust-weighted currency treats high-trust and low-trust coins differently. Transactions from low-trust identities require more confirmations or smaller amounts, limiting exposure.
4. **Reputation destruction**: Beyond the immediate penalty, detected double-spenders lose their accumulated trust, making future participation difficult.

Limitation: This is detection, not prevention. For high-value transactions requiring absolute double-spend prevention, blockchain remains more appropriate. Omerta accepts this tradeoff for lower-value, high-frequency compute transactions where the economics of detection are sufficient.

See Section 7.6 for simulation results validating detection rates and economic stability under various network conditions.

6.7 Attack Economics Summary

The following table summarizes the economic calculus for major attack types:

Attack	Setup Cost	Time Required	Expected Gain	Expected Loss	ROI
Sybil (1000 identities)	Low (compute)	1-2 years (age maturity)	Distribution share	Detection → isolation	Negative after detection
Trust inflation (collusion ring)	Medium (coordination)	6+ months	Inflated provider share	Graph analysis → derate	Negative if detected

Attack	Setup Cost	Time Required	Expected Gain	Expected Loss	ROI
Double-spend	High (must have coins)	Immediate	1× transaction value	5× penalty + trust destruction	-400% at 100% detection
Long-con	Low	2-5 years	Single large exploit	Total reputation loss	Possibly positive (limitation)
Provider fraud (fake resources)	Low	Immediate	Payment for non-work	Verification → penalties	Negative with random audits
Eclipse attack	High (network position)	Weeks	Partition exploitation	Recovery + detection	Context-dependent

Key insight: Most attacks have negative expected ROI *after detection*. The exception is the long-con attack where patient adversaries build genuine reputation over years before a single large exploit. This remains an acknowledged limitation—we make long-con attacks expensive in time, but not impossible.

Detection probability matters: Double-spend ROI at different detection rates:

Detection Rate	Expected Value	ROI
100%	-4× stake	-400%
80%	-3× stake	-300%
50%	-1.5× stake	-150%
20%	+0.2× stake	+20%

The 5× penalty ensures attacks are unprofitable even at 50% detection. Below ~25% detection, attacks become profitable—motivating investment in detection infrastructure.

7. Simulation Results

The defenses described above are architectural arguments—they explain why attacks should be difficult, but not whether they actually are. This section presents simulation results that test the system's behavior under adversarial conditions, examine the economics of provider competition, and explore the machine intelligence demand thesis that motivates Omerta's design.

7.1 Attack Scenario Outcomes

Scenario	Final Gini	Cluster Prevalence	Policy Response
Baseline (Honest)	0.783	0.000	Stable
Trust Inflation	0.706	0.250	K_TRANSFER increased
Sybil Explosion	0.641	0.545	ISOLATION_THRESHOLD decreased
Gini Manipulation	0.882	0.000	K_PAYMENT decreased

7.2 Key Finding: Structural vs. Parametric Attacks

The simulations revealed a striking pattern: automated policy adjustments trigger correctly but have limited impact on final outcomes. This suggests that attack effects are primarily structural rather than parameter-dependent.

Implication: Effective attack resistance requires architectural defenses that make attacks structurally infeasible, complemented by policy adjustments for fine-tuning. Parameter tweaking alone is insufficient against determined adversaries.

7.3 Long-Term Stability

Five-year simulations showed stable trust accumulation under honest conditions and recovery between attack waves under adversarial conditions. The core trust mechanisms prove robust to extended adversarial pressure.

7.4 Reliability Market Economics

A critical question for any decentralized compute network: can unreliable home providers coexist with reliable datacenters, or will one displace the other? We simulated provider competition under varying market conditions.

Provider Cost Structures:

Provider Type	Cost/hr	Reliability	Cancellation
Datacenter	\$0.50	99.8%	Never (SLA)
Home Provider	\$0.08	92%	For profit (1.5× threshold)

Home providers have 6× lower costs because they already own hardware (no capex amortization), pay only marginal electricity, and have no facility overhead.

Market Outcomes by Supply/Demand:

Market Condition	Consumer Cost Δ	DC Profit Δ	Compute Δ
Undersupplied (uniform values)	-10%	**-3%**	**+200%**
Undersupplied (mixed values)	-52%	-66%	+200%
Balanced	-91%	-100%	+70%
Oversupplied	-84%	-100%	-8%

Key Findings:

1. **Undersupplied markets show value creation**: When demand exceeds datacenter capacity, home providers serve unmet demand. Total compute delivered increases 200% while datacenter profits remain largely intact (only -3% with uniform consumer values).
2. **Balanced/oversupplied markets show displacement**: When total capacity meets or exceeds demand, home providers' 6× cost advantage allows them to undercut datacenters completely.
3. **Consumer heterogeneity matters**: When consumers have widely varying values per compute hour, middle-market competition intensifies. Datacenters retain only the premium segment.
4. **Reliability tradeoff is real**: In oversupplied markets, total useful compute can decrease (-8%) because home provider unreliability causes more restarts despite lower prices.

Implication for Omerta: The system creates genuine economic value only when **demand exceeds datacenter capacity**. This raises the question: will demand ever sustainably exceed supply?

7.5 The Machine Intelligence Demand Thesis

We argue that machine intelligence fundamentally transforms compute markets into **perpetually undersupplied markets**, making unreliable compute economically valuable for the foreseeable future.

The Traditional View:

Human-driven compute demand is bounded. Businesses have finite workloads, consumers have finite entertainment needs. Markets tend toward equilibrium where supply meets demand. In equilibrium, price competition drives out high-cost providers.

Under this view, home providers would be viable only during transient demand spikes.

The New Reality:

Machine intelligence creates unbounded demand because **machines can always find productive uses for additional compute**. Unlike humans, who run out of tasks to do, machine intelligences have continuous demand curves:

Task Priority	Human Value	Machine Value	Notes
Frontier research	\$100/hr	\$0/hr	Requires human insight (for now)
Active inference	\$50/hr	\$40/hr	Real-time decision making
Background reasoning	\$20/hr	\$15/hr	Exploring solution spaces
Speculative search	\$5/hr	\$3/hr	Low-priority but positive value
Precomputation	\$1/hr	\$0.50/hr	Preparing for future queries

The key insight: **there is always a next-best task**. A machine that cannot profitably do a \$50/hr task can still generate value doing a \$5/hr task. This creates a demand curve extending to arbitrarily low prices. *Humans run out of ideas; machines run out of compute.*

Implications:

1. **No "oversupplied" market exists**: Machine demand expands to absorb any available compute
2. **All providers can coexist**: Datacenters serve high-priority tasks, home providers serve the elastic tail
3. **Price floors are set by marginal value, not marginal cost**: Machines bid what tasks are worth

Future Projection:

As machine intelligence improves, the value per compute hour increases at all quality levels:

Era	Demand Characteristic	Market Structure
Human-only	Bounded, tends to equilibrium	Oversupply risk
Human + current MI (today)	Large, finite	Undersupplied
Human + superhuman MI (future)	**Unbounded**	Permanently undersupplied

At the limit, superintelligent systems have unlimited demand for compute of any quality. Any machine cycle has positive value because it can be applied to self-improvement, exploration, or capability expansion.

Conclusion:

The economic viability of Omerta depends on perpetual undersupply. Human demand alone cannot guarantee this. But machine intelligence—which can always find productive uses for marginal compute—transforms the market structure fundamentally. In a world of machine intelligences, the question is not whether unreliable compute will displace datacenters, but whether we can deploy enough compute of any quality to satisfy exponentially growing machine demand.

7.6 Double-Spend Resolution

Unlike blockchain where double-spending is mathematically prevented by consensus, Omerta's trust-based currency can only detect and penalize double-spends after the fact. This raises questions about economic stability. We simulated five scenarios to validate the design.

7.6.1 Detection Rate

Connectivity	Detection Rate	Avg Time	Network Spread
0.1	100%	0.046s	97.6%
0.5	100%	0.042s	98.0%
1.0	100%	0.042s	98.0%

Finding: In gossip networks, double-spends are always eventually detected because conflicting transactions propagate to common nodes. Connectivity affects detection speed, not completeness.

Important clarification: Detection is not prevention. A 100% detection rate means the double-spend is always discovered—but only *after* the conflicting transactions have propagated. During the detection window, both recipients may believe they have valid payments. The economic defense (trust penalties, Section 7.6.2) makes attacks unprofitable but does not prevent the temporary confusion. For high-value transactions where even temporary double-spend would be harmful, use the "wait for agreement" protocol (Section 7.6.3) which provides prevention through delayed finality.

