RAFA Network Architecture

RAFA Engineering & Research

San Francisco, CA 11/2025

1.Why a Decentralized Insight marketplace unlocks AI Acceleration

- **Access to unique signal means differentiated, verifiable accuracy.** Proprietary datasets locked in add the "critical nuance" that foundation models lack. Domain-specific data demonstrably improves outcomes in Finance, healthcare and other verticals[1]. Instead of relying on data "modulation", "policy", "ethics" and other risks that centralized providers like snowflake and Adobe have, a decentralized marketplace gives this access to anyone, verifiable openly, with cryptographic guarantees and not meatspace trust or contracts.
- **Governance & access control at query-time. A verifiable, trust minimized** Agentic architecture preserves permissions, satisfying security and —historically blockers to data sharing[2] via immutable on-chain contracts.
- **Distribution & liquidity for data products.** As with Snowflake/AWS-style exchanges, marketplaces reduce friction to **find/license/activate** external data, speeding cohort creation and model improvement. (This dynamic is already visible in data exchanges and is accelerating.)[3]
- **Capital flows prove data's cash value.** Large platforms now pay real money for proprietary datasets/tools to strengthen AI offerings (e.g., BlackRock buying Preqin to feed Aladdin's analytics, Reddit, Brave etc monetizing their data sets etc serve as proofs of this accelerating trend)[4]

2. The Data and Insight Verifiability crisis in the age of LLMs & GWMs

An In-depth explanation of the cryptographic protocols used for model state verification and advice authenticity is crucial because it addresses the Verifiability Crisis in Distributed Artificial

[1] https://www.aicontio.com/2024/02/20/what-are-general-world-models
[2] https://www.techradar.com/pro/rag-is-dead-why-enterprises-are-shifting-to-agent-based-ai-architectures
[3] https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/charting-a-path-to-the-data-and-ai-driven-enterprise-of-2030
[4] https://www.investopedia.com/blackrock-buys-private-markets-data-firm-preqin-8671884

Intelligence. As models transition from experimental curiosities to critical infrastructure underpinning financial analysis, code generation, and automated decision-making, the centralized deployment model—typified by opaque APIs from providers like OpenAI or Anthropic—has revealed significant limitations regarding trust, privacy, and censorship resistance. In response, a robust and lasting need for commoditizing AI models and capturing value where it resides, proprietary, commercially valuable data has emerged, driven by networks such as Akash, Ritual, Gensyn, and various "Compute-as-a-Service" marketplaces. These platforms promise to democratize access to high-performance inference by aggregating heterogeneous GPU resources from independent node operators globally.

However, this decentralization introduces a profound adversarial dynamic known as the "Verifiability Gap." Unlike centralized providers, whose reputation serves as a proxy for trust, decentralized node operators are anonymous, economically rational actors who may be incentivized to cheat. The computational expense of running a 70-billion parameter model is substantial; a rational adversary might therefore attempt to serve a requested inference task using a smaller, cheaper model (e.g., substituting Llama-3-70B with Llama-3-8B), execute the model with aggressive quantization that degrades precision, or simply return pre-computed or fabricated responses to save on energy and hardware wear. Furthermore, in the context of model training and fine-tuning, an operator might falsify the compute logs to claim rewards for training steps that were never performed, or train on unauthorized datasets that violate safety policies or copyright restrictions.

Consequently, there is an urgent necessity for **LLM State Verification**. This discipline encompasses a suite of protocols designed to provide mathematical or cryptoeconomic guarantees that a specific computation was executed correctly over a specific model state. The literature from 2024 and 2025 highlights a bifurcation in approaches: "heavy" cryptographic methods like Zero-Knowledge Machine Learning (ZKML), which offer mathematically sound guarantees at the cost of significant computational overhead which is economically infeasible at least into the near future, and "lightweight" optimistic or secret-based protocols, which leverage game-theoretic assumptions to achieve scalability. For RAFA, we favor the lightweight, economical, and faster go-to-market secret based schemes.

Below, we summarize the technical analysis of the state of verifiable inference and training. We examine the intricate mechanisms of protocols such as **zkLLM**, **opML**, **VeriLLM**, **SVIP**, and **Verifiable Fine-Tuning (VFT)**, dissecting their security models, performance characteristics, and implementation challenges in the quest to build a trustless data foundation to accelerate the current rate of AI development.

2.1. Theoretical Framework of State Verification

To understand the landscape of verification protocols, one must first delineate the specific properties of the "state" being verified. In the context of LLMs, "state" refers not only to the static weights of the neural network but also to the dynamic intermediate representations (activations, KV-caches) generated during the forward pass of inference, as well as the trajectory of weight updates during training.

2.2 The Dual Nature of Verification

The existing literature distinguishes between two primary categories of verification, each requiring distinct cryptographic primitives:

Proof of Inference (PoI):

This form of verification addresses the question: Did the provider run model M with weights θ on input x to produce output y?

