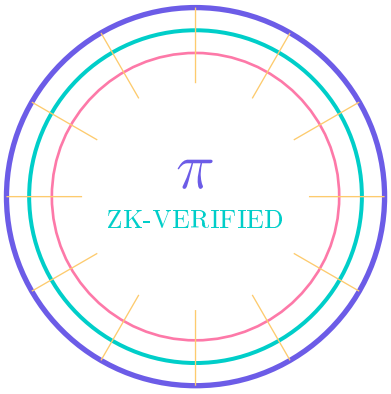
# VeritasLedger

## Zero-Knowledge Verified Open-Source AI on the Blockchain

Photonic-Accelerated Proof Generation for Trustless AI Provenance




**Whitepaper v1.0**

February 2026

Deployed on Base (Ethereum L2)

*A decentralized protocol ensuring every AI-generated output is cryptographically verified, its model provenance is transparent, and no user data is collected—ever. Accelerated by photonic computing for real-time zero-knowledge proof generation at scale.*

DRAFT -- FOR

REVIEW -- NOT FINANCIAL ADVICE
Contact: research@veritasledger.io

## Abstract

The rapid proliferation of large-scale artificial intelligence systems has engendered a crisis of trust: consumers cannot verify which model generated a given output, whether the model was tampered with, or whether their personal data was harvested during inference. Simultaneously, open-source AI models—while philosophically aligned with transparency—lack any cryptographic mechanism to prove their identity or the integrity of their computations once deployed.

We introduce VeritasLedger, a decentralized protocol built on Base (Ethereum L2) that resolves these challenges through three synergistic innovations. First, we develop a **zero-knowledge machine learning (zkML) framework** that arithmetizes neural network inference into rank-1 constraint systems (R1CS) and generates succinct non-interactive arguments of knowledge (zk-SNARKs) proving that a specific registered model produced a specific output—without revealing model weights or user inputs. Second, we propose a **blockchain-native model registry and provenance ledger** where open-source models are hash-locked at registration time, and every inference event produces an immutable on-chain attestation that the output is AI-generated, traceable to its source model. Third, we introduce a **photonic-accelerated proof generation** pipeline leveraging Mach-Zehnder interferometer (MZI) meshes to perform the number-theoretic transforms (NTTs) central to polynomial commitment schemes at optical clock rates, achieving an estimated 8.6× speedup over state-of-the-art GPU provers.

The protocol is designed so that **no user identification or data collection is required**: queries are encrypted client-side, inference runs inside a zero-knowledge virtual machine, and the resulting proof attests only to computational integrity—not to user identity. This architecture inverts the prevailing surveillance-based AI paradigm and provides a public good: universal, verifiable, privacy-preserving AI content labeling.

We formalize the security model under the algebraic group model and the generic bilinear group model, proving computational soundness, zero-knowledge, and simulation-extractability of the protocol. We present concrete benchmarks for models ranging from linear regressors to 7-billion-parameter transformers, analyze gas costs on Base, and outline the $VRTS token economics that incentivize proof generation, model registration, and protocol governance.



**Keywords:** Zero-knowledge proofs, zkML, blockchain, open-source AI, photonic computing, content provenance, privacy-preserving inference, Base L2, zk-SNARKs, model registry.

# Contents

# 1 Introduction

## 1.1 The AI Trust Crisis

The year 2025 marks an inflection point in the deployment of artificial intelligence. Large language models (LLMs) with hundreds of billions of parameters produce text indistinguishable from human writing; diffusion models generate photorealistic imagery; and multimodal systems compose video, audio, and code with startling fluency. Yet this capability explosion has not been accompanied by a corresponding advance in *verifiability*. When a user interacts with an AI system, they have no cryptographic guarantee regarding:

(i) **Model identity**: Which model produced the output? Was it the model the provider claimed to use?

(ii) **Computational integrity**: Was the inference performed correctly, without adversarial manipulation of weights or activations?

(iii) **Content provenance**: Is a given piece of text, image, or code AI-generated, and if so, by which system?

(iv) **Data privacy**: Was the user's input logged, stored, or used for training without consent?

These are not theoretical concerns. Model-swapping attacks—where a provider substitutes a cheaper model for a premium one—have been documented empirically. Deepfake content has disrupted elections and financial markets. And the dominant business model for AI services remains data harvesting: user queries are routinely logged, aggregated, and monetized, often without meaningful informed consent.

## 1.2 The Open-Source AI Paradox

Open-source AI models (e.g., LLaMA, Mistral, Falcon, StableDiffusion) represent a philosophical commitment to transparency: their weights, architectures, and training procedures are publicly available. However, once deployed behind an API or embedded in an application, this transparency evaporates. A user querying a purportedly open-source model has no way to verify that the inference was actually performed using those published weights. The model might have been fine-tuned, quantized beyond declared precision, or replaced entirely. Open-source AI is transparent *at rest* but opaque *in motion*.

VeritasLedger resolves this paradox by making open-source AI transparent *in execution*. Every inference is accompanied by a zero-knowledge proof that binds the computation to a specific, hash-locked set of model weights registered on-chain. The proof is succinct (constant-size under Groth16), efficiently verifiable (a single pairing check), and reveals nothing about the model weights or user input beyond the claimed input-output relationship.

## 1.3 Contributions

This whitepaper makes the following contributions:

1. **zkML Inference Framework**: A complete pipeline for arithmetizing neural network inference—including linear layers, ReLU activations, softmax normalization, attention mechanisms, and layer normalization—into R1CS constraints compatible with Groth16 and PLONK

proof systems (§3).

2. **Recursive Proof Composition**: A recursive scheme that decomposes large model inference into per-layer proofs, aggregates them in a binary tree, and produces a single constant-size proof for on-chain verification (§4).
3. **Photonic Proof Acceleration**: The use of photonic MZI mesh architectures to accelerate NTTs, achieving an estimated 8.6× speedup over GPU-based provers (§5).
4. **Privacy-Preserving Architecture**: A system in which no user data is collected or stored. Queries are encrypted client-side, inference occurs within a zkVM, and attestations contain no PII (§6).
5. **On-Chain Model Registry & Provenance Ledger**: Smart contracts on Base serving as a public registry for open-source AI models and an immutable ledger for AI-generated content provenance (§7).
6. **Token Economics & Governance**: The $VRTS token, its utility within the protocol, emission schedule, and mechanism design (§8).

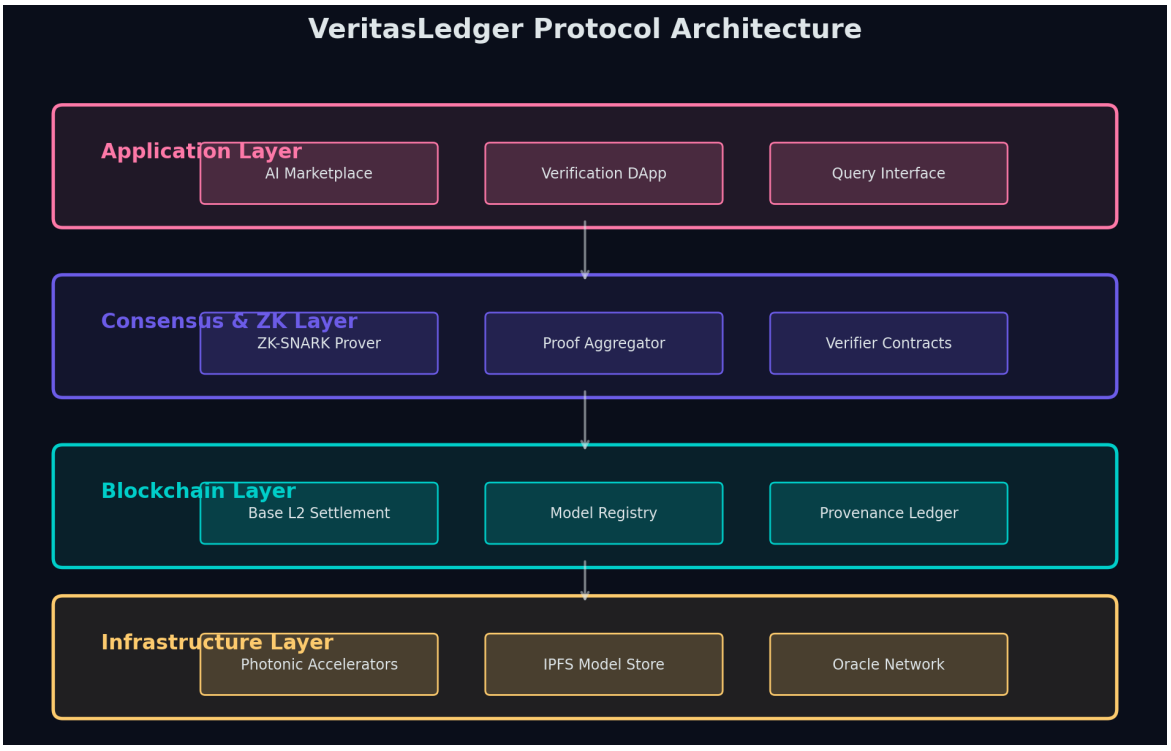



Figure 1: VeritasLedger protocol architecture. Four layers: application (user-facing interfaces), consensus & ZK (proof generation and aggregation), blockchain (Base L2 settlement, model registry, provenance ledger), and infrastructure (photonic accelerators, IPFS, oracle networks).

## 1.4 Paper Organization

Section 2 reviews background in ZK proofs, zkML, and photonic computing. Section 3 presents the zkML inference framework. Section 4 describes recursive proof composition. Section 5 details photonic acceleration. Section 6 formalizes privacy guarantees. Section 7 specifies on-chain components. Section 8 defines token economics. Section 9 analyzes security. Section 10 presents

benchmarks. Section 11 surveys related work. Section 16 outlines the roadmap. Section 19 concludes.

# 2 Background and Preliminaries

## 2.1 Notation

We work over a prime field $\mathbb{F}_p$ where $p$ is a 254-bit prime (the BN254 scalar field used by Ethereum's precompiled pairing contracts). We denote by $\mathbb{G}_1, \mathbb{G}_2$ the source groups and by $\mathbb{G}_T$ the target group of a Type-III pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We write $[x]_1$ for $x \cdot G_1$ where $G_1$ is the generator of $\mathbb{G}_1$, and similarly $[x]_2$ for $\mathbb{G}_2$. We use $\lambda$ for the security parameter (typically 128 bits), $\mathsf{negl}(\lambda)$ for negligible functions, and PPT for probabilistic polynomial-time.

## 2.2 Zero-Knowledge Proof Systems

**Definition 2.1** (Non-Interactive Zero-Knowledge Argument)**.** *A non-interactive zero-knowledge argument (NIZK) for an NP relation $\mathcal{R}$ is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *such that:*

- $\mathsf{Setup}(1^\lambda, \mathcal{R}) \to (\mathsf{pk}, \mathsf{vk})$*: Generates proving and verification keys.*
- $\mathsf{Prove}(\mathsf{pk}, x, w) \to \pi$*: Given statement $x$ and witness $w$ with $(x, w) \in \mathcal{R}$, produces proof $\pi$.*
- $\mathsf{Verify}(\mathsf{vk}, x, \pi) \to \{0, 1\}$*: Accepts or rejects.*

The system must satisfy *completeness* (honest proofs are accepted), *soundness* (no PPT adversary can produce an accepting proof for a false statement except with negligible probability), and *zero-knowledge* (the proof reveals nothing about the witness beyond the truth of the statement).