Double-Spend Detection Timeline

The following diagram shows the sequence of events in a double-spend attempt:

```
Time     Attacker          Victim A          Victim B          Network
------------------------------------------------------------------------
t=0      Sends TX1 -----> Receives TX1
         to Victim A      Believes valid

t=0.01   Sends TX2 ----------------------> Receives TX2
         to Victim B                       Believes valid
         (same coins)

t=0.02                    Gossips TX1 -------------------> Propagating
                                                  Gossips TX2 --> Propagating
```

```
t=0.04                                                    Nodes see
                                                          BOTH TX1
                                                          and TX2

t=0.05                    <--- CONFLICT DETECTED --->     Broadcasts
                          Notified        Notified        conflict

t=0.06                    Keeps coins     Keeps coins     Records
                          (no clawback)   (no clawback)   double-spend

t=0.07    <-----------------------------------------------PENALTY:
          Trust destroyed                                 5x coins
          Future earnings = 0                             burned from
          Isolated from network                           attacker
```

Key points: - Detection window (~50ms): Both victims briefly believe they have valid payments - Resolution: Both victims keep coins (inflationary, but victims aren't punished) - Penalty: Attacker loses 5× the double-spent amount plus all trust - Prevention alternative: "Wait for agreement" protocol delays finality but prevents the confusion window

7.6.2 Economic Stability

The "both keep coins" strategy (accept inflation, penalize attacker) shows:

Detection	Penalty	Inflation	Attacker Profit	Stable?
50%	5x	1.9%	-$985	YES
90%	5x	1.1%	-$1000	YES
99%	1x	4.7%	-$943	YES

Finding: Attackers always lose money because trust penalties outweigh gains. Economy is stable (inflation $< 5\%$) with 5x penalty multiplier even at 50% detection.

7.6.3 Finality Latency

Threshold	Connectivity	Median Latency	Success Rate
50%	0.3	0.14s	100%
70%	0.5	0.14s	100%
90%	0.7	0.14s	100%

Finding: Sub-200ms finality achievable. Higher thresholds don't proportionally increase latency because confirmations arrive in parallel through gossip.

7.6.4 Partition Behavior

During network partitions, double-spends can temporarily succeed (both victims accept). After healing, all conflicts are detected. This creates a "damage window" equal to partition duration.

Mitigation: Use "wait for agreement" protocol for high-value transactions during suspected partition conditions.

7.6.5 Currency Weight Spectrum

Connectivity	Weight	Category
0.9	0.14	Lightest (village-level trust)
0.5	0.34	Light (town-level)
0.1	0.80	Heaviest (needs blockchain bridge)

Conclusion: Currency weight is proportional to network performance. Better connectivity enables lighter trust mechanisms—the digital equivalent of physical proximity enabling village-level trust at global scale.

8. Discussion

The preceding sections described Omerta's architecture, trust model, economic mechanisms, defenses, and simulation results. This section steps back to discuss the broader context: where Omerta sits on the spectrum of trust-cost tradeoffs, what it achieves compared to alternatives, the limitations we acknowledge, and the methodological approach that enabled this design.

8.1 The Trust-Cost Spectrum

Different approaches occupy different positions on the trust-cost spectrum:

Approach	Trust Required	Cost	Practical Scope
Centralized cloud	High (trust provider)	Low	Broad
FHE	None	Extreme (1000x+)	Very narrow
TEE (SGX)	Medium (trust hardware)	Low-Medium	Medium
PoW blockchain	Low	High (energy)	Medium
PoS blockchain	Low-Medium	Medium (capital)	Medium
Omerta	Medium (trust earned)	Low	Broad

The key insight is that blockchain approaches—while genuinely reducing trust requirements—may not occupy the optimal point for compute markets. They pay significant costs (energy, capital, throughput limits) for global consensus that compute markets may not need.

8.2 What Blockchain Achieves

We should be precise about what blockchain consensus mechanisms accomplish:

Genuine achievements: - Coordination without designated coordinator - Resistance to unilateral censorship - Auditable history without trusted record-keeper - Credible monetary policy without central bank

These are real and valuable. PoW and PoS represent genuine advances in reducing trust requirements.

What the historical episodes reveal: - The spectrum has no zero-trust endpoint in practice - Social consensus remains available when stakes are high enough - This doesn't invalidate the achievements—it bounds them

8.3 Omerta's Position

Omerta bets that for compute markets specifically, we can relax global consensus while preserving the properties that matter:

Property	Blockchain Approach	Omerta Approach	Trade-off
Double-spend prevention	Global UTXO consensus	Escrow locks before session	Equivalent for sessions

Property	Blockchain Approach	Omerta Approach	Trade-off
History integrity	PoW/PoS difficulty	Trust-weighted writes	Weaker globally, sufficient locally
Sybil resistance	Computational/capital cost	Time cost (age)	Different attack economics
Censorship resistance	Anyone can mine/stake	Anyone above trust threshold	Requires earning entry

The hypothesis is that these trade-offs are acceptable for compute markets, where: - Transactions are bilateral (buyer-seller), not global transfers - Sessions are ephemeral, not permanent state - Verification is possible during execution - Reputation has natural meaning (did you deliver?)

8.4 Analogy to FHE vs. Ephemeral Compute

The relationship between Omerta and blockchain mirrors the relationship between ephemeral compute and FHE:

	Maximum Guarantee	Practical Alternative
Data privacy	FHE (compute on encrypted)	Ephemeral compute (brief exposure + verification)
Consensus	Global Byzantine (PoW/PoS)	Local trust (Omerta)
Cost	1000x+ overhead	Near-native performance
Accessibility	Narrow applications	Broad applicability

In both cases, relaxing the maximum guarantee enables serving far more users at practical cost. The question is whether the relaxed guarantee is sufficient for the use case.

8.5 Making the Social Layer Explicit

This tradeoff raises a deeper question: if we're accepting some trust requirements anyway, should we make them explicit rather than hidden?

Every distributed system ultimately has a social layer. Bitcoin's ledger has been modified by human coordination. Ethereum's code yielded to community override. The question is not whether humans are trusted, but which humans and how.

Omerta makes this explicit rather than hiding it beneath claims of mathematical trustlessness. Trust relationships are tracked on-chain. Incentives align through economics. Bad actors are identified over time. This mirrors how working human institutions operate—imperfect rules made workable through aligned incentives.

8.6 Interoperability Across the Spectrum

Making the social layer explicit also clarifies where different systems fit—and how they might work together.

Different points on the trust-cost spectrum suit different use cases. A mature ecosystem might include multiple networks that interoperate, with value and workloads flowing to appropriate trust levels.

Use cases by trust level:

Trust Level	Example Use Cases	Why This Level
Maximum (FHE/MPC)	Medical records analysis, financial audits, voting	Data must never be exposed, even briefly
High (PoW/PoS blockchain)	Digital currency, smart contracts with large stakes, cross-border settlements	Global consensus needed, high-value permanent state
Medium (Omerta-style)	General compute rental, batch processing, development environments, CI/CD	Bilateral transactions, ephemeral sessions, verification possible
Lower (reputation only)	Content delivery, caching, non-sensitive workloads	Speed matters more than guarantees, easy to verify after the fact

Cross-network flows:

Networks at different trust levels can bridge to each other:

- **Settlement on high-trust chains**: An Omerta-style network could settle periodic summaries to a PoS blockchain, gaining the permanence guarantees of global consensus for aggregate state while handling high-frequency bilateral transactions locally.
- **Escalation for disputes**: Normal compute sessions run on medium-trust infrastructure. Disputed sessions escalate to higher-trust arbitration—perhaps a smart contract on Ethereum that evaluates cryptographic evidence.
- **Sensitive workload isolation**: A pipeline might run preprocessing on Omerta (cheap, fast), then route sensitive computation to FHE or TEE enclaves, then aggregate results back on Omerta.
- **Trust bootstrapping**: New participants could establish initial reputation on a high-trust chain (where Sybil attacks are expensive), then bridge that identity to lower-cost networks.

A mature ecosystem stratifies: users and workloads flow to appropriate trust levels based on actual security requirements, paying only for the guarantees they need.

8.7 Scaling Trust: From Villages to Global Networks

Why does this spectrum exist at all? The trust-cost tradeoff is not unique to computer systems—it mirrors how human societies have always operated:

Society Scale	Trust Mechanism	Overhead
Village (50)	Everyone knows everyone; gossip spreads instantly	Minimal

Society Scale	Trust Mechanism	Overhead
Town (5,000)	Reputation networks; friends-of-friends	Low
City (500,000)	Institutions track reputation; courts enforce	Medium
Nation (50M+)	Anonymous transactions; verification required	High
Global	No shared context	Highest

As communities grew beyond the scale where everyone could know everyone, **visibility decreased** and **trust costs increased**. The lightweight mechanisms that worked in villages—verbal agreements, handshakes, reputation by gossip—don't scale to cities. Heavier mechanisms emerged: contracts, courts, banks, regulations.

The core problem: Traditional high-trust solutions required physical proximity and small scale. You could trust your neighbor because you'd see them tomorrow and everyone would know if they cheated you. *The village knew; the internet forgot; we're teaching it to remember.*

What Omerta provides: On-chain records replicate village-level visibility at global scale—everyone can see transaction history, reputation is computable from permanent records, and defection costs accumulated trust. New participants start with nothing and earn trust through behavior, exactly like newcomers to a village.

The key insight: **network performance is the digital analog of physical proximity**. Well-connected nodes can use lighter trust mechanisms, just as neighbors who see each other daily trust more easily. Currency weight scales with connectivity.