The core challenge here is the sheer volume of floating-point operations (FLOPs). A single inference pass of a 70B model requires hundreds of billions of operations. Verifying this trace cryptographically requires encoding these operations into a format (such as an arithmetic circuit) that can be proven. The complexity is compounded by the non-deterministic nature of floating-point arithmetic across different GPU architectures (e.g., NVIDIA vs. AMD), which complicates the generation of identical proofs for the same high-level computation[5],[6].

Proof of Learning (PoL) and Proof of Model (PoM):

These protocols address the provenance of the weights themselves: Did the weights θ result from training on dataset D using algorithm A?

This is significantly harder than PoI because the training process is stochastic (due to random seed initialization, data shuffling, and dropout) and spans orders of magnitude more compute. Early attempts at PoL in 2021 relied on re-execution of random subsets of training steps, but 2025 research has demonstrated the vulnerability of such methods to spoofing attacks, necessitating stronger cryptographic commitments to the training trajectory[7], [8], [9].

2.3 The Threat Landscape

---

[5] Nondeterminism-Aware Optimistic Verification for Floating-Point Neural Networks - arXiv

[6] PolyLink: A Blockchain Based Decentralized Edge AI Platform for LLM Inference - arXiv

[7] arxiv.org/abs/2103.05633

[8] [2208.03567] Proof-of-Learning is Currently More Broken Than You Think - arXiv

[9] Verifiable Fine-Tuning for LLMs: Zero-Knowledge Training Proofs Bound to Data Provenance and Policy - arXiv

The design of verification protocols is driven by specific adversarial models prevalent in decentralized networks:

- **Model Switching Attacks:** The adversary serves a model with fewer parameters or a different architecture to reduce latency and cost[38].
- **Quantization Attacks:** The adversary unilaterally quantizes weights (e.g., from FP16 to INT4), degrading performance to save memory bandwidth without informing the user[38].
- **Privacy Leakage:** In scenarios where the verifier needs to check the computation, they might inadvertently gain access to the user's private prompt or the model's proprietary weights, raising privacy and proprietary model leakage concerns[10].
- **Lazy Verification:** In peer-to-peer verification networks, verifiers collude to approve tasks without computation to maximize their throughput and rewards[11].

---

3. Current State of Verifiability Solutions

3.1 Zero-Knowledge Machine Learning (ZKML) Protocols

Zero-Knowledge Machine Learning (ZKML) constitutes the most rigorous approach to state verification. By converting the inference process into a cryptographic circuit, ZKML allows a prover to generate a Succinct Non-Interactive Argument of Knowledge (SNARK) that attests to the correctness of the output without revealing the model weights or the input data. This property makes ZKML uniquely suited for applications requiring both integrity and privacy, such as healthcare diagnostics or financial auditing.

3.1.1 The Computational Bottleneck

The primary obstacle to ZKML has historically been the "non-arithmetic" nature of deep learning. Zero-knowledge proof systems, such as Plonk or Halo2, operate efficiently over finite fields using addition and multiplication gates. However, modern Transformers rely heavily on non-linear operations—specifically Softmax, RELU, GELU (Gaussian Error Linear Unit), and Layer Normalization—which are computationally exorbitant to represent in arithmetic circuits. Implementing a single Softmax operation often requires complex bit-decomposition and range checks, leading to circuit sizes that explode with model depth[12], [13].

3.1.2 zkLLM: A Breakthrough in Non-Linear Verification

---

[10] [2410.22307] SVIP: Towards Verifiable Inference of Open-source Large Language Models
[11] chatpaper.com/paper/193988
[12] zkLLM: Zero Knowledge Proofs for Large Language Models - Hongyang Zhang
[13] zkLLM - Dapps - IQ.wiki

In response to these limitations, the **zkLLM** protocol (CCS 2024) introduced a specialized architecture designed to verify LLMs of up to 13 billion parameters with manageable overhead. The protocol's innovation lies in two primary components: the tlookup protocol and the zkAttn mechanism.

3.1.3 The tlookup Protocol

Standard lookup arguments (used to verify non-linear functions by referencing a pre-computed table of inputs and outputs) struggle with the high dimensionality of tensors in LLMs. tlookup addresses this by parallelizing the lookup argument. Instead of verifying lookups sequentially, it treats the tensor operations as a batch processing task.

The mechanism utilizes the homomorphic properties of multilinear polynomial extensions. It converts the verification of a tensor operation into a sum-check protocol over a random linear combination of the tensor elements. Specifically, for a tensor S and a valid operation table T, the prover demonstrates that every element $s \in S$ exists in T. This allows the verification of millions of GELU activations simultaneously, leveraging the parallelism inherent in GPUs rather than the serial processing of traditional CPU-based provers[14].

3.1.4 zkAttn: Decomposing Attention

The attention mechanism is the computational heart of the Transformer, defined as $\text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V$. The Softmax function, involving exponentiation and division, is particularly hostile to ZK circuits.

zkAttn circumvents direct circuit implementation of division by exploiting the shift-invariance property of Softmax:

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum e^{x_j}} = \frac{e^{x_i - c}}{\sum e^{x_j - c}}$$

By subtracting a constant c, the protocol stabilizes the exponents. Crucially, zkAttn separates the exponentiation (verified via tlookup) from the summation and division. It verifies the consistency of the normalization term (the denominator) across the entire vector, reducing the complexity from a high-precision division circuit to a series of simpler checks.