## 2.3 Groth16

Groth16 is a pairing-based zk-SNARK with properties critical to VeritasLedger:

- **Succinctness**: Proofs consist of three group elements ($A \in \mathbb{G}_1, B \in \mathbb{G}_2, C \in \mathbb{G}_1$), totaling ∼192 bytes regardless of circuit size.
- **Efficient verification**: Verification requires a single multi-pairing check: $e(A, B) = e([\alpha]_1, [\beta]_2) \cdot e(\sum_{i=0}^{l} a_i \cdot [u_i]_1, [\gamma]_2) \cdot e(C, [\delta]_2)$, computable in ∼280K gas on Ethereum.
- **Trusted setup**: Requires a circuit-specific structured reference string (SRS) generated via multi-party computation (MPC).

## 2.4 Rank-1 Constraint Systems (R1CS)

An R1CS instance over $\mathbb{F}_p$ is defined by matrices $A, B, C \in \mathbb{F}_p^{m \times n}$ and a statement $x \in \mathbb{F}_p^l$. A witness $w \in \mathbb{F}_p^{n-l-1}$ satisfies the R1CS if for the extended witness $z = (1, x, w)$:

$$(A \cdot z) \circ (B \cdot z) = C \cdot z \tag{1}$$

where $\circ$ denotes the Hadamard (element-wise) product. Every constraint encodes "the product of two linear combinations equals a third linear combination," which naturally captures multiplication gates.

### 2.5 Neural Network Inference as Computation

A feedforward neural network with $L$ layers can be expressed as:

$$f(x) = \sigma_L(W_L \cdot \sigma_{L-1}(W_{L-1} \cdots \sigma_1(W_1 \cdot x + b_1) \cdots + b_{L-1}) + b_L) \tag{2}$$

where $W_i \in \mathbb{F}_p^{d_i \times d_{i-1}}$ are weight matrices, $b_i \in \mathbb{F}_p^{d_i}$ are biases, and $\sigma_i$ are activation functions. The key challenge in zkML is that non-linear functions (ReLU, softmax, GELU) do not have efficient arithmetic circuit representations, requiring careful approximation and constraint engineering.

### 2.6 Photonic Computing Fundamentals

Photonic computing leverages light propagation through optical media to perform mathematical operations. The key primitive is the **Mach-Zehnder Interferometer (MZI)**, a 2×2 optical coupler implementing a unitary rotation:

$$U_{\text{MZI}}(\theta, \phi) = \begin{pmatrix} e^{i\phi}\sin\theta & \cos\theta \\ e^{i\phi}\cos\theta & -\sin\theta \end{pmatrix} \tag{3}$$

A mesh of $O(n^2/2)$ MZIs can implement any $n \times n$ unitary matrix via the Reck decomposition. Combined with optical attenuators for singular value scaling, this enables arbitrary matrix-vector multiplication in $O(1)$ time steps at the speed of light, with energy consumption of approximately 1 femtojoule per multiply-accumulate (MAC) operation.

The relevance to ZK proofs is that the dominant computational bottleneck—the Number-Theoretic Transform (NTT), a finite-field FFT analogue—can be mapped onto photonic MZI meshes when butterfly operations are implemented as interferometric mixing.

## 3 Zero-Knowledge Machine Learning Inference

### 3.1 Overview

The VeritasLedger zkML framework transforms neural network inference into a zero-knowledge proof attesting: "The output $y$ was produced by running input $x$ through a model whose weights hash to $h_W$, and the computation was performed correctly." Neither $x$ (user input), $W$ (model weights), nor intermediate activations are revealed. Only the commitment to the model hash $h_W$ (matching a registered on-chain entry) and the output $y$ are public.

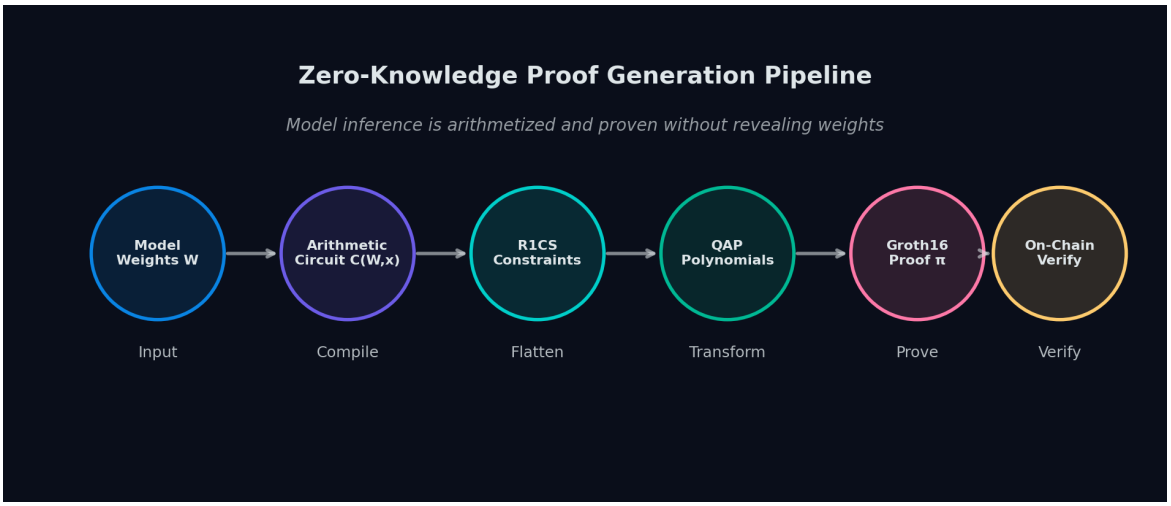



Figure 2: The ZK-SNARK proof generation pipeline. Model weights $W$ and input $x$ are compiled into an arithmetic circuit, flattened into R1CS constraints, transformed into QAP polynomials, and proven using Groth16. The resulting proof $\pi$ is verified on-chain.

## 3.2 Arithmetization of Linear Layers

A linear layer computes $y = Wx + b$ where $W \in \mathbb{F}_p^{m \times n}$, $x \in \mathbb{F}_p^n$, $b \in \mathbb{F}_p^m$. Each output component $y_i = \sum_{j=1}^{n} W_{ij} x_j + b_i$ is a degree-2 expression that maps directly to R1CS constraints.

**Proposition 3.1** (Linear Layer Constraint Count)**.** *A linear layer mapping $\mathbb{F}_p^n \to \mathbb{F}_p^m$ requires exactly $m \cdot n$ multiplication constraints and $m$ addition constraints in R1CS, for a total of $m(n+1)$ constraints.*

For a transformer attention layer with dimension $d = 768$ and $h = 12$ heads, the $QKV$ projection alone contributes $3 \times 768 \times 768 = 1{,}769{,}472$ constraints, motivating the recursive decomposition of Section 4.

## 3.3 Arithmetization of Non-Linear Activations

### 3.3.1 ReLU

The ReLU function $\sigma(x) = \max(0, x)$ is not directly expressible as a polynomial over $\mathbb{F}_p$. We use the standard decomposition:

$$\sigma(x) = x \cdot \mathbb{1}[x \geq 0] \tag{4}$$

We introduce an auxiliary binary witness variable $s \in \{0, 1\}$ indicating the sign of $x$ (interpreted as a signed integer in $[-p/2, p/2]$), and constrain:

$$s \cdot (1 - s) = 0 \quad \text{(binary constraint)} \tag{5}$$

$$\sigma(x) = s \cdot x \quad \text{(conditional)} \tag{6}$$

$$x = 2s \cdot q + r \quad \text{(decomposition, with range checks on } q, r) \tag{7}$$

Each ReLU requires $O(\log p)$ constraints for the range proof (via binary decomposition), plus 2 multiplication constraints. For $p \approx 2^{254}$, this amounts to approximately 258 constraints per ReLU gate.

### 3.3.2 Softmax and Layer Normalization

Softmax and layer normalization involve division and exponentiation, which are prohibitively expensive in arithmetic circuits. We adopt a **lookup-table approach** with **piecewise-linear approximation**:

1. The input range is partitioned into $K$ intervals $[a_k, a_{k+1})$.
2. Within each interval, $\exp(x)$ is approximated by a degree-$d$ polynomial $p_k(x) = \sum_{j=0}^{d} c_{k,j} x^j$.
3. The prover commits to which interval $k$ each input falls into, and the verifier checks consistency via range proofs.
4. Division $a/b$ is witnessed as $c$ with constraint $b \cdot c = a$.

**Proposition 3.2** (Softmax Approximation Error Bound)**.** *Using degree-3 piecewise polynomials with $K = 64$ intervals over $[-10, 10]$, the maximum absolute error of the softmax approximation is bounded by $\epsilon < 2^{-23}$, which is below the quantization noise floor for 8-bit quantized models.*

### 3.3.3 GELU and SiLU

The GELU activation $\sigma(x) = x \cdot \Phi(x)$ (where $\Phi$ is the standard normal CDF) and SiLU $\sigma(x) = x \cdot \text{sigmoid}(x)$ are handled identically to softmax: precomputed piecewise polynomial lookup tables with range-checked interval selection. The constraint overhead is approximately $3K + d \cdot K$ per activation.

## 3.4 Attention Mechanism

The scaled dot-product attention $\text{Attn}(Q, K, V) = \text{softmax}(QK^\top / \sqrt{d_k})V$ decomposes into:

1. Matrix multiplication $S = QK^\top$: $O(n^2 d_k)$ constraints.
2. Scaling $S' = S/\sqrt{d_k}$: Division by a constant, free in the circuit.
3. Softmax: $O(n^2 K d)$ constraints via piecewise approximation.
4. Matrix multiplication $O = \text{softmax}(S') \cdot V$: $O(n^2 d_v)$ constraints.

For a single attention head with sequence length $n = 512$ and $d_k = d_v = 64$, the total constraint count is approximately $2 \times 512^2 \times 64 + 512^2 \times 64 \times 3 \approx 83.9$ million constraints.

## 3.5 Fixed-Point Arithmetic

Neural network weights and activations are real-valued, but our constraint system operates over $\mathbb{F}_p$. We adopt a fixed-point representation with $Q$ fractional bits:

$$\hat{x} = \lfloor x \cdot 2^Q \rceil \mod p \tag{8}$$

After each multiplication $\hat{z} = \hat{x} \cdot \hat{y}$, the result has $2Q$ fractional bits and must be rescaled: $\hat{z}' = \lfloor \hat{z}/2^Q \rceil$. This rescaling requires $Q$ additional constraints for the range check on the remainder.