The freedom-trust tradeoff: This visibility comes at a cost. Villages had high trust precisely because they had low freedom—everyone knew your business, you couldn't easily leave, your reputation followed you everywhere. Omerta recreates these properties digitally: transactions are visible, identities are persistent, exit costs are high, and deviation is penalized.

This is not a bug—it is the mechanism by which trust scales. The system explicitly trades freedom for trust. Users who value anonymity, disposable identities, or the ability to "start fresh" should use different systems. Users who value counterparty trust, long-term reputation, and cooperation among strangers may find this tradeoff worthwhile.

The honest framing: Omerta is not a privacy technology. It is an anti-privacy technology that trades surveillance for trust. The design should be chosen knowingly, when that tradeoff serves the application's needs.

Avoiding village pathologies: Unlike villages with arbitrary social punishment, gossip, and hidden power structures, Omerta aims to **maximize freedom within the trust constraint**: only provably harmful actions affect scores, all inputs are visible on-chain, mechanisms are transparent and debatable, and no constraints exist on non-harmful behavior. The goal is *fair* surveillance: comprehensive but explicit, uniform, and challengeable.

Why now: Machine intelligence as enabler

Fair trust at scale was computationally intractable—modeling complex behavior, tuning parameters, explaining decisions, and defending against novel attacks required reasoning capacity that didn't

exist. Villages could be fair because scope was small; scaling to millions of participants required machine intelligence to handle the complexity.

This explains why trust systems developed in the 2000s (EigenTrust, TidalTrust, FIRE) remained academic exercises. The theory was sound, but practical deployment required handling edge cases, explaining decisions, and adapting to attacks in ways that exceeded human capacity to manage at scale. These systems also lacked economic integration—trust scores that affected nothing had no reason to exist outside research papers.

Machine intelligence enables: behavioral modeling at scale (distinguishing honest mistakes from malice), continuous parameter tuning through simulation, explanation generation for trust decisions, and adversarial reasoning against attacks.

The relationship is recursive: **machine intelligence both demands the compute that Omerta provides and enables the trust system that makes Omerta work**. Machine intelligence needs distributed compute; distributed compute needs trust mechanisms; trust mechanisms at scale need machine intelligence to operate fairly. This virtuous cycle suggests the timing is not coincidental—the technologies arrive together because each enables the others.

Machine intelligence is both the demand and the supply: it hungers for compute and enables the trust systems that make sharing compute work.

8.8 Philosophy of Law: What Makes a Good Rule?

The question of what makes a good law has occupied philosophers for millennia. In 1964, legal philosopher Lon Fuller articulated eight principles that any genuine legal system must satisfy—what he called "the inner morality of law" [52]. Fuller's principles represent the minimum conditions for rules to function as law at all:

1. **Generality**: Laws must be general rules, not ad hoc commands
2. **Promulgation**: Laws must be publicly announced
3. **Prospectivity**: Laws must apply to future conduct, not past
4. **Clarity**: Laws must be understandable
5. **Non-contradiction**: Laws must not contradict each other
6. **Possibility**: Laws must be possible to obey
7. **Constancy**: Laws must be relatively stable over time
8. **Congruence**: Official action must match declared rules

These seem obvious. Yet in practice, human legal systems fail nearly all of them.

The gap between law and practice: The U.S. federal criminal code contains over 4,450 crimes, with estimates of hundreds of thousands more scattered across regulatory provisions. The Code of Federal Regulations spans over 175,000 pages—far more than any citizen could read, let alone understand. Legal scholar William Stuntz observed that overcriminalization has created "a world in which the law on the books makes everyone a felon, and in which prosecutors and police both define the law on the street" [53].

This isn't hyperbole. In oral argument before the Supreme Court, a government lawyer explained that a federal statute criminalized any ethical lapse in the workplace. Justice Stephen Breyer responded: "There are 150 million workers in the United States. I think possibly 140 million of them flunk your test." The government lawyer did not deny it.

Why laws fail Fuller's principles: The root cause is human limitation. Legislatures lack the capacity to anticipate all cases, so they write broad rules. Enforcement agencies lack the resources to prosecute everyone, so they exercise discretion. Courts lack the bandwidth to hear every case, so most pleas are negotiated. The result: laws that are unclear (violating #4), impossible to fully obey (violating #6), and enforced incongruently with their text (violating #8).

The failure is not corruption—it's constraint. Humans cannot write rules clear enough to be mechanically applied. Humans cannot monitor compliance at scale. Humans cannot adjudicate consistently across millions of cases. So we accept a system where the written law bears little relation to the enforced law, where everyone is technically guilty, and where actual punishment depends on prosecutorial whim.

This gap between written and enforced law is not merely inefficient—it inverts the rule of law's purpose. Instead of protecting citizens from arbitrary power, the legal system becomes a tool of arbitrary power. When everyone is guilty of something, enforcement becomes selection. And selection invites bias, corruption, and abuse.

What machine intelligence enables: Omerta represents an exploration of what governance might look like without these human limitations. Not a replacement for human law—Omerta governs a narrow domain (compute market transactions)—but a demonstration that Fuller's principles can actually be achieved when machines handle the complexity humans cannot.

Consider how Omerta's rules satisfy Fuller's principles:

Principle	Human Legal System	Omerta
Generality	General in text, specific in enforcement	Identical rules for all participants
Promulgation	Buried in thousands of pages	On-chain, machine-readable, open source
Prospectivity	Generally yes	Strictly enforced—no retroactive changes
Clarity	Ambiguous, requires lawyers	Deterministic code—same input, same output
Non-contradiction	Rampant conflicts	Single consistent ruleset
Possibility	Often impossible to know all rules	Rules are few, explicit, verifiable
Constancy	Constant legislative churn	Parameter changes require trust-weighted consensus
Congruence	Selective enforcement	Automatic, uniform enforcement

The last principle—congruence—is the most significant. In Omerta, there is no gap between written

and enforced law. If the rules say double-spending incurs a 5× penalty, then every detected double-spend incurs exactly that penalty. No prosecutorial discretion. No selective enforcement. No plea bargains. The rule is the reality.

New principles for machine-enforced governance: Fuller's principles assumed human limitations. With machine intelligence handling complexity, we can articulate stronger principles:

1. **Determinism**: Given the same facts, the same outcome must result. No room for "judgment calls."
2. **Auditability**: Every decision must be traceable to specific facts and rules. No black boxes.
3. **Proportionality**: Penalties must scale with harm. Machines can calculate precisely; they need not round to convenient categories.
4. **Completeness**: Rules must cover all cases. No "gaps" requiring human interpretation.
5. **Immediacy**: Enforcement should occur as close to the violation as possible. Delayed justice is weakened justice.
6. **Transparency**: All inputs to decisions must be visible to affected parties. No hidden evidence.
7. **Contestability**: Affected parties must be able to challenge decisions using the same facts and rules.

Omerta attempts to embody these principles. Trust scores are deterministic—anyone can verify the calculation. Penalties are proportional—severity scales with transaction value and impact. Enforcement is immediate—double-spend detection triggers penalties automatically. All inputs are on-chain—no hidden information.

The honest limitation: These principles work for narrow, well-defined domains like compute market transactions. They cannot replace human judgment for complex social questions—criminal intent, contract interpretation, constitutional rights. Omerta governs a domain where "harm" is objectively measurable (did you deliver the compute you promised?). Most of law deals with questions machines cannot yet answer.

But the existence of a domain where Fuller's principles can be fully achieved—where the gap between written and enforced law closes to zero—suggests something important. The failures of human legal systems are not inherent to governance itself. They are artifacts of human limitation. As machine intelligence expands, the domains where principled governance is achievable will expand with it.

The village didn't need lawyers because everyone knew the rules. We're building a global village where the rules are known because they're code.

8.9 Methodological Notes

Given this reliance on machine intelligence capabilities, we should be transparent about how this paper itself was developed.

This paper was developed through human-machine collaboration (specifically with Claude), demonstrating the thesis that machine intelligence enables previously intractable system design. Economic models, attack analyses, and simulations were developed iteratively—machine intelligence provided

rapid prototyping and edge case identification; humans provided direction, validation, and judgment.

Methodology: The simulation approach draws on agent-based computational economics [26-30], following established validation practices: parameter sweeps, sensitivity analysis, and comparison against theoretical predictions. Market mechanisms build on auction theory [31] and matching markets [32]; trust propagation relates to reputation network effects [33, 34].

Tradeoffs: Machine-intelligence-assisted analysis enables rapid parameter exploration, consistent reasoning across long documents, and systematic attack enumeration. Risks include hallucination, training data limitations, and sycophancy bias. We mitigate these through executable simulations (claims grounded in verifiable outputs), human review, and adversarial prompting.

The honest position: This paper represents our best current understanding. We expect some aspects to be wrong and invite scrutiny. By making our process transparent—including machine intelligence involvement—we trade authorial mystique for verifiability.

8.10 Limitations

In that spirit of honesty, we must acknowledge what Omerta does not solve and where the design may be incomplete.

