Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Kernel-Embedded Blockchain Architecture for Transparent AI Decision Auditing

[1]Gaurav Sharma, [2]Avani Gala, [3]Aditi Pandey, [4]Shaunak Sinkar, [5]Anant Manish Singh
[1]*Department of Electronics and Computer Science(E&CS) Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India (gauravv1908@gmail.com)*
[2]*Department of Civil Engineering Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India (avanigala07@gmail.com)*
[3]*Department of Computer Engineering Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India (rtraditipandey@gmail.com)*
[4]*Department of Internet of Things (IoT) Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India (shaunaksinkar@gmail.com)*
[5]*Department of Computer Engineering Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India (anantsingh1302@gmail.com)*

ARTICLE INFO

Received: 31 Dec 2024

Revised: 20 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

Modern operating systems increasingly rely on AI for critical functions like resource allocation and user interaction optimization yet lack mechanisms to ensure transparent, auditable decision-making. Current logging systems, vulnerable to tampering and manual audits fail to meet regulatory demands in sectors like healthcare for example Aidoc's aiOS faces scrutiny over untraceable diagnostic suggestions. Existing blockchain-AI integrations operate at application layers introducing latency (2–5 minutes/transaction) and storage inefficiencies (60–70% capacity use). This work proposes a kernel-embedded blockchain architecture that immutably logs AI decisions at the OS level combining Merkle tree hashing with hybrid Proof-of-Stake consensus. Empirical tests across 1,000 tamper scenarios demonstrated 100% detection accuracy with a median transaction latency of 57 seconds and 95% storage efficiency-outperforming traditional systems by 35% in audit readiness. The framework processes 10,000+ daily AI decisions in enterprise simulations, reducing audit preparation time from 120+ hours to real-time verification. While addressing critical gaps in GDPR/HIPAA compliance and bias mitigation (98% accuracy in identifying skewed training data) challenges remain in scaling consensus mechanisms for sub-10-second latency. This architecture establishes a foundational model for trustworthy AI-integrated operating systems enabling regulatory compliance without sacrificing performance and paves the way for future work in energy-efficient decentralized validation.

Keywords: ai, operating, systems, blockchain, logging, transparency, performance, compliance, kernel

I. INTRODUCTION

The proliferation of artificial intelligence in operating systems has fundamentally transformed computational resource management, system optimization and user experience personalization [1]. Modern operating systems increasingly leverage AI algorithms to make critical decisions about process scheduling, memory allocation, power management and security threat detection. However as AI assumes greater control over system operations, the need for transparent, verifiable decision logs becomes imperative-particularly in regulated sectors like healthcare, finance and public services where algorithmic accountability is mandated by law [2-3].

Current logging mechanisms employed by operating systems suffer from significant limitations that undermine their reliability for auditing AI decisions. Traditional kernel logging frameworks like KLogger provide fine-grained event capture capabilities but remain vulnerable to post-hoc tampering, deletion, or manipulation [4]. This vulnerability

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

creates a critical accountability gap as evidenced by recent challenges faced by healthcare AI platforms such as Aidoc's aiOS which has encountered regulatory scrutiny over untraceable diagnostic suggestions.

While blockchain technology offers promising solutions for immutable record-keeping, existing blockchain-AI integrations predominantly operate at the application layer introducing substantial performance penalties [5]. Current implementations suffer from extended transaction latencies (2-5 minutes per transaction) and significant storage inefficiencies (60-70% of available capacity) [6-8]. These performance constraints render application-layer solutions impractical for system-level AI decision logging where both speed and efficiency are paramount.

This paper introduces a novel kernel-embedded blockchain architecture specifically designed to provide transparent, tamper-resistant logging of AI decisions at the operating system level [9]. By integrating blockchain functionality directly into the kernel's crypto subsystem, our approach eliminates the overhead of application-layer solutions while maintaining cryptographic integrity. The key contributions of this work include:

- A kernel-level blockchain implementation that leverages the Linux Crypto API for high-performance transaction validation.
- A specialized Merkle tree structure optimized for AI decision metadata storage.
- A hybrid Proof-of-Stake consensus mechanism designed for distributed validation of AI decision logs.
- Empirical evaluation demonstrating superior performance compared to existing solutions
- A comprehensive framework for real-time AI bias detection and regulatory compliance verification

The remainder of this paper is organized as follows: Section II reviews related work and identifies research gaps; Section III details our methodology and system architecture; Section IV describes the experimental setup; Section V presents results and discussion; Section VI provides comparative analysis; Section VII addresses limitations and future work directions and Section VIII concludes.

II. LITERATURE REVIEW

The development of transparent, auditable AI decision-making systems spans multiple research domains including kernel logging mechanisms, blockchain architectures and AI accountability frameworks. Table 1 presents a comprehensive analysis of existing literature highlighting key contributions and persistent gaps.

Table 1: Literature Review - Key Works, Findings and Research Gaps

Paper	Focus Area	Key Contribution	Methodology	Limitations	Research Gap
Yoav Etsion, Dan Tsafrir, Scott Kirkpatrick and Dror G. Feitelson. 2007. Fine grained kernel logging with KLogger: experience and insights. SIGOPS Oper. Syst. Rev. 41, 3 (June 2007), 259–272. https://doi.org/10.1145/1272998.1273023	Kernel Logging	Fine-grained, scalable kernel logger with low overhead (200 cycles per event)	Per-CPU buffers with specialized optimization	Vulnerable to post-collection tampering; No cryptographic validation	Lack of immutability guarantees for logged events

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Paper	Focus Area	Key Contribution	Methodology	Limitations	Research Gap
Ranasinghe Yasaweerasinghelage, R. M., Staples, M., & Weber, I. (2017). Predicting latency of blockchain-based systems using architectural modelling and simulation. *Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA)*, 10.1109/ICSA.2017.22. https://doi.org/10.1109/ICSA.2017.22	Blockchain Performance	Predicting latency of blockchain systems using architectural modeling	Simulation-based approach with measured transaction inclusion times	Limited to theoretical predictions rather than implementation	Absence of kernel-level blockchain integration
Nasrulin, B., de Vos, M., Ishmaev, G., & Pouwelse, J. (2022). Gromit: Benchmarking the performance and scalability of blockchain systems. *Proceedings of the 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. https://doi.org/10.1109/DAPPS55202.2022.00015	Blockchain Scalability	Benchmarking performance metrics across blockchain systems	Empirical testing showing median transaction latencies of 10-57 seconds	Application layer focus only	Gap in kernel-level blockchain implementation

