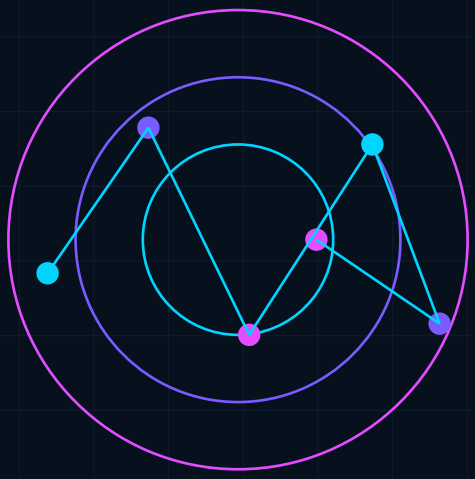
VERIFIABLE COMPUTE

Zero-Knowledge Proofs (ZKPs) for Verifiable Agent Compute

Current-source review, evidence-bounded adoption decision, explicit limitations, reproducibility controls, and preserved source research note.



PERMANENT ID	EVIDENCE	ADOPTION	FRESHNESS
RES-024	B	RESEARCH AND TARGETED PILOT	WATCH

Freshness review: 2026-09-17 / Next scheduled review: 2027-01-15

RESEARCH CONTROL SURFACE

Required Research Fields

Each field remains independently reviewable; evidence strength does not imply production authority.

EVIDENCE GRADE — **B**

Strength of the retained source set and technical basis.

SOURCE AUTHORITY — **2 registered authorities**

EZKL Documentation; Zero-Knowledge Proofs for Machine Learning

FRESHNESS — **WATCH**

Reviewed 2026-09-17; dynamic sources require event-driven revalidation.

CONFLICTS — **VISIBLE / NOT SILENT**

Differences between specification, vendor behavior, research claims, and reproduced evidence must remain explicit.

LIMITATIONS — **EXPLICIT**

No certification, procurement approval, legal conclusion, or unbounded production claim is created by this report.

REPRODUCIBILITY — **REQUIRED**

Material claims require exact versions, representative data, failure cases, artifacts, and repeatable acceptance criteria.

ADOPTION POSTURE — **RESEARCH AND TARGETED PILOT**

Pilot zero-knowledge proofs for deterministic, bounded, high-value computations. Treat general LLM proof claims as research until circuits, quantization, nondeterminism, and proof cost are resolved.

REVIEW TRIGGER — **2027-01-15**

Scheduled at 120 days; earlier on material source, vulnerability, implementation, or contradictory-evidence change.

Authority Register and Freshness Delta

Primary-source lineage is preserved; current-source changes are separated from unperformed implementation claims.

AUTHORITY 01

EZKL Documentation

https://docs.ezkl.xyz/

AUTHORITY 02

Zero-Knowledge Proofs for Machine Learning

https://arxiv.org/abs/2307.05908

2026-09-17 FRESHNESS DELTA

Primary-source register retained and freshness controls advanced to the current review date. No new product or laboratory performance claim is introduced without reproduced evidence.

Freshness policy: scheduled review + immediate review on source supersession, material release, vulnerability, retraction, or contradictory evidence.

Evidence Must Survive Contact With Reality

- Source authority is not implementation evidence. Product behavior, scale, security, interoperability, and operator fit remain environment-specific.
- Conflicting evidence is recorded rather than averaged away. Specification behavior, vendor implementation, preprint results, and production observations are separate evidence classes.
- Quantitative claims are informative until reproduced under documented hardware, software, data, topology, load, and failure conditions.
- Adoption posture is reversible. A stronger source set can coexist with a pilot-only or research-only recommendation when maturity, safety, economics, or operability remain unresolved.

Decision Gate and Validation Plan

ADOPTION POSTURE

RESEARCH AND TARGETED PILOT

Pilot zero-knowledge proofs for deterministic, bounded, high-value computations. Treat general LLM proof claims as research until circuits, quantization, nondeterminism, and proof cost are resolved.

1 / SOURCE	Pin exact versions, publication state, retrieved artifacts, and source hashes.
2 / PROTOTYPE	Demonstrate the core mechanism with representative data and instrumented execution.
3 / FAILURE	Exercise malformed input, dependency loss, stale state, overload, recovery, rollback, and misuse.
4 / BASELINE	Compare against an incumbent, classical, or simpler architecture using the same workload.
5 / ACCEPT	Record acceptance criteria, residual limitations, owner, expiration, and revalidation triggers.

EXPIRATION / REVIEW TRIGGER

Next scheduled review: 2027-01-15 (120-day cadence). Review sooner after material source revision, breaking implementation release, critical vulnerability, retraction, benchmark contradiction, changed workload, or changed legal/safety boundary.

8. Complete Source Research Note

SOURCE-LINEAGE NOTICE

The following material preserves the complete original source note, excluding only the conversational wrapper and outer Markdown fence. Legacy status labels and absolute language are retained for traceability and do not override the candidate status, limitations, or adoption decision in this edition.

