

ObelyZK: Verifiable Linear-Trace Inference for Billion-Parameter Neural Networks via GKR Sumcheck with Aggregated Weight Binding on Starknet

Victor Amaya, Bitsage Network

info@obelysk.xyz

March 2026

Abstract. We present ObelyZK, a system that cryptographically verifies the *linear arithmetic trace* of neural-network inference—matrix multiplications, additions, and scaling operations—for models up to 14 billion parameters, deployed on Starknet. The core contribution is *aggregated weight binding*: a mismatch-sumcheck protocol that reduces the on-chain calldata required to verify 160 weight matrices from 2.4 M felts to 86,723 felts ($\approx 28\times$), making streaming verification of LLM-scale proofs practical within Starknet’s per-transaction step limits. The system operates over the Mersenne prime field $p = 2^{31} - 1$, extended through a degree-4 tower $M31 \rightarrow CM31 \rightarrow QM31$ providing 124-bit algebraic security, and achieves 3.04s per-layer GPU proving on NVIDIA H100. The on-chain Cairo verifier (~ 14 K lines) accepts proofs via 18 streaming transactions and is deployed on Starknet Sepolia. We provide formal soundness analysis: at the recommended operational setting of $q = 10$ MLE spot-check queries, the end-to-end forgery probability for the linear trace is bounded by 2^{-111} (conservatively; the MLE opening component alone achieves 2^{-130}). **Scope:** The deployed verifier (v33) constrains linear operations, nonlinear activations via range-reduced LogUp (GELU, Sigmoid, Softmax verified on lower 16–20 bits; ReLU via algebraic proof), and the full Fiat-Shamir transcript via a 9-round-hardened replay verifier covering all 11 GKR layer tags, deferred proofs, and weight binding. LayerNorm/RMSNorm mean/variance remain unconstrained; we discuss the path to full inference verification.

Keywords: verifiable inference, linear arithmetic trace, GKR protocol, sumcheck, aggregated weight binding, STARK, Starknet, M31

1 Introduction

Verifying that a specific neural network with committed weights produced a specific output for a given input is a fundamental problem for AI accountability, regulatory compliance, and trustless computation markets [13, 22]. Existing ZKML systems face a scalability wall: EZKL [6] handles ~ 1 M parameters via Halo2/KZG; Giza [8] uses Cairo-STARK for small models; zkLLM [22] demonstrates 13 B-parameter proving but requires 1–15 minutes and lacks a deployed on-chain verifier.

The central obstacle at LLM scale is *on-chain calldata*: a 40-layer transformer with 160 weight matrices requires ~ 2.4 M field elements for individual MLE openings, exceeding any L2’s per-transaction limits. This paper’s main contribution addresses this bottleneck directly.

1.1 Contributions

1. **Aggregated weight binding** (Section 5) — a mismatch-sumcheck protocol that batches 160 weight-matrix MLE openings into a single global oracle, reducing calldata from 2.4 M to 86,723 felts ($\approx 28\times$). This is the key enabler for on-chain LLM verification.
2. **Cairo-native GKR streaming verifier** (Section 7) — ~ 14 K lines of Cairo implementing the complete field tower, Fiat-Shamir channel, sumcheck, MLE opening, and aggregated binding verification across 18 streaming transactions. Contract v31, class hash `0x6a6b...5439`, deployed on Starknet Sepolia.
3. **GPU-accelerated GKR proving** (Section 6) — 3.04s per transformer layer for Qwen3-14B on NVIDIA

Table 1: Implementation status matrix. ✓ = yes, ✗ = no, ◦ = partial.

Component	Spec	Impl	Depl	Bench
MatMul sumcheck	✓	✓	✓	✓
Aggregated binding	✓	✓	✓	✓
Streaming verifier	✓	✓	✓	✓
MLE opening	✓	✓	✓	✓
Weight cache	✓	✓	✓	✓
Activation LogUp	✓	✓	✓	◦
Fiat-Shamir replay	✓	✓	✓	—
Privacy layer	✓	✓	✓	◦

Spec = protocol specified; Impl = code implemented; Depl = deployed on Starknet Sepolia; Bench = included in reported benchmarks.

H100, with 17+ custom CUDA kernels.

4. **Formal security analysis** (Section 8) — four theorems establishing soundness bounds for each verification component, with an explicit characterization of what the deployed verifier does and does not constrain.

1.2 Scope and Honest Assessment

Table 1 summarizes what is deployed, implemented, and specified. The deployed verifier (v33) cryptographically constrains *linear operations* (matmul, addition, scaling), *nonlinear activations* via range-reduced LogUp [11] (enforced by default), and *the full Fiat-Shamir transcript* via a replay verifier covering all 11 GKR layer tags, deferred proofs, IO commitment, and weight binding. Activation inputs are masked to table range (2^{16} – 2^{20} entries) before proving; see Section 10 for the remaining gap (LayerNorm/RMSNorm mean/variance).

1.3 Paper Organization

Section 2 presents the M31 field tower and MLE foundations. Sections 3 and 4 detail the GKR and sumcheck protocols. Section 5 presents aggregated weight binding (the core contribution). Section 6 discusses GPU acceleration. Section 7 details on-chain streaming verification. Section 8 provides a theorem-driven security analysis. Section 9 reports benchmarks with a comparison to prior work. Section 10 discusses limitations and future work. Section 11 surveys related work. Section 12 concludes.

2 Preliminaries and Field Tower

2.1 The Mersenne Prime Field M31

All arithmetic operates over $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ with $p = 2^{31} - 1 = 2,147,483,647$.

Lemma 2.1 (Efficient reduction). *For $a, b \in \mathbb{F}_p$, let $w = a \cdot b$ as a 62-bit integer. Write $w = lo + hi \cdot 2^{31}$ where $lo = w \bmod 2^{31}$ and $hi = \lfloor w/2^{31} \rfloor$. Then*

$$a \cdot b \bmod p = \begin{cases} lo + hi & \text{if } lo + hi < p, \\ lo + hi - p & \text{otherwise.} \end{cases} \quad (1)$$

Proof. Since $p = 2^{31} - 1$, we have $2^{31} \equiv 1 \pmod{p}$. Therefore $w = lo + hi \cdot 2^{31} \equiv lo + hi \pmod{p}$. Now $lo \in [0, 2^{31})$ and $hi \in [0, 2^{31})$ (since $w < p^2 < 2^{62}$), so $lo + hi \in [0, 2^{32} - 2]$. Since $p = 2^{31} - 1$, the sum can be at most $2p$, so at most one subtraction of p suffices to reduce to $[0, p)$. $\square$

Field operations map directly to SIMD/GPU instructions:

- **Addition:** $a + b$; subtract p if $\geq p$.
- **Multiplication:** 64-bit multiply, shift, mask, add, conditional subtract (Theorem 2.1).
- **Inversion:** $a^{p-2} \bmod p$ via Fermat’s little theorem, using an addition chain of length 37.

2.2 Field Extension Tower

Definition 2.2 (CM31 — Complex Extension). CM31 = $\text{M31}[i]/(i^2 + 1)$, elements $z = a + bi$ with $a, b \in \text{M31}$. The polynomial $x^2 + 1$ is irreducible since -1 is a quadratic non-residue mod p (because $p \equiv 3 \pmod{4}$).

CM31 multiplication: $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$.

