TwAIChain: A Blockchain–IPFS Architecture for Trustworthy Artificial Intelligence in Healthcare

Mugdha Srivastava(✉), Abhishek Kaushik, Róisín Loughran, and Fergal Mc Caffery

Regulated Software Research Centre, Dundalk Institute of Technology, Dundalk, Ireland
{mugdha.srivastava,abhishek.kaushik,roisin.loughran,fergal.mccaffery}@dkit.ie

Abstract. Artificial Intelligence (AI) systems are increasingly being deployed in production healthcare environments to support diagnosis, prognosis, and clinical decision-making. However, existing deployment pipelines provide limited guarantees of model integrity, provenance, and auditability once models are placed into operation, leaving high-stakes systems vulnerable to undetected tampering and misuse. In this paper, we propose a blockchain–based architecture called TwAIChain for secure, auditable, and tamper-resistant access to AI models in healthcare. The framework combines decentralised storage via the InterPlanetary File System (IPFS) with the immutability of an Ethereum-based blockchain to keep the model trustworthy. Cryptographic hashing and smart contracts enforce integrity checks and immutably log model access events in accordance with a defined threat model. We introduce a new set of trust and performance metrics and evaluate the system through algorithmic analysis and empirical measurements. The results demonstrate reliable detection of model tampering prior to inference while maintaining low latency and operational cost, supporting the practical deployment of trustworthy AI in regulated healthcare environments.

Keywords: Blockchain · InterPlanetary File System · Machine Learning · Artificial Intelligence · Decentralised Systems · Trustworthy AI · Healthcare AI · Adversarial Attacks

1 Introduction

Traditional artificial intelligence (AI) workflows, widely used across industries, typically involve deploying a static, pre-trained model to a production server [10]. In these workflows, the model is trained, encapsulated into a serialised file, and then uploaded for operational use. Although considerable effort is invested

© IFIP International Federation for Information Processing 2027
Published by Springer Nature Switzerland AG 2027
I. Maglogiannis et al. (Eds.): AIAI 2026, IFIP AICT 794, pp. 89–103, 2027.
https://doi.org/10.1007/978-3-032-30805-4_7

in monitoring model accuracy and detecting performance drift, this limited oversight fails to address critical issues of traceability. Anomalous behaviour, such as erroneous outputs resulting from poisoning attacks, can persist even when accuracy metrics remain consistent, posing serious risks in high-stakes applications [14].

In healthcare, risks are significantly higher due to the direct impact of decisions informed by these AI (or machine learning (ML)) models on patient health and safety [13]. These models are increasingly used in a variety of critical tasks, such as diagnosing medical conditions, predicting disease progression, recommending treatment plans, and even assisting in surgical procedures [16]. Errors in these contexts can have severe consequences, including misdiagnoses that delay proper treatment, incorrect prognostic evaluations that misinform care strategies, or inappropriate treatment recommendations that can harm patients. The acceptable margin for error in healthcare is extremely narrow, making trust in the underlying AI systems non-negotiable [17].

The implications of such risks extend beyond the harm to the patient, as they can deteriorate trust in AI technologies in the healthcare domain. Healthcare, historically characterised by the cautious adoption of emerging technologies, may have to discontinue the use of AI if these systems do not achieve trustworthiness [6]. Thus, solutions that combine advanced security mechanisms with robust auditing and traceability are essential to protect these critical AI systems.

1.1 Motivation and Research Questions

The essential dependence on AI in healthcare highlights the necessity of systems that go beyond performance monitoring to establish strong trust and accountability. Despite the potential of emerging technologies, there are significant gaps in ensuring ML model integrity and detecting malicious interference [4]. Addressing these gaps is a priority for initiatives like the European Union's 2024 AI Act, which seeks to establish a secure, ethical, and human-centric AI framework [2]. By focusing on compliance with fundamental rights and societal well-being, the act aims to make AI a reliable tool that benefits people.

Blockchain technology (BC), with its immutability and transparency, and decentralised storage solutions such as the InterPlanetary File System (IPFS) offer potential options for addressing the challenges that arise with respect to trustworthy AI [1]. However, their adoption is hindered by concerns about complexity, scalability, and maintaining trust in a decentralised architecture [7].

