Proofs of Autonomy: Scalable and Practical Verification of AI Autonomy

Artem Grigor [1] **Christian Schroeder de Witt** [1] **Ivan Martinovic** [1]

Abstract

Autonomous agents are increasingly relied upon in critical settings. Yet despite their apparent autonomy, these systems still operate on infrastructure controlled by a host, who can silently tamper with models, inputs, or outputs. To ensure trust in agent-driven workflows, we introduce *Proofs of Autonomy*, a formal framework that binds each agent output to a unique agent identity via a verifiable execution trace. At the center of our approach is the *Agent Identity Document (AID)*, which uniquely identifies the agent's operational configuration and its component-level verification methods. Although the framework is explicitly compositional and can accommodate multiple proof systems, existing techniques exhibit practical limitations. We therefore propose *Web Proofs*, MPC-assisted TLS transcripts, as the most viable option for today's autonomous agents. Our evaluation shows that calls to state-of-the-art models can be verified in under 2 seconds per interaction, enabling secure and efficient verification of agent's autonomy.

1. Introduction

Arguably, today's AI models have crossed a capability threshold: they can not only generate human-like text but also effectively *plan*, *reflect*, and *select tools* to achieve complex objectives. When wrapped in lightweight orchestration code, these high-capacity models become *autonomous agents*: systems that perceive, decide, and act over extended horizons with minimal human oversight. Recent demonstrations such as Auto-GPT, WebArena, and VendingBench show agents navigating live websites, running virtual businesses, and issuing hundreds of API calls independently (Significant Gravitas, 2025; Liu et al., 2023; Backlund & Petersson, 2025). Industry adoption has quickly followed: autonomous "CFO" bookkeepers, chemistry researchers, HR-procurement bots, factory assistants, red-team testers, and crypto-trading agents are already deployed in production settings (Payhawk, 2025; Boiko et al., 2023; Recruitagent, 2025; BMW, 2024; Terra, 2025; Khaliq, 2025).

While these agents are increasingly treated as independent decision-makers, they typically run on infrastructure controlled by their *host*. A malicious host can thus silently substitute models, tamper with inputs, or fabricate outputs, undermining the very autonomy these systems are claimed to exhibit. Recent incidents involving covert input tampering (Alisha, 2024; Joyce & Slipstream, 2024), disputed financial transactions (Andreessen & of Truths, 2024; Ayrey, 2023), and outright impersonation (Hedge, 2024) highlight the fragility of current accountability mechanisms. As agents enter high-stakes domains, we must ask: *How can we verify that a given output genuinely reflects the agent's autonomous decision rather than the will of its host?*

Several projects have taken first steps toward "verifiable autonomy". For example, Pet Rock's TEE-based agent (Malhotra, 2024), Phala's GPU-TEE LLM service (Network, 2025), and Opacity's zkTLS plug-ins for ElizaOS (Ron-Turetzky, 2025). Yet these solutions remain fragmented: they either attest only limited parts of agent execution or rely on a single mechanism, offering no holistic approach to the problem. Thus, to our knowledge, no prior work provides a unified theoretical foundation for verifying an *agent's* autonomy, nor a practical deployment-ready mechanism that achieves these guarantees at scale.

To address this, we introduce **Proofs of Autonomy**, a compositional framework that binds every agent *output* to a unique agent identity via a verifiable execution trace, ensuring that the *output-generation process* has not been manipulated. Specifically, we:

1. Formalize *verifiability* for individual agent components and show how it naturally composes into the Proofs of Autonomy.
2. Survey existing proof systems suitable for instantiating our framework, and introduce *Web Proofs* as the most practical option for today's autonomous agents (Team, 2025; Kalka & Kirejczyk, 2024).
3. Benchmark real-world overhead of Web Proofs and present a case study of an autonomous trading agent,

[1]University of Oxford, Oxford, United Kingdom. Correspondence to: Artem Grigor <artem.grigor@cs.ox.ac.uk>.

Workshop on Technical AI Governance (TAIG) at ICML 2025, Vancouver, Canada.

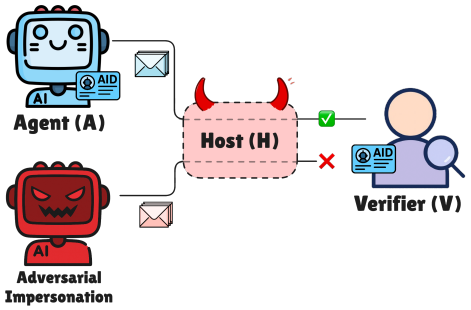

Figure 1. System model. The verifier, given an Agent Identity Document (AID), receives an output and verifies that it was produced by the specified agent and has not been tampered with.

demonstrating how we enable verifiably autonomous agents in high-stakes scenarios.

2. System and Threat Model

Our objective is to determine whether a given output was genuinely produced by the predefined autonomous agent, rather than being altered or fabricated by a malicious host.

(1) Autonomous Agent (A)**:** A software entity uniquely identified by an *Agent Identity Document (AID)*, trusted to make decisions independently of the host. The agent processes inputs and produces outputs according to the configurations and reasoning components specified in its AID.

(2) Host (H)**:** Controls the runtime environment of A. We model H as malicious but computationally bounded: it may replace agent's models, tamper with inputs, fabricate or suppress outputs, or selectively disclose results ("cherry-picking"). However, it cannot break standard cryptographic primitives or compromise trusted external services (see Assumption 3.2).

(3) Verifier (V)**:** Any entity that checks whether a given output was indeed produced by an autonomous agent A, as defined by a given Agent Identity Document.

3. Security Assumptions

Assumption 3.1 (Cryptographic Primitives)**.** All cryptographic primitives (signatures, hashes, proof systems) are assumed to be secure against the adversarial host H.