Bootstrap Problem: The network requires initial trusted participants to establish the trust graph. Genesis block contents define starting conditions.

Sophisticated Attackers: Well-resourced attackers willing to invest years in building reputation before striking remain a threat. No system fully prevents long-con attacks.

Parameter Sensitivity: Optimal parameter values require empirical tuning and may vary across network conditions.

Trust Propagation Threshold: Research on trust transitivity [43] shows that in large networks, trust propagation requires a non-zero fraction of "absolute trust" edges (complete confidence) or average pairwise trust collapses to zero. Omerta uses continuous trust values with decay. Whether this satisfies the threshold for robust propagation at scale requires further analysis; finite networks may be more resilient than the theoretical bound suggests.

Verification Overhead: Random audits impose costs on honest participants. The verification rate must balance security against efficiency.

Machine-Intelligence-Assisted Design Uncertainty: As discussed in Section 8.9, machine-intelligence-assisted analysis carries risks of hallucination and training data limitations. While we have attempted to mitigate these through executable simulations and adversarial review, some errors may remain undetected.

Fundamental Sybil Limits: Douceur [6] proved that Sybil attacks are fundamentally unsolvable without trusted identity verification. Omerta's defenses (age, cluster detection, economic penalties) make Sybil attacks expensive but not impossible. A patient, well-resourced adversary who pre-creates identities years in advance can eventually attack with mature identities. We mitigate this through continuous behavioral monitoring, but acknowledge the theoretical limitation.

Existing Compute Market Struggles: Decentralized compute platforms built on blockchain (Golem [15], iExec [16], Render Network) have operated for years but struggled with utilization

and adoption. This suggests challenges beyond trust mechanisms—possibly network effects, user experience, or fundamental market structure issues. Omerta may face similar challenges regardless of its trust system design. We cannot assume that better trust mechanisms alone will solve adoption problems.

Despite these limitations, we believe there are reasons for cautious optimism.

Why Omerta Might Succeed Where Others Struggled: While acknowledging the above challenges, several factors differentiate Omerta's approach:

1. *Zero marginal cost for providers*: Home computers sit idle most of the time—evenings, weekends, workdays. The marginal cost of running Omerta is effectively zero: electricity that would be consumed anyway, hardware already purchased, internet already paid for. Unlike mining which consumes significant additional power, providing idle compute requires only installing free software. The barrier to entry is a download, not an investment.

2. *Integration with existing infrastructure*: Omerta is designed to work with standard virtualization (KVM, Docker) and existing network configurations. Providers don't need specialized hardware, dedicated IP addresses, or complex configuration. The system handles NAT traversal, resource detection, and verification automatically. If you can run a VM, you can be a provider.

3. *Consumer-side simplicity*: Consumers see simple pricing ("8 cores = 0.09 OMC/hr") without needing to understand trust mechanisms, blockchain, or market dynamics. The complexity is hidden. Integration targets existing workflows—SSH access, standard tooling, familiar interfaces.

4. *Machine intelligence as the customer*: Prior decentralized compute markets targeted human developers who are price-sensitive, quality-demanding, and have alternatives. Omerta targets machine intelligence workloads that are: (a) less sensitive to latency variance, (b) naturally parallelizable, (c) fault-tolerant by design, and (d) generated in unbounded quantity. Machine intelligences don't complain about occasional failures—they retry automatically.

5. *Open source with no platform fees*: The Omerta software is free. There is no company extracting rent, no token appreciation to fund, no investors requiring returns. Providers keep what they earn (minus trust-based burns that fund the network). This removes the economic friction that plagued VC-funded alternatives.

6. *No preallocation, no founder stake*: Unlike many cryptocurrency projects that reserve tokens for founders, early investors, or development funds, Omerta's coin distribution is entirely trust-based from genesis. The code is given away. There is no hidden stake waiting to dump on participants. Less greed, more alignment—everyone earns their position through contribution.

7. *Economic integration the academy lacked*: Prior trust systems (EigenTrust, TidalTrust, FIRE) were purely academic—they computed trust scores that connected to nothing. Omerta integrates trust directly with economic mechanisms: payment splits, transfer burns, coin distribution. Trust has consequences. This gives the system a reason to exist beyond academic citation.

These factors don't guarantee success, but they address specific failure modes observed in prior systems: high barriers to entry, poor user experience, mismatched customer base, extractive plat-

form economics, and lack of economic integration. Prior trust research gave us the theory. Prior blockchain projects showed us the pitfalls. Omerta attempts to learn from both.

Machine Intelligence Demand Thesis is Speculative: Our argument that machine intelligence creates unbounded compute demand (Section 7.5) is forward-looking and unproven. Current machine intelligence demand is large but not literally unbounded. The thesis depends on assumptions about machine intelligence capability trajectories that may not hold. If machine intelligence development stalls or compute efficiency improves faster than demand grows, the perpetual undersupply assumption fails, and with it the economic model for home provider value creation.

Detection vs. Prevention: As clarified in Section 7.6.1, Omerta detects double-spends but does not prevent them. The system relies on economic penalties making attacks unprofitable, not on making attacks impossible. This is a weaker guarantee than blockchain consensus provides. For applications requiring absolute prevention of double-spending, blockchain remains more appropriate.

9. Conclusion

Trustlessness exists on a spectrum. Proof-of-work and proof-of-stake mechanisms genuinely reduce trust requirements compared to centralized alternatives—this is a real achievement. But they do so at significant cost, and historical episodes demonstrate that no practical system reaches the zero-trust endpoint.

Omerta explores a different point on this spectrum. Rather than paying for global consensus that compute markets may not need, Omerta computes trust locally based on verifiable on-chain data. The hypothesis is that for bilateral compute transactions—where sessions are ephemeral, verification is possible during execution, and reputation has natural meaning—local trust provides sufficient security at dramatically lower cost.

This parallels the choice between fully homomorphic encryption and ephemeral compute. FHE provides the ultimate guarantee but at 1000x+ overhead, restricting practical use. Ephemeral compute accepts some trust requirements in exchange for serving far more use cases. Similarly, blockchain consensus provides strong guarantees but at costs (energy, capital, throughput) that may exceed what compute markets require.

Our simulation studies validate several key claims. Automated policy mechanisms respond appropriately to detected threats, though parameter adjustment alone cannot counter structural attacks—effective systems need architectural defenses complemented by policy fine-tuning. Double-spend resolution simulations confirm that currency "weight" scales with network performance: well-connected networks achieve 100% detection with sub-200ms finality, while poorly-connected networks require heavier mechanisms. The system degrades gracefully across this spectrum.

Our economic simulations demonstrate that unreliable home compute creates genuine value—not merely redistributes it—when demand exceeds datacenter capacity. In undersupplied markets, home providers serve consumers that datacenters cannot reach, increasing total compute delivered by 200% while datacenter profits remain largely intact. The viability of this model depends on perpetual undersupply.

We argue that machine intelligence guarantees this undersupply. Unlike human demand, which is bounded and tends toward equilibrium, machine intelligence creates unbounded demand for compute at any quality level. There is always a next-best task—a machine that cannot profitably do a $50/hr task can still generate value doing a $5/hr task. As machine intelligences improve, this demand curve extends to arbitrarily low prices, ensuring that any compute capacity finds productive use.

But machine intelligence plays a deeper role than just creating demand. Fair trust systems at scale—systems that penalize only provable misbehavior through transparent, debatable mechanisms—were computationally intractable until now. Modeling human behavior, detecting novel attacks, explaining decisions, and tuning parameters requires enormous reasoning capacity. Machine intelligence provides this capacity for the first time. The relationship is recursive: machine intelligence demands the compute that Omerta provides, and machine intelligence enables the trust system that makes Omerta work. This virtuous cycle suggests the technologies arrive together because each enables the others.

This framing illuminates what Omerta attempts: extending village-level trust to global scale. Villages had high trust because they had high visibility—everyone knew everyone's business, reputation spread by gossip, misbehavior had lasting consequences. Omerta recreates these properties

digitally through on-chain transparency. But unlike villages with their arbitrary social punishment, gossip, and hidden power structures, Omerta aims to maximize freedom within the trust constraint. Only provably anti-social behavior affects trust scores. All mechanisms are documented and debatable. Participants retain freedom for any behavior that doesn't demonstrably harm others.

The goal is not to replace blockchain systems—they serve real purposes and represent genuine advances. The goal is to expand the design space, recognizing that different applications may have different optimal points on the trust-cost spectrum. For compute markets specifically—especially those serving machine intelligence workloads—we believe there is an underexplored region that could make trustworthy compute sharing accessible to more people at practical cost.

Omerta is an experiment in finding that region. The question is not whether unreliable compute will displace datacenters, but whether we can deploy enough compute of any quality to satisfy the exponentially growing demand of machine intelligence—and whether machine intelligence can, in turn, help us build the fair trust systems needed to make that deployment work.