To address these challenges, this paper proposes an experimental BC-based architecture (TwAIChain) designed to support the trustworthiness of ML workflows. This study aims to address the following research questions (RQs):

1. **RQ1:** How can BC and IPFS be integrated to provide secure and decentralised storage tailored to ML models?
2. **RQ2:** How can the proposed BC–IPFS framework be evaluated in terms of algorithmic complexity and scalability, trust and provenance, system performance, and feasibility?

The remainder of this paper is structured as follows: Sect. 2 reviews the background and related work. Section 3 presents the architecture, development, and implementation of the proposed system. Section 4 outlines the evaluation metrics used to assess the framework. Section 5 reports and discusses the results. Section 6 addresses the limitations and broader implications of this study. Finally, Sect. 7 concludes the paper and outlines directions for future work.

2 Background and Related Work

Recent advances in BC technology have introduced novel methods to ensure the auditability of ML models. BC is at the forefront of code as law and algorithmic governance [5]. This section examines and reports a few of the key studies, focusing on their technological implementations, specific use cases in model auditability, and their inherent limitations.

One study proposed the use of a hybrid BC to monitor modifications in ML-driven clinical trial protocols specifically for genomic data [12]. By employing smart contracts to log model updates, their system provided an immutable record of changes for regulatory compliance. However, the system faces centralisation risks and the computational cost of on-chain operations limits scalability for large-scale trials. The work also lacks implementation details of the framework and experiments.

In radiology workflows, one study developed a dual-layer auditing system in which Ethereum smart contracts track access to brain tumour detection models while IPFS stores encrypted MRI datasets and model inference logs [11]. This approach hashes all ML model versions and radiologist interactions, enabling retrospective audits of diagnostic decisions. Zero-knowledge proofs verify model accuracy claims without exposing patient data, addressing privacy concerns. However, the system does not report any results except for ML metrics and is constrained by high Ethereum gas costs and IPFS pinning expenses for large datasets.

Another study evaluated a hybrid Block-IPFS system to audit AI research in dental radiology [9]. By combining Ethereum smart contracts with IPFS storage, this approach tracks medical imaging datasets used in ML training. A proof-of-custody mechanism ensures that only authorised researchers can retrieve decryption keys for specific datasets. Although the system improves traceability compared to conventional systems, it does not mention any implementation details or an evaluation framework. Additionally, Ethereum transaction costs constrain the granularity of audit logging.

For electronic health record (EHR) analyses, another study designed a private BC and IPFS collaboration that provides granular access control over audit logs [8]. However, they only report system latency based on cloud metrics.

A recurring limitation in the reviewed studies is the prevalence of theoretical frameworks and prototype systems without substantial evaluation in real world settings. Many of the proposed solutions do well in defining mechanisms for auditability, such as immutable logging, access control, and privacy-preserving

validation, but lack thorough performance validation, scalability testing, or adaptability assessments under dynamic conditions. Furthermore, the lack of evaluation metrics or case studies hinders the ability to compare these approaches or adapt them effectively to different contexts. As a result, the research remains largely conceptual, leaving critical questions about the practical deployment of such systems unanswered. This paper addresses the gap by proposing a holistic approach that integrates BC and IPFS to secure ML models.

3 Architecture Development and Implementation

To address RQ1, this section presents the design and implementation of the proposed BC–IPFS integration, detailing how its components work together to enable secure and decentralised storage for ML models. Additionally, this section also provides details on the threat model against which this system is tested.

3.1 Technology Stack

At the core of the architecture is the Ethereum BC, which acts as an immutable ledger to store metadata about ML models, including their content identifiers (CIDs) generated by the IPFS and hash values (see Fig. 1). Solidity smart contracts are employed to enable secure and transparent interactions with the BC, facilitating model registration, verification, and usage logging. These smart contracts create an auditable trail to detect unauthorised access or adversarial activities, ensuring robust tamper resistance. The contracts are developed and tested in the Ethereum Brownie environment, using its Python-based prototyping capabilities for streamlined implementation and deployment using the Ganache-CLI.