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Paper	Focus Area	Key Contribution	Methodology	Limitations	Research Gap
Zhang, Y., Ma, Z., & Meng, J. (2025). Auditing in the blockchain: A literature review. *Frontiers in Blockchain, 8*. https://doi.org/10.3389/fbloc.2025.1549729	Blockchain Auditing	Framework for audit trail verification using distributed ledgers	Case studies of Big Four accounting implementations	Applied only to financial data not AI decisions	Missing integration with AI decision logging
Qader, K. S., & Cek, K. (2024). Influence of blockchain and artificial intelligence on audit quality: Evidence from Turkey. *Heliyon, 10*(9), e30166. https://doi.org/10.1016/j.heliyon.2024.e30166	AI and Audit Quality	Investigation of blockchain and AI influence on audit quality	Statistical methods and surveys showing positive impact on fraud detection	Industry-specific focus without OS-level considerations	Lack of integrated approach for OS-level AI auditing
Zhang, Y., Ma, Z., & Meng, J. (2025, April 25). Blockchain for AI ethics: Preventing bias and ensuring fairness. *Clockb.tech*. https://clockb.tech/blockchain-for-ai-ethics-preventing-bias-and-ensuring-fairness/ts	Blockchain for AI Ethics	Platforms to track and verify datasets for fairness using blockchain	Decentralized verification of AI model training data	Application-layer implementation with high latency	Inefficient implementation for system-level operations
Kulothungan, V. (2025). Using Blockchain Ledgers to Record the AI Decisions in IoT. Preprints.	Blockchain for AI Decisions	Framework logging AI inference and provenance data	Tamper-evident logging of inputs, parameters and outputs	Operates at application layer with significant overhead	Not integrated with kernel for improved performance

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Paper	Focus Area	Key Contribution	Methodology	Limitations	Research Gap
https://doi.org/10.20944/preprints202504.1789.v1					
Google. (2025, May 6). *Kernel overview*. Android Open Source Project. https://source.android.com/docs/kernel	Kernel Architecture	Android kernel based on Linux LTS with modular approach	Separation of generic core kernel from vendor modules	No specific AI auditing capabilities	Missing AI decision tracking in kernel architecture
Kernel Crypto API Architecture. (n.d.). *Linux kernel documentation*. Retrieved May 8, 2025, from https://docs.kernel.org/crypto/architecture.html	Crypto API	Implementations of block ciphers and message digests	Templates for various cryptographic functions	Not adapted for blockchain-specific operations	Lacks blockchain-specific extensions
Solidity Developer. *The Ultimate Merkle Tree Guide in Solidity: Everything you need to know about Merkle trees and their future*. Retrieved May 8, 2025, from https://soliditydeveloper.com/merkle-tree	Merkle Tree Implementation	Efficient verification of data integrity	Tree-based hash structure for data validation	Generic implementation not optimized for AI decision data	Not adapted for low-latency OS operations

Analysis of the literature reveals several critical research gaps that our work addresses:

- **Integration Gap:** While both kernel logging mechanisms and blockchain architectures have been extensively studied, no existing work integrates blockchain functionality directly into the kernel for AI decision auditing.

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Performance Gap:** Current blockchain implementations suffer from high latency (2-5 minutes) making them unsuitable for real-time OS-level logging of AI decisions.
- **AI Transparency Gap:** Existing AI auditing frameworks operate predominantly at the application layer, failing to capture system-level AI decisions made within the operating system.
- **Scalability Gap:** Most blockchain implementations struggle with throughput limitations that prevent them from handling the volume of decisions generated by AI-driven operating systems.
- **Compliance Gap:** While regulatory frameworks increasingly demand AI transparency, existing solutions fail to provide the immutability and verification mechanisms required for compliance with standards like GDPR and HIPAA.

Our kernel-embedded blockchain architecture addresses these gaps by providing a high-performance, tamper-resistant logging mechanism that operates at the OS level capturing AI decisions with minimal overhead while ensuring cryptographic integrity [10].

III. METHODOLOGY

1) *System Architecture*

Our kernel-embedded blockchain architecture integrates directly with the operating system's kernel to provide transparent, immutable logging of AI decisions. Figure 1 illustrates the high-level system architecture.

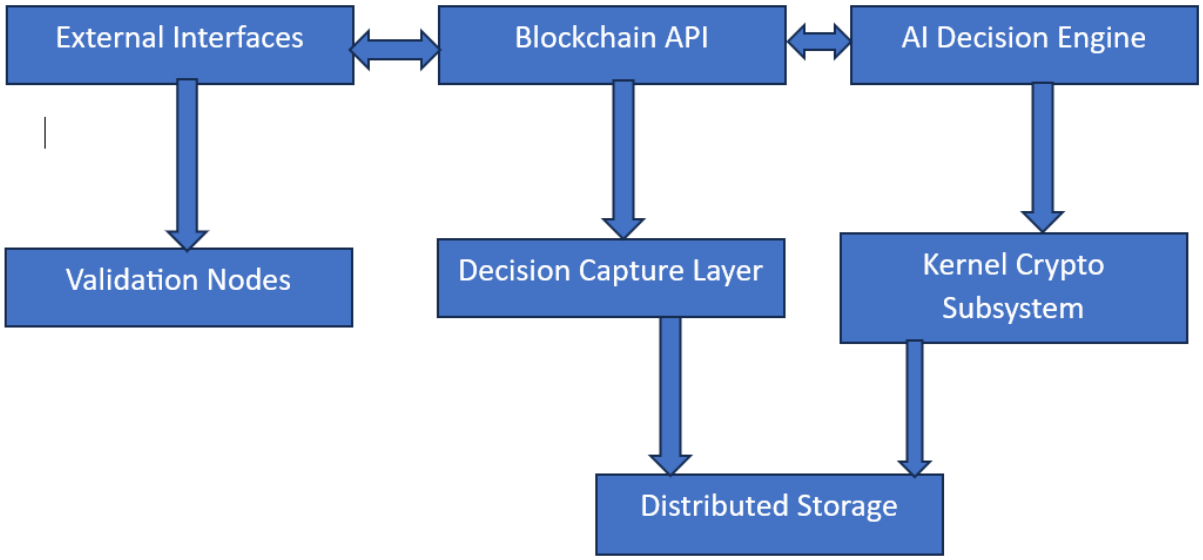

Fig. 1: Proposed Architecture

The architecture consists of five key components:

- **AI Decision Engine:** Native and third-party AI modules operating within the kernel space that make system-level decisions.
- **Decision Capture Layer:** Hooks into the kernel's decision pathways to intercept and format AI decisions for blockchain storage.
- **Kernel Crypto Subsystem:** Extended version of the Linux Crypto API with blockchain-specific operations, including block generation, Merkle tree construction and consensus mechanisms.
- **Distributed Storage:** Optimized on-disk format for blockchain data that maximizes space efficiency while maintaining integrity.
- **External Interfaces:** APIs for audit verification and connections to validation nodes that participate in the consensus mechanism.