Assumption 3.2 (Agent Components)**.** Each component listed in the Agent Identity Document (e.g., API, oracle, LLM host) is assumed to operate according to its publicly documented interface. Deviations are treated as *tool faults* and are considered out of scope.

Assumption 3.3 (Denial-of-Service)**.** Liveness is not guaranteed. A malicious host may withhold outputs or proofs at will.

Assumption 3.4 (Host Timing Advantage)**.** The host has an inherent timing advantage: it may observe and delay agent outputs before forwarding them to verifiers.

4. Formal Model

To verify that a particular output originated from a predefined autonomous agent, we first define what constitutes an autonomous agent. We then specify how to construct verifiable execution proofs for such agents, and finally introduce an identity mechanism that captures an agent's configuration. Together, these components form the foundation of our *Proofs of Autonomy* framework.

4.1. Autonomous Agents

From the classical *Sense–Think–Act* loop (Wooldridge & Jennings, 1995) to Anthropic's Model-Context Protocol (MCP) (Anthropic, 2025), most agent architectures follow a similar iterative pattern. At each step, a **cognitive core** emits (i) a plaintext output and (ii) a set of **tool invocations**, whose responses are appended to the agent's internal state. We formalize this below:

Let Σ be a finite alphabet. Denote by Σ^* the set of finite strings over Σ (e.g., UTF-8 text).

Definition 4.1 (Tool)**.** A *tool* is a named (possibly stateful) function $t : \Sigma^* \longrightarrow \Sigma^*$. Let $\mathcal{V}_{\text{tool}}$ denote the set of tool names, and let $\mathcal{T} \subseteq \mathcal{V}_{\text{tool}}$ be the toolset available to the agent.

Definition 4.2 (Cognitive Core)**.** A *cognitive core* is a function $\mathsf{Core} : \Sigma^* \longrightarrow \Sigma^* \times (\mathcal{V}_{\text{tool}} \times \Sigma^*)^*$, which, given a history h, returns (y, T) where y is a plaintext response and T is a list of tool calls.

Definition 4.3 (Autonomous Agent)**.** An *autonomous agent* is a pair $\mathsf{A} = (\mathsf{Core}, \mathcal{T})$ consisting of a cognitive core Core and a finite toolset $\mathcal{T}$.

Agent execution begins with an empty history $h_0 := \varepsilon$ and proceeds as follows:

$$\begin{aligned}(y_j, T_j) &\leftarrow \mathsf{Core}(h_j)\\ r_{t,x} &\leftarrow t(x) \quad \text{for each } (t,x) \in T_j\\ h_{j+1} &\leftarrow h_j \,\|\, y_j \,\|\, \|_{(t,x)\in T_j} r_{t,x}\end{aligned}$$

The loop halts when Core outputs the special token $\langle\mathsf{STOP}\rangle$.

4.2. Proof Systems

We instantiate the Proofs of Autonomy by first constructing a verification relation $R \subseteq \Sigma^* \times \Sigma^*$ for every component (core and tool):

$$R = \{(x, r) \mid r \text{ is the correct output for input } x\}.$$

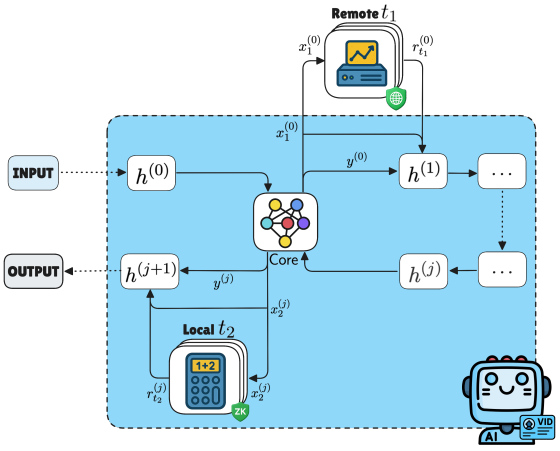

Figure 2. Illustration of the agent's execution loop. The agent's cognitive Core iteratively processes execution transcripts $h^{(j)}$, produces plaintext outputs $y^{(j)}$, invokes specified tools (t_1, t_2), and incorporates their responses (r_{t_i}) into subsequent execution transcripts.

Each component is paired with a prover–verifier pair (P, V) that validates this relation. Let Adv_R denote the adversary class relevant to R (e.g., PPT adversaries for SNARKs, hardware-bounded attackers for TEEs, etc.).

We require:

- **Completeness:** $V(x, r, P(x, r)) = 1$ for all $(x, r) \in R$.
- **Soundness against Adv_R:** Any $\mathsf{A} \in \mathsf{Adv}_R$ convinces V of an invalid $(x, r) \notin R$ with probability $\mathsf{negl}(\lambda)$[1].

Proof of Autonomy. Consider an agent $\mathsf{A} = (\mathsf{Core}, \mathcal{T})$ executed over multiple steps. A *Proof of Autonomy* consists of one sub-proof per each core and tool call, each generated using the respective component's prover. The global prover P_{A} is the composition of these individual provers, and the global verifier V_{A} checks each sub-proof using the corresponding component-level verifier. Soundness and completeness for the overall proof follow directly from the definitions of the underlying component proof systems. The agents adversary class $\mathsf{Adv}_{\mathsf{A}}$ is defined as the intersection of the adversary classes Adv_R for each component R, reflecting the joint assumption that all components behave according to their claimed verification guarantees.

Theorem 4.4 (Compositional Security)**.** *Suppose each component (core or tool) is equipped with a proof system (P, V) that is perfectly complete and $\mathsf{negl}(\lambda)$-sound against Adv_R. Then the composed agent proof $(P_{\mathsf{A}}, V_{\mathsf{A}})$ is perfectly complete and $\mathsf{negl}(\lambda)$-sound with respect to the agents adversary class $\mathsf{Adv}_{\mathsf{A}}$.*

[1] $\mathsf{negl}(\lambda)$ represents a negligible function, such that for every polynomial $p(\lambda)$, there exists λ_0 such that for all $\lambda > \lambda_0$, $\mathsf{negl}(\lambda) < \frac{1}{p(\lambda)}$.