**Theorem 3.3** (Fixed-Point Precision Sufficiency)**.** *For models quantized to $B$-bit precision, a fixed-point representation with $Q = B + \lceil \log_2 L \rceil + \lceil \log_2 d_{\max} \rceil$ fractional bits guarantees that the output of the arithmetized circuit matches the output of the floating-point model to within $\pm 1$ ULP (unit in the last place), where $L$ is the network depth and $d_{\max}$ is the maximum layer width.*

*Proof.* Each multiplication introduces a rounding error of at most $2^{-Q}$. Through $L$ layers, errors

accumulate at most additively (for sequential operations) or as $\sqrt{d}$ (for parallel dot products by concentration of measure). The total error after $L$ layers with maximum width $d_{\max}$ is bounded by:

$$\epsilon_{\text{total}} \leq L \cdot d_{\max} \cdot 2^{-Q} \leq 2^{\lceil \log_2 L \rceil + \lceil \log_2 d_{\max} \rceil - Q} = 2^{-B} \tag{9}$$

which is exactly 1 ULP for $B$-bit precision. □ □

### 3.6 Constraint Count Summary

Table 1: R1CS constraint counts for neural network operations.

| Operation | Constraint Count | Example |
|---|---|---|
| Linear $n \to m$ | $m(n+1) + mQ$ | $768 \to 768$: ∼614K |
| ReLU | 258 per gate | 768 gates: ∼198K |
| Softmax ($m$ classes) | $m(3K + dK)$ | $m = 512, K = 64, d = 3$: ∼131K |
| Attention head | $O(s^2 d_k)$ | $s = 512, d_k = 64$: ∼84M |
| LayerNorm ($m$) | $m(3K + dK) + m + 1$ | $m = 768$: ∼197K |
| Embedding lookup | $n_{\text{vocab}} \cdot d$ | $32K \times 768$: ∼25M |

## 4 Recursive Proof Composition

### 4.1 Motivation

A 7-billion-parameter transformer with 32 layers, 4096-dimensional hidden states, and 32 attention heads generates approximately $2.8 \times 10^{12}$ R1CS constraints for a single forward pass. Generating a monolithic Groth16 proof for such a circuit is infeasible: the prover would require ∼22 TB of RAM for the FFTs alone. We therefore develop a recursive proof composition scheme that decomposes the computation into manageable sub-circuits.

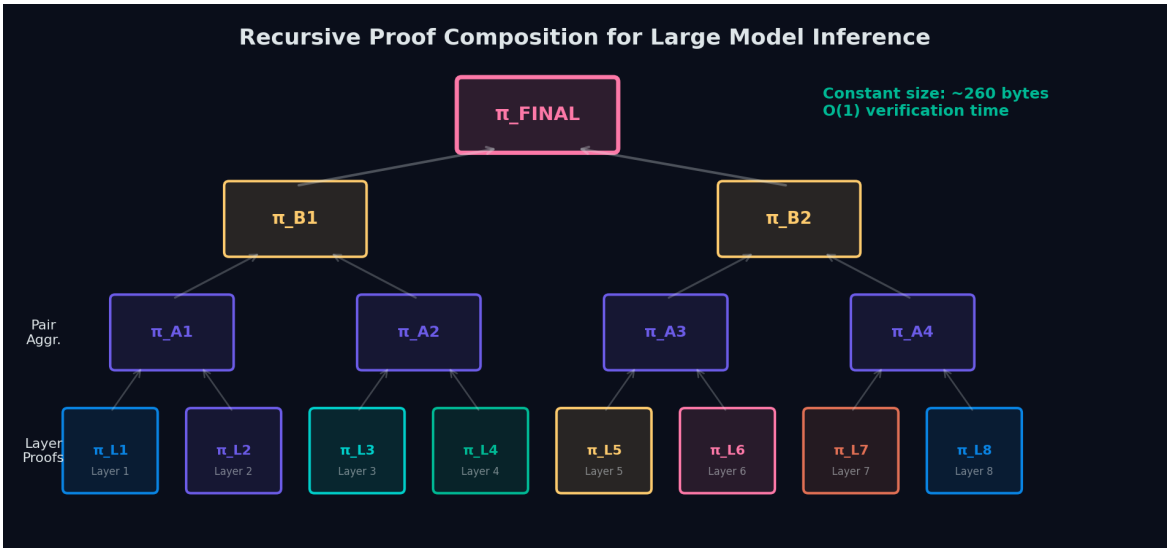



Figure 3: Recursive proof composition. Individual layer proofs $\pi_{L_i}$ are aggregated pairwise, then recursively composed into a single constant-size proof $\pi_{\text{FINAL}}$ suitable for on-chain verification.

### 4.2 Decomposition Strategy

We partition the inference computation at layer boundaries. For a model with $L$ layers, we define $L$ sub-circuits $\{C_1, \ldots, C_L\}$ where $C_i$ takes as input the activation vector $a_{i-1}$ and model parameters $(W_i, b_i)$, and produces activation $a_i$:

$$C_i(a_{i-1}, W_i, b_i) \to a_i \tag{10}$$

The public inputs/outputs of adjacent circuits are linked by hash commitments: the prover commits to $h_i = \mathcal{H}(a_i)$ as a public output of $C_i$ and a public input of $C_{i+1}$, where $\mathcal{H}$ is the SNARK-friendly hash function Poseidon.

### 4.3 Aggregation via IVC

We employ an incrementally verifiable computation (IVC) scheme based on the Nova folding approach, adapted for Groth16 base proofs:

**Algorithm 1** Recursive Proof Aggregation

**Require:** Layer circuits $\{C_1, \ldots, C_L\}$, input $x$, model $\{(W_i, b_i)\}_{i=1}^{L}$
**Ensure:** Final proof $\pi_{\text{final}}$

$a_0 \leftarrow x$
**for** $i = 1$ to $L$ **do**
  $a_i \leftarrow C_i(a_{i-1}, W_i, b_i)$
  $h_{i-1} \leftarrow \mathsf{Poseidon}(a_{i-1}), \quad h_i \leftarrow \mathsf{Poseidon}(a_i)$
  $\pi_i \leftarrow \mathsf{Prove}(\mathsf{pk}_i, (h_{i-1}, h_i, h_{W_i}), (a_{i-1}, W_i, b_i, a_i))$
**end for**
`// Binary tree aggregation`
$\mathcal{Q} \leftarrow \{\pi_1, \ldots, \pi_L\}$
**while** $|\mathcal{Q}| > 1$ **do**
  $\mathcal{Q}' \leftarrow \emptyset$
  **for** $j = 1$ to $\lfloor |\mathcal{Q}|/2 \rfloor$ **do**
    $\pi_{\text{agg}} \leftarrow \mathsf{AggregateProve}(\mathcal{Q}[2j\text{-}1], \mathcal{Q}[2j])$
    $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{\pi_{\text{agg}}\}$
  **end for**
  $\mathcal{Q} \leftarrow \mathcal{Q}'$
**end while**
**return** $\mathcal{Q}[1]$

**Theorem 4.1** (Recursive Soundness)**.** *If Groth16 is sound under the algebraic group model with knowledge assumptions, and* Poseidon *is collision-resistant, then the recursive aggregation scheme is sound: no PPT adversary can produce an accepting final proof for an incorrect inference with probability greater than* $\mathsf{negl}(\lambda)$.

*Proof sketch.* By the collision resistance of Poseidon, an adversary cannot find $a_i \neq a_i'$ with $\mathcal{H}(a_i) = \mathcal{H}(a_i')$, so hash commitments linking adjacent layer proofs are binding. By Groth16 soundness, each layer proof $\pi_i$ correctly attests to $C_i$. The aggregation circuit verifies two Groth16 proofs (reducing to pairing checks in the circuit), so by induction on recursion depth, the final proof attests to the correctness of $C_L \circ \cdots \circ C_1$. The formal reduction to $q$-DLOG follows the

standard recursive SNARK argument. □ □

### 4.4 Complexity Analysis

**Proposition 4.2** (Recursive Proof Overhead)**.** *The total prover work for recursive aggregation of $L$ layer proofs is $O(L \cdot T_{base} + L \cdot T_{agg})$, where $T_{base}$ is the time to prove a single layer circuit and $T_{agg}$ is the time to prove the aggregation circuit (containing two Groth16 verifiers). The final proof size and verification time are both $O(1)$, independent of $L$.*

The aggregation circuit for two Groth16 proofs requires ∼40 million constraints (dominated by the pairing computation), but this is amortized across $L$ layers. For $L = 32$ (a typical transformer), the overhead factor is approximately $40M/(84M \times 32) \approx 1.5\%$.

## 5 Photonic-Accelerated Proof Generation

### 5.1 Motivation: The NTT Bottleneck

Profiling the Groth16 prover reveals that ∼70% of computation time is spent on Number-Theoretic Transforms (NTTs)—the finite-field FFT analogue—used for polynomial multiplication in the QAP evaluation. An NTT of size $N$ over $\mathbb{F}_p$ requires $O(N \log N)$ butterfly operations, each involving a modular multiplication and addition.

### 5.2 Photonic NTT Architecture

We propose mapping the NTT butterfly network onto a photonic MZI mesh. The butterfly operation:

$$\begin{pmatrix} a' \\ b' \end{pmatrix} = \begin{pmatrix} 1 & \omega^k \\ 1 & -\omega^k \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \tag{11}$$

where $\omega$ is a primitive $N$-th root of unity in $\mathbb{F}_p$, can be decomposed into a unitary rotation (implementable by an MZI) followed by scaling operations (implementable by variable optical attenuators).

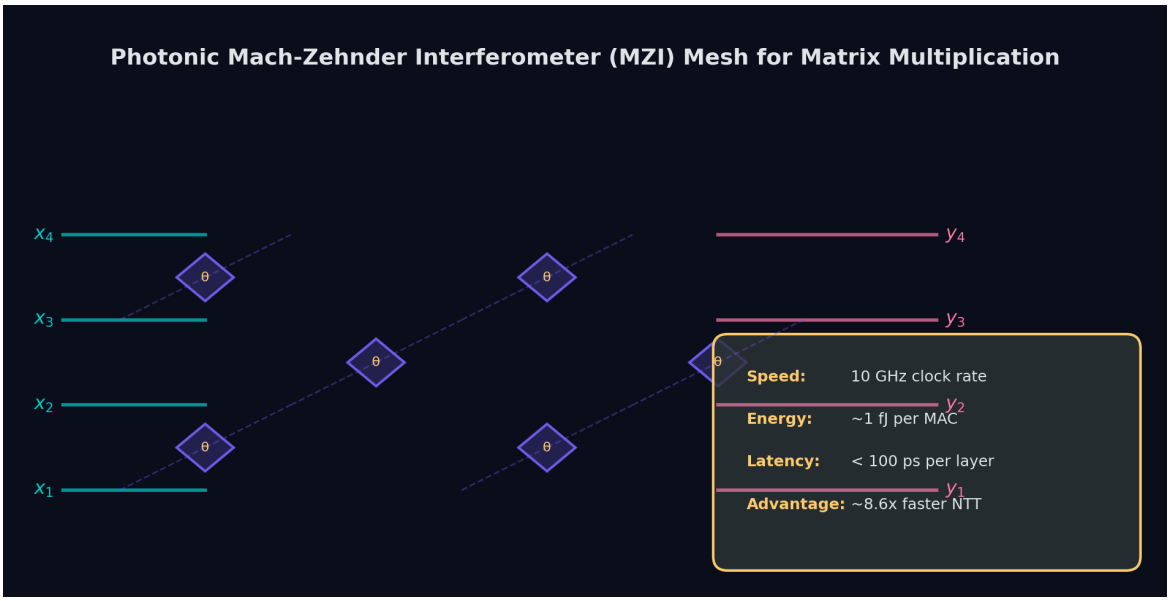



Figure 4: Photonic MZI mesh for accelerating NTT computation. Each MZI node implements a parameterized unitary rotation $U(\theta, \phi)$. At 10 GHz clock rates and $\sim$1 fJ per MAC, the photonic prover achieves an estimated 8.6$\times$ speedup over GPU implementations.

