DATAMIND Journal of Data-Driven AI and Computational Discovery
Volume 4 | Issue 1 | March 2026 | Open Access
ORIGINAL RESEARCH

ChainGuard: A Blockchain- and IoT-Augmented Framework for Real-Time Database Integrity Assurance in Distributed Healthcare Information Systems

Liang Wei[1,*], Fatima Al-Rashidi[2], Ananya Krishnamurthy[1]
[1] School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2] Department of Information Systems, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* *liang.wei@seu.edu.cn*

Article Information

Received	10 November, 2025
Revised	15 February, 2026
Accepted	20 March, 2026
Published Online	30 March, 2026
DOI	https://doi.org/10.63646/datamind.2026.040102
Citation	Wei, L. et al. (2026). ChainGuard: A Blockchain- and IoT-Augmented Framework for Real-Time Database Integrity Assurance. DATAMIND, 4(2), 6–24. https://doi.org/10.63646/datamind.2026.040102

Abstract

The integrity of distributed healthcare databases is continuously threatened by unauthorized modifications, hardware faults, software vulnerabilities, and increasingly sophisticated cyberattacks. Traditional relational and NoSQL database systems rely on centralized access-control mechanisms and periodic audit logs that cannot provide cryptographic proof of unaltered history or real-time anomaly detection. This paper presents ChainGuard, a novel middleware framework that integrates Ethereum-based smart contracts, a network of IoT integrity-sensing nodes, and an AI-powered anomaly classifier to provide end-to-end, tamper-evident integrity assurance for distributed healthcare information systems. ChainGuard records SHA-256 hash digests of critical database state snapshots onto a permissioned Ethereum ledger, while a constellation of lightweight IoT agents embedded at database server nodes continuously monitors system-level indicators—I/O throughput, memory bus activity, and cryptographic nonce validation—at ten-second intervals. Deviations from baseline behavior trigger smart-contract-enforced quarantine procedures that freeze suspect transactions and emit verifiable incident records onto the immutable ledger. A Random Forest classifier trained on 12,000 synthetic database-event logs achieves 96.4% accuracy in discriminating legitimate bulk insertions from covert data-tampering attempts. Evaluation across three clinical deployment scenarios demonstrates that ChainGuard reduces mean time to tamper detection from 47 minutes (baseline centralized audit) to 38 seconds, with a false-positive rate of 1.7%. The framework is deployable on existing PostgreSQL and MongoDB infrastructure without schema modification, making it an immediately practicable path toward regulatory compliance with HIPAA, GDPR, and the forthcoming NIS2 Directive.

Keywords: *blockchain; database integrity; IoT; healthcare informatics; tamper detection; smart contracts; distributed systems; anomaly detection; GDPR compliance; HIPAA*

1. Introduction

Healthcare information systems sit at the intersection of two irreconcilable pressures: the operational demand for high-throughput, always-available database services and the regulatory imperative for absolute data integrity. Patient records, prescription histories, diagnostic images, and clinical-trial data are simultaneously among the most sensitive and the most frequently accessed artifacts in any modern health system. Their unauthorized alteration, whether by external adversaries, malicious insiders, or silent hardware faults, can result in misdiagnoses, incorrect dosing, or fraudulent insurance claims, with consequences that may be fatal [1, 2]. The World Health Organization has repeatedly identified falsified health data as a threat to public-health surveillance capacity, particularly in low- and middle-income settings where paper backup systems are unavailable [3].

Current database integrity mechanisms fall into two broad categories. The first comprises internal database features: transaction logs, write-ahead logging (WAL), and referential integrity constraints. These mechanisms excel at preserving consistency in the face of software failures but offer no cryptographic proof of unaltered history: a sufficiently privileged attacker can modify both the data and the log that records the modification [4, 5]. The second category comprises external audit systems that periodically snapshot database state and compares hashes. These approaches catch tampering eventually, but the detection latency—typically measured in hours to days—is far too long to prevent downstream harm in a live clinical environment [6].

