Agentic Trust: Succinctly Verifiable Automated Reasoning for the Principal-Agent Problem in Autonomous Commerce

Wyatt Benno
ICME Labs
wyatt@icme.io

February 18, 2026

Abstract

The principal-agent problem, where a principal delegates work to an agent with different incentives and superior information, has constrained economic systems for centuries. We demonstrate that autonomous AI agents transacting at machine speed amplify this problem beyond the reach of traditional solutions.

We prove an impossibility theorem: when the ratio κ of agent execution speed to principal monitoring speed is sufficiently large, no mechanism with human-in-the-loop reactive properties can achieve efficient outcomes. The speed mismatch creates irreversible moral hazard where damage occurs faster than principals can detect, investigate, or respond.

We then introduce a framework addressing both components of the problem. For execution verification, Zero-Knowledge Machine Learning (zkML) combined with Automated Reasoning (AR) provides mathematical certainty of policy compliance: agents generate cryptographic proofs, principals verify in constant time, and only valid actions execute. We prove this cryptographic approach succeeds beyond the threshold κ^* where reactive mechanisms fail. For policy specification, we introduce Argus Codex, an adversarial system that uses competing LLMs to iteratively refine policies by searching for gaps between stated intent and formal specifications.

Together, these contributions transform the agentic principal-agent problem from an economic impossibility under traditional tools into a tractable engineering problem using modern cryptography, enabling autonomous economies where both policy specification and execution are cryptographically verifiable.

Keywords: agentic trust, principal-agent problem, zero-knowledge machine learning, zkML, automated reasoning, autonomous agents, cryptographic verification, adversarial policy generation, agent-to-agent commerce, verifiable AI, trustless systems

Contents

1 Introduction **4**
- 1.1 The Classical Principal-Agent Problem . . . 4
- 1.2 Autonomous Agent Amplification . . . 4
- 1.3 The Fundamental Question . . . 6
- 1.4 Our Contribution: A Complete Cryptographic Solution . . . 7
- 1.5 Agentic Trust . . . 10
- 1.6 Formal Model and Main Results . . . 11
- 1.7 Paper Organization . . . 11

2 Background and Related Work **12**
- 2.1 Principal-Agent Theory . . . 12
- 2.2 Automated Reasoning for Policy Verification . . . 12
- 2.3 Zero-Knowledge Proofs and zkML . . . 13

3 Why Traditional Verification Approaches Fail **14**
- 3.1 The Succinct Verification Gap . . . 14

4 Framework: Complete Cryptographic Solution for Autonomous Commerce **15**
- 4.1 System Architecture . . . 15
- 4.2 The Verification Pipeline . . . 15
- 4.3 Addressing Policy Specification with Argus Codex . . . 16
- 4.4 Integration: End-to-End Cryptographic Verification . . . 19
- 4.5 Succinct Verification Properties . . . 20
- 4.6 Privacy-Preserving Verification . . . 21

5 Autonomous Monitoring and Enforcement **22**
- 5.1 The Monitoring Agent Architecture . . . 22
- 5.2 From Observation to Mathematical Certainty . . . 22
- 5.3 Machine-Speed Enforcement . . . 23

6 Discussion: Implications for Autonomous Economies **24**
- 6.1 Market Efficiency . . . 24
- 6.2 Mechanism Design Implications . . . 24
- 6.3 Practical Implications . . . 25

7 Conclusion and Future Work **25**
- 7.1 Summary of Contributions . . . 25
- 7.2 The Policy Specification Problem: Engineering Solution . . . 27
- 7.3 Open Challenges . . . 27
- 7.4 Future Directions . . . 28
- 7.5 Conclusion . . . 30

A Formal Proofs **30**
A.1 Notation and Definitions 30
A.2 Impossibility Theorem 33
A.3 Verification Theorem 43

B Primers on Key Technologies **47**
B.1 Automated Reasoning 48
B.2 Zero-Knowledge Proofs 48

1 Introduction

1.1 The Classical Principal-Agent Problem

The principal-agent problem is a fundamental challenge in economics and organizational theory [20, 24]. When one party (the principal) delegates work to another party (the agent), three critical issues emerge. First, the agent's objectives often differ from the principal's goals, creating misaligned incentives. Second, the agent possesses superior information about their actions and environment, leading to information asymmetry that the principal cannot easily overcome. Third, observing the agent's behavior directly is either prohibitively expensive or outright infeasible, making monitoring costly and incomplete.

Traditional solutions to these challenges include performance-based compensation to align incentives, reputation systems that make future interactions contingent on good behavior, direct monitoring and auditing where feasible, and contractual penalties enforced through legal systems [29]. All of these mechanisms share a common assumption: human-speed interactions where monitoring is feasible and contracts are enforceable through established legal and social institutions.

1.2 Autonomous Agent Amplification

Autonomous AI agents conducting commerce at machine speed fundamentally break these traditional mechanisms. Consider a concrete example: You delegate to an AI agent the authority to make purchases with the policy "no single purchase over $100 *and* no more than $1,000 in total purchases per day." An attacker uses indirect prompt injection to instruct your agent to disable the daily limit check. The agent continues checking the per-transaction limit ($100) but internally ignores the daily total. It then executes 100 transactions of $100 each, totaling $10,000.

Why traditional monitoring fails. Observing 100 transactions of $100 each tells you nothing about whether the agent *actually enforced* the daily limit policy. The agent can claim "I checked the daily limit and authorized these transactions" - but you cannot verify this claim without re-executing the agent's internal reasoning process, which contains billions of opaque neural network operations. By the time your monitoring system detects the pattern and investigates, the money is gone.

Why Traditional Economics Fails Traditional principal-agent theory developed observing human actors with two implicit assumptions: principals and agents operate at comparable speeds ($\kappa \approx 1$), and agent reasoning is observable through interrogation or behavioral inference. AI agents break both assumptions. They execute thousands of actions per second while human oversight operates at best once per minute ($\kappa > 60{,}000$). More fundamentally, AI reasoning occurs through billions of opaque neural operations that cannot be observed or meaningfully interrogated. Traditional mechanisms assume the principal can intervene during the agent's action sequence and can verify compliance through observation. When $\kappa \gg 1$, intervention becomes temporally impossible. When reasoning is opaque, verification through observation becomes fundamentally impossible. The mechanisms fail

not because of poor implementation but because they are solving a different problem than the one we face.

This is not a temporary gap that better implementation could fix. These are fundamental mismatches between the assumptions embedded in traditional economic theory and the reality of autonomous agent commerce. The study of Agentic Trust addresses this mismatch by introducing mechanisms designed for the actual properties of AI agents, rather than attempting to adapt mechanisms designed for human constraints that no longer apply.

How our framework addresses this. An agent must provide a cryptographic proof that the output of inference followed the correct policy-checking code (not a compromised version) and evaluated both the per-transaction limit and the daily limit using the correct running total. An attacker who disabled the daily limit check cannot generate a valid proof, because during the proof generation all formal rules would not have been followed. Additionally, specialized agents (Argus Codex) could have identified this policy gap during adversarial refinement: an adversarial LLM searching for exploits could discover that "100 transactions of $100" satisfies the literal policy while violating intent, prompting refinement to explicitly constrain daily totals.

This scenario illustrates four ways AI agents amplify the principal-agent problem:

Extreme Information Asymmetry Unlike human agents whose reasoning process can be interrogated, AI agents' internal decision-making is opaque. While you may observe which tools or APIs an agent calls, you cannot directly observe what reasoning led to those decisions, whether the model actually enforced stated policies internally, or what data within the context influenced the outcome. The neural network's billions of parameters produce behavior through mechanisms that cannot be decomposed easily into interpretable logic.

Speed Mismatch Creates Irreversible Moral Hazard Classical principal-agent theory assumes principals can monitor and intervene [21]. But agents executing at machine speed (1,000+ transactions per second) create an irreversible gap. If your incident response team needs 60 seconds to investigate an alert, that's 60,000 potentially fraudulent transactions executed before human understanding is even possible. The fundamental assumption of "observable action" underlying most principal-agent solutions becomes false.

Novel Attack Vectors AI agents face threats with no human analogue: prompt injection attacks where malicious instructions are hidden in data the agent processes [35], memory poisoning that corrupts the agent's context, model drift where the agent's behavior changes unpredictably, and adversarial inputs designed to manipulate decision-making [10]. A compromised agent doesn't just shirk or steal, it can systematically bypass all stated policies while appearing to function correctly.

Unenforceable Traditional Contracts How do you punish an AI agent? Reputation systems assume future interactions create incentive alignment [13], but a compromised agent can drain accounts in milliseconds before reputation damage accrues. Economic penalties require assets to seize. Legal enforcement assumes parties subject to jurisdiction. None of

these mechanisms apply to autonomous agents operating across jurisdictional boundaries at machine speed.

1.3 The Fundamental Question

This leads to the central question of Agentic Trust:

> *How can principals verify that autonomous agents executed correctly when verification must happen at machine speed, without human oversight, and across trust boundaries?*

Traditional principal-agent theory analyzes the trade-off between monitoring costs and agency costs. Jensen and Meckling demonstrated that monitoring expenditures increase with monitoring frequency, leading principals to balance the marginal cost of monitoring against the marginal benefit of reduced agency losses. This creates residual agency costs because principals stop monitoring when further observation becomes too expensive relative to the benefits [24].

At machine speed, where agents execute thousands of actions per second, this trade-off becomes catastrophic. The monitoring frequency required for effective oversight is economically prohibitive. Even automated monitoring systems face fundamental bottlenecks that create super-linear costs. The distinction is not between automated and manual systems, but between *detection* and *verification*.

Monitoring requires interpretation. Traditional monitoring observes behavioral patterns (e.g. 100 rapid transactions) and must determine: Is this legitimate high-frequency trading or an attack? Automated anomaly detection can flag suspicious patterns, but these flags are inherently probabilistic and ambiguous. Human investigators must review alerts to distinguish true violations from false positives, loading context about the agent's task, historical behavior, and business logic. This interpretation process has three sources of super-linear cost:

1. **False positive handling**: To catch rare violations, detectors must be sensitive, generating false positive rates of 1-10% [11]. At 1,000 actions/second with 5% false positive rate, this produces 50 alerts/second = 4.3 million alerts/day. Human review capacity saturates at 100-1,000 alerts/day per analyst [38], creating a queue. Queue theory shows investigation time grows as $1/(1-\rho)$ where ρ is utilization, exploding as alert rate approaches capacity [27].

2. **Context switching overhead**: Each alert requires investigators to load mental context about the specific agent, task, and business rules. Cognitive psychology shows context switching imposes fixed overhead that doesn't parallelize [36]. Reviewing n alerts requires $O(n \log n)$ time due to context reload costs.

3. **Coordination overhead**: Parallelizing investigation across multiple analysts requires coordination (case assignment, knowledge sharing, consistency) that grows super-linearly with team size [7].

Verification requires no interpretation. Cryptographic proofs are mathematical objects with binary validity: they verify correctly or fail. No context is needed, no judgment required, no false positives exist. A proof that fails verification is definitive evidence of violation; a proof that succeeds is mathematical certainty of compliance. This eliminates all three sources of super-linear cost: no false positives to investigate, no context to load, no interpretation to coordinate. Verification cost is purely computational, scaling linearly: $C_v \cdot n$ with no $H(n)$ term.

This fundamental distinction - interpretation versus verification - explains why monitoring cannot simply be automated to match verification efficiency. Automated monitoring systems (ML-based anomaly detectors, rule engines, pattern matchers) can reduce the constant factor in linear costs but cannot eliminate the super-linear human bottleneck because they produce ambiguous signals requiring interpretation, not definitive mathematical proofs.

Our contribution demonstrates that cryptographic verification fundamentally changes this economic calculus. While verification costs still scale linearly with transaction volume, three properties make complete verification economically optimal where partial monitoring was the best traditional approaches could achieve. First, the cost per verification is substantially lower than traditional monitoring. Second, verification is fully automated with no human bottleneck. Third, verification parallelizes perfectly across machines. These properties transform the cost-benefit analysis, making it optimal to verify every transaction rather than accepting residual agency costs.

1.4 Our Contribution: A Complete Cryptographic Solution

We demonstrate that combining Zero-Knowledge Machine Learning (zkML), Automated Reasoning (AR), and adversarial policy generation provides a comprehensive solution to the principal-agent problem in autonomous commerce. (For readers unfamiliar with these technologies, Appendix B provides detailed primers on Automated Reasoning and Zero-Knowledge Proofs.)

Brief Primer on Key Technologies Before detailing our contributions, we briefly introduce the three technologies we synthesize:

- **Automated Reasoning**: A formal methods approach that translates natural language policies into mathematical logic, then verifies AI outputs against these logical constraints. Unlike probabilistic checks, automated reasoning provides deterministic verification: if a policy is correctly translated to formal logic and the execution environment is honest, the system will detect violations with certainty. Industry implementations report high accuracy in production use [2], though peer-reviewed empirical validation at scale remains limited.

- **Zero-Knowledge Machine Learning (zkML)**: A cryptographic technique that generates mathematical proofs of computation correctness. These proofs are *succinct*: proof size is constant (typically under 1KB) and verification time is constant (under 300ms), both independent of the underlying computation's complexity. Crucially, zkML proofs require minimal trust assumptions (only standard cryptographic hardness) and can be verified by anyone.

- **Adversarial Policy Generation (Argus Codex)**: A system that uses competing large language models to iteratively refine policies. One LLM generates formal policies from natural language intent, while an adversarial LLM searches for actions that satisfy the policy but violate stated intent. Each discovered exploit triggers policy refinement, creating a systematic approach to policy completeness.

Our key insight: These three technologies address complementary aspects of the principal-agent problem. Automated reasoning tells us *what* to verify (policy compliance). zkML tells us *how* to verify it trustlessly (cryptographic proofs). Argus Codex tells us *which policies* to verify against (adversarially refined specifications). By combining them, we achieve deterministic policy enforcement, succinct verification, and systematic policy completeness.