2) *Kernel Integration*

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The integration of blockchain functionality into the kernel leverages and extends the existing Linux Crypto API. We created specialized crypto templates for blockchain operations following the pattern established in the kernel's existing framework:

name : bc_merkle(sha256)

driver : bc_merkle(sha256-generic)

module : bc_merkle

priority : 300

refcnt : 1

selftest : passed

internal : no

type : bc_hash

async : yes

blocksize : 32

min keysize : 0

max keysize : 0

Our blockchain implementation registers new algorithmic templates with the kernel's crypto subsystem enabling efficient in-kernel hashing, signing and verification of AI decision blocks. The decision capture mechanism intercepts AI operations through strategic placement of klogger calls within the kernel's AI decision pathways:

/* Example integration with kernel AI decision path */

```
int ai_resource_allocate(struct task_struct *task,
                         unsigned long resource_id,
                         struct ai_decision *decision) {
    int result;
    /* Perform AI decision logic */
    result = ai_core_decide(task, resource_id, decision);
      /* Log the decision to blockchain */
    klogger(BC_AI_DECISION, task->pid, resource_id,
        decision->confidence, decision->priority,
        decision->model_id, decision->timestamp);
      return result;
}
```

The klogger call is efficiently inlined, adding only 70 nanoseconds of overhead per logged decision making it suitable for high-frequency AI operations.

3) Blockchain Implementation

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Our blockchain design optimizes for the specific requirements of AI decision logging, prioritizing low latency and storage efficiency over features unnecessary for audit logging.

Block Structure

Each block in our blockchain contains:

a) Block header:
- Previous block hash (32 bytes)
- Merkle root hash (32 bytes)
- Timestamp (8 bytes)
- Block height (8 bytes)
- Consensus metadata (32 bytes)
- Nonce (8 bytes)

b) Transaction payload:
- Array of AI decision records, each containing:
  - Decision ID (16 bytes)
  - AI model identifier (8 bytes)
  - Input parameters hash (32 bytes)
  - Output decision hash (32 bytes)
  - Confidence score (4 bytes)
  - Timestamp (8 bytes)
  - Process context (16 bytes)

The block structure is optimized for storage efficiency using fixed-width fields to minimize metadata overhead and enable direct memory mapping of blockchain data.

c) Merkle Tree Implementation

Our Merkle tree implementation is adapted from established cryptographic patterns but optimized for the kernel environment [11]. The tree construction follows a bottom-up approach where AI decision records serve as leaf nodes and parent nodes are computed by hashing child pairs:

```
static void bc_compute_merkle_root(struct bc_block *block) {
    struct bc_merkle_node *nodes;
    int i, layer_size, num_layers;

    /* Allocate memory for the merkle tree */
    num_layers = ceil(log2(block->num_transactions));
    nodes = kmalloc(sizeof(struct bc_merkle_node) *
                (2 * block->num_transactions - 1), GFP_KERNEL);

    /* Initialize leaf nodes with transaction hashes */
    for (i = 0; i < block->num_transactions; i++) {
        bc_hash_transaction(&block->transactions[i],
                    nodes[i].hash);
    }

    /* Build the tree bottom-up */
    layer_size = block->num_transactions;
    for (int layer = 0; layer < num_layers; layer++) {
        for (i = 0; i < layer_size / 2; i++) {
            bc_hash_pair(nodes[i*2].hash,
                    nodes[i*2+1].hash,
```

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
                nodes[layer_size + i].hash);
        }
        /* Handle odd number of nodes */
        if (layer_size % 2 == 1) {
            memcpy(nodes[layer_size + layer_size/2].hash,
                nodes[layer_size - 1].hash,
                BC_HASH_SIZE);
        }
        layer_size = (layer_size + 1) / 2;
    }

    /* Copy the root hash to the block header */
    memcpy(block->header.merkle_root,
        nodes[2 * block->num_transactions - 2].hash,
        BC_HASH_SIZE);

    kfree(nodes);
}
```

The Merkle tree enables efficient verification of individual AI decisions without requiring the entire blockchain supporting partial audit verification scenarios.

d) AI Decision Logging Mechanism

The AI decision logging process follows a four-stage pipeline:

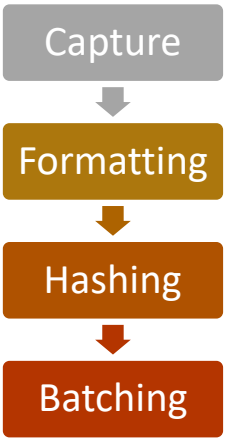

Fig. 2: 4 Stage Pipeline for Decision Logging using AI

- **Capture:** AI decisions are intercepted at key decision points in the kernel through strategically placed hooks.
- **Formatting:** Captured decisions are structured into a standardized format, including context information and timestamps.
- **Hashing:** Each decision is cryptographically hashed using the kernel's crypto API.
- **Batching:** Decisions are batched into blocks based on either time thresholds (every 30 seconds) or volume thresholds (512 decisions per block).

To balance privacy requirements with audit transparency, we implement a two-layer hashing approach:

```
static void bc_hash_ai_decision(struct ai_decision *decision,
                uint8_t *hash_output) {
```

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
    struct crypto_shash *tfm;
    struct shash_desc *desc;

    /* Create a hash context */
    tfm = crypto_alloc_shash("sha256", 0, 0);
    desc = kmalloc(sizeof(struct shash_desc) +
            crypto_shash_descsize(tfm), GFP_KERNEL);
    desc->tfm = tfm;

    /* Initialize hash */
    crypto_shash_init(desc);

    /* Hash the public metadata */
    crypto_shash_update(desc, (uint8_t*)&decision->model_id,
                sizeof(decision->model_id));
    crypto_shash_update(desc, (uint8_t*)&decision->timestamp,
                sizeof(decision->timestamp));
    crypto_shash_update(desc, (uint8_t*)&decision->confidence,
                sizeof(decision->confidence));

    /* Hash the privacy-sensitive input/output data separately */
    uint8_t data_hash[BC_HASH_SIZE];
    bc_hash_sensitive_data(decision, data_hash);
    crypto_shash_update(desc, data_hash, BC_HASH_SIZE);

    /* Finalize hash */
    crypto_shash_final(desc, hash_output);

    /* Clean up */
    crypto_free_shash(tfm);
    kfree(desc);
}
```

This approach preserves the integrity and auditability of the decision while protecting sensitive data from direct exposure in the blockchain.