To address the challenges of storing and retrieving large ML models, the system integrates IPFS, a decentralised, content-addressable file storage protocol. By storing models off-chain, IPFS ensures the BC only maintains critical metadata, such as the CIDs. The frontend is implemented using HTML, providing a simple and user-friendly interface. Communication with the backend occurs via straightforward app routing. The backend, built with Python's Flask framework, aids interactions between the user interface, the Ethereum BC, and IPFS. Additionally, it facilitates model execution by dynamically loading ML models and delivering predictions to end-users in real-time.

For demonstration purposes, a diabetes prediction model has been developed using Pandas and scikit-learn (sklearn). The model is serialised using the pickle library, enabling efficient storage and transfer. Currently, the admin uploads the serialised model file directly to IPFS through the BC using a dedicated script called *Access Handler*.

3.2 Overview of the Workflow

The workflow begins with storing the pre-trained model on IPFS. Each model is assigned a CID and a cryptographic hash, both of which are registered on the

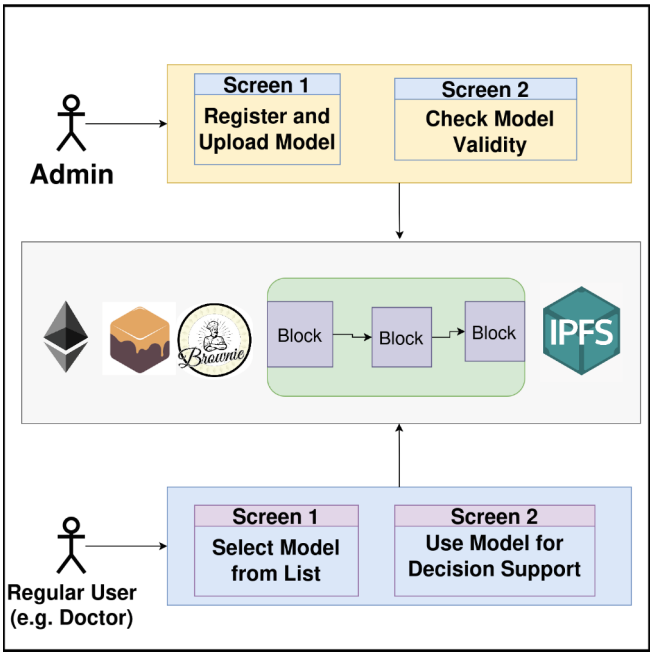


Fig. 1. Technology Stack of TwAIChain.

BC via a smart contract. Users interact with the system through a web-based frontend (see Fig. 2). They select a model and submit their input for prediction. Before any computation occurs, the model's integrity is verified by comparing its hash against the BC record. If verification is successful, the prediction is made and access details are logged on the BC for transparency and auditability. In case of a mismatch (indicating tampering or an adversarial attempt), the system rejects the access request and logs the incident as an attack.

3.3 BC Based Model Verification and Logging

When a model is accessed, its unique CID is retrieved from the BC, and the associated file is fetched from IPFS. To verify the model's integrity, a cryptographic hash (SHA-256) is computed for the retrieved file and compared to the hash stored on the BC. This process ensures that the model has not been altered since its deployment. The verification step guarantees both the integrity and authenticity of the model. Integrity is ensured when the model remains unmodified, safeguarding its accuracy and reliability. Authenticity is confirmed when the file retrieved from IPFS matches its BC metadata, establishing trust in its origin.

Every access request and its outcome are immutably logged on the BC, creating a tamper-proof record of all system interactions. Each log entry includes