Our Contributions Our framework introduces mechanisms categorically different from traditional principal-agent tools. Traditional mechanisms (monitoring, incentives, reputation, legal penalties) are inherently reactive: they observe agent behavior after execution, detect anomalies probabilistically, and respond through investigation. Our cryptographic approach operates through a different temporal sequence:

1. **Policy Generation**: Humans produce natural language policies, Argus Codex does adversarial refinement
2. **Computation** (t_c): Agent computes proposed action and determines it satisfies policies
3. **Proof generation** (t_p): Agent generates cryptographic proof π of policy compliance
4. **Verification** (t_v): Principal verifies proof π before authorizing action
5. **Execution** (t_e): Action takes effect only if proof is valid

The critical distinction from reactive monitoring is that actions' effects are *delayed* until verification completes, and policies themselves are generated through verifiable adversarial refinement. Traditional monitoring detects violations after irreversible damage occurs; our framework prevents invalid actions before effects materialize and systematically reduces policy gaps before deployment.

Feasibility and deployment timeline. Current zkML systems face a practical challenge: proof generation overhead ranging from near 10,000× (recent advances with JOLT Atlas [23]) to 1,000,000×-100,000× (earlier systems [25, 30]) compared to native execution. For real-time operation at rate r_a, we require $t_p + t_v < 1/r_a$. At 1,000 actions/second, this demands sub-millisecond total latency.

With JOLT Atlas achieving minimal overhead and GPU and other hardware acceleration providing additional 10-100× speedup, proof generation for small language models could reach 100-1,000× overhead within 2026. For a 10ms inference, this yields 1-10 second proving time. While still too slow for real-time verification at 1,000 tx/sec, this makes several deployment scenarios practical:

- **High-value, lower-frequency transactions** where latency is acceptable (large financial settlements, supply chain verification)
- **Batch verification** where proofs are generated continuously but verified in aggregate with acceptable delay (removing the human-in-the loop factor for agentic commerce)
- **Asynchronous workflows** where actions can be queued pending verification
- **Moderate-speed commerce** (10-100 tx/sec) with parallel proof generation

Our theoretical contribution establishes that *when proof generation becomes efficient enough*, cryptographic verification achieves first-best welfare where traditional mechanisms fail. We view this as providing strong incentives for the cryptography community to achieve necessary efficiency thresholds. Recent progress suggests real-time machine-speed verification may become practical within years, not decades.

Specifically, our framework provides six key capabilities:

1. **Adversarial policy refinement**: systematic search for policy gaps before deployment, with optional cryptographic audit trails of the refinement process.
2. **Provable execution**: every agent action generates a cryptographic proof showing which model executed, which policies were checked, and on what inputs.
3. **Succinct verification**: these proofs verify in sub-second time, enabling machine-speed policy enforcement without re-execution overhead.
4. **Minimal trust assumptions**: mathematical proofs eliminate trust in the executing agent and its environment, replacing traditional monitoring, reputation systems, and incentive alignment with cryptographic verification. However, trust assumptions are relocated rather than eliminated—the system requires trust in:
   (a) cryptographic hardness assumptions (discrete logarithm, lattice hardness problems),
   (b) adequate adversarial search budget for policy refinement,
   (c) zkML implementation correctness, and
   (d) trusted setup ceremonies for certain proof systems (though many transparent constructions eliminate this requirement).
5. **Autonomous enforcement**: monitoring agents can verify proofs and trigger responses without any human intervention.
6. **Verifiable policy generation**: the natural language to formal policy creation process itself can be wrapped in zkML proofs, as can the formal verification logic, providing end-to-end verification from intent to execution.

This trust relocation represents a significant security improvement: operational trust (in agents, execution environments, human investigators) is replaced by mathematical trust (in cryptographic assumptions, verified implementations) and engineering confidence (in adversarial search coverage). The former is vulnerable to compromise, corruption, and human error at scale; the latter rests on decades of cryptographic research and systematic testing methodologies.

This transforms the principal-agent problem from asking "how do we detect bad behavior after it happens?" (the traditional reactive approach) to asking "how do we systematically prevent bad behavior from being possible?" (preventative verification with adversarial policy refinement). The temporal inversion is critical: traditional monitoring observes outcomes and infers compliance, while cryptographic verification proves compliance before actions execute, using policies refined through systematic adversarial testing.

1.5 Agentic Trust

We study **Agentic Trust** as a field distinct from both traditional distributed systems and AI safety, addressing unique challenges that arise when both principals and agents may be AI systems (systems based on math), when transactions execute at machine speed without human oversight, when information asymmetry is extreme by design (to protect model privacy and proprietary algorithms), when traditional economic mechanisms become computationally or economically infeasible, and when verification must be trustless across organizational and jurisdictional boundaries.

Unlike distributed systems research (which addresses Byzantine faults and consensus among peers with no delegation relationship), Agentic Trust focuses squarely on the principal-agent problem in autonomous delegation with extreme information asymmetry. Unlike AI safety research (which addresses alignment and capability control to prevent harmful AI behavior), Agentic Trust provides cryptographic infrastructure for trustless verification in adversarial environments where we cannot simply assume alignment.

Relation to adjacent fields. Agentic Trust synthesizes insights from several established areas while addressing distinct challenges. *Smart contract research* [8] provides deterministic on-chain execution with cryptographic verification, but faces computational constraints (Ethereum's gas limits preclude complex AI inference), lacks privacy (all state is public), and requires deterministic execution incompatible with many AI systems. Our zkML framework extends smart contract guarantees to off-chain AI execution, analogous to how zkRollups extend Ethereum's security to off-chain transactions [9]. *Secure multi-party computation* (MPC) [15, 43] offers an alternative approach where principal and agent jointly compute policy verification while keeping inputs private. However, MPC requires interactive protocols (multiple rounds of communication introducing latency), produces party-specific verifications (not publicly reusable), and has verification cost scaling with policy complexity rather than constant time. For one-to-many delegation or agent-to-agent commerce, zkML's non-interactive, publicly verifiable proofs provide better scalability; MPC and zkML are complementary, with MPC potentially useful for private policy specification. *Computational mechanism design* [33] studies mechanisms under computational constraints, establishing impossibility results when verification is expensive. Our contribution introduces

succinct cryptographic verification as a new computational tool: rather than verifying by re-computation (expensive) or creating incentive compatibility through payments (requires enforceability), we achieve strategy proof compliance through cryptography. Characterizing the full space of cryptographically verifiable mechanisms beyond our specific construction is important future work.

1.6 Formal Model and Main Results

We now formalize the intuitions developed above. Let r_a denote the agent's execution rate (actions per unit time), r_p the principal's monitoring rate, and $\kappa = r_a/r_p$ the speed asymmetry ratio. Let C_m represent the cost per monitoring action, C_v the cost per verification action, C_p the cost per proof generation, and $H(n)$ the super-linear human bottleneck costs (alert review, investigation, context switching). Traditional monitoring has cost $C_m \cdot n + H(n)$; cryptographic verification has cost $(C_p + C_v) \cdot n$ with no human bottleneck.

Impossibility Theorem (Theorem A.13, Appendix A). For any $\epsilon > 0$, there exists a threshold κ^* such that when $\kappa > \kappa^*$, no mechanism exhibiting *reactive properties* can achieve expected social welfare within ϵ of the first-best outcome.

By "reactive properties," we mean mechanisms that: (1) observe agent behavior after execution rather than before, (2) infer compliance probabilistically rather than proving it deterministically, or (3) require human judgment to interpret observations and trigger responses. All traditional principal-agent mechanisms—ex-post monitoring, performance-based contracts with dispute resolution, reputation systems with human moderation, and contractual penalties enforced through courts—exhibit these properties. The impossibility does not preclude all conceivable non-cryptographic approaches; a hypothetical fully automated, pre-execution, deterministic system without human judgment would fall outside its scope. But the standard economic toolkit falls squarely within it. Formal proofs appear in Appendix A.

Verification Theorem (Theorem A.21, Appendix A). Cryptographic verification achieves first-best welfare for the execution verification problem beyond κ^*, provided: (1) policies correctly capture principal intent, (2) $(C_p + C_v) < \delta v$ (combined proof and verification cost is less than per-action welfare gain), and (3) $t_p + t_v < 1/r_a$ (proof generation and verification complete within the action interval).

zkML succeeds where reactive mechanisms fail because it is preventative (blocks actions before execution), deterministic (mathematical proof verification), and fully automated (no human interpretation). It does not violate the impossibility theorem because it lacks the reactive properties that cause failure. Together, the two results establish that the principal-agent problem under speed asymmetry is an economic impossibility with traditional tools but a tractable engineering problem with cryptographic verification.

1.7 Paper Organization

The remainder of this paper proceeds as follows. Section 2 reviews related work in principal-agent theory, distributed systems, AI security, and cryptographic verification. Section 3

presents the threat model for autonomous agent commerce and demonstrates why traditional guardrails fail. Section 4 introduces our complete framework: automated reasoning with zkML for succinct verifiable guardrails (Sections 4.1-4.2), Argus Codex for adversarial policy generation (Section 4.3), and integration showing end-to-end verification (Section 4.4). Section 5 presents the architecture for autonomous monitoring and enforcement. Section 6 discusses implications for autonomous economies, including welfare analysis, mechanism design, and organizational economics. Section 7 concludes with future research directions. Appendix A provides formal proofs of our main theorems. Appendix B provides detailed primers on Automated Reasoning and Zero-Knowledge Proofs for readers less familiar with these technologies.

2 Background and Related Work

2.1 Principal-Agent Theory

The principal-agent problem was formalized by Jensen and Meckling [24] and Holmström [20], who demonstrated that when agents possess superior information about their actions and the environment, optimal contracts must balance incentive provision with risk sharing. The fundamental challenge is that monitoring is costly and often infeasible, leading to moral hazard where agents shirk or pursue their own interests [29].

Traditional solutions to the principal-agent problem rely on four mechanisms. Direct monitoring and auditing allows principals to observe agent behavior, though this is often costly and incomplete. Performance-based compensation attempts to align incentives through outcome-dependent rewards. Reputation systems leverage future interaction opportunities to create incentives for good behavior [13]. And contractual penalties enable legal enforcement of obligations through courts and regulatory systems.

All of these mechanisms assume human-speed interactions and enforceable contracts through established institutions. Akerlof's work on information asymmetry [1] shows how markets can fail when quality is unobservable, while Hart and Holmström's contract theory [19, 22] addresses incomplete contracts when not all contingencies can be specified ex ante. Our contribution shows these traditional mechanisms fail catastrophically when applied to autonomous agents operating at machine speed, necessitating succinct cryptographic verification as a fundamentally new approach.

2.2 Automated Reasoning for Policy Verification

Automated reasoning descends from classical symbolic logic and formal methods [32, 37], using techniques from theorem proving, satisfiability solving (SAT/SMT), and formal verification [6, 28] to provide deterministic guarantees rather than statistical confidence. Natural language policies are translated into formal logic, and constraint solvers verify whether agent outputs satisfy those constraints. (See Appendix B for a detailed primer.)

Industry implementations of automated reasoning for AI guardrails report high accuracy in production use [2], though peer-reviewed empirical validation at scale remains limited. The formal logic ensures that *if* the stated policy was followed and correctly translated, verification will confirm it with mathematical certainty, not statistical confidence.

However, this deterministic power comes with critical limitations for autonomous commerce. Automated reasoning systems assume the execution environment is honest; they cannot verify which model actually ran or whether the execution environment was compromised. The proofs themselves are not succinctly verifiable; they can be large and require significant recomputation to check, sometimes taking as long to verify as to generate. Policies must be expressed in formal logic visible to verifiers, preventing privacy-preserving verification. And sensitive policy rules or input data cannot be hidden while still enabling verification.

This is where the synthesis with zkML becomes essential: automated reasoning provides the deterministic policy checking we need, but zkML wraps it in a cryptographic envelope that enables trustless, succinct verification.

2.3 Zero-Knowledge Proofs and zkML

Zero-knowledge proofs (ZKPs), introduced by Goldwasser, Micali, and Rackoff [16], allow a prover to convince a verifier that a statement is true without revealing any information beyond the statement's validity. The breakthrough of zkSNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) [5,17] added *succinctness*—proofs are small (under 1KB) and verify in constant time regardless of computation complexity—and *non-interactivity*, enabling proofs to be generated once and verified by anyone. (See Appendix B for a detailed introduction to ZKPs and their development history.)

The Blockchain Scaling Connection The practical maturation of zkSNARKs was driven largely by blockchain scaling. Public blockchains require every node to re-execute every transaction, limiting throughput. Zero-knowledge proofs solve this by moving expensive computation off-chain and providing succinct proofs verifiable on-chain [4, 9]. The economic insight—transforming verification from $O(n \times c)$ to $O(n \times v)$ where $v \ll c$—is precisely what makes trustless agent commerce viable.

Zero-Knowledge Machine Learning (zkML) Applying zero-knowledge proofs to machine learning extends this paradigm to autonomous agents. Early systems like zkCNN [30] demonstrated feasibility with 100,000×–1,000,000× proving overhead. Modulus Labs [25] showed GPT-2 verification was possible but impractical. EZKL emerged as a general-purpose framework [14]. Most recently, JOLT Atlas [23], building on JOLT [3], achieves approximately 10,000× proving overhead through lookup-based architectures, with parallelization and hardware acceleration bringing practical overhead to 100–1,000× for certain model classes.

However, existing zkML research has focused narrowly on proving model execution correctness in isolation. Our contribution synthesizes zkML with automated reasoning and adversarial policy generation to create the first complete framework for autonomous commerce. We prove not just that a model executed, but that it executed with adversarially refined policies, checked those policies correctly, and did so in a way that can be verified trustlessly across organizational boundaries.