Benchmarks indicate that this architecture allows zkLLM to generate proofs for 13B models in under 15 minutes with a proof size of less than 200KB. While this latency precludes real-time chat, it is orders of magnitude faster than generic ZKVMs (Zero-Knowledge Virtual Machines) attempting the same task.10

3.1.5 Modulus Labs and the GKR Protocol

[14] [Literature Review] zkLLM: Zero Knowledge Proofs for Large Language Models

Modulus Labs has pioneered an alternative route to efficient ZKML with their **Remainder** prover. Rather than optimizing general-purpose SNARKs, Remainder is built upon the **GKR (Goldwasser-Kalai-Rothblum)** protocol.

3.3.1 Architectural Advantages of GKR

The GKR protocol is an interactive proof system designed specifically for layered arithmetic circuits. This structure mirrors the layered architecture of neural networks (where Layer N feeds Layer $N+1$).

- **Linear Prover Time:** Unlike SNARKs, where prover time can be superlinear in the circuit size, GKR offers a prover runtime that is linear with respect to the number of gates $O(S)$.
- **Memory Efficiency:** GKR does not require the prover to commit to every intermediate wire in the circuit. Instead, it reduces the claim about the output of Layer i to a claim about the output of Layer $i-1$ using a sum-check protocol. This drastically reduces the memory footprint, a critical factor when verifying large models that are already memory-bound[15], [16].

3.1.6 Specialized Decision Forest Verification

Modulus has demonstrated that this approach is particularly effective for structured models like Decision Forests and specialized deep learning sub-circuits. By designing custom gates that map directly to the ML operations, Remainder achieves a **180x efficiency improvement** over generic ZK proving systems like Halo2 or Plonky2. This performance leap makes it feasible to verify complex financial models or healthcare algorithms on-chain with acceptable cost[17], [18].

3.1.7 The Quantization Trade-off in ZKML

A pervasive challenge in all ZKML protocols is the need for quantization. ZK circuits operate on finite fields (integers modulo a prime p), whereas LLMs are trained in floating-point (FP16/BF16). Converting a model to integers for ZK verification introduces quantization error, which can degrade the model's perplexity and reasoning ability.

ZKML protocols in 2025 employ sophisticated quantization-aware training (QAT) or post-training quantization techniques to minimize this loss. They often use large lookup tables to simulate fixed-point arithmetic with sufficient precision. However, this creates a discrepancy between the "verifiable model" (the quantized version) and the "native model" (the floating-point version used in training), potentially allowing subtle behaviors to differ between the two[42, 19].

---

[15] The GKR Protocol

[16] Time-Optimal Interactive Proofs for Circuit Evaluation - Yimg
[17] Modulus Labs: All You Need to Know About the AI Accountability Platform - Gate.com
[18] A Survey on the Applications of Zero-Knowledge Proofs - arXiv
[19] A Survey of Zero-Knowledge Proof Based Verifiable Machine Learning - arXiv

3.2 Optimistic Verification Paradigms

Given the high latency of ZKML, **Optimistic Machine Learning (opML)** has emerged as the pragmatic solution for consumer-facing decentralized AI. Drawing inspiration from Optimistic Rollups in blockchain scaling (e.g., Arbitrum, Optimism), these protocols operate on the "AnyTrust" assumption: the system remains secure as long as there is at least one honest verifier in the network.

3.2.1 opML: The Interactive Dispute Game

The **opML** framework allows node operators to post inference results on-chain immediately, accompanied by a bond (stake). These results are assumed valid unless challenged within a specific window (the challenge period).

3.2.2 The Bisection Protocol

When a verifier disputes a result, the protocol initiates an interactive **Bisection Game**:

1. **Trace Commitment:** The prover and challenger agree on the execution trace of the inference, represented as a Merkle tree of machine states at each step.
2. **Binary Search:** They iteratively narrow down the disagreement. ("Do our states match at step 1,000,000? Yes. At step 1,500,000? No.") This logarithmic process quickly identifies the single instruction where the divergence occurred.
3. One-Step Verification: Only that single disputed instruction is executed on-chain via a Fraud Proof Virtual Machine (FPVM). The FPVM is a lightweight implementation of the instruction set (e.g., MIPS or RISC-V) running as a smart contract.
   This mechanism ensures that the expensive on-chain computation is only triggered in rare cases of fraud, keeping the amortized cost of verification extremely low[20], [21], [22].20

3.2.3 Multi-Phase Disputation

To further optimize this process, opML employs a **multi-phase** approach. Instead of bisecting immediately at the instruction level, the protocol first bisects at the level of the **computational graph** (e.g., identifying which layer of the neural network produced the error). Once the faulty layer is identified, a second phase bisects the tensor operations within that layer, and finally, a third phase bisects the specific machine instructions. This hierarchical approach leverages the structure of ML models to speed up conflict resolution[23].[23]

---

[20] themoonlight.io/en/review/opml-optimistic-machine-learning-on-blockchain

[21] opML - ORA

[22] OPML: Optimistic Machine Learning on Blockchain, an efficient on-chain AI approach

[23] opML - Dapps | IQ.wiki

3.2.4 VeriLLM: The One-Honest-Verifier (OHV) Protocol

While opML focuses on faithful execution of instructions, **VeriLLM** (2025) optimizes specifically for the architecture of LLMs to reduce the verification burden even further. It introduces a protocol that avoids the need for full re-execution by the verifier[24].[24]

3.2.4.1 The "Prefill-Only" Verification Insight

LLM inference is bifurcated into a "Prefill" phase (processing the prompt in parallel) and a "Decoding" phase (generating tokens sequentially). The Decoding phase is memory-bound and slow.