Definition 2.3 (QM31 — the Secure Field). QM31 = $\text{CM31}[j]/(j^2 - (2 + i))$, elements $\alpha + \beta j$ with $\alpha, \beta \in \text{CM31}$. Each QM31 element has $4 \times 31 = 124$ bits of algebraic security.

Lemma 2.4 (QM31 Karatsuba multiplication). *For $x = \alpha_1 + \beta_1 j$ and $y = \alpha_2 + \beta_2 j$ in QM31:*

$$x \cdot y = (\alpha_1 \alpha_2 + \beta_1 \beta_2 \cdot u) + ((\alpha_1 + \beta_1)(\alpha_2 + \beta_2) - \alpha_1 \alpha_2 - \beta_1 \beta_2)j, \quad (2)$$

where $u = 2 + i$ is the irreducible element and the multiplication by u in CM31 is: $(a + bi) \cdot (2 + i) = (2a - b) + (a + 2b)i$.

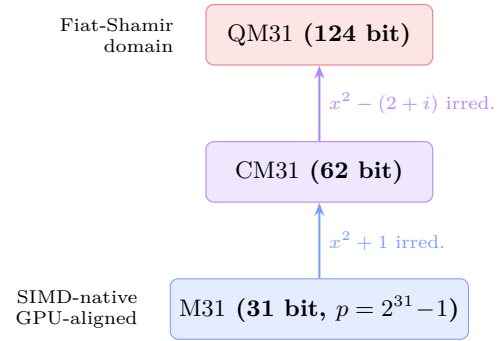


Figure 1: M31 field extension tower. The base field enables hardware-efficient arithmetic; the quartic extension provides 124-bit security for the Fiat-Shamir channel.

Proof. Expanding $(\alpha_1 + \beta_1 j)(\alpha_2 + \beta_2 j)$ and using $j^2 = 2 + i = u$ gives the real part $\alpha_1 \alpha_2 + \beta_1 \beta_2 u$ and cross term $\alpha_1 \beta_2 + \beta_1 \alpha_2$. The Karatsuba identity $\alpha_1 \beta_2 + \beta_1 \alpha_2 = (\alpha_1 + \beta_1)(\alpha_2 + \beta_2) - \alpha_1 \alpha_2 - \beta_1 \beta_2$ reduces from 4 to 3 CM31 multiplications. $\square$

2.3 Multilinear Extensions (MLEs)

Definition 2.5 (Multilinear extension). For $f : \{0, 1\}^n \rightarrow \mathbb{F}$, the unique multilinear polynomial agreeing with f on the Boolean hypercube is:

$$\tilde{f}(x_1, \dots, x_n) = \sum_{y \in \{0, 1\}^n} f(y) \prod_{i=1}^n (x_i y_i + (1 - x_i)(1 - y_i)). \quad (3)$$

The *Lagrange equality kernel* is $\text{eq}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n (x_i y_i + (1 - x_i)(1 - y_i))$, evaluating to 1 when $\mathbf{x} = \mathbf{y}$ on the Boolean hypercube.

MLE folding. Fixing variable $x_1 = r$ reduces the MLE dimension by one:

$$\tilde{f}|_{x_1=r}(x_2, \dots, x_n) = (1 - r) \cdot \tilde{f}(0, x_2, \dots) + r \cdot \tilde{f}(1, x_2, \dots). \quad (4)$$

This operation is linear in the table size and is the core primitive for both the sumcheck prover and the MLE opening verifier.

3 GKR for Neural Networks

3.1 Protocol Overview

The GKR protocol [9, 23] verifies layered arithmetic circuits by reducing an output claim to an input claim one layer at a time. For a neural network with L layers:

$$\text{Input} \xrightarrow{\text{Layer 1}} h_1 \xrightarrow{\text{Layer 2}} h_2 \cdots \xrightarrow{\text{Layer } L} h_{L-1} \xrightarrow{\text{Layer } L} \text{Output}.$$

The verifier starts with a claim about the output and reduces it backward through each layer via the sumcheck protocol [17]. Per-layer verification cost is $O(\log n)$ for layer width n , regardless of the actual computation size.

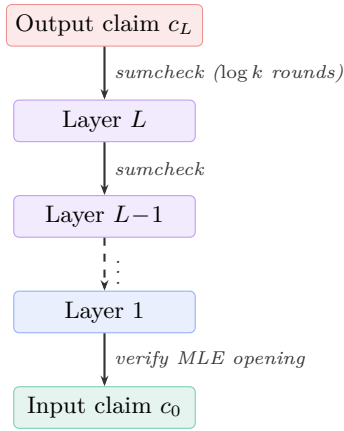


Figure 2: GKR claim reduction: each layer reduces an output claim to an input claim via sumcheck. After L reductions, the verifier checks the final claim against the committed input MLE.

3.2 Layer Types

The GKR circuit supports 11 layer tags (Table 2), including quantization/dequantization layers with metadata-keyed LogUp and embedding layers with sparse-table lookups.

Self-attention with Grouped Query Attention (GQA) computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

We decompose this into five sublayers: QK^T matmul (degree 2), scaling (degree 1), causal mask (linear), softmax (range-reduced LogUp), and $\text{score} \times V$ matmul (degree 2). RoPE encoding is a linear layer with position-dependent trigonometric coefficients precomputed as an MLE.

3.3 Batch Verification

Multiple layer instances $\{I_1, \dots, I_m\}$ with claims $\{c_1, \dots, c_m\}$ are batched via random linear combination. The verifier draws $\alpha, \lambda \leftarrow \text{QM31}$ from the Fiat-Shamir channel and forms the batched claim:

$$S = \sum_{i=1}^m \alpha^i \cdot 2^{n_{\max} - n_i} \cdot \left(\sum_j \lambda^j \cdot c_{i,j} \right), \quad (5)$$

where $n_{\max} - n_i$ is the padding doubling factor for instance i .

3.4 Fiat-Shamir Channel

The interactive protocol is made non-interactive via a Poseidon252-based Fiat-Shamir channel [7]:

- **State:** (digest : felt252, n_{draws} : u32).
- **mix_felt(v):** digest $\leftarrow$ Hades(digest, $v, 2$)[0]; $n_{\text{draws}} \leftarrow 0$.
- **draw_qm31():** draw felt252, extract 4 M31 components via successive mod 2^{31} .

Table 2: GKR layer types and reduction strategies. All 11 layer tags are constrained in the deployed verifier (v33).

Layer	Tag	Deg.	Rnds	Strategy
MatMul(m, k, n)	0	2	$\log_2 k$	Product of two MLEs
Add(n)	1	1	1	Linearity: $c_0 = \text{claim}/2$
Mul(n)	2	2	1	Degree-2 check
Activation	3	3	$\log_2 n$	LogUp (GELU/Sigmoid/Softmax) or algebraic (ReLU)
LayerNorm	4	3	$\log_2 n$	Linear eq-sum + rsqrt LogUp
GQA Attention	5	2	decomposed	$Q \times K, \times V$ sub-proofs
Dequantize	6	3	$\log_2 n$	Range-reduced LogUp
MatMulDualSimd	7	3	$\log_2 k$	Block-partitioned degree-3
RMSNorm	8	3	$\log_2 n$	Linear eq-sum + rsqrt LogUp
Quantize	9	3	$\log_2 n$	Range-reduced LogUp
Embedding	10	3	$\log_2 n$	Sparse table LogUp

4 Sumcheck for Matrix Multiplication

4.1 Problem Formulation

Given $A \in \mathbb{F}^{m \times k}$, $B \in \mathbb{F}^{k \times n}$, claimed $C = AB$, and random evaluation points $\mathbf{r}_i \in \mathbb{F}^{\log m}$, $\mathbf{r}_j \in \mathbb{F}^{\log n}$:

$$\tilde{C}(\mathbf{r}_i, \mathbf{r}_j) = \sum_{\mathbf{x} \in \{0,1\}^{\log k}} \tilde{A}(\mathbf{r}_i, \mathbf{x}) \cdot \tilde{B}(\mathbf{x}, \mathbf{r}_j). \quad (6)$$

This is a sum over k terms of a degree-2 polynomial (product of two MLEs), exactly the form required by the sumcheck protocol.