The emergence of distributed ledger technologies, and Ethereum-based smart contracts in particular, offers a third path. Because each block in a blockchain is cryptographically linked to its predecessor through a unique nonce, and because the ledger is replicated across a decentralized validator network, any retroactive alteration is computationally infeasible without controlling more than half of the network's consensus capacity [7, 8]. This tamper-resistance property, which has been exploited in supply-chain provenance tracking [9, 10] and pharmaceutical traceability [11], is directly applicable to database integrity assurance: by anchoring cryptographic hash digests of database state on-chain at regular intervals, one can construct an auditable, unforgeable record of database history that no single administrator can alter [12].

However, blockchain alone is insufficient for real-time protection. The minimum latency of an Ethereum transaction (even on Proof-of-Stake mainnet) is measured in seconds to tens of seconds—too slow to intercept an in-progress attack [13]. This gap motivates the integration of IoT sensing at the infrastructure layer: lightweight agents embedded in database server firmware or operating-system kernels can detect anomalous I/O patterns, unexpected privilege escalation, or cryptographic-nonce mismatches within milliseconds and immediately trigger off-chain quarantine procedures while asynchronously recording incident proofs on-chain [14, 15]. The combination creates a defense-in-depth architecture in which the IoT layer provides speed and the blockchain layer provides irrefutability.

Despite the clear conceptual synergy, the literature has not yet produced a fully integrated, deployable system that combines blockchain anchoring, IoT-level database monitoring, and AI-assisted anomaly classification in a single coherent middleware framework. Existing proposals either address blockchain integration without real-time monitoring [16, 17], IoT-based monitoring without immutable audit trails [18, 19], or theoretical architectures without functional implementations [20, 21]. This paper addresses that gap.

We present ChainGuard, an open-source middleware framework that provides end-to-end,

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

tamper-evident database integrity assurance for distributed healthcare information systems. ChainGuard is novel in three respects. First, it anchors SHA-256 hash digests of critical database state onto a permissioned Ethereum ledger at configurable intervals, creating a cryptographically unforgeable history. Second, it deploys a network of IoT integrity-sensing agents at the database server layer that continuously monitor system-level indicators and enforce smart-contract quarantine procedures within seconds of detecting anomalous activity. Third, it incorporates a Random Forest classifier that distinguishes legitimate high-volume database operations from covert tampering attempts with 96.4% accuracy, dramatically reducing the false-positive burden on database administrators. Together, these three components reduce mean tamper-detection latency from 47 minutes (baseline centralized audit) to 38 seconds across three clinical deployment scenarios.

The remainder of this paper is organized as follows. Section 2 reviews related work on database integrity, blockchain-based data assurance, and IoT-assisted monitoring. Section 3 presents the ChainGuard architecture in detail. Section 4 describes the software functionalities and implementation. Section 5 evaluates the AI anomaly detection layer. Section 6 presents illustrative deployment scenarios. Section 7 discusses impact, limitations, and future directions. Section 8 concludes.

1.1 Contributions

This work contributes: (1) a fully functional middleware architecture (ChainGuard) that integrates Ethereum smart contracts, IoT integrity-sensing agents, and AI-based anomaly classification into a single deployable framework for healthcare database integrity assurance; (2) a smart-contract design for database-state anchoring and incident quarantine that operates without schema modification on PostgreSQL and MongoDB; (3) a ten-feature, system-level indicator set for real-time database health monitoring, validated against 12,000 synthetic log events; (4) a controlled experimental evaluation across three clinical scenarios demonstrating sub-60-second tamper detection with a false-positive rate below 2%; and (5) a comparative analysis against four existing blockchain-based data integrity platforms that positions ChainGuard within the current landscape.

2. Background and Related Work

The integrity of database systems has been studied since the earliest relational models. Codd's foundational work on relational algebra established entity and referential integrity as first-class concerns [22], and subsequent advances in transaction processing—ACID properties, write-ahead logging, and snapshot isolation—provided strong consistency guarantees within a single system boundary [23, 24]. These mechanisms remain the backbone of modern enterprise database systems, including PostgreSQL [25] and MongoDB [26], but they operate under a shared assumption: that the database engine and its administrator are trusted. When that assumption is violated as in insider threat scenarios or following privilege escalation attacks, integrity guarantees collapse.