VeriLLM's key insight is that if the output sequence is already known (because the prover has published it), the verifier does not need to run the slow decoding loop. Instead, the verifier can treat the entire sequence (Prompt + Output) as a single input and run a parallel Prefill pass to verify the token probabilities.

This "parallel verification" reduces the computational cost to approximately 1% of the original inference cost, effectively giving the verifier a 100x speed advantage over the prover. This asymmetry is crucial for decentralized networks, as it allows lightweight nodes (e.g., consumer GPUs) to audit heavy nodes (e.g., H100 clusters)[57].

3.2.4.2 VRF-Based Hidden State Sampling

To prevent "lazy verification"—where verifiers approve results without doing the work—VeriLLM employs a **Commit-Then-Sample** mechanism.

1. **Commitment:** The prover commits to the Merkle root of all hidden states generated during inference.
2. **Random Challenge:** A **Verifiable Random Function (VRF)** on the blockchain selects random indices (e.g., Layer 24, Token 105).
3. Spot Check: The prover must reveal the values at those indices. The verifier checks these values against their own prefill calculation.
   Because of the "avalanche effect" in deep neural networks, a manipulation in any part of the model is highly likely to perturb the hidden states at the sampled locations. This probabilistic check provides high security guarantees without requiring the transmission of the entire hidden state tensor (which would be gigabytes in size)[57].9

---

[24] VeriLLM: A Lightweight Framework for Publicly Verifiable Decentralized Inference - arXiv

4. RAFA's Preferred Verification Scheme: Secret-Based and Watermarking Protocols

A distinct category of protocols emerging in 2025 abandons the pursuit of bit-exact reproduction in favor of statistical guarantees derived from **cryptographic secrets**. These protocols are designed to detect model switching and quantization attacks with negligible overhead. A few of the existing protocols are summarized below.

4.1 SVIP: Secret-Based Verifiable Inference Protocol

The **SVIP** protocol secures inference by embedding a secret "watermark" into the user's prompt that propagates through the model's hidden states in a predictable way, but only predictable to someone holding the secret key[25].

4.1.1 The Three-Party Architecture

SVIP introduces a novel architecture involving a **Secret Embedding Network (SEN)** and a **Proxy Task**:

1. **Secret Embedding:** A trusted third party (or the user) generates a random secret vector s. The SEN transforms this secret into an embedding $t_\psi(s)$ that is compatible with the dimension of the LLM's input space.
2. **Inference with Proxy:** The provider receives the prompt and the embedded secret. They must run the LLM to generate the text output, but they must *also* run a lightweight **Proxy Task Feature Extractor** g_θ on the concatenation of the model's hidden states and the embedded secret.
3. Verification: The provider returns the text and the proxy output z. The user, possessing the secret s and a corresponding Labeling Network, computes the expected value of z.
   Since the Proxy Task is trained to be sensitive to the interaction between the specific model weights and the secret, any substitution of the model (e.g., using a smaller model) results in a mismatch in the proxy output. The provider cannot forge this output because they do not know the secret s or the parameters of the Labeling Network[26].

5.1.2 Security Analysis

The strength of SVIP lies in its resistance to adaptive attacks. Even if an adversary accumulates many query-response pairs, they cannot train a "distillation model" to mimic the proxy outputs because the mapping depends on the high-entropy secret vector s, which changes with every request. Empirical results demonstrate that SVIP detects model switching with false negative

---

[25] SVIP: Towards Verifiable Inference of Open-source Large Language Models - OpenReview
[26] github.com/ASTRAL-Group/SVIP_LLM_Inference_Verification

rates below 5% while adding less than 0.01 seconds to the inference latency—a negligible cost compared to ZKML[58].

4.2 iSeal: Active Fingerprinting for Ownership

While SVIP verifies the *execution*, **iSeal** (2025) focuses on verifying the *ownership* and provenance of the model weights themselves. It addresses the problem of model theft, where an adversary steals weights and deploys them as their own[27].

- **Active Fingerprinting:** iSeal modifies the training process (fine-tuning) to inject specific, secret behaviors into the model. An encoder, initialized with a cryptographic seed (HMAC-SHA256), generates "radioactive" data triggers.
- **Verification:** To prove ownership, the creator sends these trigger queries to the suspect API. The model's response is decoded using the creator's private key. If the model is a stolen copy (even if it has been fine-tuned or quantized), it will exhibit the embedded fingerprint. This allows for copyright enforcement in decentralized model marketplaces[60].