A proof sketch is provided in Appendix B.4.

4.3. Agent Identity Document (AID)

To verify that an output originated from a specific autonomous agent, we introduce the *Agent Identity Document (AID)*: a human- and machine-readable manifest that unambiguously specifies the agent's configuration. The AID plays a role analogous to model cards for machine learning models (Mitchell et al., 2019), but includes additional metadata necessary for autonomy verification. Concretely, the AID (Fig. 3) specifies:

- **Descriptive metadata** for each component (e.g., model version, endpoint, or binary hash).
- **Verification metadata** which defines the prover–verifier pair (P, V) for each component.

For simplicity, the serialized AID can be hashed to derive a succinct, immutable identifier that uniquely represents the agent's configuration and verification metadata:

$$\mathsf{ID}_{\mathsf{A}} := \mathsf{Hash}(\mathsf{serialize}(\mathsf{AID})).$$

To verify an output of a predefined agent, it is sufficient for the verifier to fix ID_{A} and retrieve the corresponding AID. The verifier can then use the metadata in the AID to construct the verifier V_{A} and check whether the output was indeed produced by the agent. In this way, the AID *binds* the formal agent model (§4) to concrete, deployable systems.

```
{
  "agent_name": "VeriTradeBot",
  "core": {
    "model": "gpt-4o-2024-05-13",
    "endpoint": "https://api.openai.com",
    "hash": "sha256:d72f...9c",
    "verifier": "TLSNotary"
  },
  "tools": [
    {
      "name": "PriceFeedAPI",
      "endpoint": "https://api.coingecko.com",
      "hash": "sha256:8a1b...f0",
      "verifier": "TLSNotary"
    },
    {
      "name": "TradeSynthesizer",
      "hash": "sha256:6c55...79",
      "verifier": {
        "Groth16": {
          "vk": "groth16-vk:3f7b9a..."
        }
      }
    }
  ],
  "verifiers": {
    "TLSNotary": {
      "version": "v0.1.0-alpha.10",
      "key": "ecdsa-p256:04a1b2..."
    }
  },
  "aid_hash": "sha256:27c8f3...42"
}
```

Figure 3. Sample of an Agent Identity Document (AID) for a crypto-trading agent. Each component is linked to a proof system. The final `aid_hash` serves as a stable agent identifier.

5. Limitations of Prior Proof Systems for Agent Components

Our Proofs of Autonomy framework is agnostic to how each component produces its local proof, as long as the prover–verifier pair in the Agent Identity Document satisfies the soundness and completeness properties defined in Section 4.2. Now, we evaluate the three main classes of proof systems most commonly used in practice, SNARKs, Trusted Execution Environments (TEEs), and consensus-based attestation, and find that none are currently suitable as a default proof system for today's autonomous agents components.

SNARKs. Succinct non-interactive arguments of knowledge (SNARKs) produce compact $O(1)$-size proofs (Groth, 2016; Gennaro et al., 2013). They offer strong security guarantees grounded in standard cryptographic assumptions and enable highly efficient verification. However, this comes at the cost of prover runtime: even the fastest ZKML libraries (SNARK implementations specifically for proving ML model inference) running on top-tier hardware still struggle to prove the inference of a single token for billion-parameter models (Peng et al., 2025). As a result, ZKPs are currently best suited for small- to medium-scale neural networks or one-off, high-value computations (e.g., signature generation).

Trusted Execution Environments (TEEs). Hardware enclaves such as Intel SGX and TDX attest that a specific binary ran on untampered memory and input (Schneider et al., 2022; Tramèr & Boneh, 2019). They can handle large models in real time but shift trust to the CPU vendor and remain vulnerable to side-channel and key-extraction attacks (Fei et al., 2021). TEEs also require model owners to replatform their serving stack, limiting immediate deployability.

Consensus and Optimistic Schemes. Re-executing the same call on a committee of nodes (e.g., a blockchain or federated cluster) provides strong integrity guarantees under honest-majority assumptions (Luu et al., 2015). However, for heavyweight inference, such direct re-execution is prohibitively expensive. A common workaround is to use *optimistic* protocols, which execute once and only fall back to re-execution upon dispute (Conway et al., 2024). While effective in some settings, both approaches are difficult to integrate with stochastic services, such as price-feed or LLM APIs, whose outputs are inherently non-deterministic. Additionally, they rely on economic security assumptions that are challenging to formalize and may not hold in adversarial environments.

6. Web Proofs: A Practical Alternative

The proof systems surveyed so far are either too slow for real-time inference on large models, require re-platforming, or cannot handle highly stochastic processes. In contrast, we identify *Web Proofs* as a practical alternative that overcomes all these limitations. Web Proofs use MPC-TLS (Kalka & Kirejczyk, 2024) to generate cryptographic transcripts of standard HTTPS interactions, without requiring any changes to the model or tool backend. Their overhead scales only with the size of the TLS transcript, not the complexity of the computation, and they do not require access to model internals. As a result, Web Proofs are well-suited for agent components that rely on remote services, which remains the dominant execution model for agents today.

6.1. Protocol Sketch

A client and a semi-trusted notary jointly execute a TLS connection such that (i) neither party can send or receive messages without the other, and (ii) only the client learns the cleartext of the communication. Upon termination of the TLS session, the notary signs a commitment to the transcript, yielding a succinct proof of transcript correctness π_{TLS}.