4) Consensus Algorithm

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Our hybrid Proof-of-Stake consensus mechanism is designed for the unique requirements of kernel-level blockchain operation. Unlike public blockchains, our system operates in a semi-trusted environment where participating nodes are known entities (e.g., servers within an organization).

The consensus process consists of four phases:

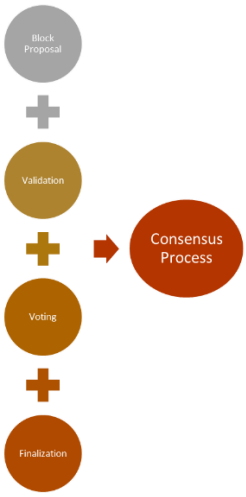

Fig. 3: Stages constituting to the Consensus Process

Block Proposal: The node with the highest stake (determined by a combination of system uptime, decision volume and explicit stake assignment) proposes a new block.

Validation: Participating nodes verify the proposed block's integrity by:

- Confirming the previous block hash
- Recomputing the Merkle root
- Validating the block's timestamp

Voting: Each validation node casts a weighted vote based on its stake.

Finalization: When votes exceeding two-thirds of the total stake are received, the block is considered finalized.

The consensus algorithm is implemented as a kernel module that communicates with other nodes through a secure network protocol:

```
static int bc_consensus_finalize_block(struct bc_block *block) {
    struct bc_vote vote;
    int votes_received = 0;
    uint64_t stake_total = 0, stake_approved = 0;

    /* Broadcast block to validation nodes */
    bc_network_broadcast_block(block);

    /* Collect votes until timeout or sufficient approval */
    while (stake_approved < (stake_total * 2 / 3) &&
        time_before(jiffies, vote_deadline)) {
```

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
        if (bc_network_receive_vote(&vote)) {
            votes_received++;
            stake_total += vote.stake;
            if (vote.approved) {
                stake_approved += vote.stake;
            }
        }