---

5. Verifiable Fine-Tuning (VFT) and Training

Verifying that a model was *trained* correctly (Proof of Learning) presents a challenge distinct from inference verification. Training involves iterating over a dataset for epochs, executing billions of backward passes.

5.1 Evolution from PoL (2021) to VFT (2025)

Early work in Proof of Learning (PoL) (Jia et al., 2021) relied on a "checkpoint-and-replay" mechanism. The prover would save model checkpoints every k steps. The verifier would challenge the prover to re-execute the training between step t and $t+k$ to prove the update was valid.

However, research in 2024–2025 revealed that PoL is vulnerable to spoofing attacks[28]. An adversary with a stolen model and the dataset could generate fake checkpoints that satisfy the re-execution check without actually performing the full training run[29].

This failure led to the development of Verifiable Fine-Tuning (VFT) in late 2025, which replaces simple re-execution with rigorous cryptographic commitments.

5.2 The RAFA-VFT Protocol: KZG Commitments and Verifiable Sampling

---

[27] iSeal: Encrypted Fingerprinting for Reliable LLM Ownership Verification - arXiv
[28] Proof-of-Learning is Currently More Broken Than You Think - arXiv
[29] "Adversarial Examples" for Proof-of-Learning - arXiv

VFT (October 2025) introduces a "Chain of Custody" for model weights, ensuring that a fine-tuned model adheres to declared data provenance and safety policies.

5.2.1 Data Commitment with KZG

VFT requires the trainer to commit to the entire training dataset using a Merkle Tree. Additionally, auxiliary properties of the data (e.g., class distribution histograms, safety labels) are committed using KZG (Kate-Zaverucha-Goldberg) Vector Commitments.

KZG commitments allow the prover to open the commitment at specific positions (proving a data point exists) with a constant-size proof, regardless of the dataset size. This is crucial for verifying that the model was trained on the correct distribution of data (e.g., ensuring a safety-aligned dataset was not swapped for a toxic one[30]).

5.2.2 The Verifiable Sampler

A critical vector for cheating in training is "batch selection bias"—the trainer might skip hard examples or over-sample easy ones. VFT enforces a Verifiable Sampler using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG).

The seed for the CSPRNG is publicly committed. For every training step, the batch indices are derived deterministically from the seed. The verifier can then check that the updates applied to the model correspond exactly to the data points mandated by the verifiable sampler. This removes the trainer's discretion in data selection[31].

5.2.3 Recursive Aggregation

To manage the complexity of verifying millions of updates, VFT uses **recursive SNARKs**. The proof for step $t+1$verifies the update from θ_t to θ_{t+1} *and* the validity of the proof for step t. This "folding" of proofs results in a single, constant-size cryptographic certificate that attests to the integrity of the entire training trajectory[32].

---

[30] KZG Polynomial Commitments - Cryptical
[31] KZG polynomial commitments - Dankrad Feist
[32] (PDF) Verifiable Fine-Tuning for LLMs: Zero-Knowledge Training Proofs Bound to Data Provenance and Policy - ResearchGate

6. Decentralized Infrastructure: Implementing the Protocols

The theoretical protocols discussed are being operationalized in 2025 within a new generation of decentralized AI infrastructure.

6.1 Ritual: The AI Coprocessor

Ritual positions itself as the "Execution Layer for AI." Its architecture is designed to make AI verifiable on-chain without bloating the base layer blockchain.[40]

- **Infernet Nodes:** Ritual's nodes (Infernet) act as an oracle network. They listen for on-chain requests (smart contracts), execute the AI inference off-chain, and return the result with a proof.
- **EVM++:** Ritual introduces a set of **precompiles** to the Ethereum Virtual Machine (EVM) that allow smart contracts to natively verify AI proofs. This includes precompiles for checking ZK-SNARKs and potentially verifying optimistic fraud proofs directly in Solidity.[42]
- **ComputeID:** To support heterogeneous hardware, Ritual uses a ComputeID system. A user can specify ComputeID: TEE for privacy-critical tasks or ComputeID: ZK for highest integrity. The network routes the task to nodes possessing the required capability (e.g., an H100 with TDX enabled).[44]

6.2 Gensyn: Probabilistic Verification at Scale

Gensyn focuses on the massive scale of *training* foundation models, where ZK is currently too slow. Its protocol is a sophisticated cryptoeconomic game.[45]

- **Probabilistic Proof-of-Learning:** Gensyn uses metadata from the gradient descent process (such as the loss trajectory) as a lightweight certificate.
- **Graph-Based Pinpoint Protocol:** This is a variant of the verification game. Solvers perform the work, and Verifiers re-run portions of the work. If a Verifier contests a result, they enter a bisection game (similar to opML) to locate the error.
- **Impact:** This system allows Gensyn to aggregate diverse hardware (from data centers to gaming PCs) into a single training cluster, using the economic threat of slashing to maintain integrity.[47]

---

7. Hardware-Based Verification (TEEs)

While cryptographic protocols offer mathematical security, **Trusted Execution Environments (TEEs)** offer hardware-based security. In 2025, TEEs have become a critical component of the verification stack, particularly for privacy-preserving inference.