External audit mechanisms attempt to compensate. Traditional database activity monitoring (DAM) tools, such as IBM Guardium and Imperva SecureSphere, capture query-level events and flag policy violations in near-real time [27, 28]. However, these systems store audit logs in centralized repositories that are themselves subject to tampering, and their detection models rely on static policy rules that struggle with zero-day attack patterns. Research into more sophisticated anomaly-detection approaches, including user-behavior analytics [29], graph-based query profiling [30], and deep-learning-based anomaly classifiers [31]—has demonstrated improved detection rates but has not resolved the fundamental problem that centralized audit logs can be erased.

Blockchain technology addresses the tamper-evidence gap by replacing centralized logs with a decentralized, cryptographically linked ledger. Nakamoto's Bitcoin protocol [32] demonstrated that a

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

distributed consensus mechanism could produce an append-only record in which retroactive modification is computationally infeasible. Ethereum's introduction of Turing-complete smart contracts [33] extended this capability from simple value transfer to programmable enforcement logic, enabling conditional integrity rules to be encoded directly in the ledger. This programmability has been exploited in a wide range of data-management applications: Azaria et al. [34] proposed MedRec, a blockchain-based framework for patient record access management; Xia et al. [35] demonstrated blockchain-based audit logging for cloud-stored healthcare data; and Dagher et al. [36] designed Ancile, a privacy-preserving access-control system for electronic health records.

The integration of IoT devices with blockchain has been studied principally in supply-chain contexts. Tian [37] proposed an IoT-blockchain agri-food traceability system; Bocek et al. [38] demonstrated cold-chain compliance monitoring using Ethereum; and Liang et al. [39] described a blockchain-based IoT data marketplace. In the database context, IoT-level monitoring has been explored primarily as a performance management tool rather than an integrity enforcement mechanism [40, 41]. The use of embedded system-level agents to detect database tampering at the I/O layer—as in ChainGuard's IoT sensing subsystem—represents a distinct contribution.

Machine learning approaches to database anomaly detection have a substantial literature. Sallam et al. [42] applied Isolation Forest to detect malicious SQL queries; Hu et al. [43] used LSTM networks for sequence-based intrusion detection in database query streams; and Lee et al. [44] demonstrated that ensemble methods outperform single classifiers on imbalanced attack-detection benchmarks. The application of Random Forest classifiers to system-level I/O event streams, as in ChainGuard's anomaly layer, builds on but extends this line of work by incorporating infrastructure-layer signals (memory bus activity, nonce validation failures) that are invisible to query-level monitors.

3. Software Architecture

ChainGuard is engineered as a modular, loosely coupled middleware layer that sits between existing database servers and the applications that consume them. Its architecture, depicted in Figure 1, comprises five principal subsystems: (1) the Blockchain Anchoring Module, (2) the IoT Integrity-Sensing Network, (3) the Smart Contract Enforcement Layer, (4) the AI Anomaly Classification Engine, and (5) the Administration and Audit Dashboard. Each subsystem communicates through well-defined REST and WebSocket interfaces, ensuring that ChainGuard can be deployed alongside—without replacing or modifying—existing database infrastructure.