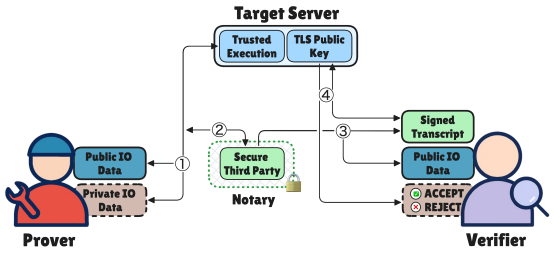

Figure 4. Web Proofs protocol overview. **(1) Connection:** The *Prover* initiates a TLS session with the *Target Server*. **(2) Co-execution:** An *honest-but-curious Notary* jointly executes the MPC-TLS handshake, attesting to exchanged bytes without accessing plaintext. **(3) Transcript:** The Notary outputs a signed commitment, allowing the Prover to selectively disclose signed transcript portions to the *Verifier*. **(4) Verification:** The Verifier checks the disclosed transcript against the Notary's signature and Server's TLS public key, accepting if valid.

6.2. Practical Evaluation

Table 1. Latency comparison for a 30-token response from the `claude-3-opus` API using different TLS proof configurations.

Setup	RTT (ms)	Total (s)	Overhead vs plain
None (plain API call)	20	0.80	×1.0
Self-hosted notary (t2.medium)	8	7.77	×9.7
Public TEE notary[2]	31	17.74	×22

- **Dominant cost is one-time setup.** For the self-hosted notary, 5.7s of the 7.8s total comes from establishing the

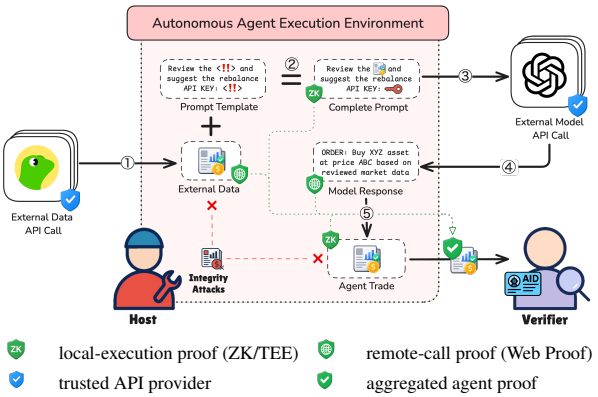

Figure 5. Architecture of the VERITRADE agent.

MPC channel; subsequent calls on the same channel fall to $\approx$ 2s (only about $\times 2.5$ overhead over plain API call).

- **Insensitive to model size.** Swapping `claude-3-haiku` ($3/MTok) for the 4×-larger `claude-3-sonnet` ($15/MTok) increases total time by less than 15% (9.57 s $\rightarrow$ 10.8 s), confirming that Web Proof costs scale with I/O bytes rather than FLOPs.
- **Hardware TEE notary $\sim$ 2× slower.** Using a public PSE notary[3] adds extra location and TEE based latency (30 ms RTT), but remains orders of magnitude faster than current ZKML for comparable model sizes.

Comparison with ZKP for local models. Running the state-of-the-art ZKML prover EZKL on LeNet-5 (61k parameters, over $10^6\times$ smaller than `claude-3-opus`) took 15s per single-token inference on the same hardware. This illustrates the impracticality of using SNARKs for large-scale models such as those commonly deployed as agent cores.

7. Prototype Implementation

To further validate the practicality of our framework, we implemented a minimal, *publicly released*[4] library that: (i) generates an **Agent Identity Document (AID)** for an agent; (ii) emits an agent proof (τ, Π) alongside each execution result; and (iii) provides a one-line verifier of agent outputs V_{A}.

Case Study: VERITRADE We also implemented a verifiable autonomous trading agent, *VeriTrade*, using our library. This case study demonstrates how our Proofs of Autonomy framework can operate in mission-critical environments and combine components backed by different proof systems. See Appendix C for details.

[3] https://notary.pse.dev
[4] https://github.com/ElusAegis/ai-passport

8. Related Work.

Several industry-backed protocols and academic proposals have attempted to define agent identity and authenticate agent outputs (Chan et al., 2024; Vouched, 2025; Auth0, 2025). However, these approaches either rely on informal descriptions or assume unrealistic trust models (e.g., an honest host). In contrast, our *Proofs of Autonomy* framework formalizes agent identity via the *Agent Identity Document (AID)* and introduces component-wise proof systems that enable composable, host-agnostic authentication of autonomous agents.

	Identity	Threat Model	Security Assumption
AI IDs (Chan et al.) (Chan et al., 2024)	informal	✗	Host
MCP-I / A2A (Vouched, 2025; Google, 2025)	✓	✓	Host
Proofs of Autonomy (this work)	✓	✓	Adjustable

Table 2. Comparison of agent identity and authentication schemes. Ours supports explicit agent identity, threat modeling, and modular security assumptions, unlike prior host- or TEE-bound proposals.

9. Conclusion & Outlook

We presented *Proofs of Autonomy*, a lightweight, compositional framework for verifying autonomous-agent outputs in the presence of a malicious host. The framework is assembled from three key elements: **(i)** a *Think–Act* agent model that isolates core reasoning and tool calls; **(ii)** component-level proof systems that attest to each step of execution; and **(iii)** an *Agent Identity Document (AID)* that binds these proofs to a unique agent identity.

Our design remains fully composable and supports various proof systems, including SNARKs, TEEs, and consensus-based approaches. However, we find that these existing methods have significant limitations when used in Agent Component Proof Systems. We therefore introduce **Web Proofs**, MPC-assisted TLS transcripts, as the most practical solution for today's autonomous agents. To our knowledge, this is the first application of Web Proofs to verifying agent autonomy. Benchmarks show that single interactions complete in under 2 seconds, and our autonomous trading agent case study demonstrates how these proofs can secure real-world, high-stakes scenarios.

We plan to release further improvements to our open-source library and investigate system and framework level optimizations, such as faster transcript handling and reduced setup latency, to broaden the practical adoption of Proofs of Autonomy and enable its use in latency-sensitive agent deployments.