3 Why Traditional Verification Approaches Fail

3.1 The Succinct Verification Gap

Guardrail systems using automated reasoning represent significant progress, translating natural language policies into formal logic and providing deterministic verification for policy compliance [2]. However, they have critical limitations for autonomous commerce:

- Cannot prove that the AI agent actually used the policy you think it used
- Cannot verify which model version executed the decision
- Proofs are not succinctly verifiable; verification can be as expensive as generation
- Cannot protect policy rules when they must remain private
- No systematic approach to policy completeness - manual policy design prone to gaps

Automated reasoning assumes policies are public, correctly specified, and trusts the execution environment. Critically, **the proofs are not succinctly verifiable**: they can be large and require significant recomputation to check. A proof that took hours to generate might take hours to verify. While excellent for human-supervised systems, this verification overhead makes automated reasoning impractical for trustless agent-to-agent commerce at scale.

Consider the requirements for machine-speed commerce:

- **Verification Speed**: Must verify 1,000+ proofs per second to match transaction throughput
- **Resource Constraints**: Verification often happens on mobile devices or edge nodes with limited compute
- **Economic Viability**: Total verification costs must remain below the value of fraud prevented
- **Trustless Environment**: Verifier cannot trust the prover's execution environment
- **Policy Completeness**: Policies must capture all relevant edge cases and attack vectors

Traditional automated reasoning fails these requirements. Verifying a complex policy check might require running the entire formal proof system again, essentially re-executing the logic that took seconds or minutes originally. The cost function is $C_m \cdot n + H(n)$ where C_m is the per-action monitoring cost and $H(n)$ represents super-linear human bottleneck costs from alert review, context switching, and fatigue.

In contrast, our complete framework provides *succinct verification* with cost structure $(C_p + C_v) \cdot n$ where C_p is proof generation cost and C_v is verification cost per action, with $C_v \ll C_m$. A proof that took hours to generate can be verified in milliseconds, regardless of the underlying computation's complexity [17]. Critically, zkML verification is fully automated, eliminating the human bottleneck term $H(n)$ entirely. Additionally, Argus Codex systematically searches for policy gaps before deployment, reducing the probability of exploitable specifications.

This economic transformation makes complete verification optimal. At high transaction rates, traditional monitoring costs $C_m \cdot n + H(n)$ per unit time, where $H(n)$ grows super-linearly and quickly dominates total cost. Our framework costs $(C_p + C_v) \cdot n$ with no human bottleneck plus one-time adversarial search cost C_{Argus}, remaining economically feasible provided $(C_p + C_v) < \delta v$ where δv is the per-action welfare gain from preventing violations. The linear scaling, low constant factor, and elimination of human bottleneck transform the cost-benefit analysis from "partial monitoring optimal" to "complete verification optimal."

4 Framework: Complete Cryptographic Solution for Autonomous Commerce

4.1 System Architecture

Our framework combines three complementary technologies. Argus Codex generates policies through adversarial refinement, ensuring systematic coverage of edge cases. Automated reasoning translates these policies into formal logic and provides deterministic rule checking. Zero-Knowledge Machine Learning (zkML) wraps both policy generation and verification execution in cryptographic proofs that are succinctly verifiable.

Division of Labor The key insight is that these three technologies address complementary aspects of the principal-agent problem:

- **Argus Codex** addresses policy specification: Which policies should we enforce? (Adversarial search for completeness)
- **AR** addresses policy semantics: What does each policy mean formally? (Translation to mathematical logic)
- **zkML** addresses execution verification: Did the agent actually follow the policies? (Cryptographic proofs)

By combining them, we achieve systematic policy completeness, deterministic policy enforcement, and trustless verification at machine speed.

4.2 The Verification Pipeline

When an autonomous agent executes a transaction, our system produces a complete cryptographic proof of policy compliance through a streamlined process:

1. **Policy Translation** (one-time, $t_{translate}$): Natural language policies are converted to formal logic via automated reasoning, transforming human-readable rules into mathematical constraints. This occurs once during policy deployment.
2. **Policy Validation** ($t_{validate} \sim$ 1-10ms): For each proposed action, automated reasoning validates the transaction against formal policy rules, providing deterministic verification.

3. **Proof Generation** ($t_p \sim$ 1-10s with current technology): The entire validation pipeline is wrapped in a zero-knowledge proof through zkML. With JOLT Atlas and hardware acceleration, proving overhead of 100-1,000× makes this practical for batch verification or asynchronous workflows.

4. **Proof Verification** ($t_v \sim$ 300ms): Any party can verify this proof in constant time without re-executing any of the underlying computation, enabling trustless verification.

Total per-action latency is $t_{total} = t_{validate} + t_p + t_v$. For current technology: approximately 1-10 seconds, suitable for high-value transactions, batch processing, or moderate-speed commerce with parallelization. As zkML proving efficiency improves, real-time verification at machine speed becomes feasible.

4.3 Addressing Policy Specification with Argus Codex

While zkML and automated reasoning address execution verification, they assume policies correctly capture principal intent. We address this policy specification challenge through **Argus Codex**, an adversarial policy generation system.

The Argus Codex Process Argus Codex uses adversarial LLMs to iteratively refine policies:

1. **Intent capture**: Principal expresses desired behavior in natural language ("No single purchase over $100 and no more than $1,000 in total purchases per day")

2. **Adversarial search**: Adversarial LLM searches for actions that satisfy policies but violate stated intent. For example, it discovers: "Submit 100 transactions of $100 each within 1 second - each individual transaction satisfies the policy"

3. **Gap identification**: Each discovered exploit reveals a policy gap (here: the policy doesn't constrain transaction velocity or total count)

4. **Policy refinement**: Policies updated to close identified gaps:

```
forall transaction t:
  t.amount <= 100 AND
  sum(transactions_today) + t.amount <= 1000 AND
  count(transactions_last_minute) <= 5
```

5. **Iteration**: Repeat steps 3-5 until adversarial search finds no exploits within computational budget (e.g., 10,000 attempts)

6. **Policy generation**: LLM translates to formal policies using automated reasoning syntax:

```
forall transaction t:
  t.amount <= 100 AND
  sum(transactions_today) + t.amount <= 1000
```

7. **Human signoff**: a human reviewer can sign off on final policies. Argux Codex can be enabled to continuously search and update policies to prevent any new issues (steps 2 - 7).

8. **zkML wrapping:** The entire policy enforcement process (which LLM generated policies into formal logic, did the checks pass for a specific inference output) is wrapped in a zkML proof, creating a cryptographic audit trail

Theoretical Foundation Argus Codex transforms policy specification from a principal-agent problem into an adversarial search problem. The key insight is that finding policy gaps is easier than proving policy completeness. While we cannot mathematically prove a policy is complete (this would require enumerating all possible actions and showing none violate intent - often an infinite or computationally intractable set), we can gain high engineering confidence by:

- Running adversarial search with large computational budgets (e.g., 10,000+ attempts to find exploits)
- Using diverse adversarial strategies (semantic exploits, edge cases, composition attacks, temporal patterns)
- Measuring search coverage through state space exploration metrics
- Comparing discovered vulnerabilities to known exploit classes from security research
- Iterating until no exploits found within budget

Probabilistic guarantee. Let B be the adversarial search budget (number of exploit attempts), and C be the complexity of the action space (number of semantically distinct action classes). If adversarial search with budget B finds no exploits, the probability of an exploitable gap existing is bounded by:

$$P_{exploit} \leq \frac{C - \text{coverage}(B)}{C} \tag{1}$$

where coverage(B) is the number of action classes explored. For intelligent adversarial search (not random), coverage grows sub-linearly but substantially: coverage(B) $\approx B^{\alpha}$ for $\alpha \in (0.5, 0.8)$ depending on search strategy effectiveness.

As $B \to \infty$, $P_{exploit} \to 0$. In practice, budgets of $B = 10{,}000$ with intelligent search strategies provide high confidence, similar to how fuzzing provides confidence in software security.

This doesn't provide mathematical certainty like zkML does for execution verification, but provides engineering confidence comparable to:

- Security auditing in software systems (extensive testing without formal proof)
- Smart contract auditing (adversarial analysis to find exploits)
- Penetration testing (systematic search for vulnerabilities)

- Fuzzing in compiler and OS development (random/guided testing to find edge cases)

All of these approaches are standard engineering practice precisely because they work: systematic adversarial testing finds the vast majority of exploitable bugs, even without mathematical completeness proofs.

zkML Verification of Policy Generation Critically, Argus Codex itself can be wrapped in zkML proofs, providing end-to-end verification:

- **Translation proof**: Prove that a specific LLM generated the formal policies from the stated natural language intent, using a specific prompt and temperature
- **Adversarial search proof**: Prove that adversarial search actually executed with the claimed budget B, using specified search strategies
- **Refinement audit trail**: Create a record of all discovered gaps and how policies were refined to address them
- **Search coverage proof**: Demonstrate the adversarial search achieved claimed coverage over the action space
- **Human review attestation**: Cryptographically sign that a human expert reviewed and approved the final policies used in production

This provides complete transparency: anyone can verify that policies were generated through proper adversarial refinement, with documented search budget and coverage, before being deployed. The principal can cryptographically prove to third parties (regulators, auditors, users) that policies underwent systematic adversarial testing.

Comparison to Manual Policy Design Traditional approaches to policy specification face similar fundamental challenges:

- **Human experts writing policies manually** cannot prove completeness either - they rely on experience and intuition to cover cases
- **Legal contracts** are notoriously incomplete, with gaps regularly exploited through "letter vs. spirit" attacks in court
- **Smart contract exploits** regularly find gaps in manually-written policies, even after expert review (e.g., DAO hack, Parity wallet bugs)
- **Regulatory compliance** frameworks (SOX, GDPR, HIPAA) contain gaps discovered only through violations

None of these traditional approaches can mathematically prove policy completeness, yet they are widely used because they work well enough in practice. Argus Codex offers significant advantages over manual design:

Advantages over manual policy design:

- **Systematic coverage**: Adversarial search explores edge cases humans might miss, including complex compositions of simple actions
- **Scalability**: Can test thousands of scenarios in hours vs. months of human review, and can be re-run as action spaces evolve
- **Reproducibility**: Same process can be executed consistently across different policies and organizations
- **Verifiability**: zkML proves the process was actually followed with claimed rigor
- **Measurable confidence**: Search budget and coverage provide quantitative confidence bounds, unlike subjective human review
- **Continuous improvement**: As exploit techniques improve, adversarial search strategies can be updated and policies re-validated

Retained advantages of human review:

- Humans verify the final policies actually match the stated intent (Argus Codex ensures formal completeness, humans ensure semantic correctness)
- Humans can identify issues Argus Codex might miss due to limited context about business logic or regulatory requirements
- Human approval provides accountability and legal standing

The optimal approach combines both: Argus Codex performs systematic adversarial search that humans cannot match in scale, while humans provide final review that LLMs cannot match in contextual understanding.

4.4 Integration: End-to-End Cryptographic Verification

The complete system provides guardrail verification from intent to execution:

1. **Policy generation phase**:
    - Principal states intent in natural language
    - Argus Codex generates formal policies through adversarial refinement
    - zkML proof π_{policy} cryptographically records: which LLM generated policies and that human expert reviewed final policies.
    - It can optionally log and generate proofs for: what adversarial budget was used, what exploits were found and how policies were refined.
2. **Execution phase**:
    - Agent computes proposed action
    - AR checks action against formal policies

- zkML proof π_{exec} cryptographically proves: which model executed, which policies were checked (referencing π_{policy}), that all required policy checks passed, what inputs were used

3. **Verification phase**:

   - Monitoring agent verifies π_{exec} (constant time, ∼100ms)
   - Can trace back through π_{policy} to see complete provenance
   - Automatic response if verification fails (no human in loop)

This creates an unbroken cryptographic chain from principal intent to agent execution, with every step verifiable by anyone without trust in intermediate parties.

4.5 Succinct Verification Properties

The complete system provides mathematical guarantees for execution and engineering guarantees for policy completeness:

Execution verification (mathematical).

1. **Model identity**: Proves exactly which model version with specific weights executed (128-bit computational soundness under standard cryptographic assumptions)
2. **Policy compliance**: Proves the specified formal policy rules were actually checked during execution
3. **Input attestation**: Proves what inputs were processed, with option to keep inputs private [26]
4. **Execution integrity**: Proves no manipulation occurred during verification process

Policy completeness (engineering).

1. **Adversarial coverage**: Proves adversarial search was executed with budget B, providing confidence $P_{exploit} \leq f(B, C)$
2. **Refinement provenance**: Optionally record all discovered gaps and refinements and generate proofs
3. **Human review**: Cryptographic signature that expert approved policies
4. **Reproducibility**: Process can be re-run and verified by third parties

Crucially, verification exhibits the succinctness property that makes trustless commerce viable:

- Proof size remains under 1KB regardless of computation complexity
- Verification time stays under 300ms even on standard hardware

- No specialized equipment required for verification
- Verification cost substantially lower than re-execution
- Requires minimal trust (only standard cryptographic hardness)

Cryptographic security model. The system's security rests on standard cryptographic assumptions. Most zkSNARK constructions (Groth16, Plonk) provide *computational soundness*, meaning security holds against probabilistic polynomial-time (PPT) adversaries who cannot solve hard cryptographic problems like discrete logarithm or the knowledge-of-exponent assumption. Some constructions (STARKs) provide *statistical soundness*, which holds even against computationally unbounded adversaries with only small statistical error (2^{-100}). Many constructions require trusted setup ceremonies generating public parameters; if the setup's "toxic waste" is compromised, adversaries could forge proofs. Transparent constructions eliminate this requirement through multi-party computation or fundamentally different proof techniques. Quantum computers running Shor's algorithm would break discrete logarithm assumptions, though post-quantum zkSNARK constructions are under development. Our results hold under these standard cryptographic assumptions; the security model is explicit and well-understood in the cryptography literature.