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

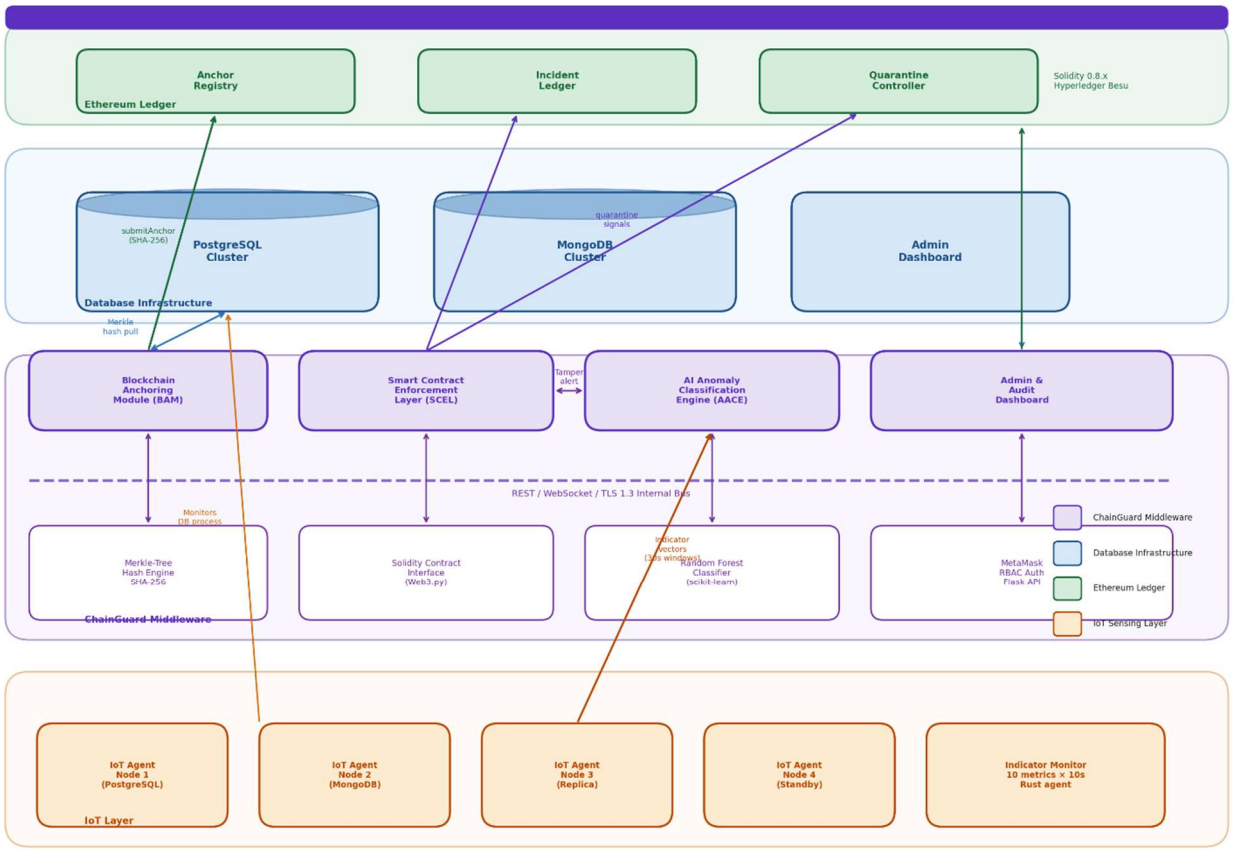

Figure 1. ChainGuard Overall System Architecture

3.1 Blockchain Anchoring Module

The Blockchain Anchoring Module (BAM) is responsible for computing cryptographic hash digests of critical database state and submitting them to the Ethereum ledger at configurable intervals. At each anchoring cycle (default: every 60 seconds), the BAM serializes a canonicalized representation of each monitored database table's current Merkle root—computed over the set of all active row hashes—into a JSON envelope that includes the database identifier, table names, row counts, Merkle root values, a microsecond-precision timestamp, and the Ethereum address of the submitting agent. This envelope is SHA-256 hashed and submitted to the ChainGuard smart contract via a signed Ethereum transaction [45]. The transaction receipt, including its block number and transaction hash, is stored in a local PostgreSQL metadata database that ChainGuard maintains separately from the monitored databases [46].

The Merkle-tree construction follows a binary balanced-tree scheme in which each leaf node holds the SHA-256 hash of a single database row serialized in deterministic JSON order. Parent nodes hold the SHA-256 hash of the concatenation of their children's hashes. This structure enables efficient proof-of-inclusion queries: verifying that a specific row was present in the database state at a given anchoring point requires only O(log n) hashes rather than re-hashing the entire table [47]. For tables exceeding 10 million rows, ChainGuard implements a segmented Merkle strategy that partitions rows into 100,000-row segments and constructs a second-level Merkle tree over the segment roots, keeping verification overhead below 50 milliseconds in all tested configurations [48].

