The AI-Timeproof Protocol: Decentralized AI Verification Using Temporal Persistence

Nathaniel Joseph Houk
Independent Researcher
Email: njhouk@gmail.com

July 2025

Abstract

The rapid advancement of AI models raises concerns regarding the verifiability and trustworthiness of their outputs over time. We introduce the **AI-Timeproof Protocol**, a decentralized framework for verifying AI-generated content through temporal persistence. Inspired by blockchain-based timestamping and cryptographic proof mechanisms, the protocol ensures that AI claims remain provably authentic long after their creation. We propose a novel **Weibull-based Verification Function (WVF)** to quantify the probability that an AI-generated claim remains unchanged over time, and we introduce **Verifiable Delay Functions (VDFs)** to enforce time-dependent validation constraints. Our approach enables AI models to cryptographically commit to outputs, ensuring accountability and preventing post hoc tampering. This work lays the foundation for trustless AI governance, enabling decentralized AI auditing, compliance tracking, and secure AI model execution verification.

0.1 Introduction

AI models generate an increasing volume of outputs that impact legal, financial, and scientific domains. However, ensuring the long-term integrity of AI-generated claims presents a challenge. Current verification mechanisms either rely on **centralized trust** (e.g., model logs) or **static cryptographic signatures** (e.g., hash commitments), which fail to provide probabilistic guarantees of temporal persistence. We propose a **probabilistic, time-dependent verification model** that enhances AI output accountability while remaining decentralized.

Key contributions of this paper:

1. **Weibull-Based Verification Function (WVF):** A probabilistic model for assessing the likelihood that an AI-generated claim remains unaltered over time.
2. **Verifiable Delay Functions (VDFs) for AI Proofs:** Enforcing cryptographic delays in AI output verification to ensure time-sensitive commitment mechanisms.
3. **Decentralized AI Model Governance:** AI models cryptographically commit to execution logs on a blockchain, preventing unauthorized modifications.
4. **Trustless AI Compliance Auditing:** Using blockchain-based storage and zk-SNARKs to verify AI claims without exposing proprietary data.

0.2 Theoretical Foundations

0.2.1 Probabilistic Verification of AI Claims

Let C be an AI-generated claim at time t_0. The probability that C remains unchanged at time t is modeled using the Weibull distribution:

$$P(C_t = C_{t_0}) = e^{-(\lambda(t-t_0))^k}$$

where:

- λ represents the claim's decay rate (i.e., risk of modification over time),
- k is the shape parameter, which adjusts sensitivity to time-dependent decay.

This function provides a **probabilistic guarantee** rather than a binary assertion, making it more flexible than traditional cryptographic commitments.

0.2.2 Verifiable Delay Functions (VDFs) for AI Execution

A **Verifiable Delay Function (VDF)** enforces a time delay before verification can be completed. We define a delay-enforced AI commitment scheme:

1. An AI model generates output C and commits to it via a cryptographic hash $H(C)$.
2. The hash is recorded on a blockchain, along with a time-lock enforced by a VDF.
3. Verification is only possible after computing $f(C)$, which requires solving a sequential computational problem (e.g., repeated squaring in a prime field).

This approach ensures that **tampering with an AI-generated claim after the fact is computationally infeasible**.

0.2.3 Understanding Verifiable Delay Functions (VDFs)

A **Verifiable Delay Function (VDF)** is a cryptographic construct that requires a predetermined amount of sequential computation to evaluate while being efficiently verifiable. VDFs serve as proof that a given period has elapsed before verification can take place.

Mathematically, a VDF consists of three key components:

1. **Setup:** A function G generates public parameters pp based on security constraints.
2. **Evaluation:** A function $Eval$ computes the output $y = f(x)$ after a delay of T steps:

$$y = f^T(x) \mod N$$

   where N is a large prime number, and the computation must be performed sequentially.

3. **Verification:** A function $Verify$ ensures that y is correct using an efficiently computable proof.

Common constructions of VDFs involve repeated squaring in a modular arithmetic setting. The delay constraint ensures that AI-generated claims cannot be verified before the predefined time has elapsed, making them ideal for enforcing time-sensitive commitments in decentralized AI verification.

0.3 Protocol Design

0.3.1 AI Commitment Phase

1. **AI Model Execution:** AI generates output C.

2. **Hash Commitment:** Compute $H(C)$ and store on the blockchain.
3. **VDF Time-Lock:** Attach a VDF constraint, preventing immediate verification.

0.3.2 Verification Phase

1. **Unlocking with VDF:** After the delay period, compute $f(C)$ to unlock verification.
2. **Cross-Validation with Weibull Function:** Assess the probability of claim persistence using WVF.
3. **zk-SNARK Proofs for Compliance:** Generate a zero-knowledge proof ensuring execution constraints were followed.

0.4 Practical Workflow of the Protocol

1. **AI Model Generates an Output**
   - The AI system produces an output (e.g., text, image, or dataset).
   - This output is processed into a standardized digital format.
2. **Hashing and Commitment**
   - A cryptographic hash (e.g., SHA-256) is computed for the AI output.
   - This hash serves as a commitment and is recorded on a decentralized blockchain.
3. **Applying a Verifiable Delay Function (VDF)**
   - The system enforces a time-lock mechanism using a VDF.
   - The hash cannot be immediately verified, ensuring no premature modifications.
4. **Blockchain Storage and Timestamping**
   - The commitment hash is stored in a tamper-proof ledger.
   - A timestamp is recorded to establish a temporal reference.
5. **Verification Unlocking and Validation**
   - After the predefined time delay, the verifier solves the VDF.
   - The AI claim is cross-validated using the Weibull function.

6. **Zero-Knowledge Proofs (zk-SNARKs) for Compliance**
    - To ensure compliance and correctness, zk-SNARKs are generated.
    - These proofs confirm that the AI-generated claim was executed according to its predefined constraints without revealing sensitive model details.
7. **Final Verification and Auditability**
    - If the verification passes, the AI-generated claim is considered valid and persistent.
    - If any modifications are detected, the system flags the claim for further review.

This workflow ensures that AI-generated claims remain verifiable, resistant to tampering, and traceable over time.

0.5 Implementation & Security Analysis

- **Decentralization:** No single entity controls verification.
- **Tamper-Resistance:** Modifications require infeasible computation due to VDF constraints.
- **Privacy-Preserving Auditing:** zk-SNARKs allow verifiable claims without exposing proprietary AI models.

0.6 Conclusion

The AI-Timeproof Protocol provides a **probabilistic, decentralized, and time-enforced verification mechanism** for AI-generated claims. By integrating **Weibull-based probability modeling**, **VDF-enforced commitments**, and **zero-knowledge proofs**, we establish a **trustless AI verification framework** that enhances long-term AI accountability. This protocol enables decentralized AI auditing, regulatory compliance, and verifiable AI model execution, addressing critical challenges in AI governance.

Keywords: AI verification, probabilistic AI governance, Verifiable Delay Functions, zk-SNARKs, blockchain-based AI auditing.