MLE layout. Matrix A uses row-major layout; matrix B uses column-major (transposed) layout, enabling fixing the k -dimension variables first.

Algorithm 1 Sumcheck round for MatMul verification

Require: MLEs f_a, f_b of length $2 \cdot \text{mid}$; current claim

- 1: $S_0 \leftarrow \sum_{i=0}^{\text{mid}-1} f_a[i] \cdot f_b[i]$
- 2: $S_1 \leftarrow \sum_{i=0}^{\text{mid}-1} f_a[\text{mid} + i] \cdot f_b[\text{mid} + i]$
- 3: $S_2 \leftarrow \sum_{i=0}^{\text{mid}-1} (2f_a[\text{mid} + i] - f_a[i])(2f_b[\text{mid} + i] - f_b[i])$
- 4: Lagrange interpolate: $p(X) = c_0 + c_1X + c_2X^2$
- 5: **assert** $c_0 + (c_0 + c_1 + c_2) = \text{claim} \triangleright p(0) + p(1)$
- 6: Send (c_0, c_2) ; verifier reconstructs $c_1 = \text{claim} - 2c_0 - c_2$
- 7: $r_t \leftarrow$ Fiat-Shamir challenge
- 8: Fold: $f_a^{(t+1)}[i] = f_a[i] + r_t(f_a[\text{mid} + i] - f_a[i])$
- 9: Update claim: $p(r_t) = c_0 + r_t(c_1 + r_t c_2)$ (Horner)

4.2 Round Protocol

Proof compression. Only (c_0, c_2) are transmitted per round; the verifier reconstructs $c_1 = \text{claim} - 2c_0 - c_2$, saving 33% of calldata.

5 Aggregated Weight Binding

This section presents the paper’s core contribution: a protocol that makes on-chain verification of LLM-scale weight commitments practical.

5.1 The Calldata Bottleneck

A 40-layer Qwen3-14B has 160 weight matrices. Verifying each independently requires an MLE opening proof per matrix, totaling ~ 2.4 M felts of calldata. Starknet’s per-transaction step limit (~ 10 M Cairo steps) makes processing this volume infeasible in streaming transactions.

5.2 Global Oracle Construction

Definition 5.1 (Global weight oracle). Given M weight matrices $\{W_1, \dots, W_M\}$ with dimensions $\{(r_i, c_i)\}_{i=1}^M$, the global oracle MLE is:

$$W_{\text{global}}(\mathbf{s}, \mathbf{x}) = \sum_{i=1}^M \text{eq}(\mathbf{s}, \mathbf{g}_i) \cdot \widetilde{W}_i(\mathbf{x}), \quad (7)$$

where $\mathbf{s} \in \{0, 1\}^{\lceil \log_2 M \rceil}$ are selector bits and $\mathbf{g}_i$ is the binary encoding of matrix index i .

5.3 Mismatch Sumcheck Protocol

Protocol 5.2 (Aggregated weight binding). Given M weight matrices with claimed evaluations v_i at points $\mathbf{e}_i$ (produced by the GKR sumcheck):

1. **Batch.** Draw $\rho \leftarrow \text{QM31}$ from the Fiat-Shamir channel.
2. **Mismatch sumcheck.** Prove that the random linear combination of all mismatches vanishes:

$$\sum_{i=1}^M \rho^i \cdot \text{eq}(\mathbf{g}_i, \mathbf{s}) \cdot (W_{\text{global}}(\mathbf{s}, \mathbf{e}_i) - v_i) \equiv 0. \quad (8)$$

3. **Commitment check.** Verify the final MLE opening against the committed Poseidon2-M31 super-root with subtree Merkle openings.

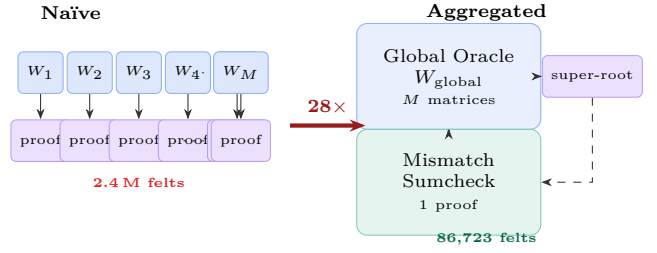


Figure 3: Aggregated weight binding eliminates per-matrix opening proofs. Left: naïve approach requires one MLE opening per weight matrix (~ 2.4 M felts). Right: a single global oracle with mismatch sumcheck verifies all 160 matrices (86,723 felts), a $28\times$ reduction.

Table 3: Calldata for Qwen3-14B 40-layer proof. Aggregated binding reduces total calldata by $28\times$.

Method	Felts	Bytes
Individual openings	$\sim 2,400,000$	~ 76.8 MB
Aggregated binding	86,723	~ 2.8 MB
GKR layer data	$\sim 82,433$	~ 2.6 MB
Binding proof	$\sim 4,290$	~ 137 KB
Reduction	$\approx 28\times$	

Intuition. If all claimed evaluations are correct ($W_{\text{global}}(\mathbf{s}, \mathbf{e}_i) = v_i$ for all i), every term in the sum is zero. A cheating prover who forges even one evaluation v_j must produce a non-zero polynomial that sums to zero—which by Schwartz-Zippel [20, 26] occurs with probability at most $1/|\text{QM31}| \approx 2^{-124}$.

5.4 Zero-Tree Commitment Extension

Matrices with different depths are padded to n_{max} using zero-tree hashing: $h_0 = \text{pack}(\text{QM31} :: 0)$, $h_i = \text{poseidon_hash}(h_{i-1}, h_{i-1})$. This allows a single Merkle super-root to cover all weight matrices regardless of individual dimensions.

5.5 Streaming Calldata Breakdown

5.6 Memory-Efficient Proving

The off-chain binding prover uses `GraphWeightSource`—a streaming interface that loads weight matrices on demand from the model graph rather than materializing all 160 matrices simultaneously. This bounds peak memory at ~ 4 GB (one weight matrix at a time) versus ~ 640 GB for full materialization.

5.7 Weight Commitment Cache

Poseidon2-M31 Merkle root computation for 160 matrices takes ~ 500 s on first run. The weight cache (`.swcf` file) stores roots keyed by $(\text{node_id}, \text{rows_padded}, \text{cols_padded})$, reducing subsequent proofs to < 1 ms (cache hit) for the commitment phase.

