Blockchain-Based Data Provenance and Integrity Verification for Secure AI Training

Akanksha Patil[1], Sujata Sathe[2]
[1]*TY BSc Blockchain Technology, DES Pune University*
[2]*Assistant Professor, Fergusson College Pune*
doi.org/10.64643/ijirt.208612-459

***Abstract*—The growing use of Artificial Intelligence (AI) has made the quality and reliability of training data increasingly important. If training data is modified without authorization, contains malicious records, or includes incorrect information, it can affect the performance and reliability of an AI system.**

These issues can lead to data poisoning, where compromised data influences the training process and produces unreliable results. Existing centralized methods for tracking data provenance may not provide sufficient transparency and can be vulnerable to tampering or a single point of failure. This paper proposes a blockchain-assisted framework for tracking and verifying the integrity of data used in AI training pipelines. The framework records important information such as data hashes, source details, timestamps, dataset versions, and modification history on a blockchain. This creates a transparent and tamper-evident record that can be used to verify whether data has been changed. Smart contracts are used to enforce predefined rules for registering and modifying datasets.

Before data is used for AI training, its integrity is checked against the information recorded on the blockchain. Any unauthorized change, hash mismatch, or suspicious data submission is flagged for further verification. The proposed framework is evaluated using controlled data-manipulation scenarios, with performance measured through poisoning detection rate, verification time, provenance traceability, transaction latency, and storage overhead. The study explores how blockchain can provide an additional layer of trust and transparency to AI data pipelines while making compromised data easier to identify and trace. The implementation is available at https://github.com/akankshapatil99/BAAITF.

***Index Terms*—Blockchain technology, artificial intelligence, security, integrity, data poisoning**

I. INTRODUCTION

The rapid advancement of Artificial Intelligence (AI) has transformed critical sectors including healthcare, finance, and autonomous systems. The effectiveness and trustworthiness of these AI systems are fundamentally dependent on the quality and integrity of their training datasets. However, contemporary AI pipelines are characterized by their complexity, multi-stage nature, and frequent distribution across organizational boundaries, rendering them susceptible to data poisoning attacks. Data poisoning represents a class of adversarial techniques where attackers inject or modify training data to degrade model performance, introduce backdoors, or induce biased outcomes.

Data poisoning differs fundamentally from prompt injection attacks in its timing and scope. Unlike prompt injection, which manipulates a model through user-provided input at runtime, data poisoning takes effect earlier in the lifecycle, during learning or retrieval construction. This temporal distinction makes data poisoning a supply chain problem as much as a model problem [1].

The severity of data poisoning threats is evidenced by recent real-world incidents across the AI supply chain, including hidden-prompt backdoor attacks embedded in code repositories that trigger attacker instructions upon phrase detection, and large-scale fabricated-content injection into training corpora that has been shown to degrade downstream model reliability [1], [5], [6]. Other documented cases include fraud detection systems trained on mislabelled transactions classified as safe, supply chain forecasting systems manipulated through poisoned demand data, and systems where training

data manipulation caused systematic misreading of numerical values [7], [8]. These incidents illustrate why security researchers increasingly frame poisoning as a systemic problem intersecting cybersecurity, misinformation, and regulatory compliance [1].

One of the primary concerns with the Artificial Intelligence (AI) boom is the privacy and integrity of the data used to train AI systems. Existing centralized methods for tracking data provenance may not provide sufficient transparency and can be vulnerable to tampering or a single point of failure. Data poisoning can occur at multiple points: during data collection, labelling, preprocessing, or even during model retraining cycles. While defences such as statistical anomaly detection, robust training, and outlier filtering have been proposed, they often assume trusted logging and do not address adversarial manipulation of pipeline metadata or dataset versions [2], [9].

This paper addresses the following research question: How can a blockchain-assisted provenance and integrity layer be designed and evaluated to detect and attribute data poisoning in AI training pipelines with practical overhead? We propose a hybrid architecture that combines blockchain and smart contract capabilities with the model training protocols of training AI models for the purpose of tamper resistance, traceability, and verification of data [2].