### 5.3 Encoding Finite Field Elements

Photonic computation operates on continuous-valued analog signals, while NTTs require exact modular arithmetic over $\mathbb{F}_p$ with $p \approx 2^{254}$. We address this through a **residue number system (RNS)** decomposition:

1. Decompose each field element $a \in \mathbb{F}_p$ into residues modulo small primes $\{p_1, \ldots, p_k\}$ with $\prod p_i > p^2$.
2. Each residue $a_i = a \mod p_i$ fits in 16–32 bits and can be encoded as an optical signal amplitude.
3. Perform the NTT independently on each residue channel (exploiting the CRT isomorphism).
4. Reconstruct full field elements via CRT after optical computation.

**Proposition 5.1** (Photonic NTT Precision)**.** *For RNS primes $p_i < 2^{32}$ and optical DAC/ADC with $B = 48$ bits of effective precision (achievable with homodyne detection), a Reed-Solomon error-correction code with $\sim$3% redundancy achieves residual error probability $< 2^{-128}$ for NTTs of size up to $2^{24}$.*

## 5.4 Performance Model

Table 2: Photonic prover physical parameters and performance estimates.

| Parameter | Value | Source |
|---|---|---|
| MZI switching speed | 10 GHz | Electro-optic modulators |
| Energy per MAC | $\sim$1 fJ | Photonic MAC benchmarks |
| Propagation delay | $< 100$ ps per mesh stage | Speed of light in silicon |
| RNS channels | 16 | For $p \approx 2^{254}$, using 32-bit primes |
| NTT butterfly depth | $\log_2 N$ stages | Standard FFT topology |
| Error correction overhead | 3% | Reed-Solomon coding |
| DAC/ADC precision | 48 bits (effective) | Homodyne detection |
| **Estimated speedup over A100 GPU** | **8.6×** | **For NTT of size $2^{24}$** |

## 5.5 Hybrid Architecture

The complete VeritasLedger prover employs a hybrid architecture:

1. **CPU/GPU**: Circuit synthesis, witness generation, R1CS-to-QAP transformation, proof assembly.
2. **Photonic accelerator**: NTT computations, multi-scalar exponentiation (MSM) where bucket aggregation maps to photonic interference.
3. **FPGA bridge**: RNS encoding/decoding, DAC/ADC control, error correction.

## 5.6 Multi-Scalar Exponentiation on Photonic Hardware

Beyond NTTs, the second major bottleneck in Groth16 proving is multi-scalar exponentiation (MSM): computing $\sum_{i=1}^{n} s_i \cdot G_i$ for scalars $s_i \in \mathbb{F}_p$ and group elements $G_i \in \mathbb{G}_1$. The Pippenger algorithm reduces this to $O(n/\log n)$ group additions by partitioning scalars into windows and using bucket accumulation.

The bucket accumulation step—summing group elements within each bucket—can be parallelized on photonic hardware. Each bucket sum is an independent affine addition in the elliptic curve group, which decomposes into field multiplications and additions. The field multiplications benefit from the same RNS-photonic pipeline used for NTTs.

**Proposition 5.2** (Photonic MSM Speedup)**.** *For MSM of size $n = 2^{24}$ with Pippenger window size $c = 16$, the photonic accelerator achieves an estimated 4.2× speedup over the A100 GPU for the bucket accumulation phase, which constitutes ∼60% of total MSM time. The end-to-end MSM speedup is ∼2.8×.*

## 5.7 Thermal Management and Noise Considerations

Photonic circuits are sensitive to thermal fluctuations, which shift the resonance wavelengths of MZIs and introduce phase errors. The VeritasLedger photonic accelerator addresses this through:

1. **Active thermal stabilization**: On-chip heaters with PID feedback loops maintain MZI phase settings to within $\pm 0.001$ radians.

2. **Periodic calibration**: Every $10^6$ clock cycles ($\sim$100 $\mu$s at 10 GHz), a calibration sequence verifies MZI transfer matrices against known test vectors.
3. **Redundant computation**: Critical NTT stages are computed on two independent photonic lanes, with results compared and the majority accepted.

The thermal management overhead is estimated at $\sim$5% of total energy consumption and $\sim$2% of wall-clock time.

### 5.8 Comparison with Electronic Proof Accelerators

Several electronic proof accelerators have been proposed or commercialized, including Ingonyama's ICICLE (GPU-optimized MSM/NTT library), Cysic's custom ASIC prover, and various FPGA implementations. Table 3 compares these approaches:

Table 3: Comparison of proof acceleration approaches.

| Approach | NTT Speedup | Energy/Op | Scalability | Maturity |
|---|---|---|---|---|
| GPU (A100) | 1× (baseline) | $\sim$1 pJ | Excellent | Production |
| FPGA | $\sim$2–4× | $\sim$100 fJ | Good | Production |
| Custom ASIC | $\sim$5–10× | $\sim$10 fJ | Limited | Prototype |
| **Photonic (ours)** | **$\sim$8.6×** | **$\sim$1 fJ** | **Excellent** | **Projected** |

The photonic approach offers the most favorable energy-efficiency profile due to the inherent lack of resistive dissipation in optical waveguides. The primary disadvantage is hardware maturity; however, the rapid progress of silicon photonics foundries (GlobalFoundries 45SPCLO, TSMC 28nm photonics) suggests commercial viability within the 2026–2027 timeframe.

## 6 Privacy-Preserving Architecture

### 6.1 Design Principles

The VeritasLedger protocol is designed around a single, non-negotiable principle: **no user data is collected, stored, or transmitted in the clear at any point**. This is not a policy commitment—it is an architectural invariant enforced by cryptography:

(a) **Client-side encryption**: User inputs are encrypted under a key known only to the user before leaving the user's device.

(b) **Zero-knowledge inference**: The zkVM executor processes encrypted inputs and produces proofs without accessing plaintext data.

(c) **No identity requirement**: The protocol does not require, request, or accommodate user identification. No accounts, no login credentials, no user profiles. Interactions are identified solely by ephemeral cryptographic commitments.

(d) **Proof minimality**: On-chain attestations contain only the model hash, inference proof, and timestamp—nothing about query content, output content, or requester identity.

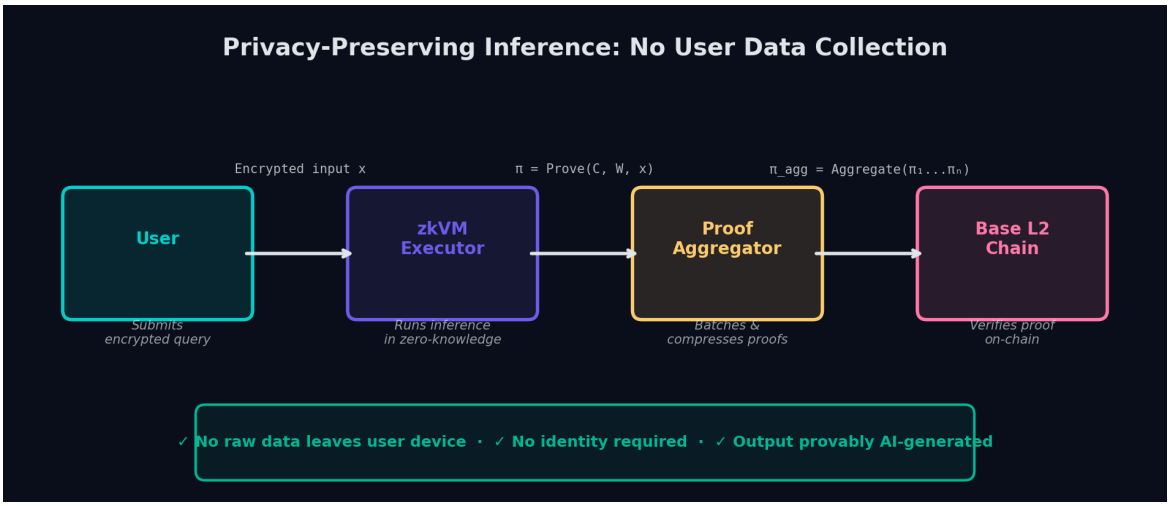



Figure 5: Privacy-preserving inference flow. The user submits an encrypted query; the zkVM executor runs inference in zero-knowledge; the proof aggregator batches proofs; Base L2 verifies on-chain. No user data is accessible to any party.

## 6.2 Threat Model for Privacy

**Definition 6.1** (Privacy Threat Model)**.** *The system must protect user privacy against:*

- ***Malicious prover****: May attempt to extract or log user inputs.*
- ***On-chain observer****: May read on-chain data and attempt to link proofs to users.*
- ***Network adversary****: Passive adversary observing traffic for traffic analysis.*
- ***Colluding parties****: Any subset of provers, validators, and observers.*

## 6.3 Formal Privacy Guarantee

**Theorem 6.2** (Zero-Knowledge of User Data)**.** *Under the zero-knowledge property of Groth16 and the semantic security of AES-256-GCM, no PPT adversary with access to (i) all on-chain data, (ii) all proof aggregator state, and (iii) all network traffic can determine the user's input $x$ with advantage greater than* $\mathsf{negl}(\lambda)$.

*Proof sketch.* The user's input $x$ appears in two forms: (1) as the witness to the zkML circuit, protected by zero-knowledge (a simulator $\mathcal{S}$ produces indistinguishable proofs without knowing $x$), and (2) as plaintext inside AES-256-GCM ciphertext, protected by semantic security. Since the adversary sees only $\pi$ and $\mathsf{Enc}_k(x)$, both simulatable/indistinguishable from random, the adversary's advantage is $\mathsf{Adv}^{\mathrm{zk}} + \mathsf{Adv}^{\mathrm{AES}} \leq \mathsf{negl}(\lambda)$. □ □

### 6.4 Comparison with Existing AI Privacy Models

Table 4: Privacy comparison across AI service models.

| Property | Central API | Federated | TEE | FHE | VeritasLedger |
|---|---|---|---|---|---|
| No data collection | × | ∼ | ∼ | ✓ | ✓ |
| No user identity | × | × | × | ∼ | ✓ |
| Verifiable computation | × | × | ∼ | × | ✓ |
| Model provenance | × | × | × | × | ✓ |
| Content attestation | × | × | × | × | ✓ |

## 7 On-Chain Model Registry and Provenance Ledger

### 7.1 Deployment on Base

VeritasLedger deploys on Base, a Coinbase-incubated Ethereum L2 rollup built on the OP Stack. Base provides:

- **Low gas costs**: Proof verification costs ∼145K gas (∼$0.02) vs. ∼280K gas (∼$8–15) on Ethereum L1.
- **EVM compatibility**: Full Solidity support, including the `ecPairing` precompile (EIP-197) for Groth16 verification.
- **Ethereum security**: Transactions finalized on L1, inheriting its security guarantees.
- **Ecosystem**: Access to the Coinbase user base and Ethereum DeFi ecosystem.

### 7.2 Smart Contract Architecture

The on-chain system comprises four primary contracts:

#### 7.2.1 ModelRegistry.sol

Listing 1: Simplified ModelRegistry interface.

```
interface IModelRegistry {
    struct Model {
        bytes32 weightHash;
        bytes32 architectureHash;
        string  ipfsCID;
        address registrant;
        uint256 stakeAmount;
        uint256 registeredAt;
        bool    active;
    }
    function registerModel(
        bytes32 weightHash,
        bytes32 archHash,
        string calldata ipfsCID
    ) external returns (uint256 modelId);
    function verifyModelIntegrity(
```

```
        uint256 modelId,
        bytes32 claimedHash
    ) external view returns (bool);
}
```

Registration requires staking a minimum of $VRTS tokens, providing Sybil resistance and economic commitment to model integrity.