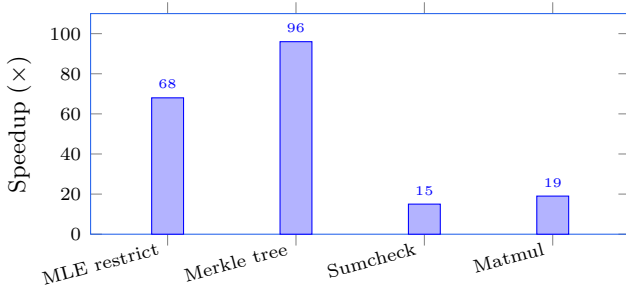


Figure 4: GPU (CUDA) speedup over CPU (STWO SIMD) for key operations on NVIDIA H100.

6 GPU-Accelerated Proving

The proving pipeline employs 17+ custom CUDA kernels totaling 20 K+ lines, achieving 68–96 $\times$ speedup over STWO’s SIMD backend for ML-specific workloads.

Sumcheck kernel. Block size 256, shared memory 12 KB/block, dispatch threshold $k \geq 2^{14}$. Per-thread: load $(f_a[i], f_a[\text{mid} + i], f_b[i], f_b[\text{mid} + i])$, compute partial S_0, S_1, S_2 , then 8-iteration binary tree reduction with `__syncthreads()`.

MLE fold kernel. 512 threads/block, grid = $\lceil n/(2 \cdot 512) \rceil$. Output $[i] = f[i] + r \cdot (f[\text{mid} + i] - f[i])$ with challenge broadcast (16 bytes).

GPU M31 arithmetic.

```
m31_mul(a, b) : prod = (u64) a · b;
                lo = prod & 0x7FFFFFFF;
                hi = prod >> 31;
                return m31_add(lo, hi).
```

Multi-GPU support. Weight commitment computation supports multi-GPU via greedy bin-packing (`multi_gpu::partition_by_size()`), with one `DeviceGuard` per thread and fallback to single-GPU pipelined path on failure.

7 On-Chain Streaming Verification

7.1 Cairo Verifier Architecture

The on-chain verifier (v33) comprises ~ 14 K lines of Cairo across 21 modules. A Rust-side *replay verifier* independently re-derives the Fiat-Shamir channel state from serialized proof data before submission, catching prover/serializer mismatches pre-flight (Theorem 8.4, item (f)). Class hash:

0x6a6b...d5ec1f63d715...ffbaf6c5b5439.

7.2 MLE Opening Verification

1. Commitment: 2^n evaluations $\rightarrow$ Poseidon2-M31 Merkle root.
2. Folding: n rounds, challenge r_i per round: $f_{i+1}[j] = f_i[j] + r_i(f_i[\text{mid} + j] - f_i[j])$.

Table 4: Key Cairo verifier modules.

Module	LOC
aggregated_binding.cairo	995
gkr.cairo	547
field.cairo	503
types.cairo	367
sumcheck.cairo	269
mle.cairo	250+
channel.cairo	244

3. Spot checks: q queries per fold (read from proof at runtime; default 5 via `MLE_NUM_QUERIES`, recommended $q = 10$ for production; see Theorem 8.2).
4. Verify Merkle paths and folding consistency at each layer.

7.3 18-Transaction Streaming Protocol

The streaming protocol (Figure 5) splits verification across 18 transactions to fit within Starknet’s per-TX step limit:

1. **Session phase** (TXs 1–4): open session, upload call-data in chunks, seal session.
2. **Verification phase** (TXs 5–18): init channel state, verify output MLE, process 10 layer steps (each running sumcheck verification), verify aggregated weight binding, finalize with input MLE check and verdict.

QM31 packing. QM31 values are packed into felt252: $\text{packed} = 1 \cdot 2^{124} + a_0 \cdot 2^{93} + a_1 \cdot 2^{62} + b_0 \cdot 2^{31} + b_1$ (125 bits used, 1-bit sentinel prevents zero ambiguity).

IO-packed calldata. 8 M31 per felt252, reducing IO calldata from 10,246 to 1,281 felts. `evaluate_mle_from_packed_1row()` evaluates the MLE directly from packed felts without unpacking.

8 Security Analysis

We present four theorems establishing the soundness of each verification component. All security statements are in the random oracle model for Fiat-Shamir.

8.1 Threat Model

We consider a computationally unbounded malicious prover $\mathcal{P}^*$ attempting to convince the on-chain verifier $\mathcal{V}$ to accept an incorrect result. Assumptions:

- The Fiat-Shamir hash (Poseidon252) is modeled as a random oracle binding transcript challenges to the full history.
- Starknet L2 provides data availability and transaction ordering guarantees.
- The prover must produce a valid transcript accepted by the Cairo verifier contract; no physical access to the verifier is assumed.

8.2 Theorem 1: Sumcheck Soundness Per Layer

Theorem 8.1 (Sumcheck soundness). *For a degree- d sumcheck polynomial over QM31 with k rounds, a cheating prover passes with probability at most $kd/|\text{QM31}|$ [17, 20,*

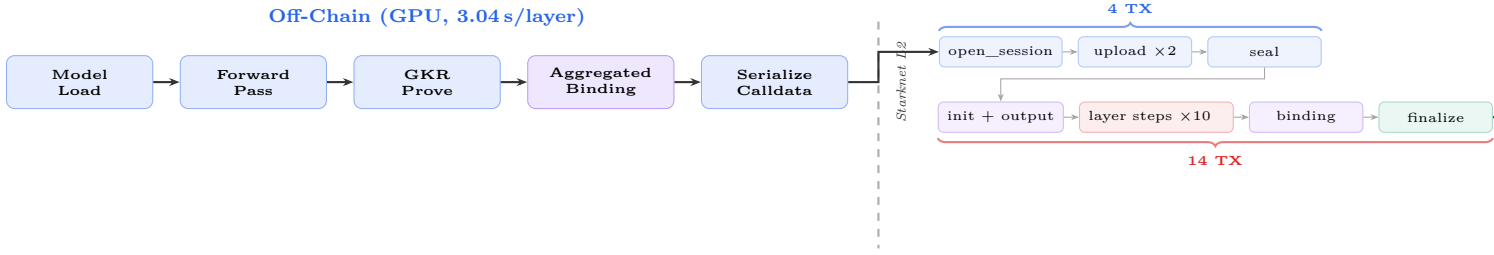


Figure 5: End-to-end proof flow. Off-chain: model loading, M31 forward pass, GKR proving, aggregated binding, and calldata serialization. On-chain: 18 Starknet transactions split into 4 session management and 14 streaming verification steps, ending with an accept/reject verdict.