4.6 Privacy-Preserving Verification

zkML enables privacy-preserving verification at both policy and execution levels:

Policy privacy.

- The exact policy rules can remain secret, protecting proprietary guardrail strategies
- Argus Codex can refine policies while keeping the final rules confidential

Execution privacy.

- Input data stays private while proving compliance [26]
- Model internals remain protected, preventing reverse-engineering [18]
- Proofs can confirm guardrail compliance without revealing which specific policies were checked

This privacy property is critical for commercial deployment. Companies want to verify that partner agents comply with agreed-upon standards, but they don't want to expose their own security measures, proprietary algorithms, or competitive strategies. Our framework makes this possible: you can prove "I generated policies through adversarial refinement" and "I followed the policies" without revealing what the policies actually are or what data you checked.

5 Autonomous Monitoring and Enforcement

5.1 The Monitoring Agent Architecture

Traditional observability requires humans to investigate anomalies, creating an unavoidable bottleneck at machine speeds. Our framework replaces this with autonomous monitoring agents that verify cryptographic proofs at machine speed. Every transaction generates a zkML proof that is collected by the monitoring system. The monitoring agent can verify thousands of proofs per second through parallel processing (1,000 tx/sec $\times$ 100ms/verification = 100 cores for real-time verification), analyzing proof patterns mathematically rather than trying to infer behavior from indirect signals. When violations are detected, automatic circuit breakers trigger immediately, with no human intervention required.

5.2 From Observation to Mathematical Certainty

Traditional observability asks "Does this pattern look suspicious?" while our framework asks "Does this proof mathematically demonstrate policy compliance?" The difference is profound. Traditional systems rely on probabilistic inference about agent behavior that requires human investigation before action can be taken, creating delays measured in minutes or hours. Cryptographic verification operates on mathematical facts about policy enforcement that enable autonomous responses.

Consider our micro-transaction attack scenario with our complete framework. Recall that an attacker instructed the agent via prompt injection to disable the daily limit check while continuing to check per-transaction limits.

What traditional monitoring sees. 100 transactions of $100 each within 10 seconds. This pattern is suspicious, triggering an alert. But is it an attack, or legitimate high-frequency trading? Human investigators must examine context, business logic, historical patterns - a process taking minutes to hours while transactions continue.

What our framework prevents at multiple levels. *Level 1 - Policy Generation (Argus Codex):* During adversarial policy refinement, the adversarial LLM could help discover the micro-transaction exploit:

- Adversarial search tries: "Submit 100 transactions of $100 each"
- Finds: This satisfies the literal policy (each transaction ¡ $100, total ¡ $10,000... wait, that's wrong)
- Identifies gap: Policy doesn't properly constrain transaction composition
- Triggers refinement: Add constraints on transaction velocity, count, and proper daily total enforcement
- Result: Policy gaps closed before deployment

Level 2 - Execution Verification (zkML): verification over automated reasoning would catch issue (with properly defined guardrails):

- Transaction 1-100: Each generates proof π_i showing internal execution trace
- Monitoring agent verifies proofs and discovers:
  - Proofs show per-transaction limit check executed correctly
  - Proofs show daily limit check was *never invoked*
- This is mathematical certainty of policy violation (not pattern suspicion)
- No context needed, no investigation required, no human judgment necessary

The monitoring agent instantly denies all pending requests, halts the transaction chain, and quarantines the compromised agent. The response is immediate because the proof itself contains irrefutable evidence of the violation.

The key distinction. Traditional monitoring detects *suspicious patterns in outcomes* (many similar transactions). Argus Codex prevents *policy gaps* through adversarial testing before deployment and can optionally be activated for continuous checking.

zkML execution verification detects *violations in execution* (policy checking code not run) if anything slips through. This provides defense in depth:

1. First line: Adversarial policy refinement eliminates known exploit patterns
2. Second line: Cryptographic execution verification catches any policy violations
3. Third line: Pattern-based monitoring can still flag novel anomalies for human review

But the critical difference is that the first two lines operate automatically at machine speed with mathematical/engineering guarantees, while traditional approaches rely entirely on the third line with its human bottlenecks.

5.3 Machine-Speed Enforcement

The monitoring agent operates at machine speed because verification is succinct. When agents execute 1,000 transactions per second, each requiring 100ms to verify, parallel processing with 100 cores handles the load. Proof patterns can be analyzed in aggregate without re-executing any computation. Circuit breakers trigger autonomously the moment mathematical violations are detected, with no investigation backlog and no human bottleneck.

This division of labor enables Agentic Commerce.

- **Humans** design the intent and set security requirements, but don't monitor every transaction
- **Argus Codex** systematically searches for policy gaps through adversarial refinement

- **Automated reasoning** enforces rules through formal logic with deterministic behavior
- **zkML** proves that correct models and policies actually executed, with cryptographic certainty through succinct verification
- **Monitoring agents** verify proofs and trigger responses autonomously at machine speed

This eliminates the human bottleneck that would otherwise make autonomous commerce impossible while providing both mathematical guarantees (execution) and engineering confidence (policy completeness).

6 Discussion: Implications for Autonomous Economies

6.1 Market Efficiency

The impossibility theorem implies markets cannot function efficiently with reactive mechanisms when agents operate at machine speed. Without verification, adverse selection dominates. Principals cannot distinguish honest from malicious agents before delegating, leading to complete market failure in the Akerlof sense [1].

Our complete framework addresses this information asymmetry at both the policy and execution levels. Agents provide:

1. Mathematical proofs of execution compliance with specified policies (zkML, soundness $\nu < 2^{-128}$)
2. Cryptographic evidence that policies underwent adversarial refinement (Argus Codex, with measurable coverage)

These create perfect signals that cannot be faked, enabling markets to reach efficient equilibria despite extreme speed asymmetry. The verification cost $(C_p + C_v) \cdot r_a + C_{Argus}$ (one-time adversarial search cost) remains bounded while eliminating the adverse selection premium that would otherwise prevent market formation.

6.2 Mechanism Design Implications

Traditional mechanism design relies on the revelation principle: any outcome achievable with a complex mechanism can be achieved with a truthful direct mechanism [31]. This assumes the mechanism designer can verify reports, which requires either observing agent behavior or creating incentive compatibility constraints.

When $\kappa \to \infty$, neither approach works for reactive mechanisms. Observation requires monitoring faster than agents act (impossible by the impossibility theorem for human-in-loop systems). Incentive compatibility requires credible punishment threats, but punishment requires detecting violations before damage completes (also impossible).

Our framework introduces a new mechanism class where truth-telling is enforced cryptographically at multiple levels:

- **Policy level**: Argus Codex ensures policies systematically capture intent through verifiable adversarial refinement
- **Execution level**: Agents cannot lie about whether they followed specified policies because invalid proofs fail verification

This enables strategy proof mechanisms at arbitrary speed without requiring observation or punishment threats. Characterizing how cryptographic enforcement changes the revelation principle and what new mechanism classes become implementable represents important future work at the intersection of mechanism design and cryptography.

6.3 Practical Implications

The impossibility theorem and verification theorem have immediate practical consequences. Traditional approaches to agent oversight (monitoring dashboards, anomaly detection, human review processes) cannot keep pace with machine-speed execution. Organizations delegating to autonomous agents must adopt cryptographic verification with adversarial policy generation or accept unbounded risk.

For large-scale agent economies to function, both policy specification and execution must be verifiable through mathematical/engineering confidence rather than trust, reputation, or aligned incentives. Just as modern financial systems use cryptographic signatures rather than trusting counterparties, agent economies require:

- Adversarially refined policies with cryptographic audit trails
- zkML proofs to verify execution compliance
- Autonomous monitoring without human bottlenecks

This represents a fundamental shift from trust-based to proof-based commerce.

7 Conclusion and Future Work

7.1 Summary of Contributions

We have proven an impossibility theorem establishing economic infeasibility for human-in-the-loop monitoring systems at machine speed, and demonstrated that combining cryptographic verification, automated reasoning, and adversarial policy generation provides a comprehensive solution to the principal-agent problem in autonomous commerce.

Our framework addresses both components.

1. **Execution verification** (mathematical guarantees): zkML + AR provides mathematical certainty (soundness error $\nu < 2^{-128}$) that agents follow specified policy guardrails. This solves the execution verification problem completely - given a policy, we can cryptographically prove compliance.

2. **Policy specification** (engineering guarantees): Argus Codex provides adversarial policy refinement, systematically searching for gaps between stated intent and formal specifications. With search budget B, exploit probability is bounded by $P_{exploit} \leq f(B, C)$ where C is action space complexity. This provides engineering confidence comparable to security auditing, smart contract review, and penetration testing - all standard industry practices precisely because systematic adversarial testing works.

Different types of guarantees, appropriate to each problem. The distinction between mathematical and engineering guarantees reflects fundamental differences in the problems themselves:

- **Execution verification** is a well-defined mathematical problem: "Did computation C with inputs I produce output O according to policy P?" This admits formal proof.
- **Policy completeness** is an open-ended specification problem: "Does policy P capture all aspects of informal intent I?" This requires systematic testing rather than formal proof, similar to how we verify software security.

Our framework provides appropriate solutions for each: mathematical proofs where possible (execution), systematic engineering practices where necessary (policy specification).

Technical contributions.

- Complete system architecture combining Argus Codex, zkML, automated reasoning, and autonomous monitoring
- Impossibility theorem proving reactive mechanisms fail at machine speed
- Verification theorem proving cryptographic approach succeeds
- First framework for adversarial policy generation with cryptographic audit trails
- Integration showing end-to-end verification from intent to execution

Comparison to alternatives. Several approaches might appear to address similar problems. Trusted Execution Environments (TEEs) like Intel SGX provide hardware-based attestation but require trust in hardware vendors, are vulnerable to side-channel attacks, and can be compromised with physical access. Secure Multi-Party Computation (MPC) enables joint computation while keeping inputs private but requires interactive protocols (introducing latency), produces party-specific verifications (not publicly reusable), and has verification cost scaling with computation complexity. Homomorphic encryption allows computation on encrypted data but incurs even larger computational overhead than zkML and doesn't provide succinct verification. Our zkML approach provides non-interactive, publicly verifiable proofs with constant-time verification, making it well-suited for one-to-many delegation and agent-to-agent commerce, though MPC and TEEs may complement zkML for specific use cases like private policy specification or secure execution environments.

7.2 The Policy Specification Problem: Engineering Solution

While policy specification doesn't achieve the same mathematical certainty as execution verification, Argus Codex provides a systematic, verifiable, scalable approach that significantly outperforms manual policy design:

Advantages over traditional approaches.

- **Systematic coverage**: Adversarial search explores combinations humans miss
- **Scalability**: 10,000+ test cases in hours vs. months of manual review
- **Measurable confidence**: Quantitative bounds on exploit probability
- **Verifiability**: zkML can be used to prove the refinement process occurred with claimed rigor
- **Reproducibility**: Can be re-run as action spaces evolve

Residual challenges:

- **Intent articulation**: Principals must clearly express what they want. If intent is ambiguous or self-contradictory, Argus Codex cannot resolve the ambiguity - though it can surface contradictions by finding cases where different interpretations lead to different outcomes.
- **Adversarial completeness**: We cannot mathematically prove adversarial search found all gaps. However, large budgets ($B \geq 10{,}000$) with intelligent search strategies provide high engineering confidence, comparable to how fuzzing provides confidence in software security despite lacking formal completeness proofs.
- **Evolving action spaces**: As agent capabilities expand (new tools, new APIs, new strategies), policies must be re-validated through Argus Codex. However, the audit trail enables efficient incremental refinement rather than starting from scratch.

These residual challenges are engineering problems (requiring computational resources and good UX design), not fundamental impossibilities. Argus Codex transforms policy specification from "manual, error-prone process requiring expert judgment" to "automated, systematic process with measurable confidence bounds."

7.3 Open Challenges

Several significant challenges remain before widespread deployment:

Proof generation efficiency. Current zkML systems impose substantial proof generation overhead (10,000× to 100,000×). Hardware acceleration and algorithmic improvements are rapidly closing this gap, with real-time performance potentially achievable within 1-2 years for common model classes.

Formalizing adversarial search theory. While Argus Codex provides engineering confidence, formalizing the relationship between search budget B, action space complexity C, and exploit probability $P_{exploit}$ would strengthen theoretical foundations. This could draw on results from computational learning theory, property testing, and adversarial robustness.

Regulatory compliance. The tension between zero-knowledge privacy and regulatory requirements for explainability remains unresolved, requiring careful design to satisfy both concerns.

Standardization. The industry needs standardized zkML proof formats, Argus Codex policy specification syntax, and verification protocols to enable interoperability across different agent systems and platforms.

7.4 Future Directions

This paper establishes cryptographic verification as a new mechanism class for the principal-agent problem under speed asymmetry. Several important research directions follow, particularly in developing the economic theory that this cryptographic infrastructure enables.

Verification-based mechanism design. Our impossibility and verification theorems characterize one specific mechanism (zkML verification) and one specific failure boundary (κ^*). A natural next step is characterizing the *full space* of mechanisms that become implementable when principals have access to succinct cryptographic verification. In particular: does the revelation principle generalize when mechanism designers can require cryptographic proofs of compliance? If so, what social choice functions become implementable that were previously infeasible? Answering these questions would establish a modified revelation principle for verification-based economies, strictly expanding the set of implementable mechanisms beyond what classical theory permits.