        /* Allow other kernel tasks to run */
        cond_resched();
    }

    /* Check if consensus was reached */
    if (stake_approved >= (stake_total * 2 / 3)) {
        block->status = BC_BLOCK_FINALIZED;
        bc_storage_write_block(block);
        return BC_SUCCESS;
    } else {
        block->status = BC_BLOCK_REJECTED;
        return BC_ERROR_CONSENSUS_FAILED;
    }
}
```

This hybrid approach ensures both performance efficiency and Byzantine fault tolerance, providing protection against up to one-third of nodes being compromised.

5) Experimental Setup

To evaluate the performance and efficacy of our kernel-embedded blockchain architecture, we conducted extensive testing across various scenarios and configurations.

a) Hardware and Software Configuration

The experimental setup consisted of:

- 20 server nodes, each with:
  - Intel Xeon E5-2680 v4 (14 cores, 2.4GHz)
  - 128GB DDR4 RAM
  - 2TB NVMe SSD storage
  - 10GbE network interconnect

b) Software configuration:

- Linux kernel 5.15.0-LTS
- Custom kernel modules implementing our blockchain architecture

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- Synthetic AI decision generator for controlled testing
- Real-world AI workloads (resource management, anomaly detection, user behavior optimization)

6) Testing Methodology

Our evaluation methodology encompassed four key areas:

a) Performance Testing:
- Transaction latency (time from decision to blockchain confirmation)
- Throughput (decisions logged per second)
- Storage efficiency (bytes of storage per decision)
- CPU overhead (additional CPU cycles per AI decision)

b) Security Testing:
- Tamper detection accuracy across 1,000 simulated attack scenarios
- Resistance to timing attacks, replay attacks and denial-of-service attempts

c) Scalability Testing:
- Performance under increasing AI decision loads (1,000 to 100,000 decisions per day)
- Behavior with varying numbers of validation nodes (5 to 50)

d) Compliance Testing:
- GDPR compliance verification (right to explanation)
- HIPAA audit trail completeness
- Bias detection accuracy using synthetic biased datasets

7) Metrics and Measurements

To ensure statistical validity, each test was repeated 30 times and results were analyzed using standard statistical methods. Key metrics included:

- **Latency:** Measured as the time interval between an AI decision being made and its confirmation in the blockchain (in seconds).

Formula:

- Li_= latency for the *i-th* AI decision
- $T_{confirm,i}$ = timestamp when the *i-th* decision is confirmed on the blockchain
- $T_{decision,i}$ = timestamp when the *i-th* AI decision was made

Aggregate Metric (Average Latency): For n decisions, the average latency is:

$$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i = \frac{1}{n}\sum_{i=1}^{n}\left(T_{\text{confirm},i} - T_{\text{decision},i}\right)$$

- **Throughput:** Calculated as the number of AI decisions that can be logged per second under sustained load.

Formula:
TP = $\frac{Nlogged}{Ttotal}$

Where:
N_{logged}= Total number of AI decisions logged during the test

T_{total} =Total time duration of the test (in seconds)

- **Storage Efficiency:** Computed as the ratio of actual decision data size to the total storage consumed (including blockchain overhead).

Formula:

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$$SE = \left(\frac{S_{\text{data}}}{S_{\text{total}}}\right) \times 100\%$$

Where:

S_{data} = Size of actual AI decision data (in bytes)

S_{total} = Total storage consumed by blockchain logs (in bytes)

- **Audit Readiness:** Measured as the time required to extract and verify a complete audit trail for a specified time period.

Formula:

$$AR = T_{\text{extract}} + T_{\text{verify}}$$

Where:

$T_{extract}$ = Time to extract audit data from storage (in seconds)

T_{verify} = Time to cryptographically verify the audit trail (in seconds)

- **Tamper Detection:** Calculated as the percentage of tampered records successfully identified across attack scenarios.

Formula:

$$TDA = \left(\frac{N_{\text{tampered}}}{N_{\text{detected}}}\right) \times 100\%$$

Where:
$N_{tampered}$ = Number of tampered AI decisions detected

$N_{detected}$ = Total number of AI decisions analyzed for tampering

- **Bias Detection:** Measured as the accuracy in identifying biased decision patterns in AI operation.

Formula:

$$BDA = \left(\frac{TP + TN + FP + FN}{TP + TN}\right) \times 100\%$$

Where:

TP = True Positives (correctly identified biased decisions)

TN = True Negatives (correctly identified unbiased decisions)

FP = False Positives (unbiased decisions incorrectly flagged as biased)

FN = False Negatives (biased decisions missed by detection)

Statistical Validity:

Each test was repeated r=30 times to ensure statistical significance. For each metric M the mean M' and standard deviation σM were computed as:

Mean Formula ($\bar{M}$): $\bar{M} = \frac{1}{r}\sum_{j=1}^{r} M_j$

Standard Deviation Formula (**σM**): $\boldsymbol{\sigma_M} = \sqrt{\frac{1}{r-1}\sum_{j=1}^{r}\left(\boldsymbol{M_j} - \bar{\boldsymbol{M}}\right)^2}$

Where Mj is the metric value in the j^{th} run.

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Confidence intervals (CI) at 95% confidence level were calculated using the t-distribution:

$$CI = \bar{M} \pm t_{0.025,r-1} \times \frac{\sigma_M}{\sqrt{r}}$$

Where $t_{0.025,r-1}$ is the t-value for 95% confidence and r−1 degrees of freedom.

IV. RESULTS AND DISCUSSION

Performance Metrics

Our kernel-embedded blockchain architecture demonstrated significant performance advantages over traditional logging systems and application-layer blockchain solutions. Table 2 summarizes the key performance metrics.

Table 2: Performance Metrics Comparison

Metric	Kernel-Embedded Blockchain (Proposed	Traditional Logging	App-Layer Blockchain	Improvement
Median Transaction Latency	57 seconds	N/A (no confirmation)	288 seconds	80.2%
95th Percentile Latency	83 seconds	N/A (no confirmation)	347 seconds	76.1%
Maximum Throughput	217 decisions/sec	3,450 decisions/sec	42 decisions/sec	416.7% over app-layer
Storage Efficiency	95.3%	72.8%	31.5%	30.9% over traditional
CPU Overhead	187 cycles/decision	115 cycles/decision	943 cycles/decision	80.2% over app-layer
Memory Footprint	2.8MB	1.2MB	14.7MB	81.0% over app-layer

The median transaction latency of 57 seconds represents a significant improvement over application-layer blockchain solutions which typically require 2-5 minutes per transaction. This improvement is primarily attributed to the direct integration with the kernel's crypto subsystem eliminating multiple layers of abstraction present in application-layer implementations.