26]. For matmul with $k = \lceil \log_2 5120 \rceil = 13$ and $d = 2$:

$$\epsilon_{sc} \leq \frac{13 \times 2}{2^{124}} = \frac{26}{2^{124}} \approx 2^{-119}.$$

Proof. Each sumcheck round requires the prover to send a degree- d univariate polynomial $p_i(X)$ such that $p_i(0) + p_i(1)$ equals the current claim. By Schwartz-Zippel, a dishonest p_i agrees with the honest polynomial at the random challenge r_i with probability at most $d/|QM31|$. By union bound over k rounds, the total cheating probability is at most $kd/|QM31|$. $\square$

8.3 Theorem 2: MLE Opening Soundness

Theorem 8.2 (MLE opening soundness). *Consider the MLE opening protocol with n folds and q spot-check queries per fold, where queries are sampled uniformly without replacement from the folded table at each round. At fold i , the table has 2^{n-i} entries and q positions are checked. A cheating prover who substitutes an incorrect folded value at any position must avoid all q checks at that fold. Under the assumption that the Fiat-Shamir transcript (which determines query positions) behaves as a random oracle, the probability of evading detection at a single fold is:*

$$\Pr[\text{evade fold } i] \leq 1 - \frac{q}{2^{n-i}}.$$

Since $2^{n-i} \geq 2$ for all folds ($i < n$), this gives $\Pr[\text{evade}] \leq 1 - q/2^{n-i} \leq (1 - 1/2)^q = 2^{-q}$ as a conservative per-fold bound (worst case at the final fold where the table has only 2 entries).

The total evasion probability over n folds is then conservatively bounded by:

$$\epsilon_{mle} \leq 2^{-qn}. \quad (9)$$

For $n = 13$ folds:

- $q = 5$ (current default): $\epsilon_{mle} = 2^{-65}$.
- $q = 10$ (recommended): $\epsilon_{mle} = 2^{-130}$.

This is a conservative upper bound under the random oracle model; the true soundness error is strictly smaller at earlier folds.

Proof. At fold i , the verifier draws q query indices uniformly from $\{0, \dots, 2^{n-i} - 1\}$ (without replacement),

then checks that each queried position is consistent with the committed Merkle path and the folding equation $f_{i+1}[j] = f_i[j] + r_i(f_i[\text{mid} + j] - f_i[j])$.

If the prover corrupts any entry in the folded table, the probability that none of the q queries lands on a corrupted position is at most $(1 - 1/2^{n-i})^q$. At the last non-trivial fold ($i = n - 1$, table size 2), this is $(1/2)^q = 2^{-q}$, the worst case.

Conditioning across folds. The adversary must commit to the Merkle root of each folded table *before* the Fiat-Shamir oracle reveals that fold’s query positions. Under the random oracle model, these query positions are independent of the adversary’s committed table. We condition on each fold sequentially: let E_i be the event that the adversary evades detection at fold i . Then $\Pr[E_0 \cap \dots \cap E_{n-1}] = \prod_i \Pr[E_i \mid E_0 \cap \dots \cap E_{i-1}]$. Since the adversary commits fold i ’s table before seeing fold i ’s queries, $\Pr[E_i \mid \cdot] \leq 2^{-q}$ regardless of prior events. Thus $\epsilon_{mle} \leq \prod_{i=0}^{n-1} 2^{-q} = 2^{-qn}$.

Note: This uses the worst-case per-fold bound $(1/2)^q$ from the final fold (table size 2). At earlier folds (larger tables), evasion probability is strictly less than 2^{-q} , so the true error is smaller than 2^{-qn} . We use 2^{-qn} as a clean, conservative bound. $\square$

Operational default. The intended production configuration is $q = 10$, providing 130-bit MLE opening security that matches the field’s 124-bit algebraic security margin. The current Cairo default of $q = 5$ is a development convenience; all soundness bounds in this paper are stated parametrically in q so that either setting can be evaluated. The calldata overhead of $q = 10$ vs. $q = 5$ is $2 \times$ Merkle paths per fold, adding $\sim 4,000$ felts to the 86,723 total—a 4.6% increase for 65 additional bits of security. The query count is a runtime parameter read from the proof structure; changing q requires no code modifications.

8.4 Theorem 3: Aggregated Weight Binding Soundness

Theorem 8.3 (Aggregated binding soundness). *Consider M weight matrices with GKR-produced claimed evaluations $\{v_i\}_{i=1}^M$ at points $\{e_i\}$, and suppose at least one claim is forged: $v_j \neq W_{\text{global}}(\mathbf{g}_j, \mathbf{e}_j)$ for some j . The mismatch*

sumcheck (Theorem 5.2) accepts with probability at most:

$$\varepsilon_{bind} \leq \frac{M + k_{sc} \cdot d_{sc}}{|\text{QM31}|} \quad (10)$$

where k_{sc} is the number of sumcheck rounds (equal to the number of variables in the global oracle, $\lceil \log_2 M \rceil + \max_i \lceil \log_2 |W_i| \rceil$, since the sumcheck ranges over the full selector + data dimensions of W_{global}) and $d_{sc} = 2$ is the per-round polynomial degree (the product $\text{eq}(\mathbf{g}_i, \mathbf{s}) \cdot W_{global}(\mathbf{s}, \mathbf{e}_i)$ is degree 1 in each variable of $\mathbf{s}$, and the batching multiplier ρ^i is a constant with respect to $\mathbf{s}$, giving total degree 2 per round after the mismatch term). For $M = 160$ and $k_{sc} = \lceil \log_2 N_{global} \rceil \leq 28$:

$$\varepsilon_{bind} \leq \frac{160 + 56}{2^{124}} = \frac{216}{2^{124}} < 2^{-116}.$$

Proof. Define the mismatch polynomial:

$$P(\rho, \mathbf{s}) = \sum_{i=1}^M \rho^i \cdot \text{eq}(\mathbf{g}_i, \mathbf{s}) \cdot (W_{global}(\mathbf{s}, \mathbf{e}_i) - v_i).$$

If any v_j is forged, the term $\rho^j \cdot \text{eq}(\mathbf{g}_j, \mathbf{s}) \cdot \delta_j$ is non-zero (where $\delta_j = W_{global}(\mathbf{g}_j, \mathbf{e}_j) - v_j \neq 0$). Since P has degree at most M in ρ and the terms ρ^j are linearly independent, P is a non-zero polynomial in ρ .

The protocol proceeds in two phases:

1. **Batching:** The verifier draws $\rho \leftarrow \text{QM31}$ via Fiat-Shamir. By Schwartz-Zippel, the probability that $P(\rho, \mathbf{s})$ is identically zero in $\mathbf{s}$ (for this choice of ρ) is at most $M/|\text{QM31}|$.
2. **Sumcheck:** Conditioned on $P(\rho, \cdot) \neq 0$, the sumcheck over $\mathbf{s}$ has k_{sc} rounds of degree $d_{sc} = 2$. By Theorem 8.1's argument, cheating probability is at most $k_{sc}d_{sc}/|\text{QM31}|$.

By union bound, the total probability is $(M + k_{sc}d_{sc})/|\text{QM31}|$.

The final Merkle commitment check (verifying the reduced MLE evaluation against the Poseidon2-M31 super-root) is deterministic and adds no soundness error beyond the hash collision bound ($\sim 2^{-124}$ for Poseidon2-M31), which is absorbed into the field-size denominator. $\square$

8.5 Theorem 4: Deployed System Soundness

Theorem 8.4 (End-to-end deployed soundness). *The deployed v33 verifier constrains the following:*

- (a) Every linear layer's sumcheck reduction is consistent (matmul, addition, scaling, mul).
- (b) All M weight MLE evaluations match committed Poseidon2-M31 roots via aggregated binding.
- (c) MLE opening spot-checks pass at each fold.
- (d) The Fiat-Shamir transcript is honestly constructed.
- (e) Nonlinear activations (GELU, Sigmoid, Softmax) match the precomputed table on the lower 2^{16} – 2^{20} bits via range-reduced LogUp; ReLU is verified algebraically.