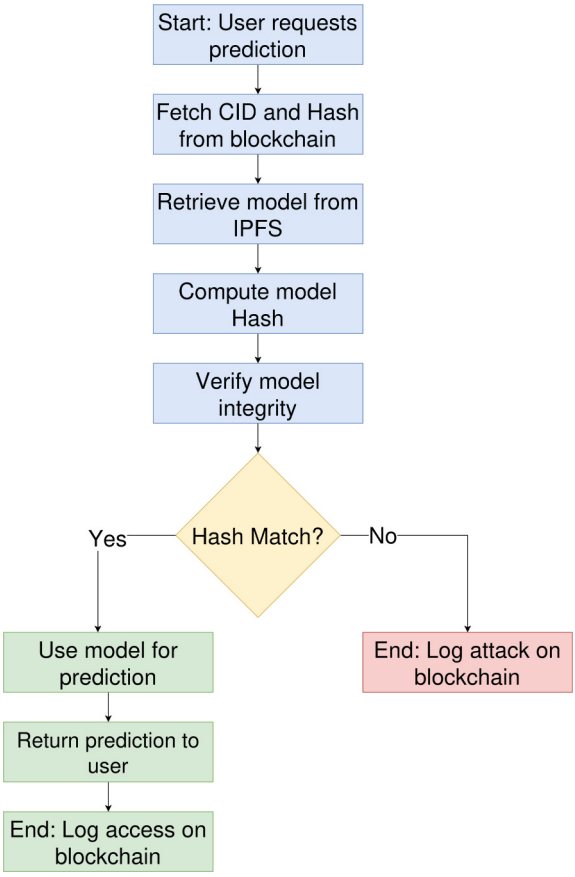


Fig. 2. User Request Workflow.

critical details such as the timestamp, model identifier, and result of the verification process.

3.4 Threat Model

This work considers an adversarial environment in which ML models and inference workflows are exposed to integrity and provenance risks typical of distributed healthcare deployments.

Adversary Capabilities. We assume an adversary capable of modifying off-chain resources, including serialised model artefacts and input features, through compromised storage, deployment pipelines, or authorised but malicious access. The adversary may attempt to tamper with models or manipulate inputs to influence inference outcomes. Attacks against BC consensus, deployed smart contracts, or cryptographic primitives are assumed to be infeasible and are out of scope.

Security Objectives. The system protects the integrity of model artefacts, the correctness of inference execution, and the completeness of audit records. Its objectives are to ensure that only registered and untampered models are executed, that integrity violations are detected prior to inference, and that all access attempts are immutably logged.

Trust Assumptions and Guarantees. The BC layer is treated as a trusted, immutable audit and verification space, while decentralised storage is assumed to be untrusted and integrity-enforced via cryptographic hash verification. Under these assumptions, the framework guarantees deterministic detection of model and feature tampering and prevents compromised artefacts from influencing predictions, while maintaining complete and verifiable auditability.

3.5 Implementation Details

Smart Contract Design. The smart contract brings several key functions to enable the system's core operations. The `storeModel()` function allows authorised users to register a new model by submitting its ID, CID, and hash. This data is stored in a mapping that associates model IDs with their corresponding metadata, ensuring fast and efficient retrieval. The `getModel()` function allows retrieval of this metadata, enabling external verification processes to access the CID and hash for integrity checks. Additionally, the contract includes a `logModelAccess()` function, which records each instance of model usage on-chain by emitting an event that includes the model ID and the timestamp of access.

IPFS Integration. By using IPFS, models are assigned unique CIDs, which act as cryptographic fingerprints for the files. These CIDs, coupled with their corresponding cryptographic hashes, are securely stored on the BC. This dual-layered approach ensures that any modifications to the models, intentional or otherwise, can be immediately detected through hash verification. Furthermore, the decentralised nature of IPFS provides fault tolerance and high availability, critical for applications like healthcare diagnostics. Even in the event of node failures or network disruptions, the redundancy built into IPFS guarantees that the models remain accessible, making it a robust storage layer for the system.

Frontend and API. The system's frontend is created with an emphasis on user experience, enabling healthcare professionals to interact with the underlying BC and storage infrastructure seamlessly. Its intuitive interface offers a range of functionalities, including selecting models for specific diagnostic tasks, submitting patient data as input, and receiving predictions (see Fig. 3). These features are designed to ensure that non-technical users can make use of the system's capabilities without requiring a deep understanding of BC or decentralised storage technologies. The backend API acts as a communication bridge, facilitating interactions between the frontend, BC, and IPFS. Through efficient endpoints, it

manages tasks such as fetching model metadata, verifying data integrity, logging access events on-chain, and retrieving stored models from IPFS.