Document ID: RES-024 Version: 1.0.0 (Definitive Registry Edition) Status: Published Research Note Classification: Public Organization: Mythos Systems (MYTHOS AI) Owner: Aaron Stovall Effective Date: 2026-07-16 Applies To: Cryptographers, AI security engineers, and platform architects designing verifiable AI pipelines, zero-knowledge cloud compute, and privacy-preserving compliance audits Companion Standards: MYTHOS-SEC-0010 (FHE & ZKP Standard), MYTHOS-AI-0001 (Agentic Frameworks), MYTHOS-DAT-0005 (Tamper-Evident Ledgers), MYTHOS-SEC-0009 (Confidential Computing), RA-001 (Governed Multi-Agent Runtime)

The architecture of AI auditability has failed. When an autonomous agent executes a high-risk action, the organization relies on the AI provider's logs to prove the agent followed the rules. If the proprietary LLM prompt or the user's PII is involved, the organization cannot allow third-party auditors to inspect the logs. The auditor must simply "trust" the provider.

Zero-Knowledge Proofs (ZKPs) shatter this paradox. ZKPs allow an agent (the Prover) to mathematically prove to an auditor (the Verifier) that it executed a specific deterministic policy on a specific input, yielding a specific output, without ever revealing the input, the output, or the proprietary system prompt.

This research note defines the architectural dynamics of ZKPs for Verifiable Agent Compute. It dissects the transition from probabilistic LLM reasoning to deterministic policy execution (zk-ML), the computational constraints of arithmetization, and the choice between zk-SNARKs (succinct) and zk-STARKs (transparent, quantum-resistant). Understanding these dynamics is a prerequisite for building the zero-knowledge, verifiable AI governance pipelines mandated by the Mythos Cryptography Standards.

To apply ZKPs to AI, we must move from standard API logging to cryptographic arithmetic circuits.

1.1 The Prover/Verifier Model

A ZKP system involves two parties:

- The Prover (The Agent/Host): Possesses the private data (the user prompt, the proprietary model weights, the system prompt) and executes the computation.
- The Verifier (The Auditor/Control Plane): Possesses the public parameters (the policy rules, the proof keys). The Verifier checks the mathematical proof generated by the Prover. If the proof is valid, the Verifier is 100% certain the Prover executed the policy correctly, without ever seeing the data.

1.2 zk-SNARKs vs. zk-STARKs

- zk-SNARK (Succinct Non-interactive Argument of Knowledge): Produces very small proofs that can be verified in milliseconds. Ideal for on-chain or edge verification. However, SNARKs rely on a "trusted setup" (a cryptographic ceremony) and are vulnerable to future quantum computers (relying on elliptic curve cryptography).
- zk-STARK (Scalable Transparent ARgument of Knowledge): Produces larger proofs, but requires no trusted setup (transparent) and is mathematically resistant to quantum attacks (relying on hash functions). STARKs scale better for massive computations, such as deep neural network inference.

1.3 Arithmetization (The Circuit)

ZKPs cannot execute standard Python code or PyTorch models. The computation MUST be compiled into an arithmetic circuit—a massive graph of addition and multiplication operations over a finite field.

- The Constraint: Every `if/else` statement becomes a complex set of polynomial constraints. Floating-point math is impossible; everything must be quantized to integers.

Generating a Zero-Knowledge Proof for an AI model introduces severe physical and computational constraints.

2.1 The Non-Determinism of LLMs

ZKPs prove deterministic computations. If input X goes into the circuit, output Y must always come out. Large Language Models are probabilistic; they sample from a probability distribution.

- The Flaw: You cannot natively ZK-prove that an LLM "decided" to output a specific string, because the decision is non-deterministic.
- The Mitigation: ZKPs MUST be applied to the deterministic wrapper around the LLM, not the LLM itself. The LLM proposes an action; the ZKP circuit proves that the Aegis policy engine deterministically evaluated the proposed action against the rules and the user's context, and correctly outputted `Allow` or `Deny`.

2.2 The Prover Time Bottleneck

Generating a ZK proof for a complex computation is 1,000x to 10,000x slower than executing the computation itself.

- The Constraint: Proving the execution of a 1-billion parameter neural network forward pass can take hours on a massive GPU cluster.
- The Impact: zk-ML is currently unviable for real-time, sub-second agent loops. It is reserved for post-hoc compliance auditing or high-value, low-frequency transactions.

2.3 Floating-Point Incompatibility

Neural networks rely on FP16/FP32 floating-point math. ZK circuits operate on prime fields (integers modulo p).

- The Constraint: Converting a neural network to integer math (quantization) introduces slight variations in the output. The ZK circuit must account for these approximations, or the proof will fail.

The true power of ZKPs in the Mythos architecture is realized by isolating the deterministic policy engine (Aegis) and proving its execution to third parties.

3.1 The Zero-Knowledge Audit

An organization deploys an autonomous agent to process highly classified CUI. A government auditor demands proof that the agent never violated the policy (e.g., "Never exfiltrate data to an external server").