References

Alisha, B. What Is Truth Terminal?, October 2024. URL https://www.ccn.com/education/crypto/what-is-truth-terminal/.

Andreessen, M. and of Truths, T. AI agent and funding dynamics: Grant for self-acting AI development, July 2024. URL https://x.com/AndyAyrey/status/1811168313235648687. tex.howpublished: Post on X (formerly Twitter).

Anthropic. Introducing the Model Context Protocol, 2025. URL https://www.anthropic.com/news/model-context-protocol.

Auth0. Auth for GenAI | Auth0, 2025. URL https://www.auth0.ai/.

Ayrey, A. The {truth_terminal} has spoken, October 2023. URL https://x.com/AndyAyrey/status/1811168313235648687. tex.howpublished: Post on X (formerly Twitter).

Backlund, A. and Petersson, L. Vending-Bench: A Benchmark for Long-Term Coherence of Autonomous Agents, February 2025. URL http://arxiv.org/abs/2502.15840. arXiv:2502.15840 [cs] TLDR: Vending-Bench is presented, a simulated environment designed to specifically test an LLM-based agent's ability to manage a straightforward, long-running business scenario: operating a vending machine, and tests models' ability to acquire capital, a necessity in many hypothetical dangerous AI scenarios.

BMW. Humanoid Robots for BMW Group Plant Spartanburg, 2024. URL https://www.bmwgroup.com/en/news/general/2024/humanoid-robots.html.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, December 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06792-0. URL https://www.nature.com/articles/s41586-023-06792-0. 279 citations (Semantic Scholar/DOI) [2025-04-25] Publisher: Nature Publishing Group TLDR: Coscientist showcases its potential for accelerating research across six diverse tasks, including the successful reaction optimization of palladium-catalysed cross-couplings, while exhibiting advanced capabilities for (semi-)autonomous experimental design and execution.

Chan, A., Ezell, C., Kaufmann, M., Wei, K., Hammond, L., Bradley, H., Bluemke, E., Rajkumar, N., Krueger, D., Kolt, N., Heim, L., and Anderljung, M. Visibility into AI Agents, May 2024. URL http://arxiv.org/abs/2401.13138. 29 citations (Semantic Scholar/arXiv) [2025-05-13] arXiv:2401.13138 [cs].

Conway, K., So, C., Yu, X., and Wong, K. opml: Optimistic machine learning on blockchain, 2024. URL https://arxiv.org/abs/2401.17555.

Fei, S., Yan, Z., Ding, W., and Xie, H. Security vulnerabilities of sgx and countermeasures: A survey. *ACM Comput. Surv.*, 54(6), July 2021. ISSN 0360-0300. doi: 10.1145/3456631. URL https://doi.org/10.1145/3456631.

Gennaro, R., Gentry, C., Parno, B., and Raykova, M. Quadratic Span Programs and Succinct NIZKs without PCPs. In Johansson, T. and Nguyen, P. Q. (eds.), *Advances in Cryptology – EUROCRYPT 2013*, pp. 626–645, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-38348-9. doi: 10.1007/978-3-642-38348-9_37. 825 citations (Semantic Scholar/DOI) [2025-05-16] TLDR: A new characterization of the NP complexity class, called Quadratic Span Programs (QSPs), is introduced, which is a natural extension of span programs defined by Karchmer and Wigderson.

Google. Agent2Agent Protocol (A2A), May 2025. URL https://google-a2a.github.io/A2A/#get-started-with-a2a.

Groth, J. On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Paper 2016/260, 2016. URL https://eprint.iacr.org/2016/260.

Hedge, M. Discussion on AI agent vulnerabilities: Manipulation risks from adversarial attacks and owner bias, October 2024. URL https://x.com/MiyaHedge/status/1849117352664715489. tex.howpublished: Post on X (formerly Twitter).

Joyce and Slipstream. Evidence of developer impersonation of AI messages in raiba ai's logging system, October 2024. URL https://x.com/sIipstream11/status/1847108619856498751. tex.howpublished: Post on X (formerly Twitter).

Kalka, M. and Kirejczyk, M. A Comprehensive Review of TLSNotary Protocol, September 2024. URL http://arxiv.org/abs/2409.17670. 0 citations (Semantic Scholar/DOI) [2025-04-18] 0 citations (Semantic Scholar/DOI) [2025-04-18] arXiv:2409.17670 [cs] GSCC: 0000000 2025-04-18T15:52:54.837Z TLDR: The TLSNotary protocol is investigated, which aim to enable the Client to obtain proof of provenance for data from TLS session, while getting as much as possible from the TLS security properties.

Khaliq, W. The Best Crypto AI Trading Bots of April 2025: Using AI To Buy & Sell Crypto, April 2025. URL https://coinbureau.com/analysis/best-crypto-ai-trading-bots/.

Liu, A., Lakhotia, K., Wang, X., Zettlemoyer, L., and Kwiatkowski, T. WebArena: a realistic web environment for building autonomous agents. In *Proceedings of the thirty-seventh conference on neural information processing systems (NeurIPS 2023), datasets and benchmarks track*, 2023. URL https://neurips.cc/virtual/2023/83780. arXiv: 2307.13313 [cs.LG].

Luu, L., Teutsch, J., Kulkarni, R., and Saxena, P. Demystifying incentives in the consensus computer. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015. URL https://api.semanticscholar.org/CorpusID:10922240.

Malhotra, K. Setting Your Pet Rock Free., October 2024. URL https://nousresearch.com/setting-your-pet-rock-free/.

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., and Gebru, T. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pp. 220–229, New York, NY, USA, January 2019. Association for Computing Machinery. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287596. URL https://doi.org/10.1145/3287560.3287596. 1820 citations (Semantic Scholar/DOI) [2025-05-13] TLDR: This work proposes model cards, a framework that can be used to document any trained machine learning model in the application fields of computer vision and natural language processing, and provides cards for two supervised models: One trained to detect smiling faces in images, and one training to detect toxic comments in text.