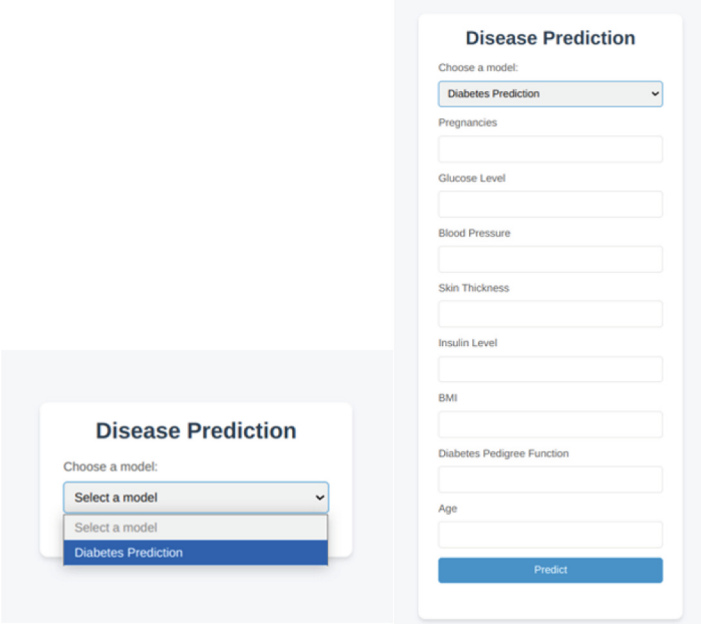


Fig. 3. Home and Prediction Pages.

4 Evaluation Metrics: Definitions and Formulae

The evaluation of BC-based trustworthiness mechanisms for ML systems is currently not supported by established benchmark suites or standardised metrics. To address RQ2, we adopt a task-driven evaluation methodology. Specifically, we define a new set of quantitative metrics spanning *trust and provenance*, *system performance*, and *feasibility*. We have **derived all these metrics directly from observable system artifacts**, including BC transaction logs, IPFS retrieval responses, and runtime execution traces. Additionally, algorithmic complexity analysis is used to characterise asymptotic scalability.

4.1 Algorithmic Complexity

To assess the computational complexity, scalability, and potential bottlenecks of the proposed system, we report an asymptotic complexity analysis. This analysis provides a theoretical foundation for understanding how the system behaves as input size grows, offering insights into its efficiency and performance under varying conditions [3,15].

4.2 Trust and Provenance Metrics

Audit Completeness. This metric measures the extent to which the ML model lifecycle is traceable through immutable BC records. Lifecycle events include model registration, model retrieval, and prediction requests. Each event is recorded as a BC transaction containing cryptographic references to IPFS content identifiers (CIDs).

$$\text{Audit Completeness } (\%) = \frac{E_{\text{recorded}}}{E_{\text{total}}} \times 100 \tag{1}$$

where E_{recorded} denotes the number of lifecycle events successfully logged on-chain, and E_{total} represents the total number of events generated by the system.

Integrity Verification Success Rate. This metric quantifies how often retrieved model artefacts (from IPFS) match their expected cryptographic fingerprint.

$$\text{Integrity Verification Success Rate } (\%) = \frac{R_{\text{valid}}}{R_{\text{total}}} \times 100 \tag{2}$$

where R_{valid} is the number of retrievals whose computed hash matches the BC-registered hash, and R_{total} is the total number of retrieval attempts.

Tamper Detection Rate. This metric evaluates the system's ability to identify and reject unauthorised or altered models. Controlled tampering is introduced by modifying model artifacts stored in IPFS while keeping the original BC record unchanged.

$$\text{Tamper Detection Rate } (\%) = \frac{T_{\text{detected}}}{T_{\text{attempts}}} \times 100 \tag{3}$$

where T_{detected} denotes the number of tampered models correctly identified via hash mismatch, and T_{attempts} is the total number of tampering attempts.