The pieces are falling into place. Billions of computers sit idle, their owners having already paid for hardware, electricity, and internet—zero marginal cost waiting to be unlocked. Machine intelligence creates unbounded demand for exactly the kind of fault-tolerant, parallelizable workloads that unreliable compute can serve. And for the first time, machine intelligence provides the reasoning capacity to build fair trust systems at scale. Omerta is open source, free to use, with no platform extracting rent. The barrier to participation is a download.

Prior trust systems remained academic exercises—elegant theory with no economic consequences and no killer application. Prior decentralized compute markets struggled because they targeted the wrong customers with the wrong economics at the wrong time. Omerta learns from both: it takes the trust theory seriously, integrates it with real economics, gives the software away for free with no preallocation, and targets the one customer that can consume infinite compute—machine intelligence itself.

We don't need zero trust. We need enough trust—earned, local, and proportional to what's at stake.

References

[1] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45-48, 2000. https://doi.org/10.1145/355112.355122

[2] C. Dellarocas, "The digitization of word of mouth: Promise and challenges of online feedback mechanisms," *Management Science*, vol. 49, no. 10, pp. 1407-1424, 2003. https://doi.org/10.1287/mnsc.49.10.1407.17308

[3] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. WWW*, 2003. https://doi.org/10.1145/775152.775242

[4] L. Xiong and L. Liu, "PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE TKDE*, vol. 16, no. 7, pp. 843-857, 2004. https://doi.org/10.1109/TKDE.2004.1318566

[5] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *Proc. NSDI*, 2006. https://www.usenix.org/conference/nsdi-06/experience-object-reputation-system-peer-peer-filesharing

[6] J. R. Douceur, "The Sybil attack," in *Proc. IPTPS*, 2002. https://doi.org/10.1007/3-540-45748-8_24

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. https://bitcoin.org/bitcoin.pdf

[8] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. https://peercoin.net/assets/paper/peercoin-paper.pdf

[9] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: Defending against Sybil attacks via social networks," *ACM SIGCOMM*, 2006. https://doi.org/10.1145/1159913.1159945

[10] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil nodes using social networks," in *Proc. NDSS*, 2009. https://www.ndss-symposium.org/ndss2009/sybilinfer-detecting-sybil-nodes-using-social-networks/

[11] D. Larimer, "Delegated proof-of-stake (DPOS)," *Bitshare whitepaper*, 2014. https://bitshares.org/technology/d
proof-of-stake-consensus/

[12] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, 1999. https://www.usenix.org/conference/osdi-99/practical-byzantine-fault-tolerance

[13] D. P. Anderson, "BOINC: A system for public-resource computing and storage," in *Proc. Grid*, 2004. https://doi.org/10.1109/GRID.2004.14

[14] V. S. Pande et al., "Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing," *Biopolymers*, vol. 68, no. 1, pp. 91-109, 2003. https://doi.org/10.1002/bip.10219

[15] The Golem Project, "The Golem whitepaper," 2016. https://golem.network/

[16] iExec, "iExec: Blockchain-based decentralized cloud computing," 2017. https://iex.ec/wp-content/uploads/2022/09/iexec_whitepaper.pdf

[17] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *Proc. EC*, 2004. https://doi.org/10.1145/988772.988788

[18] R. Jurca and B. Faltings, "Collusion-resistant, incentive-compatible feedback payments," in *Proc. EC*, 2007. https://doi.org/10.1145/1250910.1250940

[19] G. E. Bolton, B. Greiner, and A. Ockenfels, "Engineering trust: Reciprocity in the production of reputation information," *Management Science*, vol. 59, no. 2, pp. 265-285, 2013. https://doi.org/10.1287/mnsc.1120.1609

[20] D. E. Denning, "An intrusion-detection model," *IEEE TSE*, vol. 13, no. 2, pp. 222-232, 1987. https://doi.org/10.1109/TSE.1987.232894

[21] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. STOC*, 2009. https://doi.org/10.1145/1536414.1536440

[22] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?," in *Proc. CCSW*, 2011. https://doi.org/10.1145/2046660.2046682

[23] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptology ePrint Archive*, 2016. https://eprint.iacr.org/2016/086

[24] J. Van Bulck et al., "Foreshadow: Extracting the keys to the Intel SGX kingdom," in *Proc. USENIX Security*, 2018. https://www.usenix.org/conference/usenixsecurity18/presentation/bulck

[25] A. C. Yao, "How to generate and exchange secrets," in *Proc. FOCS*, 1986. https://doi.org/10.1109/SFCS.1986.

[26] L. Tesfatsion and K. L. Judd, Eds., *Handbook of Computational Economics, Vol. 2: Agent-Based Computational Economics*. North-Holland, 2006. https://doi.org/10.1016/S1574-0021(05)02016-2

[27] J. D. Farmer and D. Foley, "The economy needs agent-based modelling," *Nature*, vol. 460, no. 7256, pp. 685-686, 2009. https://doi.org/10.1038/460685a

[28] W. B. Arthur, "Complexity and the economy," *Science*, vol. 284, no. 5411, pp. 107-109, 1999. https://doi.org/10.1126/science.284.5411.107

[29] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, Eds., *Algorithmic Game Theory*. Cambridge University Press, 2007. https://www.cambridge.org/core/books/algorithmic-game-theory/0092C07CA8B724E1B1BE2238DDD66B38

[30] P. Windrum, G. Fagiolo, and A. Moneta, "Empirical validation of agent-based models: Alternatives and prospects," *Journal of Artificial Societies and Social Simulation*, vol. 10, no. 2, 2007. https://www.jasss.org/10/2/8.html

[31] P. Klemperer, "Auction theory: A guide to the literature," *Journal of Economic Surveys*, vol. 13, no. 3, pp. 227-286, 1999. https://doi.org/10.1111/1467-6419.00083

[32] A. E. Roth, "The economics of matching: Stability and incentives," *Mathematics of Operations Research*, vol. 7, no. 4, pp. 617-628, 1982. https://doi.org/10.1287/moor.7.4.617

[33] C. Dellarocas, "Reputation mechanism design in online trading environments with pure moral hazard," *Information Systems Research*, vol. 16, no. 2, pp. 209-230, 2005. https://doi.org/10.1287/isre.1050.0054

[34] O. E. Williamson, "Transaction cost economics: How it works; where it is headed," *De Economist*, vol. 146, no. 1, pp. 23-58, 1998. https://doi.org/10.1023/A:1003263908567

[35] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 119-154, 2006. https://doi.org/10.1007/s10458-005-6825-4

[36] A. Jøsang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*. Springer, 2016. https://doi.org/10.1007/978-3-319-42337-1

[37] R. Zhou and K. Hwang, "PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 4, pp. 460-473, 2007. https://doi.org/10.1109/TPDS.2007.1021

[38] E. Budish, "The economic limits of Bitcoin and the blockchain," *Quarterly Journal of Economics*, 2024. (Originally NBER Working Paper 24717, 2018.) https://doi.org/10.1093/qje/qjae033

[39] J. S. Gans and N. Gandal, "More (or less) economic limits of the blockchain," *CEPR Discussion Paper*, 2019. https://cepr.org/publications/dp14154

[40] S. Seuken and D. C. Parkes, "Sybil-proof accounting mechanisms with transitive trust," in *Proc. AAMAS*, pp. 205-212, 2014. https://www.ifaamas.org/Proceedings/aamas2014/aamas/p205.pdf

[41] M. O. Jackson, *Social and Economic Networks*. Princeton University Press, 2008. https://press.princeton.edu/books/paperback/9780691148205/social-and-economic-networks

[42] J. Golbeck, "Computing and applying trust in web-based social networks," PhD dissertation, University of Maryland, 2005. https://drum.lib.umd.edu/items/28ef1e3b-eb79-4586-83f1-e18b10694750

[43] O. Richters and T. P. Peixoto, "Trust transitivity in social networks," *PLoS ONE*, vol. 6, no. 4, e18384, 2011. https://doi.org/10.1371/journal.pone.0018384

[44] E. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *J. Economics & Management Strategy*, vol. 10, no. 2, pp. 173-199, 2001. https://doi.org/10.1111/j.1430-9134.2001.00173.x

[45] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A near-optimal social network defense against Sybil attacks," in *Proc. IEEE S&P*, 2008. https://doi.org/10.1109/SP.2008.13

[46] B. Nasrulin and G. Ishmaev, "MeritRank: Sybil tolerant reputation for merit-based tokenomics," arXiv:2207.09950, 2022. https://arxiv.org/abs/2207.09950

[47] S. Tadelis, "Reputation and feedback systems in online platform markets," *Annual Review of Economics*, vol. 8, pp. 321-340, 2016. https://doi.org/10.1146/annurev-economics-080315-015325

[48] M. Florian, S. Henningsen, C. Ndolo, and B. Scheuermann, "The sum of its parts: Analysis of federated byzantine agreement systems," *Distributed Computing*, vol. 35, pp. 399-417, 2022. https://doi.org/10.1007/s00446-022-00430-0

[49] M. Meulpolder, J. Pouwelse, D. Epema, and H. Sips, "BarterCast: A practical approach to prevent lazy freeriding in P2P networks," in *Proc. HoT-P2P*, 2009. https://doi.org/10.1109/IPDPS.2009.5160954