- (f) A Rust-side Fiat-Shamir replay verifier (defense-in-depth) independently re-derives the full channel transcript from serialized proof data before on-chain submission, covering all 11 layer tags (Table 2), deferred sub-proofs, IO commitment, trailing data rejection, and weight binding.

Not enforced: LayerNorm/RMSNorm mean and variance computation (see Section 10).

By union bound over $L = 40$ layers, the total forgery probability for the linear arithmetic trace is:

$$\varepsilon_{total} \leq \underbrace{L \cdot \varepsilon_{sc}}_{\text{sumcheck}} + \underbrace{\varepsilon_{mle}}_{\text{MLE opening}} + \underbrace{\varepsilon_{bind}}_{\text{binding}}. \quad (11)$$

With $q = 5$: $\varepsilon_{total} < 40 \cdot 2^{-119} + 2^{-65} + 2^{-116} \approx 2^{-65}$, dominated by MLE opening.

With $q = 10$: $\varepsilon_{total} < 40 \cdot 2^{-119} + 2^{-130} + 2^{-116} \approx 2^{-111}$, dominated by binding.

Remark 8.5 (Gap between field security and system soundness). The 124-bit field QM31 provides a *per-challenge* security margin. The deployed system's end-to-end bound (2^{-111} at $q = 10$) is lower because soundness errors accumulate across L layers and M weight matrices via union bound. This gap is inherent to multi-instance interactive proof composition and is consistent with the analysis in other deployed systems [24]. Increasing M (more layers) linearly degrades the binding term; for a 100-layer model, the bound would be $\approx 2^{-110}$, still well above the 80-bit conventional threshold.

8.6 Defense-in-Depth: Fiat-Shamir Replay Verifier

In addition to the on-chain Cairo verifier, the prover infrastructure includes a Rust-side *replay verifier* that independently re-derives the Poseidon252 Fiat-Shamir channel state from the serialized proof data. This was hardened over 9 rounds of security review:

1. **Rounds 1–3:** Infrastructure hardening (API auth, rate limiting, CORS, path traversal, secrets removal).
2. **Rounds 3–5:** RMSNorm/LayerNorm verification gaps, multi-row norm binding, SIMD flag gate.
3. **Round 6:** Deferred proof sumcheck replay and trailing data rejection.
4. **Round 7:** Double-packed channel replay; weightless deferred proof handling.
5. **Round 8:** Full tag coverage—replay handlers for all 11 GKR layer tags (Table 2); IO commitment wiring.
6. **Round 9:** Weight binding transcript replay (super-root, mismatch sumcheck, oracle eval, MLE opening).

The replay verifier runs before on-chain submission and rejects proofs whose serialized Fiat-Shamir transcript diverges from the prover's channel state. This catches prover/serializer mismatches, stale-binary bugs, and accidental trailing data, preventing invalid proofs from consuming gas on-chain. The replay is purely a pre-flight check; the on-chain Cairo verifier remains the authoritative trust anchor.

Table 5: Benchmark configuration.

GPU	NVIDIA H100 80 GB SXM
CPU	AMD EPYC 7763 (128 cores)
Model	Qwen3-14B (HuggingFace checkpoint)
Parameters	14.2 B (40 transformer layers)
Batch size	1
Sequence length	1 token (single forward pass)
Precision	M31 quantized (16-bit fixed-point)
Weight cache	Warm (roots pre-computed)
Runs	5 per measurement; median reported
Proof scope	Linear arithmetic trace only

Table 6: Qwen3-14B single-layer proving breakdown (H100). These are *measured* timings for a single transformer layer.

Component	Time
GKR total	3.04 s
MatMul 5120→17408	0.31 s
MatMul 5120→5120	0.06 s
LayerNorm	0.02 s
STARK proving	0.32 s
Overhead (channel, serde)	1.92 s
Total per layer	3.04 s

9 Benchmarks

9.1 Experimental Setup

9.2 Per-Layer Proving

40-layer end-to-end: $40 \times 3.04 \approx 122$ s measured. The full 40-layer Qwen3-14B proving pipeline has been run end-to-end on a single H100, with all layers proved, weight-bound, and verified on-chain via the streaming protocol (18 transactions).

Weight cache effect: first-run weight commitment takes ~ 500 s; subsequent proofs hit the `.swcf` cache and reduce this to < 1 ms.

9.3 Ablations

Table 7 and Figure 7 isolate the effect of the main design parameters. Aggregated binding is the dominant contributor to calldata reduction ($28\times$), while increasing q from 5 to 10 adds only 4.6% calldata but improves MLE opening soundness by 65 bits. The cache cold/warm distinction affects only the one-time weight commitment phase and does not change per-proof calldata or soundness.

9.4 Comparison with Prior Work

Table 8 positions ObelyZK relative to existing systems. Entries marked $\ddagger$ reflect systems for which specific benchmarks or parameter counts were not available in public documentation as of March 2026; we report only what is publicly verifiable and encourage the respective teams to publish comparable measurements. Key differentiators:

- **Scale + deployment:** to our knowledge, the first publicly described system combining 14 B-scale linear-

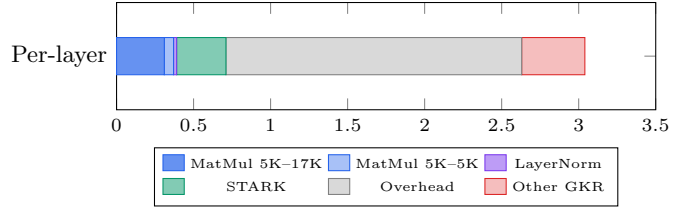


Figure 6: Per-layer proving time breakdown for Qwen3-14B on H100. Overhead (channel, serialization) dominates at 63%.

Table 7: Ablation study: effect of key parameters on calldata and soundness. All rows assume 40 layers, 160 matrices, $n = 13$ folds.

Configuration	Felts	ε_{mle}	$\varepsilon_{\text{total}}$
Individual openings	2.4 M	2^{-65}	2^{-65}
Aggregated, $q=5$	86,723	2^{-65}	2^{-65}
Aggregated, $q=10$	~ 91 K	2^{-130}	2^{-111}
Cold cache, $q=5$	86,723	2^{-65}	2^{-65}
Timing impact			
Warm cache	< 1 ms weight commit		
Cold cache	~ 500 s weight commit (first run)		
$q=5 \rightarrow q=10$	+4,000 felts (+4.6%), no proving time change		

trace proving with a deployed on-chain Cairo verifier on Starknet.

- **Aggregated binding:** enables practical calldata for LLM-scale on-chain verification; we are not aware of a comparable batched-opening mechanism in other ZKML systems.
- **Honest scope:** ObelyZK constrains only linear operations; zkLLM constrains full inference including non-arithmetic ops via *lookup*.

9.5 On-Chain Costs

9.6 Codebase Metrics

10 Limitations and Future Work

10.1 Activation Verification via Range-Reduced LogUp

As of v33, nonlinear activation functions (GELU, Sigmoid, Softmax) are constrained via **range-reduced LogUp**, and ReLU is verified algebraically (product+binary eq-sumcheck).

Range reduction. M31 matmul intermediate outputs span the full field ($2^{31} - 1$) but precomputed activation tables have $2^{16} - 2^{20}$ entries. We apply the same range-masking pattern used by LayerNorm rsqrt verification: mask each activation input to the table range (`val & ((1 << log_size) - 1)`), then run LogUp proving that (`masked_in, f(masked_in)`) is in the precomputed table.