Audit Query Response Time. This metric measures the time required to retrieve complete model lineage information, including training data reference, model version, and deployment metadata, from BC records (t_{end}) after a request has been initiated (t_{start}).

$$\text{Audit Response Time} = t_{\text{end}} - t_{\text{start}} \tag{4}$$

Response times are reported in seconds using mean, median, and 95th percentile statistics across audit queries.

4.3 Performance Metrics

Prediction Latency. This metric captures the end-to-end time required to produce a prediction, including model retrieval, integrity verification, audit logging,

and inference execution. Latency is measured in milliseconds from request initiation (lt_{start}) to prediction output (lt_{end}), enabling direct comparison of system overhead introduced by trust mechanisms.

$$\text{Prediction Latency} = lt_{\text{end}} - lt_{\text{start}} \tag{5}$$

Model Availability. This metric measures the reliability of decentralised storage by tracking successful IPFS retrievals.

$$\text{Model Availability } (\%) = \frac{A_{\text{success}}}{A_{\text{total}}} \times 100 \tag{6}$$

where A_{success} is the number of successful retrievals and A_{total} is the total retrieval attempts.

4.4 Feasibility Metrics

Transaction Cost. This metric represents the monetary overhead of BC interactions, including model registration and integrity verification transactions.

$$\text{Transaction Cost} = \sum_{i=1}^{N} \text{Gas}_i \times \text{GasPrice} \tag{7}$$

Costs are reported in USD-equivalent values using the prevailing gas price and exchange rate at the execution time.

Storage Efficiency. This metric evaluates IPFS scalability by analysing storage usage and retrieval latency as the number of registered models increases. Metrics include total storage consumed (MB) and number of blocks used.

5 Results

5.1 Algorithmic Complexity

The computational complexities associated with the system's operations are critical for understanding its performance and scalability. By examining these factors, we can better understand how the system behaves under various workloads and identify areas for optimisation to enhance efficiency and scalability.

The overall complexity of the system for a single prediction request involves both time and space considerations. The time complexity begins with fetching the CID and hash, which operates in $O(1)$, followed by fetching the model from IPFS with a complexity of $O(n)$, where n represents the model size. Hash calculation also operates in $O(n)$, as does the deserialisation of the model. The prediction step, which depends on the number of features m and the internal operations of the model k, exhibits a complexity of $O(m \cdot k)$. Consequently, the total time complexity for a single prediction request is $O(n + m \cdot k)$, with the

dominant factors being the model size n and the complexity of the prediction step.

The space complexity for a single prediction request is dominated by the size of the model. Storing the CID and hash requires $O(1)$, while fetching the model from IPFS and its subsequent deserialisation both consume $O(n)$ space. The prediction step itself requires $O(1)$ space for a single inference. Thus, the total space complexity for a single prediction request is $O(n)$.

Current algorithmic bottlenecks primarily arise from the size of the model, n, as fetching, hashing, and deserialising large models pose significant challenges. Additionally, the scalability of the system with respect to the number of models, d, may require advanced techniques such as pagination or indexing to handle thousands of models efficiently. Furthermore, the complexity of prediction, $O(m \cdot k)$, becomes a bottleneck when dealing with large, deep models, such as neural networks, which increase inference time significantly.

5.2 Empirical Evaluation and Results

Experimental Setup and Workload. The system was setup on a single-machine testbed. The registration of a single diagnostic model (diabetes) to IPFS used **152604 gas units** at a cost of **USD 9.2**. The evaluation workload consisted of 200 prediction requests at once due to system constraints and was issued through the API to the model. This was done twice, once to set up a baseline and the second time to test adversarial attacks. For each request, the system retrieved the model, verified its integrity against a BC-recorded cryptographic reference, recorded an immutable access log, and then executed the inference.

Trust and Provenance Evaluation

Audit Completeness. Across both baseline and adversarial workloads, the system achieved **100% audit completeness**. Every model access and prediction request produced a corresponding immutable audit record, with no missing or partial entries observed.

Integrity Verification Success Rate. Under baseline conditions, integrity verification succeeded for **100%** of requests, indicating that all models were retrieved without modification. During adversarial evaluation, the success rate decreased to **88.89%**, reflecting the deliberate rejection of tampered artefacts rather than failures on legitimate requests. All verification failures corresponded to confirmed integrity violations.