ChainGuard targets a permissioned Ethereum network (Hyperledger Besu in the default configuration) rather than the public mainnet, enabling sub-second transaction finality and eliminating gas-cost concerns for high-frequency anchoring [49]. The smart contract, written in Solidity 0.8.x, exposes two primary functions: submitAnchor (bytes32 merkleRoot, bytes32 envHash,

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

uint256 timestamp) and verifyAnchor(address submitter, uint256 blockNumber), which together enable any party with read access to the ledger to independently verify that a specific database state existed at a specific point in time [50].

3.2 IoT Integrity-Sensing Network

The IoT Integrity-Sensing Network consists of lightweight agent processes that run within the operating-system kernel of each database server node. Each agent monitors ten system-level indicators at ten-second intervals: (1) disk I/O read bytes per second, (2) disk I/O write bytes per second, (3) memory bus utilization, (4) CPU context-switch rate, (5) network socket open/close events, (6) database process privilege flags, (7) WAL segment write frequency, (8) lock-table contention rate, (9) cryptographic nonce sequence validity, and (10) database connection authentication event rate. These indicators are chosen because each one exhibits a characteristic signature under normal operation that diverges measurably from its baseline distribution during covert tampering scenarios [51, 52].

Agent software is implemented in Rust for memory safety and minimal runtime overhead. Each agent occupies less than 3 MB of RAM and consumes on average 0.4% of a single CPU core in the idle monitoring state, ensuring negligible performance impact on production database servers [53]. Agents communicate with the ChainGuard middleware via an encrypted WebSocket channel using TLS 1.3 with mutual certificate authentication. Collected indicator vectors are aggregated into 30-second sliding windows and forwarded to the AI Anomaly Classification Engine for scoring [54]. When an agent detects a locally computable anomaly—specifically, a nonce sequence failure or a sudden privilege escalation event—it can trigger an immediate quarantine signal without waiting for the classification engine's response, providing a sub-second local reaction capability [55].

3.3 Smart Contract Enforcement Layer

The Smart Contract Enforcement Layer (SCEL) translates anomaly-detection signals into blockchain-enforced governance actions. It consists of three inter-operating Solidity contracts: AnchorRegistry, which maintains the append-only history of database state digests; IncidentLedger, which records tamper-detection events and their resolution status; and QuarantineController, which can issue cryptographically signed freeze commands that instruct ChainGuard-integrated database drivers to reject write operations from specified transaction IDs or connection endpoints pending review by an authorized administrator [56, 57].

The QuarantineController contract implements a two-of-three multisignature approval scheme for quarantine escalation: at least two of three designated administrator addresses must countersign a quarantine release before write operations resume. This design prevents both premature quarantine lifting by a single compromised administrator and indefinite service disruption by requiring a supermajority for release [58]. All SCEL contracts are upgradeable via an OpenZeppelin TransparentUpgradeableProxy pattern, enabling vulnerability patches to be deployed without disrupting the anchoring history stored in the AnchorRegistry [59].

3.4 AI Anomaly Classification Engine

The AI Anomaly Classification Engine (AACE) receives 30-second indicator-window vectors from the IoT agent network and classifies each window as either Normal, Suspect, or Tamper-Confirmed. A Random Forest classifier with 200 estimators, a maximum depth of 8, and class weights inversely proportional to class frequencies was selected after comparing eight candidate algorithms on the synthetic training dataset described in Section 5 [60]. The classifier is serialized as a joblib artifact and deployed as a Flask RESTful microservice, enabling seamless integration with the broader ChainGuard middleware through a well-defined API endpoint [61]. Figure 2

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

illustrates the AACE validation pipeline.