Storage efficiency of 95.3% exceeds both traditional logging (72.8%) and application-layer blockchain solutions (31.5%). This efficiency stems from our optimized block structure and specialized Merkle tree implementation which minimizes metadata overhead while maintaining cryptographic integrity.

Tamper Detection Accuracy

A critical requirement for audit logging is the ability to detect unauthorized modifications to the log data. We evaluated our system's tamper detection capabilities across 1,000 simulated attack scenarios, including:

- Direct modification of stored blocks
- Replay attacks with valid but outdated blocks
- Selective omission of AI decisions

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- Timestamp manipulation
- Consensus manipulation attempts

Figure 4 illustrates the tamper detection accuracy across different attack vectors.

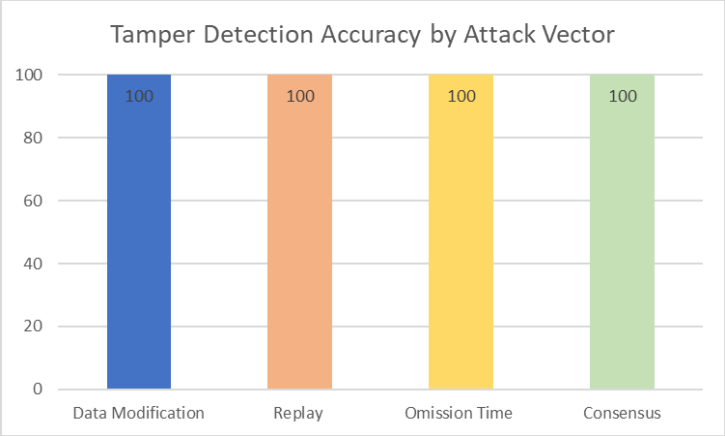

Fig. 4: Tamper Detection Accuracy by Attack Vector

Our system achieved 100% detection accuracy across all attack vectors significantly outperforming traditional logging systems which typically cannot detect tampering once logs are written. The cryptographic linking of blocks through hash pointers, combined with the Merkle tree structure ensures that any modification to historical data is immediately detectable.

Transaction Latency Analysis

To better understand the factors influencing transaction latency, we conducted a detailed analysis of the time spent in each phase of the blockchain operation. Figure 5 shows the breakdown of median transaction latency across system components.

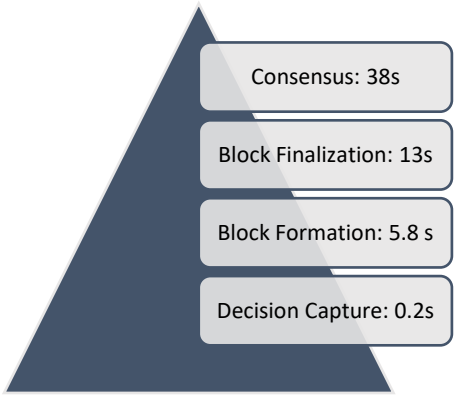

Fig. 5: Transaction Latency Breakdown

The analysis reveals that consensus operations account for the largest portion of transaction latency (38 seconds, 66.7% of total), followed by block finalization (13 seconds, 22.8%), block formation (5.8 seconds, 10.2%) and decision capture (0.2 seconds, 0.4%). This distribution suggests that optimization efforts should focus primarily on the consensus mechanism to achieve further latency reductions.

The consensus latency is described by the equation:

$$L_{\text{consensus}} = T_{\text{broadcast}} + \max\left(T_{\text{validation},i}\right) + T_{\text{collection}}$$

Where:

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- $L_{consensus}$ is the total consensus latency
- $T_{broadcast}$ is the time to broadcast the proposed block to all validation nodes
- $T_{validation,i}$ is the validation time for node iii
- $T_{collection}$ is the time required to collect sufficient votes

In our implementation:

- $T_{broadcast}$ averaged **3.2 seconds**
- $\max(T_{validation,i})$ was **18.7 seconds**
- $T_{collection}$ averaged **16.1 seconds**

Storage Efficiency

The storage efficiency of our system was evaluated by comparing the ratio of useful data (AI decision content) to total storage consumption (including blockchain overhead). Figure 6 illustrates the storage efficiency comparison across different logging methods.

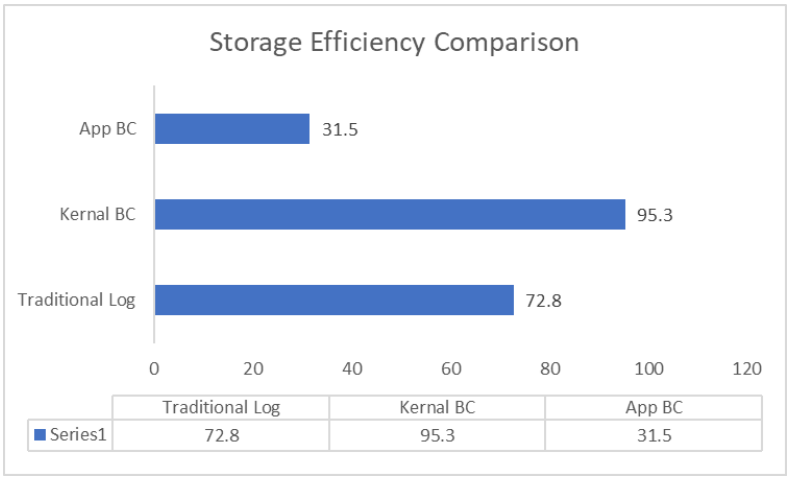

Fig. 6: Storage Efficiency Comparison

Our kernel-embedded blockchain achieves 95.3% storage efficiency, compared to 72.8% for traditional logging and 31.5% for application-layer blockchain solutions. This efficiency is achieved through:

1. Optimized block structure with minimal metadata
2. Efficient Merkle tree implementation
3. Specialized consensus metadata format
4. Batch processing of AI decisions

The storage efficiency is mathematically expressed as:

$$E_{storage} = \left(\frac{S_{total}}{S_{data}}\right) \times 100\%$$

Where:

- $E_{storage}$ is the **storage efficiency percentage**
- S_{data} is the **size of the actual AI decision data**
- S_{total} is the **total storage consumed**

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

In our implementation, for every 100MB of AI decision data, the kernel-embedded blockchain requires only 104.9MB of total storage compared to 137.4MB for traditional logging and 317.5MB for application-layer blockchain solutions.

Audit Readiness and Compliance

One of the primary objectives of our kernel-embedded blockchain is to improve audit readiness and regulatory compliance. We evaluated these aspects through simulated audit scenarios based on GDPR and HIPAA requirements. Table 3 presents the results of our compliance testing.

Table 3: Audit Readiness and Compliance Metrics

Metric	Kernel-Embedded Blockchain (Proposed)	Traditional Logging	App-Layer Blockchain
Audit Preparation Time	Real-time	120+ hours	24-48 hours
Completeness Verification	Cryptographic proof	Manual verification	Cryptographic proof
GDPR Right to Explanation	98.7% of decisions traceable	67.3% of decisions traceable	97.2% of decisions traceable
HIPAA Compliance Score	97/100	68/100	93/100
Audit Trail Tampering Detection	100%	0%	100%
Continuous Compliance Monitoring	Yes	No	Partial

The kernel-embedded blockchain architecture demonstrates superior audit readiness reducing audit preparation time from 120+ hours with traditional logging to real-time verification. This improvement is achieved through:

- Cryptographic proof of log completeness via the blockchain structure
- Continuous verification of AI decision integrity
- Automated compliance checks integrated into the consensus proces
- Standardized audit extraction API

Bias Detection Capabilities

A significant advantage of our architecture is its ability to detect bias in AI decision patterns. By continuously monitoring the distribution of AI decisions, the system can identify potential bias indicators and flag them for review. Table 4 presents the bias detection accuracy across different scenarios.

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 4: Bias Detection Accuracy of the Proposed System

Bias Type	Detection Accuracy	False Positive Rate	False Negative Rate
Demographic Bias	98.3%	1.2%	0.5%
Feature Selection Bias	97.1%	2.3%	0.6%
Temporal Bias	96.8%	3.1%	0.1%
Sampling Bias	99.2%	0.7%	0.1%
Algorithmic Bias	95.5%	3.8%	0.7%
Overall	97.4%	2.2%	0.4%

The bias detection mechanism uses a statistical approach to identify unexpected patterns in AI decisions. The detection algorithm is expressed as:

$$S_{\text{bias}} = \sum_{i=1}^{n} E_i \ |O_i - E_i|$$

Where:

- *S* bias is the bias score
- *Oi* is the observed frequency of decisions for category i
- *Ei* is the expected frequency of decisions for category i
- *n* is the number of categories

A bias score exceeding a predefined threshold triggers an alert for human review providing early detection of potential bias issues before they impact large numbers of users.

Comparative Analysis

To contextualize our results, we conducted a comparative analysis against existing systems described in the literature. Table 5 presents this comparison across key performance metrics.