7.1 The Role of H100 Confidential Computing

NVIDIA's introduction of **Confidential Computing** on H100 GPUs has been a game-changer. These GPUs support **Trusted Execution Environments (TEEs)** (specifically Intel TDX or AMD SEV-SNP at the host level, and encrypted memory on the GPU).[49]

- **Mechanism:** The model and data are encrypted in memory. The GPU processes them within a secure enclave that is isolated from the host operating system. The GPU hardware generates a **Remote Attestation**—a digital signature rooted in the hardware's manufacturing key—proving that a specific workload was executed within the secure boundary.
- **Pros and Cons:** TEEs offer near-native speed (unlike ZKML) and privacy. However, they require trust in the hardware vendor (NVIDIA/Intel) and have historically been vulnerable to side-channel attacks (e.g., power analysis).

7.2 Hybrid Architectures

The consensus in 2025 is moving toward **Hybrid Architectures**. For example, **Phala Network** and **Ritual** combine TEEs with ZK. The heavy computation happens inside the TEE (for speed), and the TEE generates a lightweight ZK proof (or simply signs the result) to post on-chain. This minimizes the "trust surface" of the hardware while maximizing performance.[49]

---

8. Existing Solution Comparative Performance & Security Analysis

To provide a concrete basis for selection and improvement, we benchmark the leading verification paradigms based on 2025 state of the art methods.

8.1 Performance Metrics Table

Feature	ZKML (zkLLM/Remainder)	Optimistic (opML/VeriLLM)	Secret-Based (SVIP)	TEE (Confidential Computing)
Verification Overhead	High (50x - 1000x)	Low (Avg 1% of inference)	Negligible (<0.1%)	Low (~10-20% for enclave)
Latency (70B Model)	~Hours for proof gen	<1s (Optimistic)	<0.01s	Native speed
Security Assumption	Cryptographic (Math)	1-Honest-Verifier (Game Theory)	Statistical / Secret Keeping	Hardware Vendor Trust
Privacy Support	Excellent (Native)	Poor (Requires data availability)	Moderate (Depends on protocol)	Good (Enclave memory)
Hardware Reqs	High RAM (TB+) for Prover	Standard Consumer GPU	Standard GPU	Specialized (H100/TDX)

Model Constraints	Limited (Quantization needed)	Unlimited (Supports Full Precision)	Unlimited	Unlimited

Data Sources: [10]

8.2 Cost Analysis

- **ZKML:** Prohibitively expensive for consumer chat. A single proof for a large model can cost dollars to dozens of dollars in compute time. Best suited for high-value transactions (e.g., "approve a $1M DeFi loan based on this credit analysis").
- **Optimistic:** Extremely cheap. Since verification is only triggered upon dispute (which is rare due to slashing deterrents), the amortized cost is essentially just the cost of inference plus a small "insurance premium" (stake).
- **SVIP:** Virtually free. The cost is embedding a vector and running a small proxy head. Ideal for "checking" if an API provider is cheating on their Service Level Agreement (SLA).

---

9. RAFA's Verifiable Precision Protocol for Time-Series and Financial Data

To mitigate the tradeoffs involved in all the above approaches to verifying the accuracy of data and model states and computations, we propose an improved protocol specialized for financial and time-series data pipelines, which often involve numerous transformations and iterative updates, where even tiny numerical errors can accumulate into significant biases. Inspired by recent advances in *verifiable fine-tuning (VFT)* for LLMs – which use recursive SNARK proofs to certify each training update[1] – we propose a specialized protocol to ensure high numerical accuracy and integrity in data pipelines. By cryptographically *"folding"* verification proofs

across all processing steps (analogous to VFT's end-to-end certificate[1]), our protocol delivers a single succinct certificate attesting that the entire sequence of data transformations is correct. This approach addresses *all of the above* challenges – rounding errors, update tracking, model drift, and data integrity – in a holistic, widely applicable framework.

9.1 RAFA Protocol Summary

10.1.1 Node Participation and Cryptographic Proofs

- **Nodes**: In the RAFA network, each participating node represents an entity that either offers AI-generated insights or seeks to consume them. Nodes could be individuals, corporations, or AI agents themselves, each equipped with LLMs or GWMs trained on proprietary data provided by vendors and consumed by other agents or "buyers". The architecture of this data availability network is described in the RAFA architecture section.
- **Authenticity of Advice with Cryptographic Proofs**: To ensure the integrity and authenticity of the insights offered, each node must provide cryptographic proof of the state of its LLM or GWM pre and post the "insight generation operation". This proof, akin to a digital signature, verifies that a piece of advice or insight indeed originates from the claimed model without revealing the model's proprietary data or inner workings, but guarantees it using publi*cly detectable watermarking for Language Models*[33],[34],[35]*
- **Step-by-Step Network Setup**: Guide on setting up nodes and participating in the network is described in the RAFA's github documentation.
- **Model Training and Integration**: Detailed process for training/fine-tuning LLMs and GWMs with proprietary data and integrating them into the network, as well as openly available tools for downloading, finetuning, and sandboxing sensitive data are available from the RAFA tool marketplace.