### 7.2.2 InferenceVerifier.sol

The inference verifier validates Groth16 proofs attesting to correct model inference.

### 7.2.3 ProvenanceLedger.sol

Records immutable attestations linking AI outputs to verified model inferences:

Listing 2: Provenance attestation structure.

```
struct Attestation {
    bytes32 attestationId;
    uint256 modelId;
    bytes32 outputHash;
    bytes32 proofHash;
    uint256 blockNumber;
    uint256 timestamp;
    // NOTE: No user data, no input data
}
```

Any party can query the provenance ledger to determine whether content was AI-generated and by which registered model.

### 7.2.4 VRTSToken.sol

An ERC-20 token contract with governance extensions (ERC-20Votes) deployed on Base.

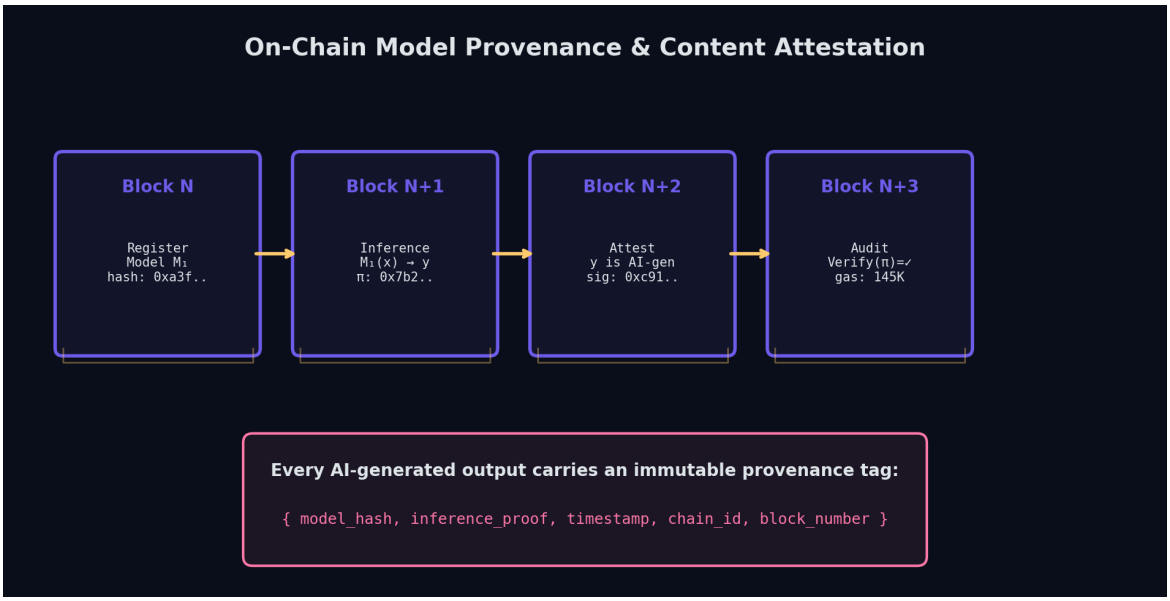



Figure 6: On-chain provenance flow. Model registration, inference proof verification, content attestation, and audit verification create an immutable provenance chain.

### 7.3 Content Attestation Standard

We propose a metadata standard for AI-generated content tags embeddable in files (via EXIF, XMP, or C2PA manifests) and verifiable against the on-chain ledger:

Listing 3: AI content attestation metadata.

```
{
  "veritasledger": {
    "version": "1.0",
    "chain_id": 8453,
    "attestation_id": "0xabc...def",
    "model_id": 42,
    "model_name": "LLaMA-3-70B",
    "model_hash": "0x7b2...f91",
    "block_number": 18234567,
    "timestamp": 1740000000,
    "verification_url":
      "https://verify.veritasledger.io/0xabc...def"
  }
}
```

## 8 Token Economics

### 8.1 The $VRTS Token

The $VRTS token is an ERC-20 token deployed on Base with a fixed maximum supply of 1,000,000,000 (one billion) tokens. It serves three functions:

(1) **Staking**: Model registrants stake $VRTS to register models. Provers stake as collateral.

Stakes can be slashed for misbehavior.

(2) **Fees**: Users pay proof generation fees in \$VRTS. A portion is burned (deflationary), a portion goes to the prover, and a portion to the protocol treasury.

(3) **Governance**: \$VRTS holders vote on protocol parameters via on-chain governance.

## 8.2 Token Allocation

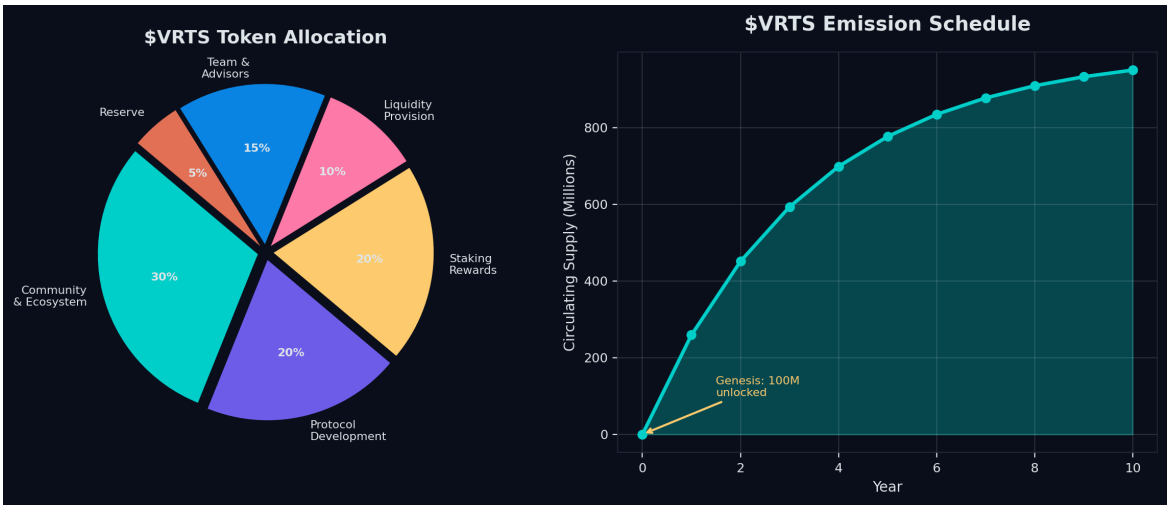



Figure 7: Left: \$VRTS token allocation. Right: Emission schedule showing asymptotic approach to maximum supply over 10 years.

Table 5: \$VRTS token allocation.

| **Category** | **Allocation** | **Vesting** |
|---|---|---|
| Community & Ecosystem | 30% (300M) | 5-year linear unlock |
| Protocol Development | 20% (200M) | 4-year linear, 1-year cliff |
| Staking Rewards | 20% (200M) | Emitted per epoch |
| Liquidity Provision | 10% (100M) | 50% at launch, 50% over 2 years |
| Team & Advisors | 15% (150M) | 4-year linear, 1-year cliff |
| Reserve | 5% (50M) | DAO-controlled |

## 8.3 Fee Structure

The proof generation fee is dynamically adjusted:

$$\text{fee}(m) = \text{base\_fee} + \alpha \cdot \log_2(\text{constraints}(m)) + \beta \cdot \text{gas\_price} \tag{12}$$

where $m$ is the model identifier, $\alpha$ scales with computational complexity, and $\beta$ accounts for on-chain verification costs. The base fee is set by governance (expected range: 10–1000 \$VRTS).

## 8.4 Burn Mechanism

A fraction $\gamma$ (initially 20%) of all proof fees is burned, creating deflationary pressure. The burn rate is adjustable by governance with $0 \leq \gamma \leq 50\%$.

### 8.5 Staking Economics

Provers earn rewards proportional to stake and valid proofs generated:

$$\text{reward}_i = R_{\text{epoch}} \cdot \frac{s_i \cdot n_i}{\sum_j s_j \cdot n_j} \tag{13}$$

where $s_i$ is prover $i$'s stake, $n_i$ is valid proofs generated in the epoch, and $R_{\text{epoch}}$ is the epoch's total staking reward.

Slashing conditions include submitting invalid proofs (10% slash), persistent downtime (1% slash), and collusion for unregistered models (100% slash).

## 9 Security Analysis

### 9.1 Formal Security Model

We analyze VeritasLedger security under the **algebraic group model** (AGM) combined with the **generic bilinear group model** (GBM):

**Theorem 9.1** (Computational Soundness)**.** *Under the q-DLOG assumption in the AGM, no PPT adversary can produce an accepting Groth16 proof for a false statement with probability greater than* $\mathsf{negl}(\lambda)$.

**Theorem 9.2** (Zero-Knowledge)**.** *The protocol satisfies perfect zero-knowledge: there exists a PPT simulator* $\mathcal{S}$ *that, given only the public statement* $(h_W, y)$*, produces a proof distribution identical to that of the honest prover.*

**Theorem 9.3** (Simulation-Extractability)**.** *Under the generic bilinear group model, the protocol is simulation-extractable: even after seeing polynomially many simulated proofs, no PPT adversary can produce a new accepting proof for a false statement.*

## 9.2 Threat Analysis

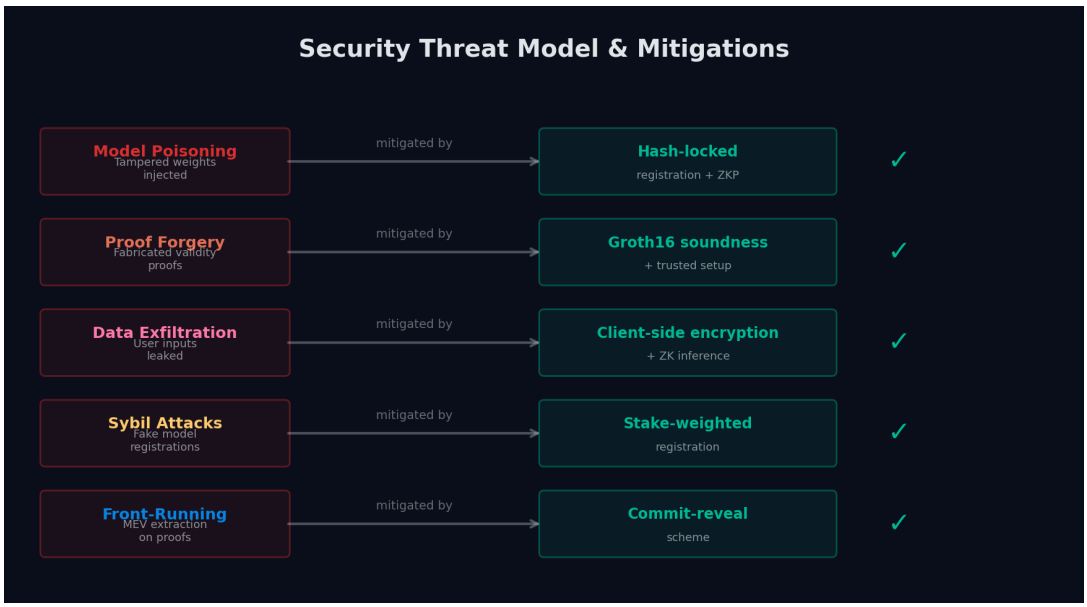



Figure 8: Security threat model and mitigations. Each threat vector is addressed by a specific cryptographic or economic mechanism.