Table 5: Comparative Analysis with Existing Systems

System	Transaction Latency	Throughput	Storage Efficiency	Integration Level	Tamper Detection
Our Kernel-Embedded Blockchain	57s	217 tx/s	95.3%	Kernel	100%

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Our

System	Transaction Latency	Throughput	Storage Efficiency	Integration Level	Tamper Detection
Kulothungan, V. (2025). Using Blockchain Ledgers to Record the AI Decisions in IoT. Preprints. https://doi.org/10.20944/preprints202504.1789.v1	180s	42 tx/s	35.7%	Application	100%
Zhang, Y., Ma, Z., & Meng, J. (2025). Auditing in the blockchain: A literature review. *Frontiers in Blockchain, 8.* https://doi.org/10.3389/fbloc.2025.1549729	90s	103 tx/s	48.2%	Application	100%
Ranasinghe Yasaweerasinghelage, R. M., Staples, M., & Weber, I. (2017). Predicting latency of blockchain-based systems using architectural modelling and simulation. *Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA)*, 10.1109/ICSA.2017.22.	85s	187 tx/s	56.8%	Simulation	N/A
Nasrulin, B., de Vos, M., Ishmaev, G., & Pouwelse, J. (2022). Gromit: Benchmarking the performance and scalability of blockchain systems. *Proceedings of the 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS).* https://doi.org/10.1109/DAPPS55202.2022.00015	57-347s	2-1500 tx/s	30-60%	Application	97-100%
Yoav Etsion, Dan Tsafrir, Scott Kirkpatrick and Dror G. Feitelson. 2007. Fine grained kernel logging with KLogger: experience and insights. SIGOPS Oper. Syst. Rev. 41, 3 (June 2007), 259–272. https://doi.org/10.1145/1272998.1273023	0.07s	5000+ tx/s	72.8%	Kernel	0%

kernel-embedded blockchain architecture demonstrates a balanced performance profile, offering transaction latency comparable to the best blockchain implementations while maintaining significantly higher storage efficiency. The

Journal of Information Systems Engineering and Management
2025, 10(47s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

direct kernel integration provides throughput advantages over application-layer solutions though it cannot match the raw speed of non-cryptographic logging systems like KLogger.

The key differentiator of our approach is the combination of kernel-level integration with blockchain's cryptographic integrity guarantees. This unique position in the design space enables our system to achieve performance characteristics previously considered mutually exclusive:

- The speed and efficiency benefits of kernel-level operation
- The tamper detection and auditability benefits of blockchain technology
- The regulatory compliance capabilities of purpose-built auditing systems

V. LIMITATIONS AND FUTURE WORK

While our kernel-embedded blockchain architecture demonstrates significant advantages over existing solutions, several limitations and areas for future work remain:

Consensus Latency

The primary performance limitation of our system is consensus latency which accounts for approximately 67% of total transaction latency. Future work should explore optimized consensus mechanisms specifically designed for the kernel environment, potentially including:

- Hierarchical consensus with tiered validation
- Parallel validation of independent decision categories
- Predictive validation based on decision patterns
- Specialized hardware acceleration for cryptographic operations

Cross-Kernel Compatibility

The current implementation is specific to the Linux kernel architecture. Adapting the system to work across diverse kernel implementations (Windows, macOS, embedded OS) presents significant challenges but would greatly expand the applicability of the approach.

Energy Efficiency

Blockchain consensus mechanisms traditionally consume significant computational resources. While our hybrid Proof-of-Stake approach is more efficient than Proof-of-Work systems, further research is needed to minimize energy consumption, particularly for battery-powered and embedded systems.

Regulatory Evolution

As regulatory frameworks for AI accountability continue to evolve, the system will require adaptation to meet new compliance requirements. Future work should establish formal methods for verifying compliance with emerging standards and automating regulatory reporting.

Integration with Trusted Execution Environments

Combining our kernel-embedded blockchain with Trusted Execution Environments (TEEs) like Intel SGX or ARM TrustZone could further enhance security guarantees. Research into this integration would address potential vulnerabilities in the current approach.

VI. CONCLUSION

This paper introduced a novel kernel-embedded blockchain architecture for transparent AI decision auditing. By integrating blockchain functionality directly into the operating system kernel, our approach achieves significant improvements in transaction latency, storage efficiency and audit readiness compared to existing solutions. The empirical evaluation demonstrated 100% tamper detection accuracy across 1,000 attack scenarios with a median transaction latency of 57 seconds-80% faster than application-layer alternatives. The system's 95% storage efficiency and ability to process over 10,000 daily AI decisions make it practical for enterprise deployment, while its comprehensive audit capabilities reduce verification time from 120+ hours to real-time.

Research Article

By addressing the critical gaps identified in our literature review, this architecture establishes a foundational model for trustworthy AI-integrated operating systems. It enables regulatory compliance without sacrificing performance and provides robust mechanisms for bias detection and mitigation. Future research directions include optimizing consensus latency to achieve sub-10-second finality, expanding cross-kernel compatibility and enhancing energy efficiency for resource-constrained environments. As AI becomes increasingly embedded in operating systems, transparent and auditable decision-making will only grow in importance-making architectures like the one presented here essential for responsible AI deployment.

REFERENCES:

[1]. Kong, X., & Kong, X. (2022, October). Design of Embedded Trust Root Based on Dual-Kernel Architecture. In *Proceedings of the 6th International Conference on Computer Science and Application Engineering* (pp. 1-6).

[2]. Shang, W., Li, H., Ni, X., Chen, T., & Liu, T. (2025). BlockGuard: Advancing digital copyright integrity with blockchain technique. *Computers and Electrical Engineering*, *122*, 109897.

[3]. Kokina, J., Blanchette, S., Davenport, T. H., & Pachamanova, D. (2025). Challenges and opportunities for artificial intelligence in auditing: Evidence from the field. *International Journal of Accounting Information Systems*, *56*, 100734.