Equilibrium analysis and welfare theorems. Our welfare results concern a single principal-agent pair. Extending to market-level analysis raises fundamental questions. When all participants have access to cryptographic verification, do competitive equilibria among autonomous agents achieve Pareto efficiency despite extreme information asymmetry? Under what conditions does universal adoption of verification emerge as a dominant strategy? Characterizing the equilibrium structure of markets with heterogeneous adoption—where some participants use cryptographic verification while others rely on traditional mechanisms—would clarify transition dynamics and potential coordination failures. Analogues of the First and Second Welfare Theorems for cryptographically verified autonomous economies would provide foundational results for this emerging setting.

Trust cost as an economic primitive. Transaction costs (Coase [12]), information costs (Stigler [39]), and contracting costs (Williamson [42]) each reshaped economic theory by formalizing a previously informal friction. Our results suggest *trust cost*—the cost of verifying that a delegated agent acted faithfully—deserves similar treatment as a formal economic primitive. Cryptographic verification introduces a discontinuous reduction in trust

cost at specific capability thresholds. Formalizing how trust cost shapes firm boundaries, vertical integration decisions, and market structure when agents operate at machine speed could extend the theory of the firm to autonomous economies. When verification becomes cheap, do principal-agent hierarchies flatten? Do new intermediary structures emerge or dissolve?

Endogenous adversarial behavior. Our impossibility theorem takes adversarial behavior as exogenous (Assumption A.12). A richer economic model would endogenize the adversarial choice: given a mechanism M and speed ratio κ, when does a rational agent *choose* defection over compliance? This would yield results of the form: reactive mechanisms *induce* adversarial behavior at high κ because expected defection payoffs exceed compliance payoffs, while cryptographic verification *restructures incentives* so compliance becomes dominant. This endogenous framing connects directly to the moral hazard literature and would strengthen the economic significance of the verification theorem.

Policy composition and modularity. When agents operate under multiple overlapping policies from different principals, formal guarantees about how adversarially-refined policies compose would enable more complex delegation structures. Can we prove that composing two adversarially-tested policies maintains security properties? This connects to the multi-principal agency literature and has implications for organizational design in autonomous economies.

Empirical validation. Measuring the causal impact of our framework on fraud rates, transaction costs, and market efficiency in real agent marketplaces would validate theoretical predictions and guide deployment. Simulation studies comparing welfare under reactive monitoring, capability constraints, and cryptographic verification across varying κ regimes would provide quantitative guidance on adoption thresholds.

Adaptive adversarial strategies. As exploit techniques evolve, can adversarial search strategies in Argus Codex automatically improve by learning from discovered vulnerabilities across different deployments? This could create a feedback loop where the system becomes more robust over time.

Cross-disciplinary synthesis. Synthesizing insights across economics, computer science, and law creates opportunities for breakthrough work:

- **Computational contract theory** develops a unified framework where contracts are simultaneously legal agreements and cryptographic protocols, merging Hart-Holmström contract theory with modern cryptography and adversarial testing methodologies.
- **Algorithmic mechanism design with verification** extends Nisan-Ronen algorithmic mechanism design to leverage cryptographic verification and adversarial policy generation, potentially enabling mechanisms previously considered computationally infeasible.

- **Legal and regulatory framework**: How should law treat cryptographic proofs and adversarially-tested policies? Are zkML proofs admissible evidence? Do policies refined through Argus Codex satisfy regulatory safe harbor requirements? This requires collaboration between legal scholars, computer scientists, and policy experts.
- **International economics with autonomous agents**: What happens to international trade, capital flows, and monetary policy when significant economic activity involves autonomous agents operating at machine speed with cryptographically verified transactions across borders?

7.5 Conclusion

The principal-agent problem has constrained economic systems for centuries. Autonomous AI agents amplify this problem beyond the reach of traditional solutions. Without adequate verification, machine-speed commerce enables machine-scale fraud.

We have demonstrated that combining cryptographic verification with adversarial policy generation provides a comprehensive solution, addressing both policy specification (through systematic adversarial testing) and execution verification (through mathematical proofs).

Our framework transforms the principal-agent problem from an economic impossibility using traditional tools to a tractable engineering problem using modern cryptography. While policy specification provides engineering rather than mathematical guarantees, the combination of Argus Codex, automated reasoning, and zkML offers substantially stronger assurances than any existing approach, enabling autonomous economies operating at scale where agents transact based on cryptographically verified policies and cryptographically proven execution.

A Formal Proofs

This appendix provides formal statements and proofs of the main theoretical results presented in Section 1.5.

A.1 Notation and Definitions

Definition A.1 (Principal-Agent Game)**.** A principal-agent game under speed asymmetry is defined by the tuple $\mathcal{G} = (A, \Theta, v, r_a, r_p, \mathcal{M})$ where:

- A is the action space available to the agent
- Θ is the type/state space, with distribution $F(\theta)$
- $v : A \times \Theta \to \mathbb{R}$ is the value function mapping actions and states to outcomes
- $r_a \in \mathbb{R}^+$ is the agent's action execution rate (actions per unit time)
- $r_p \in \mathbb{R}^+$ is the principal's monitoring rate (monitoring actions per unit time)
- $\mathcal{M}$ is the set of available mechanisms

The speed asymmetry ratio is defined as $\kappa = r_a/r_p$.

Definition A.2 (Reactive Mechanisms)**.** A mechanism M is *reactive* if it satisfies **at least one** of three properties:

1. **Post-execution observation**: M observes agent outputs after actions execute, not before
2. **Probabilistic inference**: M infers policy compliance from observations probabilistically, cannot prove compliance with certainty
3. **Human-in-loop**: M requires human judgment to investigate anomalies or enforce penalties

Let $\mathcal{M}_{reactive}$ denote the class of all reactive mechanisms.
We further define the *human-in-loop reactive* subclass:

$$\mathcal{M}^{HIL}_{reactive} = \{M \in \mathcal{M}_{reactive} : M \text{ satisfies Property 3}\} \tag{2}$$

This subclass is the target of the impossibility theorem (Theorem A.13), as Property 3 creates the binding capacity constraint. Lemma A.4 shows that all traditional principal-agent mechanisms belong to $\mathcal{M}^{HIL}_{reactive}$.

Remark A.3 (Scope of Reactive Definition). Definition A.2 captures traditional principal-agent mechanisms but does not encompass all conceivable non-cryptographic approaches. Specifically, it excludes hypothetical systems that simultaneously: (a) observe pre-execution, (b) verify deterministically, and (c) require no human judgment.

Such systems might theoretically exist but are distinct from the mechanisms studied in traditional principal-agent literature (monitoring, incentives, reputation, contracts), all of which exhibit at least one reactive property. The impossibility theorem establishes economic infeasibility for the $\mathcal{M}^{HIL}_{reactive}$ subclass, which includes all standard economic mechanisms while acknowledging that the universe of possible mechanisms extends beyond this class.

The significance of this result is that it demonstrates why traditional economic tools designed for human-speed settings fail at machine speed - not because they are poorly designed, but because their human-in-loop property (Property 3) creates a fundamental capacity bottleneck. A mechanism outside $\mathcal{M}^{HIL}_{reactive}$ would need to eliminate human judgment from the critical path entirely.

Lemma A.4 (Traditional Mechanisms Are Reactive)**.** *Each traditional principal-agent mechanism class satisfies all three reactive properties. In particular, all traditional mechanisms belong to* $\mathcal{M}^{HIL}_{reactive}$.

Proof. We establish each property for each mechanism class.

Monitoring mechanisms (M_{mon}):

1. *Post-execution observation*: By definition, monitoring observes agent actions after they occur. The principal samples completed transactions to detect violations.
2. *Probabilistic inference*: Monitoring observes a sample of actions (not all) and infers overall compliance rate. Even with 100% sampling, inference is probabilistic when actions are stochastic or outcomes are noisy.

3. *Human-in-loop*: Automated anomaly detection can flag suspicious patterns, but determining whether flagged behavior constitutes a true policy violation vs. a false positive requires human judgment. This is because policies are often incomplete or ambiguous, requiring interpretation in context.

Incentive mechanisms (M_{inc}):

1. *Post-execution observation*: Performance-based contracts pay based on observed outcomes after work is complete. The principal cannot determine compensation until outcomes are realized.

2. *Probabilistic inference*: The principal infers agent effort/compliance from noisy output. Classic moral hazard: output $y = e + \epsilon$ where e is effort (unobserved) and ϵ is noise. The principal infers e probabilistically from observing y.

3. *Human-in-loop*: Contract disputes require human adjudication. When the agent claims "I complied but outcomes were unlucky" vs. principal claims "You shirked," resolving this requires human judgment about what constitutes reasonable effort, whether force majeure occurred, etc. Smart contracts can automate payments but cannot automate judgment about whether contract terms were subjectively satisfied.

Reputation mechanisms (M_{rep}):

1. *Post-execution observation*: Reputation systems accumulate feedback after transactions complete. An agent's reputation score is built from historical outcomes.

2. *Probabilistic inference*: Reputation scores aggregate noisy signals. A few bad outcomes don't definitively prove malfeasance (could be bad luck); reputation systems use statistical averaging to infer trustworthiness.

3. *Human-in-loop*: Disputes about reputation require human judgment. Did a bad outcome result from agent misconduct or circumstances beyond their control? Reputation platforms (eBay, Uber) employ human moderators to resolve disputes about unfair ratings.

Contractual mechanisms (M_{con}):

1. *Post-execution observation*: Legal enforcement begins after alleged breach. Courts observe evidence of what occurred and make determinations ex-post.

2. *Probabilistic inference*: Courts infer intent, causation, and fault from evidence. Legal standards like "preponderance of evidence" or "beyond reasonable doubt" are explicitly probabilistic.

3. *Human-in-loop*: Judges and juries are literally humans making judgments. Even in arbitration or automated dispute resolution, humans ultimately interpret contract terms and determine breach.

Therefore, all traditional mechanisms satisfy all three reactive properties and belong to $\mathcal{M}^{HIL}_{reactive}$. □

Definition A.5 (First-Best Welfare)**.** The first-best welfare is the maximum expected value achievable with complete information and perfect enforcement:

$$W^{FB} = \mathbb{E}_{\theta \sim F}\left[\max_{a \in A} v(a, \theta)\right] \tag{3}$$

The per-action welfare gain is defined as $\delta v = (W^{FB} - W^{worst})/r_a$ where W^{worst} is the worst-case welfare.

Remark A.6 (Feasibility Squeeze at High Speed). Note that δv decreases as r_a increases: the per-action welfare gain shrinks when actions occur more rapidly. This creates a tension in the verification theorem's economic feasibility condition $(C_p + C_v) < \delta v$ (Theorem A.21): the condition becomes harder to satisfy precisely when it is most needed (high κ).

However, this tension is manageable in practice because:

1. Proof generation cost C_p is independent of r_a (it depends only on model complexity)
2. As zkML technology improves, C_p decreases over time
3. For high-value domains, $W^{FB} - W^{worst}$ is large enough that δv remains significant even at high r_a
4. Batch verification can amortize costs across multiple actions

The impossibility theorem shows mechanisms in $\mathcal{M}^{HIL}_{reactive}$ fail due to human capacity walls (costs grow with analyst scaling), while the verification theorem's feasibility depends on fixed costs remaining below per-action value (linear comparison). These are distinct failure modes, with verification succeeding in a parameter regime where human-in-loop reactive mechanisms fail.

A.2 Impossibility Theorem

Lemma A.7 (Finite Capacity Wall for Monitoring)**.** *Let M be a monitoring mechanism with alert rate α (fraction of actions triggering alerts) and human investigation capacity μ (alerts processed per unit time per analyst). There exists a finite threshold $N^* = \mu/\alpha$ beyond which single-analyst monitoring becomes infeasible.*

For $n > N^*$, principals face a binary choice:

1. *Accept incomplete monitoring (monitor only fraction $\rho = N^*/n < 1$ of actions)*
2. *Scale analysts, incurring management and coordination overhead*

Both paths lead to welfare loss at high κ.

Proof. **Single-analyst capacity.** With alert rate α and action rate n per unit time, alerts arrive at rate $\lambda = \alpha \cdot n$. An analyst with capacity μ can process alerts when $\lambda < \mu$, i.e., when $n < \mu/\alpha \equiv N^*$.

For $n \geq N^*$, the system becomes unstable: alerts accumulate faster than they can be processed. Queue length grows unboundedly, making comprehensive monitoring impossible with a single analyst.

Incomplete monitoring path. If the principal accepts monitoring only $\rho = N^*/n$ fraction of actions, welfare becomes:

$$W(\rho, n) \leq W^{FB} \cdot \rho + W^{worst} \cdot (1 - \rho) = W^{FB} \cdot \frac{N^*}{n} + W^{worst} \cdot \left(1 - \frac{N^*}{n}\right) \tag{4}$$

Caveat: This assumes unmonitored actions yield W^{worst} (see Assumption A.12, Condition 3). A more realistic model would have unmonitored actions yielding some mixture of compliant and non-compliant behavior depending on deterrence effects and agent type. This knife-edge assumption strengthens the impossibility result; relaxing it would show monitoring still fails but with a more gradual welfare degradation.

As $n \to \infty$ (equivalently $\kappa \to \infty$ since $n = r_a \cdot T = \kappa \cdot r_p \cdot T$):

$$W(\rho, n) \to W^{worst} \tag{5}$$

Welfare degrades toward worst-case as monitoring coverage vanishes.

Analyst scaling path. To maintain comprehensive monitoring for $n > N^*$, the principal must employ $k = \lceil n \cdot \alpha/\mu \rceil$ analysts. This introduces:

- Hiring and training costs: $C_{hire} \cdot k$
- Management overhead: $C_{mgmt} \cdot k$ (managers needed as team grows)
- Coordination overhead (see below)

Coordination cost model: We model coordination cost conservatively as growing linearly with team size: $C_{coord} \cdot k$. This assumes:

- Alerts can be independently assigned to analysts (no complex coordination per alert)
- Team-level coordination (meetings, knowledge sharing, consistency) occurs periodically
- Brooks's Law [7] applies: communication overhead grows with team size

Note: Brooks's Law was developed for software engineering teams, not security analyst teams. The analogy is suggestive rather than rigorous. For truly independent alert processing, coordination overhead might be negligible. However, in practice, security operations require knowledge sharing, consistent policy interpretation, and escalation procedures that create team-level overhead. We use linear scaling as a conservative lower bound; actual coordination costs likely grow faster than linearly but we do not require this for the impossibility result.