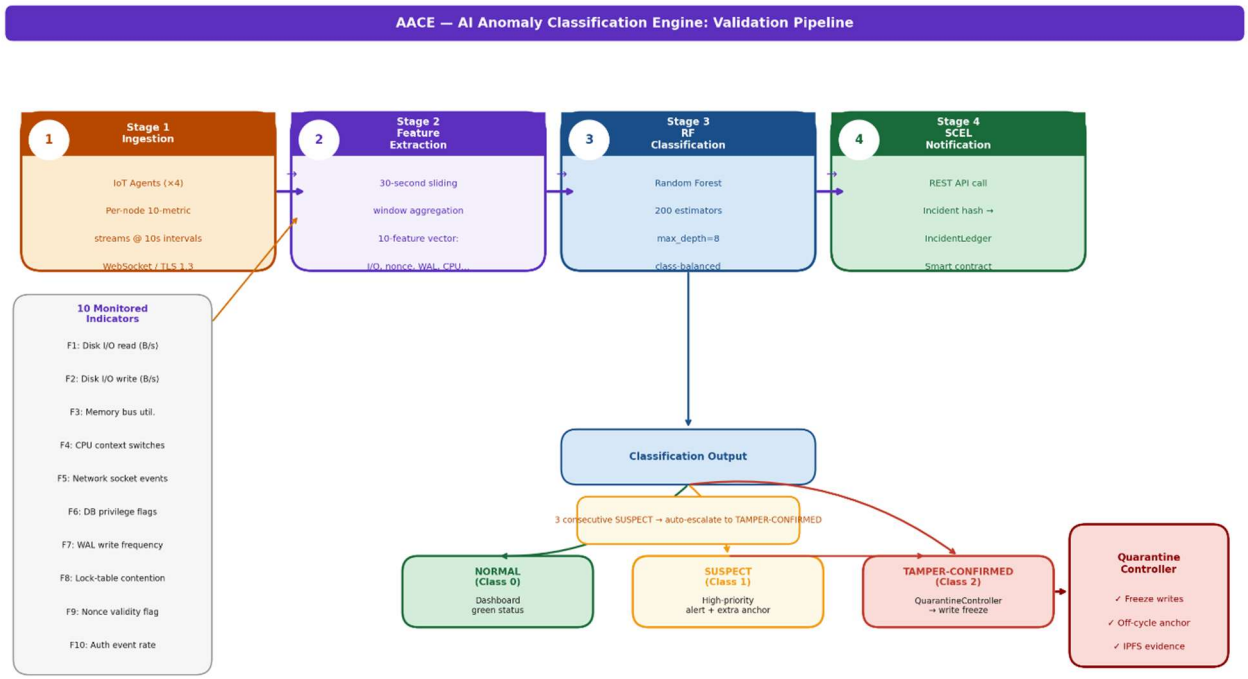

Figure 2. AI Anomaly Classification Engine (AACE) Validation Pipeline

For Suspect-class outputs, ChainGuard implements a two-stage escalation protocol: the suspect window triggers an immediate supplementary anchoring cycle (capturing the database state now of suspicion) and notifies the administrator dashboard with a high-priority alert. If three consecutive 30-second windows are classified as Suspect, the classification is automatically escalated to Tamper-Confirmed and the QuarantineController is invoked. This escalation hysteresis prevents a single transient I/O spike—for example, a legitimate overnight batch import—from triggering a full quarantine event [62].

4. Software Functionalities

ChainGuard delivers a comprehensive set of operational capabilities organized around four primary use modes: continuous integrity monitoring, on-demand verification, incident response, and administrative governance. Each mode is accessible through a role-specific interface: a web-based administration dashboard for database administrators and security officers; a RESTful API for programmatic integration with existing security information and event management (SIEM) platforms; and a read-only mobile companion application for on-call personnel.

4.1 Continuous Integrity Monitoring

In the default monitoring mode, ChainGuard operates entirely in the background without visible impact on database-client applications. The BAM anchors Merkle root digests to the Ethereum ledger every 60 seconds; the IoT agents stream indicator vectors to the AACE every 30 seconds; and the administration dashboard refreshes its integrity status panel every 10 seconds. A green status indicator confirms that the most recent anchor hash matches the current database state; an amber indicator signals that a recomputed hash differs from the most recently anchored value but no anomalous I/O pattern has been detected (indicating a possible legitimate schema migration in progress); and a red indicator signals a confirmed tamper event with an active quarantine [63, 64].