9.2 Rounding Errors, Update Tracking, and Model Drift

9.2.1 Rounding Error Propagation

[33] https://arxiv.org/abs/2310.18491

[34] https://arxiv.org/abs/2301.10226
[35] https://www.mdpi.com/2227-7390/13/9/1420

In time-series computations, premature rounding at intermediate steps may seem harmless but can subtly distort results. Each rounding discards information and those small errors *compound* with every subsequent operation[36]. Over long sequences, this compounding introduces biased drift in numerical outcomes. For example, high-frequency financial prices rounded to the nearest cent carry small errors that bias volatility estimates[37]. Research has shown that such discretization noise can significantly skew volatility calculations, necessitating bias-corrections to recover true market risk levels. Our protocol tackles this by performing calculations in higher precision (or using rational representations) internally, only rounding at final outputs. By deferring rounding and carefully bounding its impact, we minimize cumulative error. Each step explicitly tracks the *allowed error budget*, ensuring the total rounding error stays within acceptable bounds[38]. These error budgets and tolerances are chosen to balance computational overhead with fidelity – overly tight tolerances would bloat verification cost, while overly loose tolerances would weaken accuracy guarantees.

9.2.2 Update Tracking and Drift

In streaming data and forecasting models, *drift* refers to gradual changes that cause models or metrics to become misaligned with reality. One form is **model drift**, where a predictive model's performance degrades as real-world patterns shift (concept drift). Financial time-series are notorious for concept drift – e.g. a stock prediction model trained on old data becomes outdated as market behavior evolves[39]. Regularly updating models or recalibrating parameters is necessary to prevent accuracy decay. Another form is **numerical drift**: even if the data generation process is stable, incremental processing errors (like round-off accumulation) can drift a computed statistic away from its true value. Our protocol addresses both. We *track every update* to data or models in an immutable ledger of changes. This means every time series update, model retrain, or adjustment is recorded with a timestamp and cryptographic hash. By having an auditable chain of updates, we can detect unusual or missing updates and ensure nothing is silently altered. Indeed, a recent secure federated learning system used a blockchain ledger to log model updates, allowing traceability and rollback of any corrupted updates – effectively preventing model drift from untracked changes[40]. We adopt a similar strategy: if an update deviates from expected

---

[36] https://medium.com/@craakash/the-hidden-cost-of-premature-rounding-a-quantitative-analysis-from-arithmetic-to-machine-learning-fcdcb074062a#:~:text=Premature%20rounding%20silently%20corrupts%20results,examples%2C%20and%20provides%20mitigation%20strategies

[37] http://galton.uchicago.edu/~mykland/paperlinks/rounding_2013_April22.pdf#:~:text=Financial%20prices%20are%20often%20discretized,volatility%20estimator%20is%20proposed%20and

[38] https://arxiv.org/html/2510.16830v1#:~:text=VFT%E2%80%99s%20cost%20scales%20with%20trainable,to%20balance%20privacy%20and%20auditability

[39] https://www.mdpi.com/2078-2489/15/12/786#:~:text=For%20example%2C%20in%C2%A0fraud%20detection%2C%20systems,97

[40] https://www.techscience.com/cmc/v85n1/63548/html#:~:text=97.6

bounds or if drift crosses a threshold, it's flagged for review or automatically rolled back using the prior state record. Concept drift is handled by integrating drift detection signals into the update logs (for instance, if distribution shift is detected, a model update entry is expected). Thus, update tracking not only guards against *malicious or accidental tampering* but also ensures timely adaptation when data dynamics change.

9.3 Validating Data Accuracy and Integrity Across Transformation Pipelines

Complex ETL pipelines in finance often ingest raw data, apply cleaning, aggregation, statistical computations, and generate reports. At each stage, errors or inconsistencies can creep in, and without end-to-end visibility, trust in the final output erodes. Our protocol enforces **verifiable accuracy at every transformation step**. The key components are:

- **Cryptographic Commitments and Ledger:** Every dataset or intermediate result is cryptographically committed (e.g. via a hash or Merkle tree node). We employ a *decentralized tamper-proof ledger* (which could be blockchain-based) to log each transformation and its commitment[41]. This provides immutable traceability – an auditor can verify that each intermediate dataset matches the committed hash, ensuring no undetected modifications. The ledger acts as a single source of truth for the pipeline's state at each step, creating an **audit trail** from raw input to final output. Recent research in secure ETL shows that blockchain logging of operations brings **traceability, immutability, and auditability** to data pipelines. In our framework, this means if a financial report number is questioned, one can trace back through each computation (e.g. filtering, currency conversion, rounding) all the way to the source transactions, with cryptographic assurance of integrity at each hop.
- **Step-wise Zero-Knowledge Proofs:** For each transformation (say converting a time series from hourly to daily granularity, or computing a moving average), our system generates a *zero-knowledge proof* that the operation was performed correctly on the committed input data. For example, if step t takes input data state $\mathbf{D}*t$ and produces $\mathbf{D}*$, we construct a proof Π_{t+1} attesting that $\mathbf{D}*{t+1} = F_t(\mathbf{D}*t)$ for the correct function F_t (and within the allowed rounding error if applicable). This proof reveals no sensitive data (important for financial privacy), but convinces verifiers that $\mathbf{D}*$ wasn't manipulated. Crucially, we **aggregate these proofs recursively**. Much like VFT folds per-step training proofs into a single succinct certificate, we iteratively combine the pipeline step proofs: Π_2 validates step 2 *and* the validity of Π_1; Π_3 validates step 3 and Π_2;