Tamper Detection Rate. During the adversarial evaluation, 20 model-level tampering attempts and 20 feature-level tampering attempts were issued. The system achieved a **100% detection rate** for both attack types, correctly identifying all modifications before inference execution.

Audit Query Response Time. In the baseline evaluation, the mean (average) audit query response time was **34.1 ms**, with a median (typical case) of **29.6 ms** and a 95th-percentile latency (near-worst-case) of **58.5 ms**. Under adversarial conditions, mean query latency decreased to **26.5 ms**, with a 95th-percentile latency of **37.2 ms**. The reduction in latency under adversarial conditions is due to early detection and rejection mechanisms that prevent malicious audit queries from triggering full proof verification or ledger traversal. These results indicate that full audit reconstruction can be performed efficiently and does not interfere with inference execution.

Performance Evaluation

Prediction Latency. Under baseline conditions with full auditing enabled, the system exhibited a mean latency of **176.99 ms**, a median latency of **152.96 ms**, and a 95th-percentile latency of **263.80 ms**. During adversarial evaluation, mean latency decreased to **89.97 ms**, with a 95th-percentile latency of **114.58 ms**, as tampered requests were rejected before inference. Overall, the system maintained predictable and bounded latency across both scenarios.

Model Availability. Across all baseline and adversarial workloads, the system achieved **100% model availability**, with no failed retrievals or timeouts observed. This indicates reliable access to model artefacts even under sustained request rates.

Feasibility Metrics

Transaction Cost. During the both baseline and adversarial evaluations, the average cost per audited request was **25,378 gas**, corresponding to a mean transaction cost of **USD 1.52** under the configured test-network pricing. Smart contract deployment and registration to the BC incurred a one-time cost of **434145** corresponding to a transaction cost of **USD 26.0487**. These results show that integrity enforcement introduces a stable and measurable operational cost.

Storage Efficiency. The diagnostic model was stored as a content-addressed directed acyclic graph consisting of **eight blocks**, with a total logical size of **1,701,874 bytes** (approximately 1.62 MB). All blocks were unique within the local IPFS repository, resulting in zero shared storage and a deduplication ratio of **1.0**. As the evaluation focused on a single-model deployment scenario, storage usage scaled linearly and retrieval latency remained stable across repeated accesses. While no deduplication benefits were observed in this configuration, the results highlight the suitability of content-addressed storage for managing versioned or related ML models in larger deployments.

6 Limitations and Discussion

The system enhances model trustworthiness but has several limitations. First, it has been tested on auto-mining and auto-consensus which means that the real-world performance may vary from the current one when deployed on a mainnet. Second, scalability issues can arise due to gas fees and latency in real-world scenarios, which can hinder performance in large-scale deployments. Third, while the system detects tampering, it remains vulnerable to sophisticated adversarial attacks targeting the model's learning process. Finally, the absence of role-based access control mechanisms leaves the system vulnerable to unauthorised manipulation, particularly if it is deployed in the public domain without supplementary security measures.

7 Conclusion and Future Work

We present a BC-based architecture to ensure the trustworthiness of healthcare ML models by using BC as a protective and auditing layer around model artefacts. Through this work we address two research questions around the development of the proposed architecture and its evaluation. Combined with IPFS for decentralised storage, the system enables secure storage, integrity verification, and transparent logging of all model interactions. Integrity checks are enforced prior to model execution, preventing tampered or compromised models from affecting predictions. Evaluation indicates that the framework provides strong auditability and integrity guarantees with predictable performance overhead, supporting practical use in regulated healthcare settings where transparency, accountability, and robustness are critical, thus keeping AI trustworthy. Therefore, this work contributes a novel integration of BC and IPFS for secure ML model storage, a working system architecture for real-time integrity verification and audit logging, and a new set of metrics to evaluate complexity, scalability, and performance to validate feasibility.