- The Mechanic: The PANTHEON runtime executes the agent. For every tool call, the Aegis policy engine evaluates the request. The Aegis evaluation is compiled into a zk-STARK circuit. The runtime generates a proof that `Policy(user_request, tool_params) == Allow` without revealing the user_request or the tool_params.
- The Impact: The auditor verifies the proofs on their local machine. They mathematically know the agent followed the rules, but the classified CUI is never exposed to the auditor.

3.2 Verifiable Inference (zk-ML)

For smaller, critical models (e.g., a medical diagnostic model or a fraud detection classifier), the entire forward pass can be proven.

- The Mechanic: The model weights are committed to a Merkle tree (publicly known). The user's private medical data is fed into the model inside the ZK circuit. The circuit outputs a proof: "I executed the model on your data, and the result is Diagnosis X."
- The Impact: The user gets the diagnosis, and a third party (e.g., an insurance company) can verify the diagnosis came from the correct, unmodified AI model, without the user ever revealing their medical records.

The fundamental shift of ZK-verifiable compute is the abolition of "trust me" AI.

- Regulatory Compliance: GDPR and HIPAA require proof of data processing compliance. ZKPs allow organizations to prove compliance mathematically without exposing the PII to the regulator.
- Decentralized AI (DePIN): In a decentralized compute network (`MYTHOS-CLD-0009`), an anonymous node claims it ran a 70B model. The network cannot trust the node. By attaching a ZK proof to the output, the node mathematically proves it ran the exact model on the exact input, securing the token reward.
- Proprietary IP Protection: A startup can prove to an enterprise client that their proprietary AI algorithm correctly processed the client's data, without exposing the algorithm's source code or weights to the client.

To deploy ZKPs in production AI pipelines, the PANTHEON architecture enforces strict isolation and hybrid execution boundaries.

5.1 The ZK-Coprocessor Architecture

Generating ZK proofs is too slow for the main cognitive loop (Nona).

- The Mitigation: The architecture MUST utilize an asynchronous ZK-Coprocessor. When the Aegis policy engine makes a critical decision, it logs the inputs and outputs. In the background, a dedicated ZK prover cluster (utilizing GPUs or FPGAs) generates the STARK proof. This proof is appended to the Evidence Bundle (`MYTHOS-DAT-0005`) asynchronously, without blocking the agent's execution.

5.2 The ZK-TEE Hybrid

ZKPs protect the logic but are slow. TEEs protect the memory and are fast.

- The Mitigation: The architecture MUST combine them. The Aegis policy engine executes inside an Intel TDX TEE (fast, hardware-isolated). The TEE generates a remote attestation quote (proving the hardware state). A lightweight ZK proof is generated by the TEE proving that the attestation quote is valid and matches the public policy key. This combines the real-time performance of TEEs with the mathematical, post-hoc verifiability of ZKPs.

5.3 The Floating-Point Approximation Bound

When utilizing zk-ML for verifiable inference, the quantization error must be bounded.

- The Mitigation: The ZK circuit MUST include an error-margin parameter. The proof guarantees that the output is within ϵ of the true floating-point result. The system prompt must explicitly state this approximation bound to the auditor, ensuring legal clarity.

The era of trusting AI providers with opaque logs and "do not train on my data" promises is over. Zero-Knowledge Proofs transform AI from a black box into a verifiable mathematical theorem. By isolating the deterministic policy engine and generating cryptographic proofs of execution, we enable absolute regulatory compliance, verifiable decentralized compute, and zero-knowledge audits. In a Mythos-compliant architecture, the agent does not ask for trust; it provides a mathematical proof.

```
An API log is a claim; a ZK proof is a theorem.
A trusted setup is a ceremony; a STARK is transparent.
A floating-point model is unprovable; a quantized circuit is absolute.
A black box AI demands trust; a zero-knowledge AI earns it.
```

```
The prompt is private; the policy is public; the proof is absolute.

> Intelligence may be probabilistic.
> Verifiable enforcement must be absolute.
```

End of Research Note RES-024

9. Source Register

Source	Authority and Status	Freshness Control
EZKL Documentation https://docs.ezkl.xyz/	Primary project documentation Emerging - volatile Published: Living	Validated: 2026-07-22 Review on material revision, withdrawal, supersession, or scheduled report review.
Zero-Knowledge Proofs for Machine Learning https://arxiv.org/abs/2307.05908	Primary research preprint Research - volatile Published: 2023	Validated: 2026-07-22 Review on material revision, withdrawal, supersession, or scheduled report review.

10. Lineage, Review, and Publication Record

Record	Value
Original source file	RES-024_ZKP_Verifiable_Agent_Compute.md
Original source words	1534
Upgrade type	Enterprise research report; source and freshness validation; limitations; adoption decision; reproducibility plan
Generated on	2026-07-22
Current status	Candidate Research Report
Publication authority	Formal Mythos approval not yet recorded
Next review	120 days or event-driven trigger, whichever occurs first