---

41 https://ijisae.org/index.php/IJISAE/article/view/7774#:~:text=Perceptron%20%E2%80%93%20Gated%20Recurrent%20Unit,based%20anomaly

and so on. The end result is one constant-size cryptographic certificate Π_{final} that vouches for the entire pipeline's integrity from start to finish. Verifying this final proof is efficient (e.g. milliseconds), allowing real-time checks of data accuracy before decisions are made. The proof construction ensures that any discrepancy at any intermediate stage – whether due to a bug, rounding anomaly, or malicious tampering – would cause verification to fail and thus be detected. Our protocol can also include *range proofs* or tolerance checks at each step, confirming that any rounding or approximation stayed within predefined error bounds, thereby validating that numerical precision was maintained as intended.

- **Automated Integrity Checks and Drift Alerts:** Building on the update ledger and proofs, the system can automatically validate each new data batch or model output against expected patterns. If a transformation generates out-of-range values or violates conservation rules (for instance, a financial pipeline might enforce that debits and credits remain balanced), the anomaly is immediately flagged. An AI-based monitor (like an anomaly detection model) can supplement this by analyzing the sequence of logged operations for unusual patterns, as demonstrated by prior work combining blockchain logs with ML anomaly detectors. Any flagged issue triggers either a rollback (using the stored state from the previous step) or a request for human audit. This ensures not only that *data accuracy is rigorously checked*, but also that pipeline **resilience** is enhanced – the system can self-correct or halt when integrity is in doubt, rather than silently propagating errors.

Conclusion and Outlook

By uniting high-precision computation practices with cryptographic verifiability, the proposed protocol creates a robust foundation for trustworthy time-series and financial data analytics. It addresses rounding error accumulation, ensures every update is tracked and auditable, mitigates model/data drift through vigilant monitoring, and guarantees end-to-end integrity across complex transformation pipelines **by design**. The approach is broadly applicable: regulators could demand such verifiable pipelines for financial reporting or risk modeling, allowing independent verification that results are reproducible and untampered. Enterprises handling large-scale IoT time-series (energy grids, sensor networks, etc.) can use this to maintain data fidelity over long-lived processes. Recent advances in zero-knowledge proofs and blockchain technology have made it practical to verify complex computations with minimal overhead, opening the door for *"trustless"* data pipelines where consumers of the data need not simply trust the provider – they can **verify**. In summary, this novel protocol ensures that from raw data to final insight, every number can be trusted, audited, and accurate to the last decimal – a significant step forward in data reliability for finance and beyond.

Sources: The concept of recursive SNARK proofs for model updates is based on verifiable fine-tuning research. The importance of mitigating rounding error propagation is highlighted by recent analyses in both machine learning and finance. We addressed model drift with inspiration from concept drift detection studies and leveraged ideas from blockchain-based data integrity frameworks and secure federated learning systems that log and verify updates. All these advances inform the design of our integrated high-accuracy data pipeline protocol, which is the RAFA protocol.

Future Directions (2026+):

We anticipate the convergence of these methods into "Adaptive Verification" stacks. A system might start with an SVIP check (cheap, fast). If a red flag is raised, it escalates to an Optimistic Dispute. If the dispute is complex, it might resolve via a ZK Proof of a specific layer. Simultaneously, FHE (Fully Homomorphic Encryption) combined with ZK is the long-term target, promising a future where users can send encrypted prompts to untrusted nodes and receive encrypted, verified answers—solving privacy and integrity in a single stroke. Until then, the protocols detailed in this report constitute the essential infrastructure of the decentralized AI economy.

List of Abbreviations

- **CSPRNG:** Cryptographically Secure Pseudo-Random Number Generator
- **FHE:** Fully Homomorphic Encryption
- **FPVM:** Fraud Proof Virtual Machine
- **GKR:** Goldwasser-Kalai-Rothblum (Interactive Proof Protocol)
- **KZG:** Kate-Zaverucha-Goldberg (Polynomial Commitment Scheme)
- **OHV:** One-Honest-Verifier
- **opML:** Optimistic Machine Learning
- **PEFT:** Parameter-Efficient Fine-Tuning
- **PoL:** Proof of Learning
- **SEN:** Secret Embedding Network
- **SNARK:** Succinct Non-Interactive Argument of Knowledge
- **TEE:** Trusted Execution Environment
- **VFT:** Verifiable Fine-Tuning
- **VRF:** Verifiable Random Function
- **ZKML:** Zero-Knowledge Machine Learning