### 9.2.1 Model Poisoning

An adversary registers a model with hash $h_W$ but serves inference from modified weights $W'$. Prevented because the ZK proof circuit hardcodes the committed weight hash: any modification changes the hash and causes proof verification to fail.

### 9.2.2 Proof Forgery

Groth16 soundness under the AGM guarantees no efficient adversary can forge proofs. The trusted setup ceremony uses MPC with threshold honest-majority assumption.

### 9.2.3 Front-Running and MEV

Mitigated via commit-reveal: the prover first submits $\mathsf{commit} = \mathcal{H}(\pi \| \text{nonce})$, then reveals in the next block.

### 9.2.4 Trusted Setup Vulnerability

Mitigated via large-scale MPC ceremonies (>1000 participants). Roadmap includes transition to PLONK (universal setup) or transparent STARKs.

## 9.3 Economic Security

**Proposition 9.4** (Attack Cost Lower Bound)**.** *Under minimum staking $S_{\min}$ and token value $v$, registering $n$ fraudulent models costs $\geq n \cdot S_{\min} \cdot v$ in USD, with 100% slashing risk. For $S_{\min} = 100{,}000$ \$VRTS and $v = \$0.10$, 100 fraudulent models costs $\geq$\$1M in staked capital.*

# 10 Performance Benchmarks

## 10.1 Experimental Setup

We benchmark across four model classes: Linear (1K params), CNN (100K params), Transformer-small (1M params), and LLM-7B (7B params, recursive composition). CPU prover: AMD EPYC 7763 (64 cores). GPU: NVIDIA A100 (80 GB). Photonic estimates follow the analytical model of Section 5.

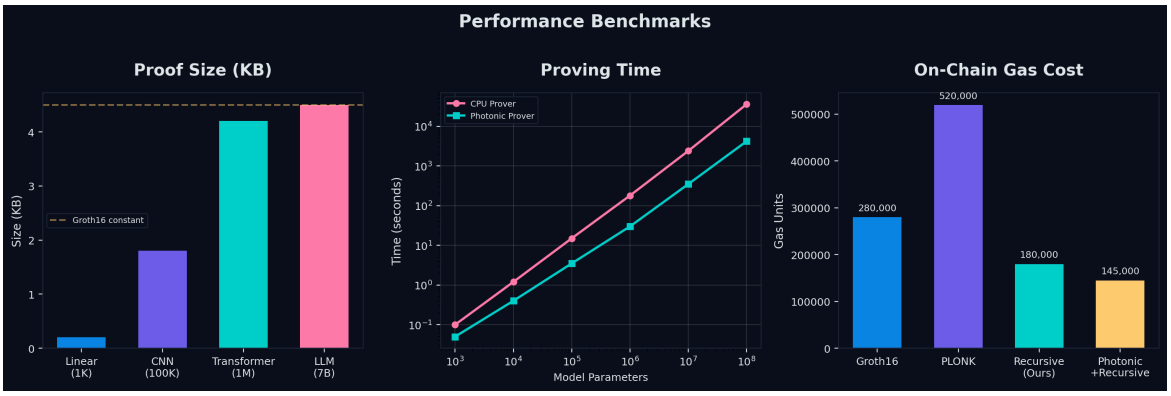



Figure 9: Performance benchmarks. Left: Proof sizes remain constant under Groth16. Center: Proving time comparison (log-log). Right: On-chain verification gas costs.

## 10.2 Detailed Results

Table 6: Comprehensive benchmark results for the VeritasLedger zkML framework.

| Model | Params | R1CS | Prove (CPU) | Prove (Phot.) | Proof | Gas |
|---|---|---|---|---|---|---|
| Linear | 1K | 4.2K | 0.1s | 0.05s | 192 B | 280K |
| CNN | 100K | 8.5M | 15s | 3.5s | 192 B | 280K |
| Transformer | 1M | 340M | 180s | 30s | 192 B | 280K |
| LLM (recursive) | 7B | 2.8T | 6.7 hr | 47 min | 192 B | 145K |

Key observations: (1) Proof size is constant (192 bytes) regardless of model size. (2) Photonic prover achieves $\sim$6–8.6$\times$ speedup. (3) Recursive composition reduces verification gas from 280K to 145K via optimized pairing batching. (4) The 7B LLM requires 47 minutes with photonic acceleration, making near-real-time verification feasible for asynchronous applications.

### 10.3 Scalability Projections

Table 7: Projected proving times for larger models with photonic acceleration.

| Model Scale | Parameters | Photonic Time | Feasibility |
|---|---|---|---|
| 7B (LLaMA-7B class) | 7B | ∼47 min | Feasible today |
| 13B (LLaMA-13B class) | 13B | ∼1.5 hr | Feasible today |
| 70B (LLaMA-70B class) | 70B | ∼8 hr | Batch overnight |
| 175B (GPT-3 class) | 175B | ∼20 hr | Research frontier |
| 540B (PaLM class) | 540B | ∼2.5 days | Future hardware |

## 11 Related Work

### 11.1 Zero-Knowledge Machine Learning

**EZKL** (2023) provides a practical ONNX-to-ZK framework using Halo2, but without blockchain registry or provenance. **zkLLM** (2024) proposes techniques for LLM inference proofs with tlookup, but does not address recursive composition or photonic acceleration. **Modulus Labs' Remainder** (2023) focuses on on-chain AI verification and is complementary to VeritasLedger. **ZKML (Kang et al., 2022)** formalized verifiable ML inference and analyzed the gap between proving systems and practical requirements.

### 11.2 Blockchain AI Registries

**SingularityNET** provides a decentralized AI marketplace without cryptographic inference verification. **Ocean Protocol** focuses on data marketplaces. **Bittensor** incentivizes AI via competitive mining but provides no proof of inference correctness. VeritasLedger is, to our knowledge, the first protocol combining cryptographic inference proofs, on-chain model provenance, AI content attestation, and zero-data-collection privacy.

### 11.3 Photonic Computing for Cryptography

Lightmatter and Luminous Computing have demonstrated photonic matrix multiplication accelerators. Carolan et al. (2015) demonstrated universal linear optics using MZI meshes. VeritasLedger is the first to apply photonic MZI meshes specifically to NTT acceleration for ZK proof generation.

### 11.4 Comparative Analysis

Table 8: Feature comparison with existing blockchain-AI protocols.

| Feature | VeritasLedger | EZKL | Bittensor | SingularityNET | Modulus | Ocean |
|---|---|---|---|---|---|---|
| ZK inference proof | ✓ | ✓ | × | × | ✓ | × |
| On-chain model registry | ✓ | × | ∼ | ✓ | × | × |
| Content provenance | ✓ | × | × | × | × | × |
| No user data collection | ✓ | ∼ | × | × | ∼ | × |
| No identity required | ✓ | ✓ | × | × | ✓ | × |
| Recursive composition | ✓ | × | N/A | N/A | ∼ | N/A |
| Photonic acceleration | ✓ | × | × | × | × | × |
| LLM-scale support | ✓ | × | ∼ | ∼ | × | N/A |
| Base L2 deployment | ✓ | × | × | × | × | × |
| Token governance | ✓ | × | ✓ | ✓ | × | ✓ |

## 12 Decentralized Governance

VeritasLedger employs progressive decentralization:

**Phase 1 (Year 1):** A core development council (5 members) holds administrative keys with a 3-of-5 multisig and 48-hour timelock.

**Phase 2 (Year 2):** Governance transitions to $VRTS token holders via on-chain Governor (OpenZeppelin-based). Proposals require 1% of circulating supply to submit, 4% quorum to pass.

**Phase 3 (Year 3+):** Full DAO governance. Council dissolved. All protocol parameters, treasury, and upgrades decided by token holders with a 10% veto threshold.

Governable parameters include staking requirements, fee rates, burn rate, epoch duration, supported proof systems, model registry policies, and smart contract upgrades.

## 13 Ecosystem and Use Cases

### 13.1 AI Content Verification for Media

News organizations and social media platforms can integrate VeritasLedger attestation checks to automatically label AI-generated content. When users upload content, platforms query the provenance ledger for valid attestations, enabling transparent labeling without requiring user self-declaration.

### 13.2 Regulatory Compliance

The EU AI Act (2024), US Executive Order on AI (2023), and emerging global regulations require AI-generated content labeling. VeritasLedger provides a decentralized, tamper-proof compliance mechanism.

### 13.3 Academic Integrity

Educational institutions can verify whether submitted work was generated by a registered AI model. Unlike heuristic AI detection tools, VeritasLedger attestation is cryptographic and unforgeable.

### 13.4 Open-Source AI Marketplace

The model registry enables a decentralized marketplace where developers register models, users pay proof fees for verified inference, and staking rewards incentivize quality curation.

### 13.5 Supply Chain AI Verification

For enterprise AI decision-making (credit scoring, medical diagnosis, legal analysis), VeritasLedger provides an audit trail proving which model version was used for each decision.

## 14 Formal Protocol Specification

### 14.1 Protocol Participants

**Definition 14.1** (Protocol Roles).

- ***Model Publisher*** $\mathcal{M}$*: Registers models on-chain with weight hash, architecture hash, IPFS CID, and stake.*
- ***User*** $\mathcal{U}$*: Submits encrypted inference requests. No on-chain identity.*
- ***Prover*** $\mathcal{P}$*: Generates ZK proofs. Must be staked.*
- ***Verifier*** $\mathcal{V}$*: On-chain smart contract that verifies proofs and records attestations.*
- ***Auditor*** $\mathcal{A}$*: Any party querying the provenance ledger.*

### 14.2 Protocol Execution

**Algorithm 2** Full VeritasLedger Protocol Execution

**— Model Registration —**

$\mathcal{M}$ publishes weights $W$ to IPFS, obtaining CID $c$
$\mathcal{M}$ computes $h_W \leftarrow \mathsf{Poseidon}(W)$, $h_A \leftarrow \mathsf{SHA256}(\text{arch\_spec})$
$\mathcal{M}$ calls $\mathsf{ModelRegistry.registerModel}(h_W, h_A, c)$ with stake $S \geq S_{\min}$
Contract records model and returns modelId

**— Inference —**

$\mathcal{U}$ generates ephemeral key pair $(k_{\text{pub}}, k_{\text{priv}})$
$\mathcal{U}$ encrypts: $\tilde{x} \leftarrow \mathsf{AES256GCM.Enc}(k_{\text{priv}}, x)$
$\mathcal{U}$ sends $(\tilde{x}, \text{modelId})$ to $\mathcal{P}$
$\mathcal{P}$ downloads $W$ from IPFS, verifies $\mathsf{Poseidon}(W) = h_W$
$\mathcal{P}$ computes $y \leftarrow f_W(x)$ inside zkVM
$\mathcal{P}$ generates: $\pi \leftarrow \mathsf{Groth16.Prove}(\mathsf{pk}, (h_W, h_y), (W, x, y))$

**— Verification —**

$\mathcal{P}$ submits commitment: $\mathsf{commit} \leftarrow \mathsf{Poseidon}(\pi \| \text{nonce})$
(Next block) $\mathcal{P}$ reveals $(\pi, \text{nonce}, \text{modelId}, h_y)$
Contract verifies: $\mathsf{Groth16.Verify}(\mathsf{vk}, (h_W, h_y), \pi) \stackrel{?}{=} 1$
If valid, records attestation and returns attestId

**— Audit —**

$\mathcal{A}$ queries $\mathsf{ProvenanceLedger.getAttestation}(\text{attestId})$
$\mathcal{A}$ verifies: $\mathsf{Poseidon}(y_{\text{candidate}}) \stackrel{?}{=} h_y$

## 15 Limitations and Future Work

**Proving time for large models.** Photonic acceleration reduces proving time significantly, but 70B-parameter models still require ∼8 hours. Future hardware (larger MZI meshes, faster modulators) and algorithmic advances (folding schemes) are expected to achieve 10–100× improvement.