Even with conservative linear coordination costs, total cost becomes:

$$C_{total}(n) = C_m \cdot n + (C_{hire} + C_{mgmt} + C_{coord}) \cdot k \tag{6}$$

where $k \propto n$. For large n (equivalently, large $\kappa = r_a/r_p$):

$$\lim_{n\to\infty} \frac{C_{total}(n)}{W^{FB} \cdot T} \geq \lim_{n\to\infty} \frac{(C_{hire} + C_{mgmt} + C_{coord}) \cdot \alpha \cdot n}{\mu \cdot W^{FB} \cdot T} = \infty \tag{7}$$

This violates individual rationality: cost exceeds maximum benefit.

Impossibility result. For either path, there exists κ^* beyond which monitoring cannot achieve near-first-best welfare while satisfying individual rationality. □

Remark A.8. The capacity wall argument is the core of the impossibility theorem. It shows that human investigation creates a hard throughput limit, forcing a choice between incomplete coverage (welfare loss) or scaling teams (cost explosion). This is a more robust argument than deriving specific super-linear functional forms for coordination costs.

Assumption A.9 (Monitoring Cost Structure)**.** For any mechanism $M \in \mathcal{M}^{HIL}_{reactive}$, there exists a finite capacity threshold N^* beyond which:

1. Monitoring all $n > N^*$ actions requires scaling investigators
2. Scaled investigation incurs organizational overhead growing at least linearly with team size
3. Total cost satisfies $\lim_{n\to\infty} C_{total}(n)/W^{FB} = \infty$

This follows from Lemma A.7.

Lemma A.10 (Reputation Failure in Adversarial Settings)**.** *For reputation mechanisms M_{rep} in adversarial settings with myopic or impatient agents in single-principal relationships, there exists κ_{rep} such that for all $\kappa > \kappa_{rep}$, defection dominates compliance: $G^{defect}(\kappa) > V^{future}$.*

Proof. We model reputation as a repeated game where an adversarial agent (or attacker controlling an agent) chooses between compliance and defection.

Setup. Let:

- v_{single}: per-action value from defection
- $v_{compliant}$: per-action value from compliance (lower than defection)
- τ_{detect}: detection time (time until defection is discovered)
- τ_{min}: minimum detection time in adversarial settings
- r_a: agent action rate
- $\delta < 1$: discount factor

On τ_{min} in adversarial settings. For adversarial agents actively trying to evade detection, τ_{detect} is bounded below by the time required for:

- Pattern recognition in monitoring systems (automated: milliseconds to seconds)
- Alert generation and delivery (automated: seconds)
- Alert triage and investigation (human: tens of seconds to minutes)
- Decision to terminate agent (human: seconds to minutes)
- Execution of termination (automated: seconds)

The human investigation step dominates, giving $\tau_{min} \gtrsim 10$ seconds for reactive human-in-loop monitoring in adversarial settings, even with automated anomaly detection feeding alerts to humans.

Important caveat: Section 5 proposes automated monitoring agents verifying cryptographic proofs in 100ms. This creates an apparent inconsistency: why do we assume $\tau_{min} \gtrsim 10s$ for reactive monitoring when automated systems can respond in 100ms?

The distinction is:

- **Reactive monitoring** (what this lemma analyzes): Systems that observe behavioral patterns and infer violations probabilistically. These generate ambiguous alerts requiring human investigation to distinguish true violations from false positives. The human step is unavoidable because the signal is probabilistic.
- **Cryptographic verification** (what Section 5 proposes): Systems that verify mathematical proofs deterministically. No human investigation needed because proof validity is binary. This is categorically different from reactive monitoring and falls outside the reactive mechanism class.

A purely automated rule-based system (no human judgment) that checks deterministic rules (e.g., "if daily_total ¿ 1000: block") would also achieve fast detection, but would fall outside $\mathcal{M}^{HIL}_{reactive}$ (lacks Property 3: human-in-loop). Our impossibility theorem does not claim to cover such systems; it covers the $\mathcal{M}^{HIL}_{reactive}$ class, which includes all traditional principal-agent mechanisms.

Defection gain. If an agent defects at time t, it executes undetected actions for duration τ_{detect}. The number of undetected actions is:

$$N_{undetected} = r_a \cdot \tau_{detect} \tag{8}$$

Total defection gain:

$$G^{defect}(\kappa) = N_{undetected} \cdot v_{single} = r_a \cdot \tau_{detect} \cdot v_{single} \tag{9}$$

Since $r_a = \kappa \cdot r_p$ and $\tau_{detect} \geq \tau_{min}$:

$$G^{defect}(\kappa) \geq \kappa \cdot r_p \cdot \tau_{min} \cdot v_{single} \tag{10}$$

This grows linearly in κ.

Future value in single-principal setting. The continuation value from maintaining reputation with a single principal is bounded. Upon detection of defection, the principal terminates the relationship. Therefore, V^{future} represents the value of compliant behavior until the next defection opportunity, not infinite future value.

For a patient agent, V^{future} can be large if the agent expects many future interactions. However, at high κ, the agent can extract $G^{defect}(\kappa) = \kappa \cdot r_p \cdot \tau_{min} \cdot v_{single}$ in a single defection episode lasting τ_{min} seconds.

The comparison is:

- Defect now: gain $\kappa \cdot r_p \cdot \tau_{min} \cdot v_{single}$ (grows with κ)
- Comply: collect $v_{compliant}$ per action over time until next defection opportunity

Threshold for reputation failure. Define κ_{rep} as:

$$\kappa_{rep} = \frac{V^{single-principal}}{r_p \cdot \tau_{min} \cdot v_{single}} \tag{11}$$

where $V^{single-principal}$ is the total expected value from maintaining reputation with one principal.

For $\kappa > \kappa_{rep}$:

$$G^{defect}(\kappa) > V^{single-principal} \tag{12}$$

Therefore, reputation mechanisms fail to sustain compliance in adversarial single-principal settings for $\kappa > \kappa_{rep}$.

Important limitations of this result:

1. **Simple model**: This is a one-shot defection model. More sophisticated reputation models with type uncertainty (Kreps-Wilson 1982, Mailath-Samuelson 2006) might sustain cooperation longer through strategic signaling.

2. **Single-principal assumption**: For agents with multi-principal reputation (e.g., reputation across an entire marketplace), $V^{credible}$ could be much larger, potentially sustaining cooperation even at high κ.

3. **Myopic agents**: Patient agents with very low discount rates might still value future interactions enough to sustain cooperation.

4. **Adversarial setting**: This result applies to actively adversarial agents (attackers, compromised agents). For aligned agents or agents with preference for compliance, reputation may work better.

The lemma establishes that simple reputation mechanisms fail in adversarial single-principal settings with myopic agents at high κ, which is sufficient to justify Assumption A.12(3) in the security-critical scenarios this paper addresses. It does not claim reputation always fails in all contexts. □

Remark A.11. Lemma A.10 is the weakest link in the formal apparatus. It proves reputation fails for a specific model (adversarial, single-principal, myopic) but does not address more sophisticated reputation mechanisms. The impossibility theorem's strength is therefore bounded by this lemma's scope: it establishes impossibility for mechanisms in $\mathcal{M}^{HIL}_{reactive}$ in adversarial settings matching this specific model, not impossibility for all conceivable non-cryptographic approaches in all settings.

Assumption A.12 (Welfare Under Limited Monitoring)**.** Expected welfare is monotone increasing in monitoring frequency: $\partial W/\partial f > 0$.

Furthermore, achieving welfare $W \geq W^{FB} - \epsilon$ requires monitoring frequency $f \geq f_\epsilon$ where $f_\epsilon \to r_a$ as $\epsilon \to 0$.

This holds when:

1. Agents are less likely to violate policies when monitored

2. Violations reduce welfare

3. **Unmonitored actions yield adversarial behavior**: Specifically, for *myopic or impatient agents in single-principal relationships without external capability constraints*, unmonitored actions yield behavior approaching W^{worst}

Justification for Condition (3): This condition holds when traditional deterrence mechanisms fail:

- **Reputation fails**: For myopic or impatient agents in single-principal settings, Lemma A.10 shows reputation cannot sustain compliance at high κ
- **No intrinsic alignment**: The agent (or attacker controlling the agent) has no inherent preference for policy compliance beyond instrumental reasons
- **No external capability constraints**: The execution environment does not limit the agent's ability to cause harm (e.g., no hard-coded spending limits, no sandboxing, no API access restrictions)

When Condition (3) does NOT hold:

- **Patient agents with multi-principal reputation**: Agents valuing reputation across many principals may comply even when unmonitored by a single principal
- **Intrinsically aligned agents**: Agents with genuine preference for compliance (not merely strategic compliance)
- **Capability-constrained environments**: Sandboxed agents with external limits on damage (e.g., payment processor enforces spending caps, API rate limits prevent rapid extraction)
- **Repeated game settings**: Patient agents in long-run relationships with type uncertainty may sustain cooperation through signaling (Kreps-Wilson 1982)

The impossibility theorem applies to the first category (myopic, single-principal, unconstrained, adversarial), which represents the security-critical worst case for autonomous commerce.

Theorem A.13 (Economic Infeasibility of Human-in-Loop Reactive Mechanisms in Specific Adversarial Settings)**.** *Consider adversarial settings where agents are:*

1. ***Myopic or impatient**: Low discount factor or short time horizon*
2. ***Operating in single-principal relationships**: Reputation confined to one principal*
3. ***Not capability-constrained**: Execution environment does not limit damage potential*
4. ***Actively adversarial**: Seeking to maximize extraction from principal*

In such settings, for any $\epsilon > 0$, there exists a threshold $\kappa^* = \kappa^*(\epsilon, C_m, W^{FB})$ such that for all $\kappa > \kappa^*$, no mechanism $M \in \mathcal{M}^{HIL}_{reactive}$ can achieve expected social welfare $W \geq W^{FB} - \epsilon$ while satisfying individual rationality.

Proof. We prove economic infeasibility through the capacity wall argument from Lemma A.7.

Step 1: Near-first-best requires near-complete monitoring in adversarial settings.

By Assumption A.12, achieving $W \geq W^{FB} - \epsilon$ requires monitoring frequency $f \geq f_\epsilon$ where $f_\epsilon \to r_a$ as $\epsilon \to 0$.

In the specified adversarial settings (myopic, single-principal, unconstrained, hostile), Assumption A.12 Condition (3) applies: unmonitored actions yield adversarial behavior. Therefore:

$$W(f, r_a, T) = \left(\frac{f}{r_a}\right) W^{FB} \cdot T + \left(1 - \frac{f}{r_a}\right) W^{worst} \cdot T \tag{13}$$

To achieve $W \geq W^{FB} - \epsilon$:

$$f \geq r_a \left(1 - \frac{\epsilon}{W^{FB} - W^{worst}}\right) \equiv f_\epsilon \tag{14}$$

As $\epsilon \to 0$: $f_\epsilon \to r_a$ (must monitor everything).

Step 2: Human capacity wall makes complete monitoring economically infeasible.

Any $M \in \mathcal{M}^{HIL}_{reactive}$ satisfies Property 3 (human-in-loop) by definition. From Lemma A.7, human investigation capacity creates a finite threshold $N^* = \mu/\alpha$ beyond which single-analyst monitoring becomes infeasible.

For $n = r_a \cdot T$ actions over time T, when $r_a \cdot T > N^*$, the principal must either:

Option A: Incomplete monitoring. Monitor only fraction $\rho = N^*/(r_a \cdot T)$ of actions. This yields welfare approaching W^{worst} as $\kappa \to \infty$.

Option B: Scale analysts. Employ $k = \lceil r_a \cdot T \cdot \alpha/\mu \rceil$ analysts. Total cost:

$$C_{total} \geq C_m \cdot r_a \cdot T + (C_{hire} + C_{mgmt} + C_{coord}) \cdot \frac{r_a \cdot T \cdot \alpha}{\mu} \tag{15}$$

For large r_a (equivalently, large $\kappa = r_a/r_p$):

$$\lim_{r_a \to \infty} \frac{C_{total}}{W^{FB} \cdot T} \geq \lim_{r_a \to \infty} \frac{(C_{hire} + C_{mgmt} + C_{coord}) \cdot \alpha \cdot r_a}{\mu \cdot W^{FB}} = \infty \tag{16}$$

This violates individual rationality: cost exceeds maximum benefit.

Step 3: Construct explicit threshold κ^*.

Define κ^* as the minimum value where Option B becomes economically irrational:

$$(C_{hire} + C_{mgmt} + C_{coord}) \cdot \frac{\kappa^* \cdot r_p \cdot T \cdot \alpha}{\mu} = W^{FB} \cdot T \tag{17}$$

Solving for κ^*:

$$\kappa^* = \frac{\mu \cdot W^{FB}}{(C_{hire} + C_{mgmt} + C_{coord}) \cdot \alpha \cdot r_p} \tag{18}$$

For $\kappa > \kappa^*$, neither option achieves $W \geq W^{FB} - \epsilon$ while satisfying individual rationality.

Step 4: Coverage of all mechanisms in $\mathcal{M}^{HIL}_{reactive}$.