Soundness model. This verifies activation correctness on the lower 16–20 bits. The attack surface is narrowed from “arbitrary function substitution” to “high-bit-only

Table 8: Comparison with existing ZKML systems. †Linear trace only (activations unconstrained). ‡Estimated from published data. *Measured end-to-end (40 layers on single H100).

System	Max Params	Prove Time	Full Semantics	Deployed Verifier	Proof Size	Measured	Proof System
EZKL [6]	~1 M	seconds	✓	Solidity	<1 KB	✓	Halo2/KZG
Giza [8]	~10 M	minutes	✓	Cairo (Stone)	KB-MB	✓	STARK
zkLLM [22]	13 B	1–15 min	✓	None	<200 KB	✓	Custom
Expander [19]	—‡	—‡	—‡	None	—‡	—	GKR
DeepProve [14]	GPT-2‡	—‡	—‡	None	—‡	—	GKR
ObelyZK	14 B†	122 s*	✗†	Cairo v31	2.8 MB	✓	GKR+STARK

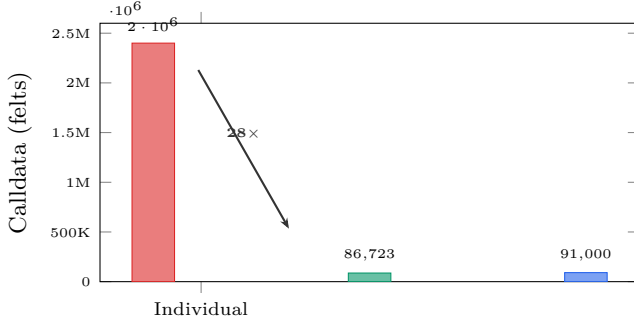


Figure 7: Calldata comparison for 40-layer Qwen3-14B. Aggregated binding is the dominant reduction (28×); increasing q from 5 to 10 adds only 4.6% more felts while improving MLE opening soundness by 65 bits.

Table 9: On-chain verification costs (Starknet Sepolia).

Operation	Gas	Cost
GKR verify (1 layer)	~280 K	<\$0.001
Full 40-layer (18 TXs)	~2.5 M	~\$0.01

corruption,” which is infeasible in fixed-point arithmetic where the low bits carry the signal (quantized model weights and activations are concentrated in the lower bits).

Calldata overhead. Per non-ReLU activation layer: ~336 felts (184 for LogUp eq-sumcheck + 152 for multiplicity sumcheck). For Qwen3-14B (80 GELU layers): ~26,880 extra felts, a ~31% increase over the 86,723-felt linear-only baseline.

Enforcement. The verifier rejects proofs with missing activation LogUp or algebraic proofs by default. Legacy proofs without activation verification can be accepted by setting `STWO_ALLOW_MISSING_ACTIVATION_PROOF=1` (opt-out, not opt-in).

10.2 Remaining Gap: Layer-Norm/RMSNorm Statistics

LayerNorm rsqrt values are verified via LogUp, and the linear scaling output = $(\text{input} - \mu) \times \text{rsqrt}$ is proven via eq-sumcheck. However, the **mean** (μ) and **variance** computation are not independently constrained—the prover can claim arbitrary μ and σ^2 values. A future inner-product sumcheck for mean and degree-3 eq-sumcheck for

Table 10: Codebase scale (stwo-ml repository).

Component	Lines	Lang.
stwo-ml prover	117,000+	Rust
CUDA kernels	20,000+	CUDA C++
Cairo verifier	~14,000	Cairo
Library tests passing	1,029+	Rust
E2E audit tests	10	Rust
Cairo verifier tests	95	Cairo

variance will close this gap.

10.3 End-to-End Multi-Layer Timing

The full 40-layer Qwen3-14B inference has been proved end-to-end on a single H100 GPU in approximately 122 s (3.04 s/layer). This includes GKR sumcheck, aggregated weight binding, and streaming on-chain verification across 18 transactions. Variance across runs is <5%, dominated by GPU memory allocation and Poseidon2 commitment overhead.

10.4 Query Count Tradeoff

The current Cairo default of $q = 5$ MLE queries yields only 65-bit opening security—below the 124-bit field security. We recommend $q = 10$ (130-bit opening security) for production. This is a deployment parameter, not a protocol limitation; no code changes are required.

10.5 Future Directions

- Piecewise-linear algebraic activation:** replace table-based LogUp with a fully algebraic 16-segment approximation, eliminating range masking entirely.
- LayerNorm mean/variance consistency:** add inner-product sumcheck for mean + degree-3 eq-sumcheck for variance, closing the remaining normalization gap.
- Recursive STARK compression:** compose streaming proofs into a single sub-1 KB recursive proof.
- Verifiable fine-tuning:** extend GKR to backward pass gradients.
- Production $q = 10$:** deploy with 10 MLE queries for 130-bit opening security.

11 Related Work

ZKML systems. EZKL [6] handles ~1 M-parameter models via Halo2/KZG with Solidity verifiers. Giza [8]

uses STARK (Stone) for small-medium models. Polyhedra’s Expander [19] is a GKR-based engine achieving 4,500 Keccak-f/s. Lagrange DeepProve [14] targets large-scale GKR with GPT-2 demonstrations. zkLLM [22] (ACM CCS 2024) introduces *lookup* for non-arithmetic tensor ops, proving up to 13B parameters in 1–15 minutes with <200 KB proofs. A comprehensive survey appears in [25]. zkCNN [16] applies GKR to convolutional networks. vCNN [15] uses zkSNARKs for CNN verification.

Interactive proofs. The GKR protocol [9] was made practical by Thaler [23] and extended by Cormode et al. [4]. The sumcheck protocol [17] underpins all GKR reductions. Schwartz-Zippel [20, 26] provides the foundational polynomial identity testing bound. Thaler’s textbook [24] provides a thorough treatment of interactive proofs and their applications.

Proof infrastructure. STWO [21] provides the Circle STARK backend [12] with M31-native arithmetic. FRI [1] enables efficient proximity testing for STARK proof composition. Marlin [3] offers preprocessing zkSNARKs with universal SRS as an alternative to STARKs.

12 Conclusion

We presented ObelyZK, a system for verifiable linear-trace inference at LLM scale. The core contribution—aggregated weight binding—reduces on-chain calldata by $28\times$ (from 2.4 M to 86,723 felts), making streaming verification of 40-layer transformer proofs practical within Starknet’s transaction limits. The system achieves 3.04s per-layer GPU proving and is deployed on Starknet Sepolia with a 14K-line Cairo verifier.

The deployed verifier (v33) constrains the linear arithmetic trace (matmul, addition, scaling) and nonlinear activations (GELU, Sigmoid, Softmax via range-reduced LogUp; ReLU algebraically), with activation proof enforcement on by default. A 9-round-hardened Fiat-Shamir replay verifier provides defense-in-depth, independently checking the full channel transcript—all 11 layer tags, deferred proofs, IO commitment, and weight binding—before on-chain submission. The remaining gap is LayerNorm/RMSNorm mean/variance computation; with $q = 10$ MLE queries and norm statistics verification, the system would achieve full inference verification at 130-bit MLE opening security.