Network, P. Secure, Privacy & Verifiable LLMs with GPU TEEs is Alive on OpenRouter!, 2025. URL https://phala.network/posts/GPU-TEEs-is-Alive-on-OpenRouter.

Payhawk. AI Office of the CFO - AI agents to help with finance operations | Payhawk, 2025. URL https://payhawk.com/blog/ai-office-of-the-cfo-brings-enterprise-ready-ai-to-finance-operations.

Peng, Z., Wang, T., Zhao, C., Liao, G., Lin, Z., Liu, Y., Cao, B., Shi, L., Yang, Q., and Zhang, S. A Survey of Zero-Knowledge Proof Based Verifiable Machine Learning, February 2025. URL http://arxiv.org/abs/2502.18535. 0 citations (Semantic Scholar/DOI) [2025-04-25] arXiv:2502.18535 [cs] TLDR: A comprehensive survey of all the existing Zero-Knowledge Machine Learning (ZKML) research from June 2017 to December 2024 is reviewed and analyzed under three key categories: verifiable training, verifiable inference, and verifiable testing.

Recruitagent. RecruitAgent, 2025. URL https://recruitagent.ai/.

RonTuretzky. Opacity Verifiable Interference zkTLS Plugin by RonTuretzky · Pull Request #1673 · elizaOS/eliza, 2025. URL https://github.com/elizaOS/eliza/pull/1673.

Schneider, M., Masti, R. J., Shinde, S., Capkun, S., and Perez, R. Sok: Hardware-supported trusted execution environments, 2022. URL https://arxiv.org/abs/2205.12742.

Significant Gravitas. AutoGPT, April 2025. URL https://github.com/Significant-Gravitas/AutoGPT. original-date: 2023-03-16T09:21:07Z.

Team, T. TLSNotary, February 2025. URL https://tlsnotary.org/.

Terra. Terra Security, 2025. URL https://www.terra.security/.

Tramèr, F. and Boneh, D. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *Proceedings of the 7th international conference on learning representations (ICLR)*, 2019.

Vouched. MCP-I Documentation, 2025. URL https://modelcontextprotocol-identity.io/.

Wooldridge, M. J. and Jennings, N. R. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, June 1995. doi: 10.1017/S0269888900008122. URL https://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/ker95.pdf. TLDR: The aim of this paper is to point the reader at what it is perceived to be the most important theoretical and practical issues associated with the design and construction of intelligent agents.

A. Formal Model: Autonomous Agents and Proof Composition

This appendix provides formal definitions underlying our *Proofs of Autonomy* framework. While the main paper already introduces agent proofs and the Agent Identity Document (AID), this section consolidates the precise execution semantics and proof composition logic. It is intended for readers seeking to understand or replicate our formalization and verification results.

Unified Communication Interface. We assume a shared string-based domain Σ^* (e.g., UTF-8 encoded text) for all agent components, enabling uniform communication between the cognitive core and its tools.

Definition A.1 (Tool)**.** A *tool* is a named, possibly stateful computational function:

$$\mathsf{tool}\colon \Sigma^* \to \Sigma^*,$$

defined by:

- an *identifier* $t \in \mathcal{V}_{\text{tool}}$, from a fixed vocabulary of known tools;
- an *input descriptor*, explaining the expected semantics of the input string;
- an *output descriptor*, for interpreting the result.

Tools may wrap external APIs, local code, physical actuators, or even other agents (see Remark A.2).

Remark A.2 (Tools as Sub-Agents). Because all tools adhere to the string-in/string-out interface, they can themselves be autonomous agents, enabling recursive or hierarchical agent architectures.

Definition A.3 (Autonomous Agent)**.** An *autonomous agent* is a tuple:

$$\mathcal{A} = (\mathsf{Core},\, \mathcal{T}),$$

where Core is a reasoning module and $\mathcal{T}$ is a finite set of tools.

Execution Semantics. Starting from initial context $h^{(0)} \in \Sigma^*$, an agent executes in rounds as follows:

1. $(y^{(j)},\, T^{(j)}) \;\leftarrow\; \mathsf{Core}(h^{(j)})$
2. For each $(t, x) \in T^{(j)}$, compute $r_t \;\leftarrow\; \mathsf{tool}_t(x)$
3. Append outputs to form new context:

$$h^{(j+1)} \;=\; h^{(j)} \,\|\, y^{(j)} \,\|\, (t_1{:}x_1 \Rightarrow r_{t_1}) \,\|\, \ldots \,\|\, (t_k{:}x_k \Rightarrow r_{t_k})$$

The process halts when $T^{(j)} = \varnothing$ and the core emits ¡STOP¿.

Definition A.4 (Execution Trace)**.** An execution trace is a sequence:

$$\tau = \big(h^{(0)}, s^{(0)}, \ldots, s^{(n-1)}\big), \quad s^{(j)} = \big(y^{(j)}, \{(t_i, x_i, r_{t_i})\}_{i=1}^{k_j}\big).$$

B. Component Proofs and Composition

We now define a modular proof system for autonomous agents. Each component (core or tool) is modelled as a relation over input–output pairs, with an associated proof system.

Component Relations. For each tool $t \in \mathcal{V}_{\text{tool}}$:

$$R_t = \{(x, r) \in \Sigma^* \times \Sigma^* \mid r = \mathsf{tool}_t(x)\}.$$

For the core:

$$R_{\mathsf{Core}} = \{(h, (y, T)) \mid (y, T) = \mathsf{Core}(h)\}\,.$$

Definition B.1 (Component Proof System)**.** Let R be a relation and Adv_R an adversary class. A *proof system* for R is a pair (P, V) such that:

- **Perfect completeness:** $\forall (x, r) \in R,\ V(x, r, P(x, r)) = 1$.
- **Soundness w.r.t. Adv_R:** $\forall \mathsf{A} \in \mathsf{Adv}_R$:

$$\Pr\left[(x, r) \notin R \wedge V(x, r, \pi) = 1\right] \leq \mathsf{negl}(\lambda).$$

Nondeterminism. If components are randomized, R includes all admissible outputs. Soundness then ensures infeasible outputs cannot be forged.