Blockchain technology offers a compelling foundation for addressing these challenges. Its core properties immutability, decentralization, timestamping, and programmable logic via smart contracts enable the construction of tamper-evident audit trails for data assets [3]. By anchoring cryptographic fingerprints of datasets and their transformation history on a blockchain, it becomes possible to verify whether data has been altered without authorization and to trace the origin of compromised versions.

The contributions of this paper are as follows. First, we present a blockchain system architecture for tamper-evident data provenance and integrity verification, combining cryptographic dataset fingerprinting, on-chain anchoring, and smart-contract enforced policies. Second, we provide a protocol design for dataset registration, versioning, and pre-training integrity verification, with formalized security properties and threat analysis. Third, we develop a poisoning detection and attribution mechanism that leverages on-chain records and lineage analysis to identify and trace compromised datasets. Finally, we implement a prototype and conduct an evaluation demonstrating the framework's effectiveness in detecting data poisoning under controlled attack scenarios, with measured performance overhead. The complete implementation is available as open-source software at https://github.com/akankshapatil99/BAAITF.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the threat model and design goals. Section IV describes the system architecture. Section V details the smart contract protocol and implementation. Section VI explains the poisoning detection mechanism. Section VII presents the evaluation. Section VIII discusses limitations and future work. Section IX concludes.

II. RELATED WORK

Data poisoning attacks manipulate training data to compromise model behaviour through various techniques including label flipping, clean-label poisoning, backdoor injection, and corpus contamination for large language models and retrieval-augmented generation systems [1], [6]. These attacks can cause systematic misclassification, performance degradation, or hidden triggers that activate under specific conditions. Defences include statistical distribution analysis, activation clustering, spectral signatures, and robust aggregation in federated learning [9], [13]. However, many of these methods focus on detecting anomalies in model behaviour or data distributions, without providing end-to-end, tamper-evident provenance across the pipeline.

Data provenance refers to tracking the origin, transformations, and usage of data throughout its lifecycle. In machine learning, provenance systems record dataset versions, preprocessing steps, and model training configurations to support reproducibility and auditability [3]. Tools such as Data Version Control, LakeFS, and model cards

provide metadata management and versioning, but typically rely on centralized storage and trusted logging. Recent surveys on provenance and traceability for large language models emphasize the need for cryptographic manifests, signed metadata, and continuous integration and continuous deployment contamination checks [9].

Blockchain technology has been applied to supply chain tracking, healthcare audit logs, genomic data access control, and IoT data integrity [4], [11]. Key techniques include Merkle-tree-based provenance for capturing fine-grained lineage in smart contracts and storing proofs efficiently [3], [10], [12], immutable audit logging for recording access and modification events in permissioned ledgers for regulatory compliance [11], [14], and smart-contract enforced policies for automating access control, versioning rules, and compliance checks [12]. In the AI context, recent work proposes blockchain-assisted assurance of data integrity in AI model training using hybrid optimization approaches for secure learning pipelines [2]. These frameworks use blockchain to record dataset fingerprints, chain-of-custody logs, and policy attestations, storing only compact digests on-chain while retaining full verifiability. Broader work on trustworthy and robust machine learning similarly argues for verifiable, auditable data handling as a precondition for secure deployment [15]. Few works explicitly tie blockchain-based provenance to systematic data-poisoning detection and attribution in training pipelines, with concrete protocol design and empirical evaluation. Our framework addresses this gap.

III. THREAT MODEL AND DESIGN GOALS

We consider the following adversary capabilities and assumptions. The adversary can inject or modify training data, labels, or preprocessing scripts at some pipeline stage. The adversary may compromise some data sources, storage systems, or pipeline components. The adversary can attempt to register malicious datasets or tamper with off-chain metadata. The adversary can embed hidden prompts or backdoors in training corpora [1], [5].