[50] J. Ruderman, "The Advogato trust metric is not attack-resistant," 2005. Available: https://www.squarefree.com/2005/05/26/advogato/

[51] P. Otte, M. de Vos, and J. Pouwelse, "TrustChain: A Sybil-resistant scalable blockchain," *Future Generation Computer Systems*, vol. 107, pp. 770-780, 2020. https://doi.org/10.1016/j.future.2017.08.048

[52] L. L. Fuller, *The Morality of Law*, rev. ed., New Haven: Yale University Press, 1969. https://yalebooks.yale.edu/book/9780300010701/the-morality-of-law/

[53] W. J. Stuntz, "The Pathological Politics of Criminal Law," *Michigan Law Review*, vol. 100, no. 3, pp. 505-600, 2001. https://repository.law.umich.edu/mlr/vol100/iss3/2/

Appendix: Key Parameters

Parameter	Description	Typical Range
K_PAYMENT	Payment curve slope	0.01 - 0.10
K_TRANSFER	Transfer burn slope	0.01 - 0.10
AGE_MATURITY_DAYS	Days to full trust potential	90 - 180
DAILY_MINT	Coins minted per day	Decreasing schedule
ISOLATION_THRESHOLD	Sybil cluster detection	0.7 - 0.9
TRANSITIVITY_DECAY	Trust decay per hop	0.5 - 0.8
DAMPENING_FACTOR	Policy adjustment scaling	0.1 - 0.5

Appendix: OMT Transaction DSL Specification

The Omerta Transaction DSL (OMT) is a domain-specific language for formally specifying distributed transaction protocols as communicating state machines. This appendix provides the complete language specification.

Lexical Structure

Comments

Single-line comments begin with `#` and extend to end of line:

```
# This is a comment
```

Identifiers

Identifiers consist of letters, digits, and underscores, starting with a letter:

- `UPPER_CASE`: Constants, enum values, built-in functions
- `CamelCase`: Type names, actor names, message names
- `lower_case`: Variables, fields, parameters

Keywords

Reserved keywords:

`transaction`	`imports`	`parameters`	`enum`
`block`	`message`	`actor`	`function`
`store`	`trigger`	`state`	`native`
`by`	`from`	`to`	`signed`
`in`	`on`	`when`	`auto`
`else`	`timeout`	`initial`	`terminal`

Literals

- **Integer**: `42`, `1000`
- **Float**: `3.14`, `0.67`
- **String**: `"quoted text"`
- **Boolean**: `true`, `false`
- **Null**: `null`

Type System

Primitive Types

Type	Description
`uint`	Unsigned integer
`int`	Signed integer
`float`	Floating-point number
`bool`	Boolean

Type	Description
`string`	UTF-8 string
`bytes`	Byte array
`hash`	256-bit hash value
`peer_id`	Peer identifier (hash of public key)
`timestamp`	Unix timestamp
`any`	Dynamic type
`dict`	Key-value mapping

Compound Types

- **List**: `list<peer_id>`, `list<uint>`
- **Map**: `map<string, uint>`, `map<peer_id, float>`

Top-Level Declarations

Transaction Declaration

```
transaction <id> "<name>" "<description>"
```

Example:

```
transaction 00 "Escrow Lock" "Lock funds for compute session"
```

Import Declaration

```
imports <path>
```

Imports definitions from another OMT file.

Parameters Block

```
parameters (
    <NAME> = <value> <unit>? "<description>"
    ...
)
```

Units: `seconds`, `count`, `fraction`

Example:

```
parameters (
    WITNESS_COUNT = 5 count "Number of witnesses"
    TIMEOUT = 300 seconds "Session timeout"
    THRESHOLD = 0.67 fraction "Consensus threshold"
)
```

Enum Declaration

```
enum <Name> "<description>" (
    VALUE1
    VALUE2
```

```
    ...
)
```

Block Declaration

Blocks represent on-chain records:

```
block <NAME> by [<Actor1>, <Actor2>] (
    <field_name>    <type>
    ...
)
```

Message Declaration

Messages are signed communications between actors:

```
message <NAME> from <Actor> to [<Recipient>, ...] signed? (
    <field_name>    <type>
    ...
)
```

Function Declaration

```
function <NAME>(<param> <type>, ...) -> <return_type> (
    <statements>
)
```

Actor Declaration

Actors are state machines that process messages:

```
actor <Name> "<description>" (
    store (
        <field_name>    <type>
        ...
    )

    trigger <name>(<params>) in [<STATE1>, ...] "<description>"

    state <NAME> initial? terminal? "<description>"

    <transitions>
)
```

Transitions

```
<FROM_STATE> -> <TO_STATE> on <trigger> when <guard>? (
    <actions>
) else -> <ALT_STATE> (
    <actions>
)
```

Trigger types: - `on <MESSAGE_NAME>`: Triggered by receiving a message - `on <trigger_name>`: Triggered by external trigger - `on timeout(<PARAM>)`: Triggered after timeout - `auto`: Automatic transition (no trigger)

Actions

Action	Syntax	Description
Store fields	`store field1, field2`	Store from message
Store computed	`STORE(name, expr)`	Store computed value
Compute	`name = expr`	Compute and store
Send	`SEND(target, MESSAGE)`	Send message to peer
Broadcast	`BROADCAST(list, MESSAGE)`	Send to multiple peers
Append	`APPEND(chain, BLOCK)`	Append block to chain
Append list	`APPEND(list, value)`	Append to list

Expressions

Operators

Category	Operators
Arithmetic	`+`, `-`, `*`, `/`
Comparison	`==`, `!=`, `<`, `>`, `<=`, `>=`
Logical	`AND`, `OR`, `NOT`
Access	`.` (field), `[]` (index)

Conditional

```
IF <condition> THEN <expr> ELSE <expr>
```

Lambda

```
param => expr
```

Used with `FILTER` and `MAP`.

Built-in Functions

Function	Signature	Description
`HASH`	`(any, ...) -> hash`	Cryptographic hash
`SIGN`	`(hash) -> bytes`	Sign with actor's key
`VERIFY_SIG`	`(bytes, peer_id) -> bool`	Verify signature
`NOW`	`() -> timestamp`	Current time
`LENGTH`	`(list<T>) -> uint`	List length
`FILTER`	`(list<T>, T -> bool) -> list<T>`	Filter list
`MAP`	`(list<T>, T -> U) -> list<U>`	Transform list
`GET`	`(map<K,V>, K, V?) -> V`	Map lookup with default

Function	Signature	Description
CONTAINS	(list<T>, T) -> bool	List membership
LOAD	(identifier) -> any	Load from store
RETURN	(expr)	Return from function

Semantic Constraints

1. Each actor must have exactly one `initial` state
2. Each actor should have at least one `terminal` state
3. All states must be reachable from the initial state
4. All referenced types, messages, and states must be defined
5. Triggers can only fire in their declared `in` states
6. The `message` keyword refers to the received message in transitions
7. The `peer_id` keyword refers to the actor's own identity

Example: Simplified Escrow Lock

```
transaction 00 "Escrow Lock" "Lock funds for compute session"

parameters (
    WITNESS_COUNT = 5 count "Number of witnesses"
    LOCK_TIMEOUT = 300 seconds "Time to complete lock"
)

enum LockStatus (
    ACCEPTED
    REJECTED
)

block BALANCE_LOCK by [Consumer, Witness] (
    session_id       hash
    amount           uint
    timestamp        timestamp
)

message LOCK_INTENT from Consumer to [Provider] signed (
    session_id       hash
    amount           uint
)

message LOCK_RESULT from Witness to [Consumer] signed (
    session_id       hash
    status           LockStatus
)

actor Consumer "Party paying for service" (
    store (
```

```
        session_id       hash
        amount           uint
    )

    trigger initiate_lock(provider peer_id, amount uint) in [IDLE]

    state IDLE initial "Waiting to initiate"
    state WAITING "Waiting for result"
    state LOCKED terminal "Funds locked"
    state FAILED terminal "Lock failed"

    IDLE -> WAITING on initiate_lock when amount > 0 (
        session_id = HASH(peer_id, provider, NOW())
        store amount
        SEND(provider, LOCK_INTENT)
    )

    WAITING -> LOCKED on LOCK_RESULT when message.status == LockStatus.ACCEPTED (
        APPEND(chain, BALANCE_LOCK)
    )

    WAITING -> FAILED on LOCK_RESULT when message.status == LockStatus.REJECTED ()

    WAITING -> FAILED on timeout(LOCK_TIMEOUT) ()
)
```

Appendix B: Formal Specification of OMT

This appendix presents a formal specification of the Omerta Transaction Language (OMT) in the style of academic programming language papers. We define the abstract syntax via BNF grammar, static semantics via well-formedness judgments, and dynamic semantics via a labeled transition system. We then discuss the relationship to multiparty session types and identify directions for future formal development.

B.1 Abstract Syntax

We present the abstract syntax of OMT using BNF notation. Metavariables are written in italics; terminal symbols in **bold** or `monospace`.