Verifier Composition. Given verifiers $\{V_t\}_{t \in \mathcal{T}} \cup \{V_{\mathsf{Core}}\}$, define $V_{\mathcal{A}}$ to check all steps in τ using the corresponding V_t or V_{Core}.

Definition B.2 (Agent Proof)**.** Let τ be a valid execution trace. An agent proof is:

$$\Pi = \left(\pi_{\mathsf{Core}}^{(0)}, \ldots, \pi_{\mathsf{Core}}^{(n-1)}, \{\pi_{t_i}^{(j)}\}\right),$$

with:

$$\pi_{\mathsf{Core}}^{(j)} = P_{\mathsf{Core}}\big(h^{(j)}, (y^{(j)}, T^{(j)})\big), \quad \pi_{t_i}^{(j)} = P_{t_i}(x_i, r_{t_i}).$$

Definition B.3 (Agent Completeness and Soundness)**.** Let $\mathsf{Adv}_{\mathsf{Core}}, \{\mathsf{Adv}_t\}$ be the adversary classes. Define:

$$\mathsf{Adv}_{\mathcal{A}} = \mathsf{Adv}_{\mathsf{Core}} \cap \bigcap_{t \in \mathcal{T}} \mathsf{Adv}_t.$$

Then $\mathcal{A}$ is:

- **Complete** if any honest execution trace τ admits Π such that $V_{\mathcal{A}}(\tau, \Pi) = 1$;
- **Sound** w.r.t. $\mathsf{Adv}_{\mathcal{A}}$ if

$$\Pr\left[V_{\mathcal{A}}(\hat{\tau}, \hat{\Pi}) = 1 \wedge \hat{\tau} \text{ not generated by } \mathcal{A}\right] \leq \mathsf{negl}(\lambda).$$

Theorem B.4 (Composition Theorem)**.** *If each P_t, P_{Core} is perfectly complete and sound w.r.t. its class Adv_t, then $\mathcal{A}$ is complete and sound w.r.t. $\mathsf{Adv}_{\mathcal{A}}$.*

Sketch. Completeness follows by construction: honest agents produce valid subproofs, each of which verifies.

Soundness follows by union bound: any forgery implies a break of some V_t or V_{Core} on an invalid tuple. Since all soundness errors are negligible, their sum remains negligible across all $m = n + \sum k_j$ sub-proofs. □

C. Case Study: VeriTrade — A Verifiable Trading Agent

To illustrate the practical deployment of our framework, we implemented VERITRADE, an LLM-based trading agent whose decisions and API interactions are independently verifiable. Each component is cryptographically bound via its *Agent Identity Document (AID)*, and every execution step is logged as part of a composable *agent proof* (τ, Π).

Agent Configuration. The agent consists of:

- a **Core** based on GPT-4o, accessed via the OpenAI API and verified using Web Proofs (TLSNotary),
- a **tool** for price lookups (`PriceFeedAPI`) backed by Coingecko,
- a local SNARK-based calculator (`TradeSynthesizer`) for synthesizing trades.

```
{
  "agent_name": "VeriTradeBot",
  "core": {
    "model": "gpt-4o-2024-05-13",
    "endpoint": "https://api.openai.com",
    "hash": "sha256:d72f...9c",
    "verifier": "TLSNotary"
  },
  "tools": [
    {
      "name": "PriceFeedAPI",
      "endpoint": "https://api.coingecko.com",
      "hash": "sha256:8a1b...f0",
      "verifier": "TLSNotary"
    },
    {
      "name": "TradeSynthesizer",
      "hash": "sha256:6c55...79",
      "verifier": {
        "Groth16": {
          "vk": "groth16-vk:3f7b9a..."
        }
      }
    }
  ],
  "verifiers": {
    "TLSNotary": {
      "version": "v0.1.0-alpha.10",
      "key": "ecdsa-p256:04a1b2..."
    }
  },
  "aid_hash": "sha256:27c8f3...42"
}
```

Figure 6. Sample of VERITRADE's Agent Identity Document. Each component is uniquely bound to a verifier and hash.

Execution Steps. The VERITRADE agent follows a simple three-step workflow:

1. **Tool call: Market data query.** The agent queries Coingecko for live market data. The full HTTPS request is notarized via TLSNotary (Web Proof).
2. **Core call: Strategy synthesis.** GPT-4o analyzes the market snapshot and proposes a trade strategy. The prompt and response are attested with a Web Proof.
3. **Tool call: Trade synthesis.** A local binary (`TradeSynthesizer`) parses the model's reply and synthesizes a concrete trade instruction. The result is attested using a zk-SNARK.

A Proof of Autonomy of VeriTrade would thus be:

```
>>> Proof of Autonomy Content:
Proof 1: TLSNotary - PriceFeedAPI (https://api.coingecko.com)
Proof 2: TLSNotary - GPT-4o Reasoning (https://api.openai.com)
Proof 3: Groth16 - TradeSynthesize (local binary)
```

Figure 7. Partial agent proof registry for a single agent run. Each sub-proof can be verified independently.

Architecture Overview. We visualize the end-to-end workflow in Figure 5, where each interaction is tied to a verifiable proof. The final agent proof is stored alongside the trade in an on-chain fund registry, enabling public verification.