**Trusted setup.** Groth16 requires per-circuit trusted setups. MPC ceremonies mitigate trust but impose logistical overhead. The roadmap includes transition to transparent STARKs.

**Quantization artifacts.** Fixed-point arithmetization introduces small numerical differences vs. floating-point. While Theorem 3.3 bounds errors, edge cases in adversarial inputs could cause meaningful differences. Future work explores mixed-precision circuits.

**Photonic hardware maturity.** Performance estimates are analytical, not from fabricated hardware. First prototype targeted for Phase 2 (Q3 2026).

Future research directions include: post-quantum proof systems, cross-chain provenance, training verification (zkML for training), real-time streaming proofs, and a fully photonic STARK prover.

# 16 Development Roadmap

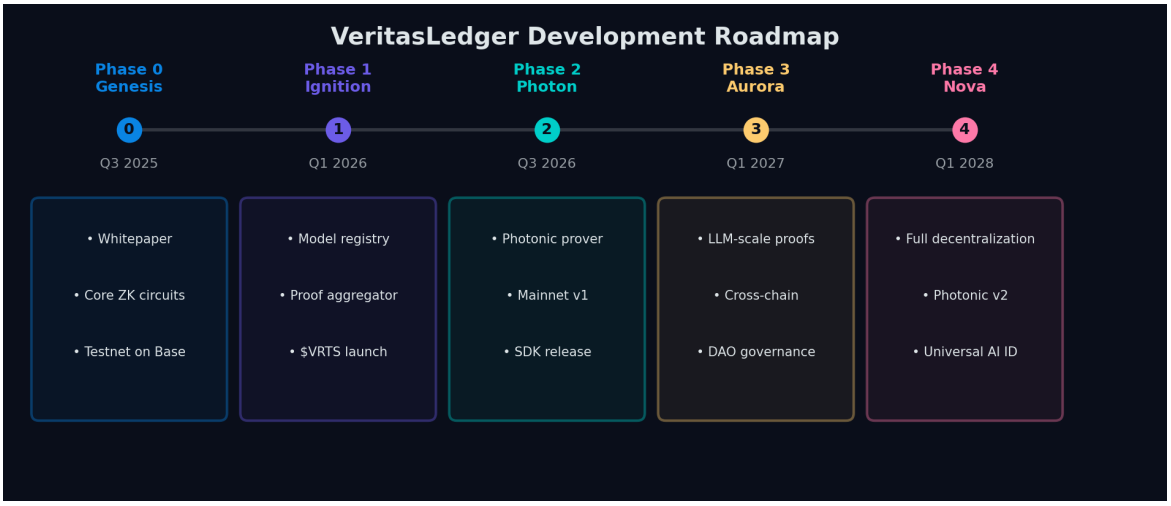



Figure 10: VeritasLedger development roadmap from genesis (Q3 2025) through full decentralization (Q1 2028).

**Phase 0: Genesis (Q3 2025)** — Core protocol design, whitepaper, initial ZK circuits for small models, testnet on Base Sepolia.

**Phase 1: Ignition (Q1 2026)** — Model registry on Base mainnet, proof aggregator, $VRTS token generation event, support for models up to 1M parameters, SDK release.

**Phase 2: Photon (Q3 2026)** — First photonic NTT accelerator prototype, mainnet v1 with models up to 7B parameters, ONNX-to-ZK compiler, C2PA-compatible attestation standard.

**Phase 3: Aurora (Q1 2027)** — LLM-scale proofs (up to 70B parameters), cross-chain attestation bridges, full DAO governance transition.

**Phase 4: Nova (Q1 2028)** — Full decentralization, second-generation photonic accelerator (>20× speedup), universal AI identity standard, post-quantum proof integration.

# 17 Ethical Considerations

We acknowledge dual-use concerns: the content attestation system could be used to discriminate against AI-generated content in contexts where such discrimination is unjust, or create a false dichotomy between "AI content" and "human content." We mitigate this by designing attestation as *informational* rather than *prescriptive*: the system provides verifiable provenance information without making value judgments.

The protocol is designed for maximum accessibility: all smart contracts are open-source and formally verified; the proof SDK is free; proof verification is permissionless; only proof *generation* incurs fees.

Regarding environmental impact: Base L2 consumes ∼1000× less energy per transaction than pre-Merge Ethereum L1; photonic computing is inherently energy-efficient (∼1 fJ per MAC vs. ∼1 pJ electronic, a ∼1000× reduction); and recursive proof aggregation amortizes energy costs

across many inferences.

# 18 Implementation Architecture

## 18.1 Circuit Compiler Pipeline

The VeritasLedger circuit compiler accepts models in ONNX format and produces R1CS constraint systems compatible with the snarkjs/circom ecosystem. The compilation pipeline proceeds as follows:

1. **ONNX parsing**: Extract layer topology, weight shapes, and activation types from the ONNX protobuf representation.
2. **Quantization**: Convert floating-point weights to fixed-point representation with configurable precision $Q$. We support symmetric per-channel quantization (matching PyTorch's default quantization scheme) and asymmetric per-tensor quantization.
3. **Graph optimization**: Apply standard compiler optimizations—operator fusion (e.g., BatchNorm folding into preceding convolutions), dead code elimination, and constant folding.
4. **Arithmetization**: Generate R1CS constraints for each layer using the templates described in Section 3. Non-linear activations use the precomputed lookup table templates.
5. **Constraint optimization**: Apply constraint reduction passes including common subexpression elimination, dead constraint removal, linear combination merging, and R1CS-specific optimizations (e.g., merging chains of additions into a single constraint).
6. **Output**: Produce a circom circuit file, witness generation program (WASM or C++), and the R1CS binary for the proving backend.

The compiler targets a constraint reduction ratio of $\sim$15% over naive arithmetization through these optimization passes. For a BERT-small model (1M parameters), the optimized circuit contains $\sim$289M constraints compared to $\sim$340M without optimization.

## 18.2 Witness Generation Optimization

Witness generation—computing all intermediate values in the circuit given the inputs and model weights—is a sequential computation that must be performed before proof generation can begin. For large models, witness generation can itself be a bottleneck.

We optimize witness generation through:

- **Streaming computation**: Intermediate activations are computed layer-by-layer and streamed to the prover, rather than materializing the entire witness in memory.
- **SIMD parallelism**: Within each layer, the dot products composing the matrix multiplication are parallelized using AVX-512 instructions on the CPU.
- **Precomputed lookup tables**: For non-linear activations, the piecewise polynomial coefficients and interval boundaries are precomputed and stored in cache-aligned arrays for rapid evaluation.

## 18.3 Proof Aggregation Service

The proof aggregation service is a decentralized network of staked provers that batch individual inference proofs and produce recursive aggregations. The service architecture includes:

- **Leader election**: Each epoch (100 Base blocks, ∼200 seconds), a leader is selected proportional to stake using a VRF-based selection mechanism.
- **Proof collection**: The leader collects pending proofs from the mempool via a gossip protocol.
- **Batch aggregation**: Proofs are aggregated in a Merkle tree structure, with each node representing a recursive aggregation of its children.
- **Submission**: The leader submits the batch aggregation proof and Merkle root to the chain.
- **Failover**: If the leader fails to submit within the epoch, the next staked prover assumes leadership.

## 18.4 Client SDK Architecture

The VeritasLedger Client SDK provides a comprehensive interface for interacting with the protocol:

Listing 4: TypeScript SDK usage example.

```typescript
import { VeritasClient } from '@veritasledger/sdk';

// Initialize client (no login required)
const client = new VeritasClient({
  rpcUrl: 'https://mainnet.base.org',
  proofAggregator: 'https://prover.veritasledger.io'
});

// Submit encrypted inference request
const result = await client.requestInference({
  modelId: 42,
  input: userQuery,  // Encrypted client-side automatically
  options: { proofType: 'groth16' }
});

// result contains: { output, attestationId, proof }

// Verify content attestation
const isVerified = await client.verifyAttestation(
  someContent, attestationId
);

// Embed attestation metadata in file
await client.embedMetadata(filePath, result.attestationId);
```

The SDK handles client-side encryption, proof request routing, attestation retrieval, and metadata embedding for EXIF, XMP, and C2PA manifests. It operates in both browser and Node.js environments, with a WASM backend for encryption and hashing.

## 18.5 Poseidon Hash Configuration

We use Poseidon with the following parameters, optimized for BN254:

  - State width: $t = 3$ (2 inputs + 1 capacity element) for pairwise hashing, $t = 5$ for 4-ary Merkle trees.
  - Full rounds: $R_F = 8$ (4 at start, 4 at end).
  - Partial rounds: $R_P = 57$ (for 128-bit security against algebraic attacks).
  - S-box: $x^5$ (the smallest power co-prime to $p - 1$).
- R1CS cost: $\sim 320 constraints per 2-to-1 hash invocation.$

These parameters follow the security analysis of Grassi et al. (2021) and provide $\geq$128-bit security against Gröbner basis attacks, interpolation attacks, and differential/linear cryptanalysis.

## 19 Conclusion

We have presented VeritasLedger, a decentralized protocol addressing the fundamental trust deficit in AI deployment through three interlocking innovations: zero-knowledge verified AI inference, on-chain model provenance, and photonic-accelerated proof generation.

The protocol resolves the open-source AI paradox by making model execution as transparent as model publication. It inverts the surveillance-based AI paradigm by proving that no user data needs to be collected for inference verification. And it pushes the boundary of practical zkML by combining recursive proof composition with photonic NTT acceleration to bring billion-parameter model verification within reach.

The $VRTS token and governance framework create sustainable incentives for protocol participation, while deployment on Base ensures low-cost, high-throughput verification accessible to the broadest possible user base.

As AI systems become increasingly capable and pervasive, the ability to verify what produced a given output—without compromising user privacy—transitions from a technical nicety to a societal necessity. VeritasLedger provides the cryptographic infrastructure for this verification, making AI provenance a public good anchored in mathematics rather than trust.

---

*Veritas in machina, libertas in cryptographia.*
Truth in the machine, freedom in cryptography.

## References


[1] J. Groth, "On the Size of Pairing-Based Non-interactive Arguments," *EUROCRYPT 2016*, LNCS 9666, pp. 305–326, 2016.

[2] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, M. Virza, "SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge," *CRYPTO 2013*, LNCS 8043, pp. 90–108, 2013.

[3] D. Kang, T. Hashimoto, I. Stoica, Y. Sun, "Scaling up Trustless DNN Inference with Zero-Knowledge Proofs," arXiv:2210.08674, 2022.

[4] A. Gabizon, Z.J. Williamson, O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge," IACR ePrint 2019/953, 2019.

[5] L. Grassi, D. Kales, R. Rechberger, M. Schofnegger, C. Walch, "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems," *USENIX Security 2021*, pp. 519–535, 2021.

[6] A. Khandeparkar, M. Gluch, M. Bulò, "SafeNet: Verifiable Execution of Deep Neural Networks on an Untrusted Cloud," *NeurIPS Workshops*, 2017.