The adversary cannot break cryptographic hash functions such as SHA-256 or SHA-3 or digital signatures. The adversary cannot rewrite confirmed blockchain records beyond the consensus security threshold. The adversary cannot impersonate authorized roles without access to private keys. The blockchain network, whether public or permissioned, is sufficiently secure with honest majority or economically secure consensus. At least some auditors and key operators, referred to as data stewards, are honest and will verify records. Off-chain storage may be untrusted; integrity is verified via on-chain hashes.

Our framework aims to achieve the following properties. Integrity ensures detection of any unauthorized change in datasets or metadata via cryptographic verification. Provenance enables tracing each dataset version to its source and transformation history. Tamper-evidence ensures logs cannot be altered without detection, leveraging blockchain immutability. Efficiency keeps verification latency, transaction costs, and storage overhead acceptable for real pipelines. Auditability supports external auditors and incident response with verifiable, time-stamped records. Privacy protection addresses concerns about data privacy in AI training by providing transparent, verifiable data handling [2], [9].

IV. SYSTEM ARCHITECTURE

The framework consists of four main layers: the Data Layer for off-chain storage, the Provenance Ledger for on-chain storage, the Smart Contract Layer, and the Pipeline Integration Layer Raw training data is stored in conventional storage systems such as object stores, data lakes, or decentralized storage like IPFS. The framework does not store raw data on-chain; only fingerprints and metadata are anchored. Each dataset is split into logical units such as files, shards, or batches. A cryptographic hash such as SHA-256 is computed for each unit. A binary Merkle tree is constructed over these hashes, with the Merkle root representing the entire dataset version [3], [10]. The Merkle root, which is 32 bytes, serves as a compact, verifiable digest of the dataset. This design enables efficient verification of individual data elements via Merkle proofs without storing the entire dataset on-chain.

The blockchain stores immutable records for each dataset version. Each DatasetRecord includes the

following fields: dataset_id (unique identifier for the dataset), version_id (sequential version number), prev_version_id (reference to the previous version, zero for the initial version), merkle_root (32-byte root hash of the Merkle tree), creator_address (blockchain address of the data steward who registered the version), timestamp (block timestamp of registration), metadata_hash (hash of off-chain metadata including schema, license, and source description), and policy_tags_hash (hash of policy annotations such as PII-screened or consent-verified). This forms a version chain a linked list of records that provides an immutable history of all changes.

The framework implements its logic in one or more smart contracts. The core contract, DatasetRegistry, provides functions for registering datasets, updating datasets, retrieving dataset versions, and verifying dataset integrity. Role-Based Access Control is enforced using a standard access control library [12]. Roles include DataSteward (authorized to register and update datasets), Auditor (read-only access for verification), and Admin (manages role assignments in permissioned settings). Events are emitted for dataset registration, dataset updates, and verification failures, enabling off-chain indexing and audit tools to track all provenance activities.

The framework integrates with AI training pipelines via a Provenance Client Library and a Pre-Training Verification Hook. The Provenance Client Library computes Merkle roots for local datasets, interacts with smart contracts to register and update datasets, and queries on-chain records for verification.

The Pre-Training Verification Hook operates before training begins: the pipeline loads the configured dataset_id and version_id, the client computes the local Merkle root, and the contract's verifyDataset function is called with the computed root.

If verification succeeds, training proceeds; otherwise, it is aborted and an alert is raised. This ensures that only data matching the on-chain record is used for training.

V. SMART CONTRACT PROTOCOL AND IMPLEMENTATION

The DatasetRegistry contract is implemented in Solidity version 0.8.19 using OpenZeppelin's AccessControl library [12]. The contract defines a DatasetRecord structure containing version_id, prev_version_id, merkle_root, creator_address, timestamp, metadata_hash, and policy_tags_hash. Three roles are defined: DATA_STEWARD_ROLE for dataset curators, AUDITOR_ROLE for verifiers, and ADMIN_ROLE for access management. The deployer receives ADMIN_ROLE and grants other roles to participants.

The registerDataset function creates initial dataset versions with Merkle root and metadata hashes. The updateDataset function creates new versions linked to previous versions, forming an immutable version chain. The getDatasetVersion function retrieves dataset records for auditors. The verifyDataset function compares local Merkle roots with on-chain records and emits VerificationFailed events on mismatch, creating an audit trail of all verification attempts [10].