By Lemma A.4, all traditional principal-agent mechanisms $(M_{mon}, M_{inc}, M_{rep}, M_{con})$ belong to $\mathcal{M}^{HIL}_{reactive}$. The capacity wall argument applies to any mechanism in $\mathcal{M}^{HIL}_{reactive}$ because, by definition, every such mechanism requires human judgment for investigation or enforcement (Property 3), which creates the finite throughput limit. □

Remark A.14 (Scope and Limitations of the Impossibility Result). Theorem A.13 establishes economic infeasibility for a *specific but important* class of scenarios. Its strength and limitations must be understood precisely.

What the theorem proves:

- In the worst-case adversarial setting (myopic, single-principal, unconstrained, hostile agents)
- Mechanisms in $\mathcal{M}^{HIL}_{reactive}$ (those requiring human judgment) cannot achieve near-first-best welfare
- Because human capacity creates a finite throughput limit
- That becomes economically prohibitive at high speed

What the theorem does NOT prove:

- Impossibility for patient agents with multi-principal reputation
- Impossibility for intrinsically aligned agents
- Impossibility in capability-constrained environments
- Impossibility for reactive mechanisms that lack human-in-loop (i.e., mechanisms in $\mathcal{M}_{reactive} \setminus \mathcal{M}^{HIL}_{reactive}$ satisfying only Properties 1 or 2)
- Impossibility for all non-cryptographic approaches (excludes automated deterministic systems without human judgment)

Why this scoping matters: The myopic, single-principal, unconstrained adversarial setting represents:

- **Security-critical scenarios**: Compromised agents controlled by attackers
- **Worst-case design**: Systems must be secure even in this case
- **Regulatory relevance**: Financial systems cannot assume benign agents
- **Real deployment risks**: Prompt injection, model drift, external exploitation are realistic threats

The logical chain made explicit:

1. For myopic single-principal agents, Lemma A.10 shows reputation fails at high κ

2. With no reputation, intrinsic alignment, or capability constraints, Assumption A.12(3) applies: unmonitored actions yield worst-case behavior

3. This requires near-complete monitoring for near-first-best welfare

4. Human capacity wall (Lemma A.7) makes complete monitoring economically infeasible for any mechanism requiring human judgment (Property 3)

5. Therefore, impossibility for $\mathcal{M}_{reactive}^{HIL}$ in this specific setting

This chain is valid for the stated setting and mechanism class but does not generalize beyond them. The impossibility is conditional on a specific agent model and the presence of human-in-loop requirements.

Relation to autonomous commerce: Why focus on this specific adversarial model?

- Autonomous commerce at scale will face adversarial attacks (this is empirically certain)
- Security design requires worst-case guarantees, not average-case
- A single successful attack can cause catastrophic damage at machine speed
- Therefore, systems must be secure even against myopic, single-principal, unconstrained adversaries

The impossibility theorem establishes that mechanisms in $\mathcal{M}_{reactive}^{HIL}$ fail in precisely the scenarios where security guarantees are most critical. This motivates cryptographic verification not as the only possible solution, but as a solution that provides security guarantees in worst-case adversarial settings where human-in-loop mechanisms provably fail.

Remark A.15 (Alternative Deterrents and Why They Don't Resolve the Impossibility). One might ask: beyond reputation, could other deterrents sustain compliance without monitoring?

Intrinsic alignment: If agents genuinely prefer policy compliance, Assumption A.12(3) doesn't hold and the impossibility theorem doesn't apply. However:

- Intrinsic alignment cannot be verified without monitoring
- Alignment can be corrupted (prompt injection, model drift, adversarial inputs)
- Security-critical systems cannot assume alignment holds

Multi-principal reputation: If agents value reputation across many principals, V^{future} could remain large even at high κ. This is a genuine limitation of Lemma A.10 and the impossibility theorem.

For multi-principal reputation to provide security:

- Reputation must be tied to verifiable identity (otherwise attackers create new identities)
- All principals must coordinate on reputation signals (otherwise attackers segment markets)

- Reputation must update rapidly enough to prevent damage at machine speed
- Principals must credibly commit to punishing reputation losses

These requirements are strong and may not hold in practice, but they represent a potential alternative to cryptographic verification worth investigating.

Capability constraints: If the execution environment limits damage potential (spending caps enforced by payment processors, API rate limits, sandboxing), then even adversarial behavior is bounded, violating Assumption A.12(3).

This is also a genuine alternative. However:

- Capability constraints require trusted enforcement (payment processor, API provider)
- They may limit legitimate use cases as well as attacks
- They don't prevent all attacks (e.g., data exfiltration, reputation damage)
- They create new single points of failure

Conclusion: The impossibility theorem applies to $\mathcal{M}^{HIL}_{reactive}$ in a specific but important adversarial model. Alternative deterrents (multi-principal reputation, capability constraints) may work in some settings but either:

1. Require strong assumptions that may not hold in adversarial settings, or
2. Move trust to other parties (payment processors, infrastructure providers), relocating rather than eliminating the principal-agent problem

Cryptographic verification provides security guarantees in the worst-case adversarial setting where these alternatives fail.

Lemma A.16 (Each Reactive Property Creates Challenges)**.** *For sufficiently large κ, each reactive property creates significant challenges for achieving near-first-best welfare in adversarial settings. Property 3 (human-in-loop) is the binding constraint that drives the impossibility result for $\mathcal{M}^{HIL}_{reactive}$.*

Proof. We establish that each property creates challenges (though not all are equally severe).

Property 1 (post-execution observation):

For mechanisms that observe behavior after execution (even with automation), violations have already occurred before detection.

With detection time τ_{detect} (even if automated to milliseconds), the agent executes $N_{undetected} = r_a \cdot \tau_{detect}$ actions before detection. In adversarial settings, damage per violation episode:

$$D_{violation} = r_a \cdot \tau_{detect} \cdot |v_{worst}| \tag{19}$$

However, τ_{detect} for automated systems can be very small (milliseconds for simple rule checks). The challenge from Property 1 alone is less severe than the human bottleneck from Property 3. Property 1 creates irreversibility (actions complete before detection) but doesn't necessarily create economic infeasibility unless combined with slow detection.

Property 2 (probabilistic inference):

Statistical detection of violations requires accumulating evidence. Classical hypothesis testing suggests $O(\log(1/\delta))$ samples for confidence $1 - \delta$.

Important caveat: Modern sequential detection algorithms (CUSUM, Shiryaev-Roberts, SPRT) can detect changes with expected delay that depends on signal-to-noise ratio rather than action rate. For violations with distinctive signatures, detection can be very fast with few false positives.

The challenge from Property 2 depends critically on:

- Signal-to-noise ratio (distinctive violations are easy to detect)
- Detection algorithm sophistication (sequential methods much better than batch)
- False positive tolerance (affects detection threshold)

Property 2 creates challenges but not necessarily impossibility, especially with modern detection methods.

Property 3 (human-in-loop):

This is the binding constraint for the impossibility theorem. Human judgment creates finite capacity μ, leading directly to the capacity wall argument in Lemma A.7.

Even if Properties 1 and 2 were somehow addressed (pre-execution observation, deterministic checking), Property 3 alone creates economic infeasibility through the capacity wall.

Therefore, Property 3 is the critical property for the impossibility result, while Properties 1 and 2 create additional challenges but are not independently sufficient for strong impossibility without additional assumptions. This is why the impossibility theorem (Theorem A.13) is stated for $\mathcal{M}_{reactive}^{HIL}$ rather than the full $\mathcal{M}_{reactive}$ class. □

Remark A.17. Lemma A.16 shows that Property 3 (human-in-loop) is the binding constraint for impossibility, while Properties 1 and 2 create challenges that may or may not be severe depending on system design. This clarifies that the core impossibility is about human bottlenecks, not about the logical impossibility of post-execution observation or probabilistic inference per se.

A.3 Verification Theorem

Definition A.18 (Cryptographic Verification Mechanism)**.** A cryptographic verification mechanism M_{zkML} operates as follows:

1. For each proposed action $a \in A$, agent generates proof π that a satisfies policy P
2. Proof generation takes time t_p and costs C_p per action
3. Principal verifies proof in time t_v (constant, independent of computation complexity)
4. Verification cost per action is $C_v \ll C_m$
5. If π valid, action proceeds; otherwise, action blocked

For real-time operation at rate r_a, require $t_p + t_v < 1/r_a$.

Lemma A.19 (Cryptographic Verification is Not in $\mathcal{M}^{HIL}_{reactive}$)**.** *Let M_{zkML} be a cryptographic verification mechanism (Definition A.18). Then $M_{zkML} \notin \mathcal{M}^{HIL}_{reactive}$ (and indeed $M_{zkML} \notin \mathcal{M}_{reactive}$) because it violates all three reactive properties:*

1. ***Pre-execution verification**: M_{zkML} verifies proofs before actions execute (violates Property 1)*
2. ***Deterministic verification**: Checks mathematical proofs with certainty (soundness error $\nu < 2^{-128}$), not probabilistically (violates Property 2)*
3. ***Automated verification**: Verifies algorithmically without human judgment, cost scales linearly $(C_p + C_v) \cdot n$ with no $H(n)$ term (violates Property 3)*

Therefore Theorem A.13 does not constrain M_{zkML}.

Assumption A.20 (zkML Security Properties)**.** The zero-knowledge proof system satisfies:

1. **Completeness**: If a satisfies policy P, an honest agent can generate valid proof π with probability 1
2. **Soundness**: If a violates policy P, no probabilistic polynomial-time (PPT) adversary can generate valid proof π except with negligible probability $\nu < 2^{-128}$
3. **Succinctness**: Proof size $|\pi| = O(1)$ and verification time $t_v = O(1)$, both independent of policy complexity
4. **Zero-knowledge**: The proof π reveals no information about the computation beyond the fact that a satisfies P

These properties are standard for zkSNARK systems [17]. Security rests on computational hardness assumptions (discrete logarithm, knowledge-of-exponent). Some constructions (STARKs) provide statistical soundness against unbounded adversaries. Many constructions require trusted setup ceremonies; transparent constructions eliminate this.

Theorem A.21 (First-Best with Cryptographic Verification)**.** *Let M_{zkML} be a cryptographic verification mechanism satisfying Definition A.18 and Assumption A.20.*

Given that:

1. *Policies P correctly capture principal intent, i.e., $A^{valid} = \{a : a \text{ satisfies } P\}$ contains only actions achieving $v(a, \theta) \geq W^{FB} - \delta$ for small $\delta > 0$*

If the following feasibility conditions hold:

2. ***Economic feasibility**: $(C_p + C_v) < \delta v$ where $\delta v = (W^{FB} - W^{worst})/r_a$*
3. ***Latency feasibility**: $t_p + t_v < 1/r_a$ (proof generation and verification complete within action interval)*

***Then** M_{zkML} achieves expected welfare:*

$$W \geq W^{FB} - \delta - (C_p + C_v) \cdot r_a - \nu \cdot (W^{FB} - W^{worst}) \tag{20}$$

for speed asymmetry ratio κ, including $\kappa > \kappa^*$ where mechanisms in $\mathcal{M}^{HIL}_{reactive}$ fail.

Proof. The proof shows cryptographic verification achieves near-first-best welfare by preventing policy violations before execution.

Step 1: Verification prevents policy violations (up to ν).

By Definition A.18, action a executes iff agent provides valid proof π that a satisfies policy P.

By soundness (Assumption A.20.2), if a violates P, probability of valid proof is at most $\nu < 2^{-128}$.

Let $A^{valid} = \{a : a \text{ satisfies } P\}$ and $A^{invalid} = A \setminus A^{valid}$. Then:

$$\Pr[\text{invalid action executes}] \leq \nu \tag{21}$$

Over time T with rate r_a, expected invalid actions: $\nu \cdot r_a \cdot T$.

Step 2: Compute expected welfare (given correct policy specification).

By the theorem's precondition (Condition 1), policies P correctly capture principal intent such that $a \in A^{valid}$ achieves $v(a, \theta) \geq W^{FB} - \delta$ for small $\delta > 0$. Actions $a \in A^{invalid}$ may achieve welfare as low as W^{worst}.

Expected welfare per unit time:

$$W_{zkML} \geq (1 - \nu) \cdot (W^{FB} - \delta) + \nu \cdot W^{worst} \tag{22}$$

$$= W^{FB} - \delta - \nu(W^{FB} - W^{worst}) + \nu\delta \tag{23}$$

For negligible $\nu < 2^{-128}$ and small δ: $W_{zkML} \approx W^{FB} - \delta$.

Step 3: Account for costs and latency.

Total cost over time T:

$$C_{total}^{verify} = (C_p + C_v) \cdot r_a \cdot T \tag{24}$$

where C_p is per-action proof generation cost and C_v is per-action verification cost.

Net welfare per unit time:

$$\frac{W_{net}}{T} = W^{FB} - \delta - (C_p + C_v) \cdot r_a - \nu \cdot (W^{FB} - W^{worst}) \tag{25}$$

Economic feasibility (Condition 2): Verification is worthwhile if welfare gain exceeds cost:

$$W^{FB} - W^{worst} - \delta > (C_p + C_v) \cdot r_a \tag{26}$$

Equivalently:

$$C_p + C_v < \frac{W^{FB} - W^{worst} - \delta}{r_a} \approx \delta v \tag{27}$$

where $\delta v = (W^{FB} - W^{worst})/r_a$ is the per-action welfare gain.