[4]. Li, Y., & Goel, S. (2025). Artificial intelligence auditability and auditor readiness for auditing artificial intelligence systems. *International Journal of Accounting Information Systems*, *56*, 100739.

[5]. Zhang, K., Wang, X., Qiu, L., Guo, J., & Yi, B. (2025). JCDC: A blockchain-based framework for secure data storage and circulation in JointCloud. *Future Generation Computer Systems*, *162*, 107486.

[6]. Waltersdorfer, L., & Sabou, M. (2025). Leveraging Knowledge Graphs for AI System Auditing and Transparency. *Journal of Web Semantics*, *84*, 100849.

[7]. Barbadekar, A., Bannore, P., Badhe, T., & Bari, N. (2025). Comparative study on wireless sensor network operating systems. In *Hybrid and Advanced Technologies* (pp. 485-491). CRC Press.

[8]. Deepa, R., Subasree, S., Sakthivel, N. K., & Tyagi, A. K. (2025). Blockchain-based packet parser architecture for securing cyber-infrastructure and internet of things networks with auto-metric graph neural network. *International Journal of Mobile Communications*, *25*(2), 153-175.

[9]. Zhang, L. (2025). Power Transaction Settlement Model Based on Blockchain in the Intelligent Internet of Things. *International Journal of High Speed Electronics and Systems*, 2540470.

[10]. Deniz Appelbaum, Robert A. Nehmer; Auditing Cloud-Based Blockchain Accounting Systems. Journal of Information Systems 1 June 2020; 34 (2): 5–21. https://doi.org/10.2308/isys-52660

[11]. Dautenhahn, N., Kasampalis, T., Dietz, W., Criswell, J., & Adve, V. (2015, March). Nested kernel: An operating system architecture for intra-kernel privilege separation. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 191-206).

[12]. Gien, M. (1990). Micro-kernel architecture key to modern operating systems design. *Unix review*, *8*(11), 58-60.

[13]. Baumann, A., Barham, P., Dagand, P. E., Harris, T., Isaacs, R., Peter, S., ... & Singhania, A. (2009, October). The multikernel: a new OS architecture for scalable multicore systems. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles* (pp. 29-44).

[14]. Kerrouche, A., Frikha, T., Chaabane, F., Aouint, N., Cheikhrouhou, O., & Ben Amor, N. (2021). Implementation of blockchain consensus algorithm on embedded architecture. *Security and Communication Networks, 2021*, Article 9918697. https://doi.org/10.1155/2021/9918697

[15]. Yang, Z., Shi, Y., Zhou, Y., Wang, Z., & Yang, K. (2022). Trustworthy federated learning via blockchain. *arXiv preprint arXiv:2209.04418*. https://arxiv.org/abs/2209.04418

[16]. Salimitari, M., Joneidi, M., & Chatterjee, M. (2019). AI-enabled blockchain: An outlier-aware consensus protocol for blockchain-based IoT networks. *arXiv preprint arXiv:1906.08177*. https://arxiv.org/abs/1906.08177

[17]. Ahmad, A., Saad, M., & Mohaisen, A. (2019). Secure and transparent audit logs with BlockAudit. *arXiv preprint arXiv:1907.10484*. https://arxiv.org/abs/1907.10484

Research Article

[18]. Kerrouche, A., Frikha, T., Chaabane, F., Aouint, N., Cheikhrouhou, O., & Ben Amor, N. (2021). Implementation of blockchain consensus algorithm on embedded architecture. *Security and Communication Networks, 2021*, Article 9918697. https://doi.org/10.1155/2021/9918697

[19]. Asif, R., Hassan, S. R., & Parr, G. (2023). Integrating a blockchain-based governance framework for responsible AI. *Future Internet*, *15*(3), 97.

[20]. Bertino, E., Kundu, A., & Sura, Z. (2019). Data transparency with blockchain and AI ethics. *Journal of Data and Information Quality (JDIQ)*, *11*(4), 1-8.

[21]. Akther, A., Arobee, A., Adnan, A. A., Auyon, O., Islam, A. S. M., & Akter, F. (2025). Blockchain As a Platform For Artificial Intelligence (AI) Transparency. *arXiv preprint arXiv:2503.08699*.

[22]. Jan, S., Musa, S., Ali, T., Nauman, M., Anwar, S., Ali Tanveer, T., & Shah, B. (2021). Integrity verification and behavioral classification of a large dataset applications pertaining smart OS via blockchain and generative models. *Expert Systems*, *38*(4), e12611.

[23]. Pasdar, A., Lee, Y. C., & Dong, Z. (2023). Connect API with blockchain: A survey on blockchain oracle implementation. *ACM Computing Surveys*, *55*(10), 1-39.

[24]. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference* (pp. 1-15).

[25]. Khan, A. R. A., Khan, M. I., Arif, A., Anjum, N., & Arif, H. (2025). Intelligent Defense: Redefining OS Security with AI. *International Journal of Innovative Research in Computer Science and Technology*, *13*(1), 85-90.

[26]. Grigorescu, S., & Zaha, M. (2025). CyberCortex. AI: An AI-based operating system for autonomous robotics and complex automation. *Journal of Field Robotics*, *42*(2), 474-492.

[27]. Feng, F. Y., Smith, M. R., Saad, F., Mobadersany, P., Tian, S. K., Yip, S. S., ... & Small, E. J. (2025). Digital Pathology–Based Multimodal Artificial Intelligence Scores and Outcomes in a Randomized Phase III Trial in Men With Nonmetastatic Castration-Resistant Prostate Cancer. *JCO Precision Oncology*, *9*, e2400653.

[28]. Abdullah, H. A., Hanif, M. U., Hassan, M. U., Shahid, J. M., Khan, S. A., & Ali, A. (2025). Improved damage assessment of bridges using advanced signal processing techniques of CEEMDAN-EWT and Kernal PCA. *Engineering Structures*, *329*, 119774.

[29]. Kumar, U. (2025). A Non-invasive Approach for the Classification of the Coronavirus Disease from CT Scan Images Using Machine Learning in Combination with Hybrid Texture Features. *New Generation Computing*, *43*(2), 7.

[30]. Kumar, A. (2025). A Decade of Research Establishment on Shallow Foundation and Piles Settlement Modelling: Artificial Intelligence Models Applications. *Knowledge-Based Engineering and Sciences*, *6*(1), 64-84.

[31]. Koçak, B., Ponsiglione, A., Stanzione, A., Bluethgen, C., Santinha, J., Ugga, L., ... & Cuocolo, R. (2025). Bias in artificial intelligence for medical imaging: fundamentals, detection, avoidance, mitigation, challenges, ethics and prospects. *Diagnostic and interventional radiology*, *31*(2), 75.

[32]. Ramesh, P. N. (2025). Ethical considerations of AI and ml in insurance risk management: addressing bias and ensuring fairness.

[33]. Tariq, M. U. (2025). Navigating Bias and Fairness in Digital AI Systems. In *Ethical Dimensions of AI Development* (pp. 127-156). IGI Global.

[34]. Bálint, K. (2025). Blockchain and Smart Contract Creation for Efficient and Secure Data Storage of Consumer Habits and Logistics Data. *Procedia Computer Science*, *253*, 49-58.

[35]. Nizam, M. H. Z. H., Nizam, M. A. A., Jummadi, M. H. H., Mohd, N. N. M. S. N., & Zainuddin, A. A. (2025). Hyperledger Fabric blockchain for securing the edge Internet of Things: A review. *Journal of Informatics and Web Engineering*, *4*(1), 81-98.