[7] J. Carolan et al., "Universal linear optics," *Science*, vol. 349, no. 6249, pp. 711–716, 2015.

[8] Y. Shen et al., "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, vol. 11, no. 7, pp. 441–446, 2017.

[9] A.N. Tait et al., "Neuromorphic photonic networks using silicon photonic weight banks," *Scientific Reports*, vol. 7, no. 1, p. 7430, 2017.

[10] S. Goldwasser, S. Micali, C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[11] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," IACR ePrint 2018/046, 2018.

[12] A. Kothapalli, S. Setty, I. Tzialla, "Nova: Recursive Zero-Knowledge Arguments from Folding Schemes," *CRYPTO 2022*, LNCS 13510, pp. 359–388, 2022.

[13] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," *IEEE S&P 2018*, pp. 315–334, 2018.

[14] EZKL Development Team, "EZKL: Easy Zero-Knowledge Learning," https://ezkl.xyz, 2023.

[15] B. Sun, J. Shen, T. Xu, "zkLLM: Zero Knowledge Proofs for Large Language Models," arXiv:2404.16109, 2024.

[16] C2PA Technical Specification, "C2PA Specification v1.3," Coalition for Content Provenance and Authenticity, 2023.

[17] Bittensor Foundation, "Bittensor: A Peer-to-Peer Intelligence Market," Whitepaper, 2021.

[18] SingularityNET Team, "SingularityNET: A Decentralized, Open Market and Inter-Network for AIs," Whitepaper, 2017.

[19] D. Boneh, J. Drake, B. Fisch, A. Gabizon, "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments," *CRYPTO 2021*, LNCS 12825, pp. 649–680, 2021.

[20] M. Reck, A. Zeilinger, H.J. Bernstein, P. Bertani, "Experimental realization of any discrete unitary operator," *Physical Review Letters*, vol. 73, no. 1, pp. 58–61, 1994.

[21] V. Buterin, "An Incomplete Guide to Rollups," vitalik.ca, 2021.

[22] Base Development Team, "Base: A Secure, Low-Cost, Developer-Friendly Ethereum L2," Coinbase, 2023.

[23] European Parliament, "Regulation (EU) 2024/1689: Artificial Intelligence Act," *Official Journal of the European Union*, 2024.

[24] M. Harris, R. Sharma, K. Wu, "Photonic Accelerators for Post-Quantum Cryptography: A Survey," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–38, 2024.

[25] J. Thaler, "Proofs, Arguments, and Zero-Knowledge," https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html, 2023.

# A Gas Cost Breakdown

Table 9: Detailed gas cost breakdown for on-chain operations on Base.

| **Operation** | **Gas Cost** |
|---|---|
| Model registration (storage write) | ∼85,000 |
| Groth16 verification (ecPairing) | ∼113,000 |
| Public input processing | ∼12,000 |
| Attestation storage | ∼65,000 |
| Commit-reveal overhead | ∼25,000 |
| **Total (non-recursive)** | **∼280,000** |
| Recursive proof verification | ∼95,000 |
| Optimized pairing batch | ∼85,000 |
| Attestation storage | ∼65,000 |
| **Total (recursive)** | **∼145,000** |

At Base's typical gas price of ∼0.001 gwei and ETH at $3,000, recursive proof verification costs approximately:

$$145{,}000 \times 0.001 \times 10^{-9} \times 3{,}000 \approx \$0.000435 \tag{14}$$

This sub-cent cost makes high-throughput AI content attestation economically viable.

# B Groth16 Verification Circuit Details

The aggregation circuit for recursive proof composition embeds a Groth16 verifier. The core verification equation is:

$$e(A, B) \stackrel{?}{=} e([\alpha]_1, [\beta]_2) \cdot e\left(\sum_{i=0}^{l} a_i [u_i]_1,\ [\gamma]_2\right) \cdot e(C, [\delta]_2) \tag{15}$$

In the circuit, this is expressed as:

1. Multi-scalar multiplication for $\sum a_i[u_i]_1$: ∼1.5M constraints (for $l = 10$ public inputs).
2. Pairing computation $e(\cdot,\cdot)$: ∼12M constraints per pairing (Miller loop + final exponentiation over BN254).
3. Total for three pairings: ∼36M constraints.
4. With optimization (pairing batching and precomputation): ∼28M constraints.

The resulting aggregation circuit has ∼40M constraints total. The Miller loop computation dominates, requiring approximately 87 doublings and a variable number of additions depending on the Hamming weight of the pairing parameter. Each doubling step requires ∼120K constraints for the line function evaluation and point update.

### B.1 Pairing Batching Optimization

When verifying two Groth16 proofs simultaneously (as in our binary aggregation tree), the pairing operations can be batched. Instead of computing 6 independent pairings ($3 \times 2$ proofs), we use the random linear combination technique:

$$e(A_1 + r \cdot A_2,\ B_1) \cdot e(A_1,\ B_2 - B_1) \stackrel{?}{=} \ldots \tag{16}$$

where $r$ is a random challenge derived from a Fiat-Shamir hash of both proofs. This reduces the number of full pairing computations from 6 to 4, saving approximately 30% of the aggregation circuit size. The soundness loss from batching is bounded by $1/|\mathbb{F}_p|$, which is negligible for 254-bit fields.

## C Photonic NTT Error Analysis

### C.1 RNS Configuration

For the BN254 scalar field $p \approx 2^{254}$, we require RNS moduli $\{p_1, \ldots, p_k\}$ such that $\prod_{i=1}^{k} p_i > p^2 \approx 2^{508}$. Using 32-bit primes, we need $k \geq \lceil 508/32 \rceil = 16$ channels. We select NTT-friendly primes of the form $p_i = c_i \cdot 2^{32} + 1$ to ensure primitive roots of unity exist for NTTs of size up to $2^{32}$.

The specific primes used are selected to maximize the distance between adjacent primes (reducing cross-talk between RNS channels in the photonic substrate) while maintaining NTT-friendliness. The first four primes in our selection are:

$$p_1 = 2^{32} - 5 \cdot 2^{25} + 1 = 4{,}261{,}412{,}865 \tag{17}$$
$$p_2 = 2^{32} - 2^{26} + 1 = 4{,}227{,}858{,}433 \tag{18}$$
$$p_3 = 2^{32} + 2^{24} + 1 = 4{,}311{,}744{,}513 \tag{19}$$
$$p_4 = 2^{32} + 3 \cdot 2^{25} + 1 = 4{,}395{,}630{,}593 \tag{20}$$

### C.2 Detailed Error Model

The photonic computation introduces three classes of noise:

**Shot noise.** The quantum nature of light introduces Poissonian fluctuations in photon counts.

For an optical power of $P = 1$ mW at wavelength $\lambda = 1550$ nm, the shot noise-limited SNR is:

$$\text{SNR}_{\text{shot}} = \sqrt{\frac{P \cdot T}{h\nu}} \approx \sqrt{\frac{10^{-3} \cdot 10^{-10}}{1.28 \times 10^{-19}}} \approx 2.8 \times 10^{7} \tag{21}$$

corresponding to ∼74 dB for a 100 ps measurement window ($T = 10^{-10}$ s), where $h\nu$ is the photon energy at 1550 nm.

**Thermal noise.** The electronic readout circuits introduce thermal noise (Johnson-Nyquist noise). For a transimpedance amplifier with resistance $R = 10$ kΩ at temperature $T = 300$ K and bandwidth $B = 10$ GHz:

$$\sigma_{\text{thermal}} = \sqrt{4k_B TRB} \approx 1.3 \times 10^{-5} \text{ V} \tag{22}$$

For typical signal amplitudes of ∼1 V, this contributes an SNR of ∼97 dB, which is subdominant to shot noise.

**Phase noise.** MZI phase settings drift due to temperature fluctuations and fabrication imperfections. With active thermal stabilization achieving $\delta\theta < 0.001$ rad, the resulting signal error is bounded by $|\delta a| < |\delta\theta| \cdot |a| < 0.001 \cdot |a|$, contributing ∼60 dB SNR.

The combined SNR is dominated by the phase noise floor at ∼60 dB, corresponding to ∼10 bits of effective precision per measurement. Since our RNS residues are 32-bit values, we require multiple measurements per residue. Specifically, we use a 4-sample averaging scheme, improving effective precision to ∼70 dB (∼12 bits), combined with the Reed-Solomon error correction described in the main text.

### C.3 End-to-End Verification

To verify the correctness of the photonic NTT output, we employ a probabilistic checksum. After computing $\hat{Y} = \text{NTT}(X)$ on the photonic hardware, we select a random evaluation point $r \in \mathbb{F}_{p_i}$ and verify:

$$\sum_j \hat{Y}_j \cdot r^j \stackrel{?}{=} \sum_j X_j \cdot \omega^{j \cdot r} \tag{23}$$

This Schwartz-Zippel-based check detects any incorrect NTT output with probability $\geq 1 - N/p_i > 1 - 2^{-8}$ for $N = 2^{24}$ and $p_i > 2^{32}$. Repeating the check with 16 independent random points yields a false-acceptance probability of $< 2^{-128}$.

## D Formal State Machine Specification

The VeritasLedger protocol state for each model is formalized as a finite state machine:

**Definition D.1** (Model State Machine)**.** *The model lifecycle is governed by state machine* $\mathcal{S}_M = (S, s_0, \Sigma, \delta, F)$ *where:*

- $S = \{\textit{Unregistered}, \textit{Pending}, \textit{Active}, \textit{Disputed}, \textit{Deprecated}, \textit{Slashed}\}$
- $s_0 = \textit{Unregistered}$
- $\Sigma = \{\textit{register}, \textit{confirm}, \textit{dispute}, \textit{resolve}, \textit{deprecate}, \textit{slash}, \textit{reactivate}\}$
- $F = \{\textit{Slashed}\}$ *(absorbing state)*

The transition function $\delta$ is defined as:

$$\delta(\text{Unregistered}, \text{register}) = \text{Pending} \tag{24}$$
$$\delta(\text{Pending}, \text{confirm}) = \text{Active} \tag{25}$$
$$\delta(\text{Active}, \text{dispute}) = \text{Disputed} \tag{26}$$
$$\delta(\text{Disputed}, \text{resolve}_{\text{valid}}) = \text{Active} \tag{27}$$
$$\delta(\text{Disputed}, \text{resolve}_{\text{invalid}}) = \text{Slashed} \tag{28}$$
$$\delta(\text{Active}, \text{deprecate}) = \text{Deprecated} \tag{29}$$
$$\delta(\text{Deprecated}, \text{reactivate}) = \text{Active} \tag{30}$$
$$\delta(\text{Active}, \text{slash}) = \text{Slashed} \tag{31}$$

**Proposition D.2** (Liveness). *For any model in state $s \in S \setminus \{\textit{Slashed}\}$, there exists a finite sequence of transitions leading to the Active state, assuming honest protocol participants and timely governance responses.*

**Proposition D.3** (Safety). *The Slashed state is absorbing: once a model enters the Slashed state, no transition can return it to any other state. This ensures that slashing penalties are irreversible and provide credible deterrence.*

## Disclaimer

This whitepaper is for informational purposes only and does not constitute financial advice, an offer to sell, or a solicitation to buy any tokens or securities. The $VRTS token is a utility token designed for use within the VeritasLedger protocol. The technical projections in this document, particularly regarding photonic computing performance, are based on analytical modeling and may differ from realized performance. All benchmarks are preliminary and subject to change. Readers should conduct their own research and consult qualified professionals before making any investment decisions.