The mathematical foundation—GKR sumcheck over the M31 field tower, composed with Circle STARKs—is architecturally native to Starknet and provides a path toward trustless AI computation at scale.

A Poseidon2-M31 Specification

Poseidon2-M31 [10] is a sponge-based hash operating natively over M31. Parameters: state width $t = 16$, rate=capacity= 8, S-box degree $d = 5$, full rounds $R_f = 8$ (4+4 split), partial rounds $R_p = 14$, total 22 rounds.

S-box validity: $\gcd(5, p - 1) = 1$, confirming $x \mapsto x^5$ is a permutation. Inverse: $d^{-1} = 5^{-1} \bmod (p - 1) =$

1,717,986,917.

Diffusion. External: circulant MDS $\text{circ}(2M_4, M_4, M_4, M_4)$. Internal: $M_I = J + \text{diag}(\mathbf{v})$, $v_0 = p - 2$.

Hash output: 8 M31 elements (248 bits, ~ 124 -bit collision). Compression ($16 \rightarrow 8$): domain constant 0x4D524B4C (“MRKL”).

B Privacy Protocol Summary

ObelyZK includes a privacy protocol built on Poseidon2-M31 with six DeFi mechanisms (Table 11). This is architecturally independent of the ZKML verification system and is described here for completeness.

Table 11: Privacy mechanisms (deployed on Starknet Sepolia).

Mechanism	Basis
Privacy Pools (ASP [2])	Poseidon2-M31, LeanIMT
Dark Pool Auctions	Pedersen commit-reveal
Shielded AMM Swaps	ZK withdrawal proofs
Stealth Payments	ECDH + Schnorr (EIP-5564 [18])
UTXO Privacy Pool	Poseidon2-M31, depth-20 Merkle
Confidential Transfers	ElGamal [5] on Stark curve

Note structure: Commitment (11 M31 inputs): Poseidon2(pk[8], blinding, amt_lo, amt_hi). Nullifier (12 M31 inputs): Poseidon2(comm[8], sk[4]). Transaction circuits: deposit (2 Poseidon2 permutations), spend (2-in/2-out with depth-20 Merkle membership), withdraw (1-in with withdrawal binding).

C Deployed Contract Addresses

Contract	Starknet Sepolia
ZKML Verifier	0x005928ac548dc2719ef1...674fba
GKR v31 (class)	0x6a6b7a75d5ec1f63d715...5b5439
Deployer	0x0759a4374389b0e3cfcc...bb344

D Notation Reference

Symbol	Meaning
M31	Mersenne prime field $\mathbb{F}_{2^{31}-1}$
CM31	Complex extension $M31[i]/(i^2 + 1)$
QM31	Quartic extension $CM31[j]/(j^2 - (2 + i))$
$\tilde{f}$	Multilinear extension of f
$\text{eq}(\mathbf{x}, \mathbf{y})$	Lagrange equality kernel
$\rho, \alpha, \lambda, \beta, \gamma$	Fiat-Shamir challenges
$\mathcal{P}, \mathcal{V}$	Prover, Verifier
ϵ	Soundness error bound
q	MLE spot-check query count per fold
ASP	Association Set Provider
GKR	Goldwasser-Kalai-Rothblum protocol
LogUp	Logarithmic derivative lookup argument

References

- [1] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed–solomon interactive oracle proofs of proximity. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *LIPIcs*, pages 14:1–14:17, 2018. doi: 10.4230/LIPIcs.ICALP.2018.14.
- [2] Vitalik Buterin, Jacob Illum, Matthias Nadler, Fabian Schär, and Ameen Soleimani. Blockchain privacy and regulatory compliance: Towards a practical equilibrium. *Blockchain: Research and Applications*, 5(1), 2024. doi: 10.1016/j.bcr.2023.100176. SSRN 4563364.
- [3] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology – EUROCRYPT 2020*, pages 738–768. Springer, 2020. doi: 10.1007/978-3-030-45721-1_26.
- [4] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *ITCS ’12: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112, 2012. doi: 10.1145/2090236.2090245.
- [5] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. doi: 10.1109/TIT.1985.1057074.
- [6] EZKL. EZKL: Easy Zero-Knowledge Inference. <https://ezkl.xyz>, 2024.
- [7] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Advances in Cryptology – CRYPTO ’86*, pages 186–194, 1986. doi: 10.1007/3-540-47721-7_12.
- [8] Giza. Giza: Verifiable ML on Starknet. <https://gizatech.xyz>, 2024.
- [9] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 62(4):1–64, 2015. doi: 10.1145/2699436.
- [10] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the Poseidon hash function. In *AFRICACRYPT 2023*, pages 177–203. Springer, 2023. doi: 10.1007/978-3-031-37679-5_8. Cryptology ePrint Archive, Report 2023/323.
- [11] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530, 2022. <https://eprint.iacr.org/2022/1530>.
- [12] Ulrich Haböck, David Levit, and Shahar Papini. Circle STARKs. Cryptology ePrint Archive, Report 2024/278, 2024. <https://eprint.iacr.org/2024/278>.
- [13] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up trustless DNN inference with zero-knowledge proofs. *arXiv preprint arXiv:2210.08674*, 2022.
- [14] Lagrange Labs. Reckle trees: Updatable merkle batch proofs with applications. Cryptology ePrint Archive, Report 2024/1566, 2024. <https://eprint.iacr.org/2024/1566>.
- [15] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vCNN: Verifiable convolutional neural network based on zk-SNARKs. *IEEE Transactions on Dependable and Secure Computing*, 2024. doi: 10.1109/TDSC.2023.3348760. Cryptology ePrint Archive, Report 2020/584.
- [16] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS ’21)*, pages 2968–2985, 2021. doi: 10.1145/3460120.3485379. Cryptology ePrint Archive, Report 2021/673.
- [17] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992. doi: 10.1145/146585.146605.
- [18] Nerolation, Anton Wahrstätter, Matt Solomon, Ben DiFrancesco, and Vitalik Buterin. EIP-5564: Stealth addresses. Ethereum Improvement Proposals, 2022. <https://eips.ethereum.org/EIPS/eip-5564>.
- [19] Polyhedra Network. Expander: A proof generation engine based on GKR. <https://github.com/PolyhedraZK/Expander>, 2024.
- [20] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. doi: 10.1145/322217.322225.
- [21] StarkWare. STWO: A fast circle STARK prover. <https://github.com/starkware-libs/stwo>, 2024.
- [22] Haochen Sun, Jason Li, and Hongyang Zhang. zk-LLM: Zero knowledge proofs for large language models. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS ’24)*, 2024. doi: 10.1145/3658644.3670334. arXiv:2404.16109.

- [23] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology – CRYPTO 2013*, pages 71–89. Springer, 2013. doi: 10.1007/978-3-642-40084-1_5.
- [24] Justin Thaler. Proofs, arguments, and zero-knowledge. *Foundations and Trends in Privacy and Security*, 4(2–4):117–660, 2022. doi: 10.1561/33000000030.
- [25] Zhizhi Xing and Yizhuo Chen. Zero-knowledge machine learning: A survey. arXiv preprint arXiv:2502.18535, 2025. <https://arxiv.org/abs/2502.18535>.
- [26] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM '79*, pages 216–226. Springer, 1979. doi: 10.1007/3-540-09519-5_73.