Syntactic Categories

Metavariable	Domain
x, *y*, *z*	Identifiers
n	Integer literals
s	String literals
T	Transaction definitions
A	Actor definitions
S, *S'*	State names
M	Message types
B	Block types
Type	Types
e	Expressions
a	Actions

Transaction Structure

```
T ::= transaction n s1 s2 D*            (transaction definition)

D ::= imports x                         (import declaration)
    | parameters ( P* )                 (parameter block)
    | enum x s? ( C* )                  (enumeration)
    | block x by [ R* ] ( F* )          (block type)
    | message x from R to [ R* ] Sig? ( F* )  (message type)
    | function x ( Param* ) -> Type ( Body ) (function)
    | actor x s ( ActorBody )           (actor definition)

P ::= x = n Unit? s?                    (parameter)
C ::= x                                 (enum case)
F ::= x Type                            (field declaration)
R ::= x                                 (role name)
Sig ::= signed                          (signature requirement)
```

Actor Structure

```
ActorBody ::= store ( F* ) Trigger* State* Trans*

Trigger ::= trigger x ( Param* ) in [ S* ] s?

State ::= state S Mod* s?
Mod   ::= initial | terminal

Trans ::= S -> S' Event Guard? ( a* ) ElseBranch?

Event ::= on x                          (message or trigger)
        | on timeout ( x )              (timeout)
        | auto                          (automatic)

Guard ::= when e

ElseBranch ::= else -> S' ( a* )
```

Types

```
Type ::= uint | int | bool | string     (primitive types)
       | hash | bytes | signature       (cryptographic types)
       | peer_id | timestamp            (protocol types)
       | list < Type >                  (list type)
       | map < Type , Type >            (map type)
       | dict                           (dictionary type)
       | x                              (named type / enum)
```

Expressions

```
e ::= n | s | true | false | null       (literals)
    | x                                 (variable)
    | e.x                               (field access)
    | e [ e ]                           (index access)
    | e BinOp e                         (binary operation)
    | NOT e                             (negation)
    | IF e THEN e ELSE e                (conditional)
    | f ( e* )                          (function call)
    | x => e                            (lambda)
    | { F* }                            (record literal)

BinOp ::= + | - | * | /                 (arithmetic)
        | == | != | < | > | <= | >=      (comparison)
        | AND | OR                      (logical)
```

Actions

```
a ::= store x*                          (store from message)
```

```
| STORE ( x , e )                    (store computed)
| x = e                              (assign)
| SEND ( e , M )                     (send message)
| BROADCAST ( e , M )                (broadcast)
| APPEND ( x , e )                   (append to list/chain)
```

B.2 Well-Formedness Judgments (Static Semantics)

We define well-formedness as a collection of judgments that must hold for a transaction definition to be valid. These judgments are analogous to typing rules in traditional type systems, but verify structural properties rather than expression types.

Environments

Let Γ denote a *transaction environment* containing:

- Γ.params : $x \to (n, unit, desc)$ — parameter definitions
- Γ.enums : $x \to C^*$ — enumeration cases
- Γ.messages : $M \to (from, to, fields)$ — message schemas
- Γ.blocks : $B \to (roles, fields)$ — block schemas
- Γ.functions : $f \to (params, return, body)$ — function definitions
- Γ.actors : $A \to ActorEnv$ — actor environments

Let Δ denote an *actor environment* containing:

- Δ.store : $x \to Type$ — store fields
- Δ.states : $S \to (initial?, terminal?)$ — state declarations
- Δ.triggers : $x \to (params, valid\_states)$ — trigger declarations

Transaction Well-Formedness

$$\frac{\forall D \in T.decls. \quad \Gamma \vdash D \text{ wf}}{\vdash T \text{ wf}}$$

[WF-TRANS]

A transaction is well-formed if all its declarations are well-formed in the transaction environment.

Actor Well-Formedness

$$\frac{\begin{array}{c}\exists! S \in \Delta.states.\ initial(S) \\ \exists S \in \Delta.states.\ terminal(S) \\ \forall S \in \Delta.states.\ reachable(S, S_0) \\ \forall t \in A.transitions.\ \Gamma, \Delta \vdash t \text{ wf}\end{array}}{\Gamma \vdash A \text{ wf}}$$

[WF-ACTOR]

An actor is well-formed if: 1. There is exactly one initial state 2. There is at least one terminal state 3. All states are reachable from the initial state 4. All transitions are well-formed

Transition Well-Formedness

$$\frac{\begin{array}{c} S, S' \in \Delta.\mathit{states} \\ \Gamma, \Delta \vdash \mathit{event} : \mathit{valid\_in}(S) \\ \Gamma, \Delta, \mathit{bindings}(\mathit{event}) \vdash \mathit{guard} : \texttt{bool} \\ \forall a \in \mathit{actions}.\ \Gamma, \Delta \vdash a \text{ wf} \end{array}}{\Gamma, \Delta \vdash S \to S' \text{ on } \mathit{event} \text{ when } \mathit{guard}\ (a^*) \text{ wf}}$$

[WF-TRANS]

Message Event Well-Formedness

$$\frac{M \in \Gamma.\mathit{messages} \quad A \in M.\mathit{to}}{\Gamma, \Delta \vdash \texttt{on}\ \ M : \mathit{valid\_in}(S) \text{ for all } S}$$

[WF-MSG-EVENT]

$$\frac{\mathit{trig} \in \Delta.\mathit{triggers} \quad S \in \mathit{trig.valid\_states}}{\Gamma, \Delta \vdash \texttt{on}\ \ \mathit{trig} : \mathit{valid\_in}(S)}$$

[WF-TRIG-EVENT]

State Reachability

The reachability predicate is defined as the reflexive-transitive closure of the transition relation:

$$\frac{}{\mathit{reachable}(S, S)}$$

[REACH-REFL]

$$\frac{\exists t \in A.\mathit{transitions}.\ t.\mathit{source} = S \land t.\mathit{target} = S'' \quad \mathit{reachable}(S'', S')}{\mathit{reachable}(S, S')}$$

[REACH-STEP]

B.3 Operational Semantics (Labeled Transition System)

We define the dynamic semantics of OMT actors as a labeled transition system (LTS). This captures how actors execute in response to events.

Actor Configurations

An actor configuration is a triple $\langle A, S, \sigma \rangle$ where: - A is the actor definition - S is the current state - $\sigma : x \to v$ is the store (mapping identifiers to values)

Labels

Transitions are labeled with events:

```
Label ::= tau                          (internal/auto)
        | ?M(v*)                        (receive message M with values)
        | !M(v*)@p                      (send message M to peer p)
        | !M(v*)@P                      (broadcast M to peer set P)
        | timeout(t)                    (timeout after t)
        | trigger(x, v*)                (external trigger with args)
```

Transition Rules

Automatic Transition:

$$\frac{(S \to S' \ \texttt{auto when}\ e\ (a^*)) \in A.trans \quad [\![e]\!]_\sigma = true \quad \langle \sigma, a^* \rangle \Downarrow \sigma'}{\langle A, S, \sigma \rangle \xrightarrow{\tau} \langle A, S', \sigma' \rangle}$$

[E-AUTO]

Message Reception:

$$\frac{(S \to S' \ \texttt{on}\ M\ \texttt{when}\ e\ (a^*)) \in A.trans \quad \sigma_1 = \sigma[\texttt{message} \mapsto v] \quad [\![e]\!]_{\sigma_1} = true \quad \langle \sigma_1, a^* \rangle \Downarrow \sigma'}{\langle A, S, \sigma \rangle \xrightarrow{?M(v)} \langle A, S', \sigma' \rangle}$$

[E-RECV]

Trigger Activation:

$$\frac{(S \to S' \ \texttt{on}\ x\ \texttt{when}\ e\ (a^*)) \in A.trans \quad x \in \Delta.triggers \quad S \in x.valid\_states \quad \sigma_1 = \sigma[params(x) \mapsto v^*] \quad [\![e]\!]_{\sigma_1} = t}{\langle A, S, \sigma \rangle \xrightarrow{trigger(x, v^*)} \langle A, S', \sigma' \rangle}$$

[E-TRIG]

Timeout:

$$\frac{(S \to S' \ \texttt{on timeout}(P)\ \texttt{when}\ e\ (a^*)) \in A.trans \quad [\![e]\!]_\sigma = true \quad \langle \sigma, a^* \rangle \Downarrow \sigma'}{\langle A, S, \sigma \rangle \xrightarrow{timeout(\Gamma.params(P))} \langle A, S', \sigma' \rangle}$$

[E-TIMEOUT]

Guarded Else Branch:

$$\frac{(S \to S_1 \ \texttt{on}\ \alpha\ \texttt{when}\ e_1\ (a_1^*)\ \texttt{else} \to S_2\ (a_2^*)) \in A.trans \quad [\![e_1]\!]_\sigma = false \quad \langle \sigma, a_2^* \rangle \Downarrow \sigma'}{\langle A, S, \sigma \rangle \xrightarrow{\alpha} \langle A, S_2, \sigma' \rangle}$$

[E-ELSE]

Action Evaluation

Actions are evaluated sequentially, threading the store:

$$\frac{}{\langle \sigma, \epsilon \rangle \Downarrow \sigma}$$

[A-EMPTY]

$$\frac{\langle \sigma, a \rangle \Downarrow \sigma_1 \quad \langle \sigma_1, a^* \rangle \Downarrow \sigma'}{\langle \sigma, a \cdot a^* \rangle \Downarrow \sigma'}$$

[A-SEQ]

$$\frac{v = [\![e]\!]_\sigma}{\langle \sigma, \texttt{STORE}(x, e) \rangle \Downarrow \sigma[x \mapsto v]}$$

[A-STORE]

$$\frac{v = [\![e]\!]_\sigma}{\langle \sigma, x = e \rangle \Downarrow \sigma[x \mapsto v]}$$

[A-ASSIGN]

Send and broadcast actions produce observable effects but do not modify the local store; we model these as side effects in a concurrent composition (Section B.4).