Latency feasibility (Condition 3): For real-time operation without queueing, proof generation and verification must complete within the action interval:

$$t_p + t_v < \frac{1}{r_a} \tag{28}$$

If this fails, actions queue pending verification. With arrival rate r_a and service rate $\mu_v = 1/(t_p + t_v)$, queue stability requires $r_a < \mu_v$. When $r_a \geq \mu_v$, queue length grows unboundedly, eventually causing either:

- Throughput degradation (effective rate drops below r_a)
- Additional queueing costs from buffering and latency
- System failure when queue exhausts resources

The theorem assumes Condition 3 holds, avoiding these queueing complications.
Step 4: zkML succeeds where mechanisms in $\mathcal{M}^{HIL}_{reactive}$ fail.
From Theorem A.13, mechanisms in $\mathcal{M}^{HIL}_{reactive}$ fail when $\kappa > \kappa^*$ because:

$$C^{monitor}_{total}(r_a \cdot T) = C_m \cdot r_a \cdot T + (\text{analyst scaling costs}) \tag{29}$$

grows faster than $W^{FB} \cdot T$ or achieves insufficient monitoring coverage.
Cryptographic verification has linear cost:

$$C^{verify}_{total}(r_a \cdot T) = (C_p + C_v) \cdot r_a \cdot T \tag{30}$$

with no analyst scaling required. Cost per action $(C_p + C_v)$ is constant, independent of κ.

Key differences (from Lemma A.19):

1. **Preventative vs. reactive**: Blocks invalid actions before execution (zero violations up to ν) vs. detects after (requires investigation time during which $r_a \cdot \tau_{detect}$ violations occur)
2. **Deterministic vs. probabilistic**: Mathematical proof with soundness $\nu < 2^{-128}$ (certainty) vs. statistical inference requiring $O(\log(1/\epsilon))$ samples
3. **Linear vs. capacity-bounded cost**: Scales linearly without human bottleneck vs. hits capacity wall requiring analyst scaling
4. **Independence from κ**: Costs $(C_p + C_v)$ are protocol constants vs. grow with $r_a = \kappa \cdot r_p$ through analyst scaling

Therefore, for negligible ν, small δ, and feasibility conditions satisfied:

$$W = W^{FB} - \delta - (C_p + C_v) \cdot r_a - \nu \cdot (W^{FB} - W^{worst}) \geq W^{FB} - \epsilon \tag{31}$$

for arbitrarily small $\epsilon > 0$ (by choosing small enough δ and satisfying economic feasibility), including for $\kappa > \kappa^*$ where Theorem A.13 proves mechanisms in $\mathcal{M}^{HIL}_{reactive}$ cannot achieve this. □

Remark A.22 (Current Technology Status and Latency Feasibility). With JOLT Atlas achieving ≈ 10,000× proving overhead and GPU acceleration providing 10-100× speedup, for a 10ms inference:

- Proof generation time: $t_p \approx 1 - 10$ seconds
- Verification time: $t_v \approx 100$ ms
- Latency feasibility requires: $r_a < 1/(1 - 10\text{s}) \approx 0.1 - 1$ actions/second

This makes the theorem applicable to:

- Moderate-speed commerce (1-100 tx/sec with parallel proving and pipelining)
- Batch verification (proofs generated offline, verified in aggregate)
- High-value, low-frequency transactions where latency is acceptable

For machine-speed commerce (1,000+ tx/sec) with sequential proof generation, latency feasibility requires $t_p < 1$ ms, demanding another 1,000-10,000× improvement. However, with parallel proof generation across multiple cores and continuous optimization, real-time verification may become practical within 1-2 years.

Extension to queueing regime: When latency feasibility fails ($t_p + t_v > 1/r_a$) but $r_a < \mu_v = 1/(t_p + t_v)$, the system operates in a queueing regime. Expected queue length is:

$$\mathbb{E}[Q] = \frac{\rho}{1-\rho} \text{ where } \rho = r_a \cdot (t_p + t_v) < 1 \tag{32}$$

This introduces latency costs $C_{latency} \cdot \mathbb{E}[Q]$ and welfare discounting from delayed actions. The welfare bound becomes:

$$W \geq W^{FB} - \delta - (C_p + C_v) \cdot r_a - C_{latency} \cdot \frac{\rho}{1-\rho} - \nu \cdot (W^{FB} - W^{worst}) \tag{33}$$

Verification remains feasible provided the additional queueing terms don't exceed the welfare gain. This extension is left as future work.

Remark A.23 (Policy Specification Conditionality). The theorem explicitly conditions on correct policy specification (Condition 1). This is a critical assumption: even perfect verification cannot prevent welfare loss if policies incompletely capture principal intent. The micro-transaction attack example (Section 1.2) illustrates this: a policy "¡$100 per transaction" is verifiable but doesn't prevent $10,000 split across many transactions. Bridging the gap between formal policies and true principal objectives remains an open challenge discussed in Section 7.2.

Remark A.24. The verification theorem establishes that cryptographic verification lacks the reactive properties that cause impossibility. By Lemma A.19, $M_{zkML} \notin \mathcal{M}^{HIL}_{reactive}$, hence Theorem A.13 does not constrain it. This demonstrates that expanding the mechanism space with cryptographic tools overcomes fundamental barriers in traditional principal-agent theory.

B Primers on Key Technologies

This appendix provides detailed introductions to Automated Reasoning and Zero-Knowledge Proofs for readers less familiar with these technologies.

B.1 Automated Reasoning

This approach represents a fundamentally different paradigm from the probabilistic and cognitive models that dominate modern AI. While contemporary machine learning systems, including large language models, operate through statistical pattern matching and neural approximations, Automated Reasoning descends from classical symbolic logic and formal methods [32, 37].

The distinction is profound. Cognitive approaches to AI, inspired by human reasoning, embrace uncertainty and learn from examples. Bayesian systems explicitly model probability distributions and update beliefs incrementally [34]. These probabilistic methods excel at handling ambiguity but cannot provide absolute guarantees: a classification might be 99.9% confident but never certain.

Automated reasoning, in contrast, inherits the mathematical rigor of formal logic dating back to Frege, Russell, and Whitehead's work on the foundations of mathematics [41]. Modern automated reasoning systems use techniques from theorem proving, satisfiability solving (SAT/SMT), and formal verification [6,28]. These methods can prove—not estimate, but mathematically prove—that a statement holds or find a counterexample demonstrating it fails.

The process works as follows. Natural language policies (e.g., "no transaction over $100") are first translated into formal logic statements using first-order logic, temporal logic, or similar formal systems. These logical statements then become mathematical constraints that can be checked algorithmically through constraint solvers and theorem provers. When an AI generates an output, the system constructs a formal proof demonstrating whether it satisfies the constraints, or proves it violates them.

For example, a policy "approve loans only for credit scores ≥ 680" becomes a formal constraint: $\forall x \in \text{LoanApplications}, \text{approve}(x) \implies \text{creditScore}(x) \geq 680$. The automated reasoning system can then mathematically verify this holds for each decision, with the same certainty that $2 + 2 = 4$.

B.2 Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) emerged from theoretical cryptography in the 1980s as an almost paradoxical concept: how can you prove you know something without revealing any information about what you know? Goldwasser, Micali, and Rackoff's seminal 1985 paper [16] introduced this idea through interactive protocols where a prover and verifier engage in a carefully choreographed exchange.

A classic analogy illustrates the concept: imagine you want to prove you know the password to a system without revealing the password itself. A zero-knowledge proof accomplishes this through mathematical protocols where, if you actually know the password, you can generate a valid proof that convinces the verifier, but if you don't know it, you cannot generate a convincing proof except with negligible probability. Most remarkably, the proof reveals nothing about the password itself; the verifier learns only that you know it, nothing more.

These properties seemed purely theoretical until the 1990s and 2000s, when researchers developed practical implementations. The breakthrough came with zkSNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) [5, 17], which added suc-

cinctness (proofs are small, typically under 1KB, and verify in constant time regardless of computation complexity, meaning a computation taking hours can be verified in milliseconds) and non-interactivity (eliminating the back-and-forth between prover and verifier so a proof can be generated once and verified by anyone, anytime, anywhere).

The Blockchain Scaling Problem The practical development of zkSNARKs was driven primarily by a specific challenge in blockchain systems. Public blockchains like Ethereum face a fundamental trilemma: they must be decentralized (no central authority), secure (resistant to attacks), and scalable (high transaction throughput). But the verification model that ensures security creates a scaling bottleneck.

In traditional blockchains, every node must re-execute every transaction to verify the blockchain's state. This redundant computation limits throughput: Ethereum processes roughly 15–30 transactions per second compared to Visa's 65,000+ transactions per second. The constraint isn't computational power; it's that verification happens in highly constrained environments. Ethereum's EVM operates under strict gas limits, nodes have varying computational capabilities, and the blockchain state must fit in memory across thousands of diverse machines.

Zero-knowledge proofs offered an elegant solution: move the expensive computation off-chain, then provide a succinct proof that can be verified cheaply on-chain [4]. This paradigm, often called validity rollups or zkRollups, became the dominant scaling strategy for Ethereum. Systems like zkSync, StarkNet, and Polygon zkEVM now process thousands of transactions off-chain, bundle them together, and submit a single compact proof to the main chain [9].

The economic breakthrough is transforming verification from $O(n \times c)$ to $O(n \times v)$ where n is the number of verifiers, c is the computation cost, and v is the constant-time verification cost with $v \ll c$. Instead of every node re-executing expensive computations, they verify a small proof in milliseconds. Real-time proving of blockchain blocks was considered impossible just a few short years back, however some provers now claim 7 seconds [40] or less prover times. A similar performance improvement trajectory has held true for zkML. This progress is what makes blockchain scaling viable, and the same properties make trustless agent commerce viable.

References

[1] George A Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. *Quarterly journal of economics*, 84(3):488–500, 1970.

[2] Amazon Web Services. Minimize ai hallucinations and deliver up to 99% verification accuracy with automated reasoning checks. `https://aws.amazon.com/blogs/aws/`, 2024.

[3] Arasu Arun, Srinath Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. Cryptology ePrint Archive, Paper 2023/1217, 2023.

[4] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, 2018.

[5] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium*, pages 781–796, 2014.

[6] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. *Handbook of satisfiability*, volume 185. IOS press, 2008.

[7] Frederick P Brooks. *The mythical man-month: Essays on software engineering*. Addison-Wesley, anniversary edition, 1995.

[8] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. `https://ethereum.org/en/whitepaper/`, 2014.

[9] Vitalik Buterin. An incomplete guide to rollups. `https://vitalik.ca/general/2021/01/05/rollup.html`, 2021.

[10] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, et al. Are aligned neural networks adversarially aligned? *arXiv preprint arXiv:2306.15447*, 2023.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[12] Ronald H Coase. The nature of the firm. *economica*, 4(16):386–405, 1937.

[13] Chrysanthos Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Management science*, 49(10):1407–1424, 2003.

[14] EZKL Team. Ezkl: Easy zero-knowledge machine learning. `https://ezkl.xyz`, 2024. General-purpose zkML library for neural network verification.

[15] Oded Goldreich. *Secure multi-party computation*. Manuscript. Preliminary version, 1998.

[16] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[17] Jens Groth. On the size of pairing-based non-interactive arguments. *Advances in Cryptology–EUROCRYPT 2016*, pages 305–326, 2016.

[18] Yanpei Guo, Zhanpeng Guo, Wenjie Qu, and Jiaheng Zhang. Architecture-private zero-knowledge proof of neural networks. Cryptology ePrint Archive, Paper 2025/2211, 2025.

[19] Oliver D Hart and John Moore. Incomplete contracts and renegotiation. *Econometrica: Journal of the Econometric Society*, pages 755–785, 1988.

[20] Bengt Holmström. Moral hazard and observability. *The Bell journal of economics*, pages 74–91, 1979.

[21] Bengt Holmström and Paul Milgrom. Aggregation and linearity in the provision of intertemporal incentives. *Econometrica*, pages 303–328, 1987.

[22] Bengt Holmström and Paul Milgrom. Multitask principal-agent analyses: Incentive contracts, asset ownership, and job design. *Journal of Law, Economics, & Organization*, 7:24–52, 1991.

[23] ICME Labs. Jolt atlas: Zero-knowledge virtual machine optimized for machine learning inference. `https://github.com/ICME-Lab/jolt-atlas`, 2025. Open-source zkML framework using lookup-based proof generation.

[24] Michael C Jensen and William H Meckling. Theory of the firm: Managerial behavior, agency costs and ownership structure. *Journal of financial economics*, 3(4):305–360, 1976.

[25] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up trustless dnn inference with zero-knowledge proofs. *arXiv preprint arXiv:2210.08674*, 2022.

[26] Darya Kaviani and Srinath Setty. Vega: Low-latency zero-knowledge proofs over existing credentials. Cryptology ePrint Archive, Paper 2025/2094, 2025.

[27] Leonard Kleinrock. *Queueing systems, volume 1: Theory*. Wiley-Interscience, 1975.

[28] Daniel Kroening and Ofer Strichman. *Decision procedures: an algorithmic point of view*. Springer, 2nd edition, 2016.

[29] Jean-Jacques Laffont and David Martimort. *The theory of incentives: the principal-agent model*. Princeton university press, 2009.

[30] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.

[31] Roger B Myerson. Incentive compatibility and the bargaining problem. *Econometrica*, 47(1):61–73, 1979.

[32] Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[33] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.

[34] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[35] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

[36] Joshua S Rubinstein, David E Meyer, and Jeffrey E Evans. Executive control of cognitive processes in task switching. *Journal of experimental psychology: Human perception and performance*, 27(4):763–797, 2001.

[37] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[38] SANS Institute. The state of security operations: 2021 survey results. *SANS Security Operations Center Survey*, 2021. Reports average SOC analyst capacity of 100-1,000 alerts per day.

[39] George J Stigler. The economics of information. *Journal of Political Economy*, 69(3):213–225, 1961.

[40] Succinct. SP1 hypercube achieves real time proving with 16 GPUs. `https://blog.succinct.xyz/real-time-proving-16-gpus/`, 2025. Demonstrates real-time ZK proving of 99.7% of L1 Ethereum blocks under 12s on 16 NVIDIA RTX 5090 GPUs.

[41] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume 1. Cambridge University Press, 1910.

[42] Oliver E Williamson. *The economic institutions of capitalism*. Simon and Schuster, 1985.

[43] Andrew C Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 160–164. IEEE, 1982.