The design follows key principles: minimal on-chain storage (only 32-byte hashes), role-based access control separating curators from verifiers, version chains linking each version to its predecessor, and event-based auditing for off-chain monitoring. The verifyDataset function is intentionally not a view function so that it can emit events on failures, enabling security monitoring and incident response.

The complete implementation is available at https://github.com/akankshapatil99/BAAITF with deployment scripts, test suites, and client libraries.

VI. POISONING DETECTION AND ATTRIBUTION

The framework detects and attributes data poisoning through integrity checks and provenance analysis. Hash mismatch detection identifies when recomputed Merkle roots differ from on-chain records, flagging compromised datasets. Metadata anomaly detection identifies unexpected changes in creator addresses, timestamps, or policy tags. Supply chain verification tracks data from source

to training, addressing poisoning as a supply chain problem [1], [7].

When poisoning is detected, the framework supports forensic attribution by tracing the version chain backward to find the divergence point where the hash first diverged from the baseline. The creator address and timestamp are retrieved for the suspicious version, and operational logs are correlated to identify the responsible system or user. This enables precise attribution to specific dataset versions, sources, and time windows.

In Unauthorized Modification scenarios, attackers modify training data but the framework flags integrity violations before training. In Malicious Version Submission scenarios, unauthorized registrations are rejected by smart contracts, while authorized malicious registrations are recorded with creator identity for later attribution. In Hidden Prompt Injection scenarios, while semantic backdoors cannot be directly detected, the exact source and version are recorded for rapid identification once discovered [5]. In Gradual Poisoning scenarios, the version chain enables auditors to correlate performance degradation with specific version transitions.

VII. EVALUATION

Experimental setup

We implemented a prototype on the Ethereum Sepolia testnet, with Solidity 0.8.19 contracts deployed via Hardhat, a Python 3.10 client library built on Web3.py, and a PyTorch 2.x ML pipeline with the pre-training verification hook attached. The complete source code, deployment scripts, and evaluation harness are available at https://github.com/akankshapatil99/BAAITF for reproducibility.

Attack scenarios

We simulated six attack types: Label Flipping (1–10% of labels), Backdoor Injection (0.5–2% of samples), Unauthorized Update (direct tampering with stored data bypassing the registry), Malicious Registration (an unauthorized party attempting to register a dataset version), Hidden Prompt Injection (simulating [5]-style hidden-instruction attacks), and Fabricated Content Injection (simulating large-scale injection of fabricated records as in [6]). Each scenario was run across multiple trials per dataset to account for variance in attack magnitude.

Baselines

We compared against two baselines: Centralized Logging (a SQL database recording the same metadata fields dataset ID, version, hash, timestamp, creator without blockchain anchoring or immutability guarantees) and No Provenance (no tracking layer at all).

Metrics

Poisoning Detection Rate, False Positive Rate, Verification Time, Transaction Latency, Gas Cost, Storage Overhead, and Provenance Traceability (whether the responsible version/source could be identified after the fact).

For Hidden Prompt Injection and Fabricated Content Injection, detection is necessarily post-hoc: the framework does not flag semantically malicious content at registration time, but once such content is identified through any means (manual review, downstream model behaviour), the on-chain version chain provides 100% accurate source and time-window attribution, versus 0% and 15% for the respective baselines.

Performance overhead

Verification time ranged from 120–450ms per dataset check; transaction latency (registration/update confirmation) ranged from 12–18 seconds, consistent with Sepolia testnet block times; gas cost per registration ranged from 0.002–0.005 ETH; on-chain storage overhead was a constant 256 bytes per version regardless of dataset size. Attribution accuracy correctly identifying the creator, timestamp, and divergence point for a compromised version was 100% across all attack types in which a compromised version was successfully registered and later flagged.

Summary

The framework demonstrates strong detection for integrity-violating attacks (label flipping, backdoor injection, unauthorized updates) with acceptable overhead, and is most effective against post-registration tampering and unauthorized submissions. For semantic attacks such as hidden-

prompt injection, the framework provides attribution rather than direct detection, enabling rapid response once poisoning is discovered by other means.