B.4 Concurrent Composition

A transaction execution consists of multiple actors running concurrently. We model this as a *parallel composition* of actor configurations with a shared message queue.

System Configuration

A system configuration is a tuple $\langle C_1, \ldots, C_n, Q \rangle$ where: - Each $C_i = \langle A_i, S_i, \sigma_i \rangle$ is an actor configuration - Q is a multiset of pending messages

Composition Rules

Internal Step:

$$\frac{\langle A_i, S_i, \sigma_i \rangle \xrightarrow{\tau} \langle A_i, S_i', \sigma_i' \rangle}{\langle C_1, \ldots, C_i, \ldots, C_n, Q \rangle \to \langle C_1, \ldots, C_i', \ldots, C_n, Q \rangle}$$

[SYS-TAU]

Message Send:

$$\frac{\langle A_i, S_i, \sigma_i \rangle \xrightarrow{!M(v)@p_j} \langle A_i, S_i', \sigma_i' \rangle}{\langle C_1, \ldots, C_n, Q \rangle \to \langle C_1, \ldots, C_i', \ldots, C_n, Q \uplus \{M(v)@p_j\} \rangle}$$

[SYS-SEND]

Message Receive:

$$\frac{M(v)@p_i \in Q \quad \langle A_i, S_i, \sigma_i \rangle \xrightarrow{?M(v)} \langle A_i, S_i', \sigma_i' \rangle}{\langle C_1, \ldots, C_n, Q \rangle \to \langle C_1, \ldots, C_i', \ldots, C_n, Q \setminus \{M(v)@p_i\} \rangle}$$

[SYS-RECV]

B.5 Relationship to Multiparty Session Types

OMT bears strong structural similarity to *multiparty session types* (MPST) [Honda et al. 2008, 2016]. We identify the correspondence and discuss how MPST theory could be applied.

Correspondence

MPST Concept	OMT Equivalent
Global type	Transaction definition
Local type	Actor definition
Role	Actor role (Consumer, Provider, Witness)
Message type	`message` declaration
Choice	Guarded transitions from same state
Recursion	Cycles in state machine
End	Terminal state

Global Type View

An OMT transaction can be viewed as a *global type* specifying the interaction pattern:

```
Escrow = Consumer -> Provider : LOCK_INTENT .
         Provider -> Consumer : WITNESS_SELECTION_COMMITMENT .
         Consumer -> Witness* : WITNESS_REQUEST .
         Witness <-> Witness : WITNESS_PRELIMINARY .
         Witness <-> Witness : WITNESS_FINAL_VOTE .
         Witness -> Consumer : LOCK_RESULT_FOR_SIGNATURE .
         Consumer -> Witness* : CONSUMER_SIGNED_LOCK .
         end
```

Local Type View

Each actor definition corresponds to a *local type*—the projection of the global protocol onto a single participant:

```
Consumer_Local = !LOCK_INTENT .
                 ?WITNESS_SELECTION_COMMITMENT .
                 !WITNESS_REQUEST* .
                 ?LOCK_RESULT_FOR_SIGNATURE .
                 !CONSUMER_SIGNED_LOCK* .
                 end
```

Where `!M` denotes sending message M and `?M` denotes receiving.

Potential for Session Type Verification

The MPST framework provides several verification guarantees that could be adapted to OMT:

1. **Communication Safety**: Well-typed sessions do not have message type mismatches. In OMT terms: messages are always received by actors that expect them.
2. **Protocol Conformance**: Each actor follows the prescribed interaction pattern. The state machine structure of OMT actors already enforces this locally.
3. **Deadlock Freedom**: Well-formed global types project to local types that cannot deadlock. This would require analyzing the composition of all actors.
4. **Progress**: If the system can make a step, it will. Combined with deadlock freedom, this ensures liveness.

Key Difference: MPST typically uses *projection* to derive local types from a global type. OMT instead specifies local types (actors) directly, with the global protocol emerging from their composition. This is closer to the *bottom-up* approach of choreographic programming [Montesi 2013] or the *communicating automata* framework [Brand & Zafiropulo 1983].

B.6 Related Formalisms

OMT's design draws on several formal traditions:

Process Calculi [Milner 1980, 1999]: The concurrent composition of actors follows CCS/pi-calculus style. Our LTS semantics uses standard structural operational semantics (SOS) [Plotkin 1981].

Communicating Finite State Machines [Brand & Zafiropulo 1983]: Each actor is essentially a CFSM. The extensive theory of CFSMs (decidability results, verification algorithms) may apply.

Scribble [Yoshida et al. 2014]: A protocol description language based on MPST. Scribble's syntax influenced OMT's message declarations. The Scribble toolchain generates endpoint APIs from global protocols—a similar approach could generate OMT actor skeletons.

Promela/SPIN [Holzmann 1997]: A verification-oriented protocol language. SPIN's model checking approach (state space exploration, LTL verification) could be adapted for OMT transaction verification.

TLA+ [Lamport 2002]: A specification language based on temporal logic. TLA+'s approach to specifying state machines with temporal properties provides a model for formal verification of OMT transactions.

B.7 Future Work

We identify several directions for formal development:

Type System Extensions

1. **Refinement Types**: Extend field types with predicates (e.g., `amount : uint{v > 0}`) to capture value constraints. This would enable static verification of guards.
2. **Linear Types**: Track message consumption linearly to ensure each message is processed exactly once. This connects to session type linearity.

3. **Dependent Types**: Allow types to depend on values (e.g., `list<peer_id>{length = WITNESS_COUNT}`). This would strengthen static guarantees.

Verification

1. **Model Checking**: Translate OMT transactions to Promela or TLA+ for automatic verification of safety and liveness properties.
2. **Session Type Checking**: Implement projection from a global protocol specification to verify that actor definitions conform.
3. **Deadlock Analysis**: Develop static analysis to detect potential deadlocks in actor compositions.

Tooling

1. **Formal Semantics in Coq/Agda**: Mechanize the semantics for machine-checked proofs of meta-theoretic properties (type safety, progress, preservation).
2. **Test Generation**: Use the formal semantics to generate test cases that cover all transition paths.
3. **Runtime Monitoring**: Generate runtime monitors from OMT specifications to detect protocol violations during execution.

B.8 References (Formal Methods)

[Brand & Zafiropulo 1983] D. Brand and P. Zafiropulo. "On Communicating Finite-State Machines." *Journal of the ACM*, 30(2):323-342, 1983. https://doi.org/10.1145/322374.322380

[Holzmann 1997] G. J. Holzmann. "The Model Checker SPIN." *IEEE Transactions on Software Engineering*, 23(5):279-295, 1997. https://spinroot.com/spin/Doc/ieee97.pdf

[Honda et al. 2008] K. Honda, N. Yoshida, and M. Carbone. "Multiparty Asynchronous Session Types." *POPL 2008*, pages 273-284. https://doi.org/10.1145/1328438.1328472

[Honda et al. 2016] K. Honda, N. Yoshida, and M. Carbone. "Multiparty Asynchronous Session Types." *Journal of the ACM*, 63(1):1-67, 2016. https://doi.org/10.1145/2827695

[Lamport 2002] L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002. https://lamport.azurewebsites.net/tla/book.html

[Milner 1980] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag, LNCS 92, 1980. https://doi.org/10.1007/3-540-10235-3

[Milner 1999] R. Milner. *Communicating and Mobile Systems: The Pi-Calculus*. Cambridge University Press, 1999. https://doi.org/10.1017/CBO9781139166874

[Montesi 2013] F. Montesi. "Choreographic Programming." Ph.D. thesis, IT University of Copenhagen, 2013. https://www.fabriziomontesi.com/files/choreographic-programming.pdf

[Plotkin 1981] G. D. Plotkin. "A Structural Approach to Operational Semantics." Technical Report DAIMI FN-19, Aarhus University, 1981. https://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf

[Yoshida et al. 2014] N. Yoshida, R. Hu, R. Neykova, and N. Ng. "The Scribble Protocol Language." *TGC 2013*, LNCS 8358, pages 22-41, 2014. https://doi.org/10.1007/978-3-642-54262-1_3