To further improve the system, several directions may be explored. First, mainnet deployment followed by scalability enhancements, such as layer-2 BC solutions, to reduce costs and increase throughput. Second, advanced adversarial defences can be developed to detect and mitigate attacks beyond simple model tampering. Third, the development of a complete performance validation framework for this system may aid in improving and advancing the architecture. Finally, the framework may be adapted for other high-stakes domains, such as finance and autonomous systems, broadening its impact. These advancements would strengthen the reliability and applicability of this system in real-world deployments.

Acknowledgments. This publication has emanated from research conducted with the financial support Research Ireland under Grant number 21/FFP-A/9255. For the purpose of open access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bin Saif, M., Migliorini, S., Spoto, F.: Efficient and secure distributed data storage and retrieval using interplanetary file system and blockchain. Future Internet **16**(3), 98 (2024)
2. Cabrera, B.M., Luiz, L.E., Teixeira, J.P.: The artificial intelligence act: insights regarding its application and implications. Procedia Comput. Sci. **256**, 230–237 (2025)
3. Chaudhry, N., Yousaf, M.M.: Consensus algorithms in blockchain: comparative analysis, challenges and opportunities. In: 2018 12th International Conference on Open Source Systems and Technologies (ICOSST), pp. 54–63. IEEE (2018)
4. Devineni, S.K., Kathiriya, S., Shende, A.: Machine learning-powered anomaly detection: Enhancing data security and integrity. J. Artif. Intell. Cloud Comput. **184**, 2–9 (2023). https://doi.org/10.47363/JAICC/2023(2)
5. Elsden, C., Manohar, A., Briggs, J., Harding, M., Speed, C., Vines, J.: Making sense of blockchain applications: a typology for HCI. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–14 (2018)
6. Goktas, P., Grzybowski, A.: Shaping the future of healthcare: ethical clinical challenges and pathways to trustworthy AI. J. Clin. Med. **14**(5), 1605 (2025)
7. Hammad, A., Abu-Zaid, R.: Applications of AI in decentralized computing systems: harnessing artificial intelligence for enhanced scalability, efficiency, and autonomous decision-making in distributed architectures. Appl. Res. Artif. Intell. Cloud Comput. **7**, 161–187 (2024)
8. Nguyen, D.C., Pathirana, P.N., Ding, M., Seneviratne, A.: Blockchain for secure EHRs sharing of mobile cloud based e-health systems. IEEE Access **7**, 66792–66806 (2019)
9. Orhan, K., Bayrakdar, I.S., Celik, O., Ayan, B., Polat, E.: Can the blockchain-enabled interplanetary file system (Block-IPFS) be a solution for securely transferring imaging data for artificial intelligence research in oral and maxillofacial radiology? Imaging Sci. Dent. **51**(3), 337 (2021)
10. Pahune, S., Akhtar, Z.: Transitioning from MLOps to LLMOps: navigating the unique challenges of large language models. Information **16**(2), 87 (2025)
11. Santhi, V., Santhosh, D., Sachitha, C., Samyuktha, N., Maran, A., et al.: Electronic health records reimagined: a blockchain approach combined with IPFS and brain tumor detection through machine learning. In: 2024 International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering (ICSSEECC), pp. 315–320. IEEE (2024)
12. Shae, Z., Tsai, J.J.: On the design of a blockchain platform for clinical trial and precision medicine. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1972–1980. IEEE (2017)
13. Shahid, A.H., Khattak, W.A.: Improving patient care with machine learning: a game-changer for healthcare. Appl. Res. Artif. Intell. Cloud Comput. **5**(1), 150–63 (2022)
14. Srivastava, M., Kaushik, A., Loughran, R., McDaid, K.: Data poisoning attacks in the training phase of machine learning models: a review (2024)

15. Tiwari, U.K., Malviya, A.K.: Complexity estimation technique for large scale software based on communicating parameters. In: Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) (2019)
16. Varghese, C., Harrison, E.M., O'Grady, G., Topol, E.J.: Artificial intelligence in surgery. Nat. Med. **30**(5), 1257–1268 (2024)
17. Wadden, J.J.: Ethics and healthcare artificial intelligence: some problems and solutions for advanced diagnostic systems. Ph.D. thesis, University of British Columbia (2023)