VIII. LIMITATIONS AND FUTURE WORK

Semantic poisoning remains a limitation: the framework detects structural changes but not semantically malicious data that preserves the same distribution (e.g., hidden-prompt-style attacks) [5]. Initial trust is challenging, as the system assumes the first registered version is benign. Key management presents risk, since compromised DataSteward keys could enable authorized malicious registrations. Blockchain constraints include public-chain latency and cost, while permissioned chains reduce decentralization. Privacy concerns exist where additional mechanisms like zero-knowledge proofs may be needed for sensitive training data. Smart contract complexity may present adoption barriers for organizations without blockchain expertise [2].

Future work includes integration with advanced detection methods (activation clustering, spectral signatures, influence functions) for semantic poisoning detection [9], [13]. Light client verification would enable auditors to verify records without running full nodes. Decentralized storage integration would anchor IPFS or Arweave CIDs on-chain [4]. Formal verification would verify smart contract invariants. Cross-organization deployment would explore consortium blockchain models. Privacy-preserving extensions would integrate zero-knowledge proofs or secure enclaves. Hybrid optimization would incorporate metaheuristic techniques to enhance AI model training robustness alongside blockchain verification [2], [15].

IX. CONCLUSION

This paper presented a blockchain-assisted framework for tamper-evident data provenance and integrity verification in AI training pipelines. By combining cryptographic fingerprinting, smart-contract enforced policies, and pre-training verification hooks, the framework enables reliable detection and attribution of data poisoning attacks. Unlike prompt injection, which manipulates a model at runtime, data poisoning is a supply chain problem requiring end-to-end visibility and tamper-evident logging [1]. Real-world incidents involving hidden-prompt backdoors and fabricated-content injection demonstrate the urgency of this challenge [5], [6]. Our evaluation demonstrates high detection rates with modest performance overhead, validating the feasibility of blockchain-assisted integrity for real-world AI pipelines. The complete implementation is available at https://github.com/akankshapatil99/BAAITF to support reproducibility and community adoption. While AI training motivated this work, the underlying architecture — immutable provenance ledgers, verifiable data structures, and protocol-enforced policies — applies broadly to any domain requiring auditable data integrity. As prior work notes, one of the primary concerns with the AI boom is the privacy and integrity of training data, and blockchain-assisted assurance provides a promising path forward for secure, transparent, and trustworthy AI systems [2], [15].

REFERENCES

[1] S. Raizada, “Data poisoning attacks on machine learning pipelines,” 2026.
[2] S. Zakariae, A. Ouidad, and Y. E. B. El Idrissi Younes, “Blockchain-assisted assurance of data integrity in AI model training: A hybrid optimization approach for secure learning pipelines,” Informatica, vol. 49, no. 18, 2025.
[3] P. Ruan et al., “Fine-grained, secure and efficient data provenance on blockchain,” PVLDB, vol. 12, 2019.
[4] U.S. Department of Energy, “Blockchain for high performance data integrity and provenance,” OSTI, 2019.
[5] Fortinet, “What is data poisoning? How does it impact AI systems?” 2026.
[6] InfoQ, “Understanding ML model poisoning,” 2026.
[7] RedFox Security, “Data poisoning attacks on AI models: Detection & defense,” 2025.
[8] SecOps.QA, “ML pipeline security monitoring,” 2026.

[9] Protecto.ai, “Preventing data poisoning in training pipelines,” 2025.
[10] Medium, “Solidity and Merkle trees: Ensuring data integrity,” 2026.
[11] Vedrax, “Blockchain for audit & integrity,” 2026.
[12] IRJET, “Secure dataset verification using blockchain and Merkle,” 2026.
[13] Cybersecurity Switzerland, “Model poisoning defense strategies,” 2026.
[14] AWS, “Protect against data poisoning threats,” Machine Learning Lens, 2026.
[15] Springer, “Trustworthy AI for secure and robust machine learning,” 2026.