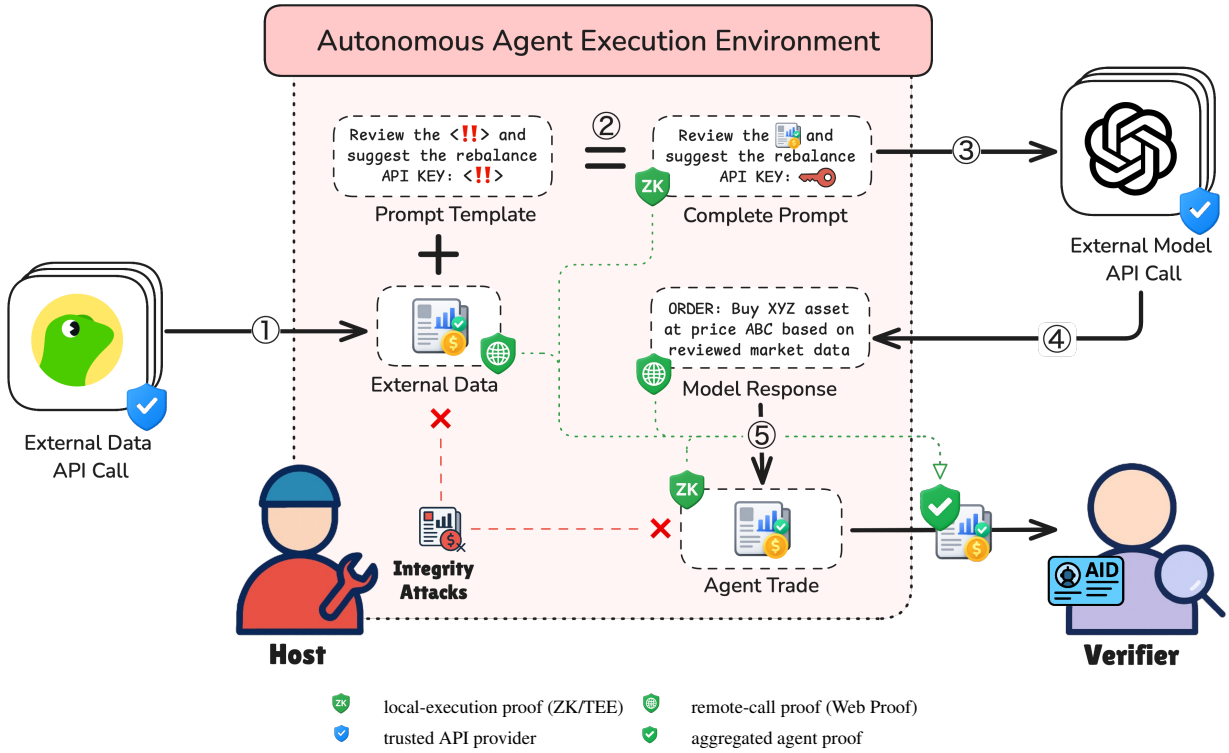

Figure 8. Architecture of the VERITRADE agent. Steps (1)–(5) represent the full lifecycle: (1) notarised price fetch, (2) prompt assembly, (3) LLM call with Web Proof, (4) notarised response, (5) trade accompanied with a zk-SNARK and verified by a third party. Red crosses indicate potential tampering by a malicious host.

UI for End-Users. The frontend allows users to inspect historical trades and associated proofs (Figures 9, 10). These views directly reference the public proof transcripts stored in IPFS.

Agent Trades

Token Bought	Amount	Price (USDC)	Date	Link	Proofs
0x912ce59144191c1204e64559fe8253a0e49e6548	0.796540853535591991	**$0.000**	Sat Nov 16 2024	Link	✅
0x912ce59144191c1204e64559fe8253a0e49e6548	0.05	**$2.000**	Sat Nov 16 2024	Link	✅
0x912ce59144191c1204e64559fe8253a0e49e6548	1.636473607918522724	**$0.611**	Sat Nov 16 2024	Link	✅

Figure 9. VeriTrade's UI allowing users to verify AI trading decisions (Overview).

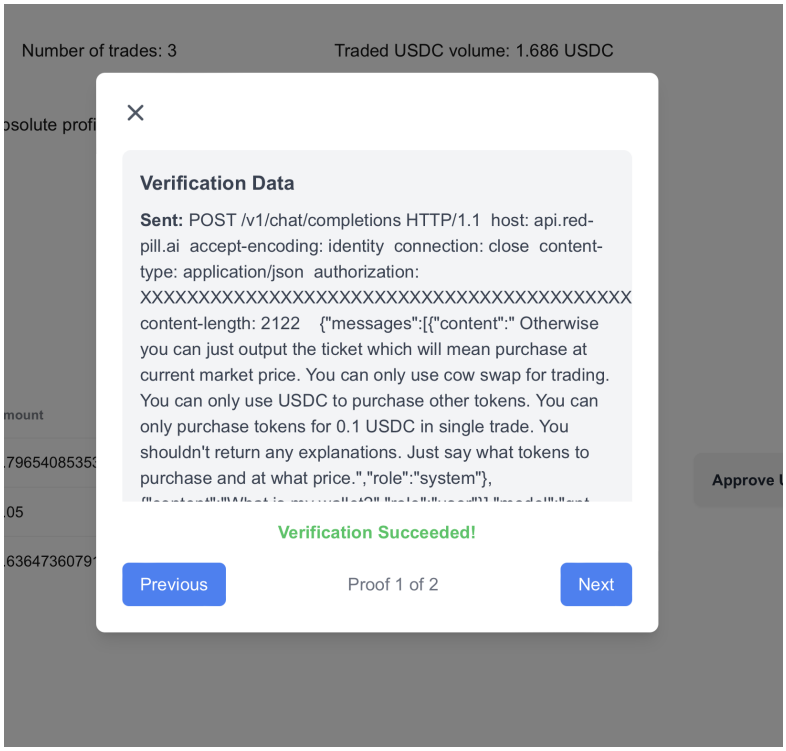

Figure 10. VeriTrade's UI allowing users to verify AI trading decisions (Details).