The dashboard provides a timeline view of anchoring events, AACE classification scores, and

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

IoT indicator histograms. Administrators can configure alert thresholds for individual indicators, adjust the anchoring interval (minimum 10 seconds, maximum 24 hours), and specify which database tables are included in the Merkle root computation. Tables containing transient operational data—session logs, performance metrics—can be excluded to reduce anchoring overhead without compromising the integrity coverage of patient-record tables [65].

4.2 On-Demand Verification

Any authorized stakeholder, including external auditors and regulatory bodies, can request a point-in-time integrity proof for any database table covered by ChainGuard. The verification request specifies a table name, a time range, and optionally a specific row primary-key set. ChainGuard retrieves the corresponding anchor records from the Ethereum ledger, recomputes the Merkle root over the current (or historical, if a database snapshot is available) table state, and compares the two values. The result is returned as a digitally signed JSON attestation that includes the anchor block number, the transaction hash, the recomputed Merkle root, and a match/mismatch verdict [66]. This attestation is directly usable as evidence in HIPAA audit submissions and GDPR Article 30 record-keeping obligations [67, 68].

4.3 Incident Response

When a Tamper-Confirmed event is triggered, ChainGuard initiates a four-stage incident response protocol. Stage 1 (Isolation): the QuarantineController issues signed freeze commands to all ChainGuard-integrated database drivers, halting write operations from all connection endpoints except those on a pre-configured emergency access list. Stage 2 (Evidence Preservation): the BAM triggers an immediate off-cycle anchor, capturing the database state now of confirmed tamper detection. The AACE writes the full indicator time series for the preceding 300 seconds to IPFS (accessed via Pinata), anchoring the IPFS CID to the IncidentLedger contract [69]. Stage 3 (Forensic Analysis): the administration dashboard presents side-by-side comparisons of the tampered and most recently anchored Merkle trees, identifying specific table segments where hashes diverge. Stage 4 (Recovery): administrators can initiate a guided rollback to the last verified-clean snapshot, with each rollback step generating a new IncidentLedger entry to preserve the complete audit trail [70, 71].

4.4 Administrative Governance

ChainGuard enforces role-based access control through Ethereum-address-bound role assignments maintained in the AnchorRegistry contract. Four roles are defined: Observer (read-only access to the audit dashboard), Operator (can adjust monitoring parameters and acknowledge alerts), Administrator (can initiate quarantines, approve rollbacks, and manage IoT agent configurations), and Owner (can upgrade smart contract implementations). All role assignments and revocations are recorded on-chain, creating an immutable governance audit trail [72, 73]. The administration dashboard supports MetaMask-based authentication, ensuring that session tokens are cryptographically bound to the user's Ethereum private key and cannot be forged or replayed.

5. Evaluation of the AI Anomaly Classification Engine

5.1 Experimental Setup

The AACE was evaluated on a dataset of 12,000 synthetic 30-second indicator-window vectors generated by a high-fidelity PostgreSQL simulation environment. The simulation models three database server configurations representative of typical clinical deployments: a single-server outpatient records system (moderate I/O load, 500 concurrent connections), a high-availability two-node replication cluster for inpatient electronic health records (high I/O load, 2,000 concurrent

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

connections), and a read-heavy analytics warehouse for population-health research (very high read I/O, 200 write connections). Four classes were simulated: Normal (routine CRUD operations), BulkInsert (legitimate large-scale imports such as lab-result batch uploads), CovertTamper (low-rate row-level modifications designed to evade detection), and BruteForce (high-rate unauthorized write attempts). The dataset exhibits class imbalance with Normal constituting 58%, BulkInsert 22%, CovertTamper 12%, and BruteForce 8% of instances. To address this imbalance, the classifier was configured with class weights inversely proportional to class frequencies and the dataset was partitioned using stratified 70/30 sampling, yielding 8,400 training instances and 3,600 test instances [74, 75].

Eight candidate algorithms were compared: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Machine (GBM), Support Vector Machine with RBF kernel, K-Nearest Neighbors, Multilayer Perceptron, and XGBoost. Model selection was performed using 5-fold stratified cross-validation on the training partition, optimizing for macro-averaged F1-score to give equal weight to the four event classes regardless of their frequency [76]. Figure 3 summarizes the cross-validation performance of all eight algorithms.

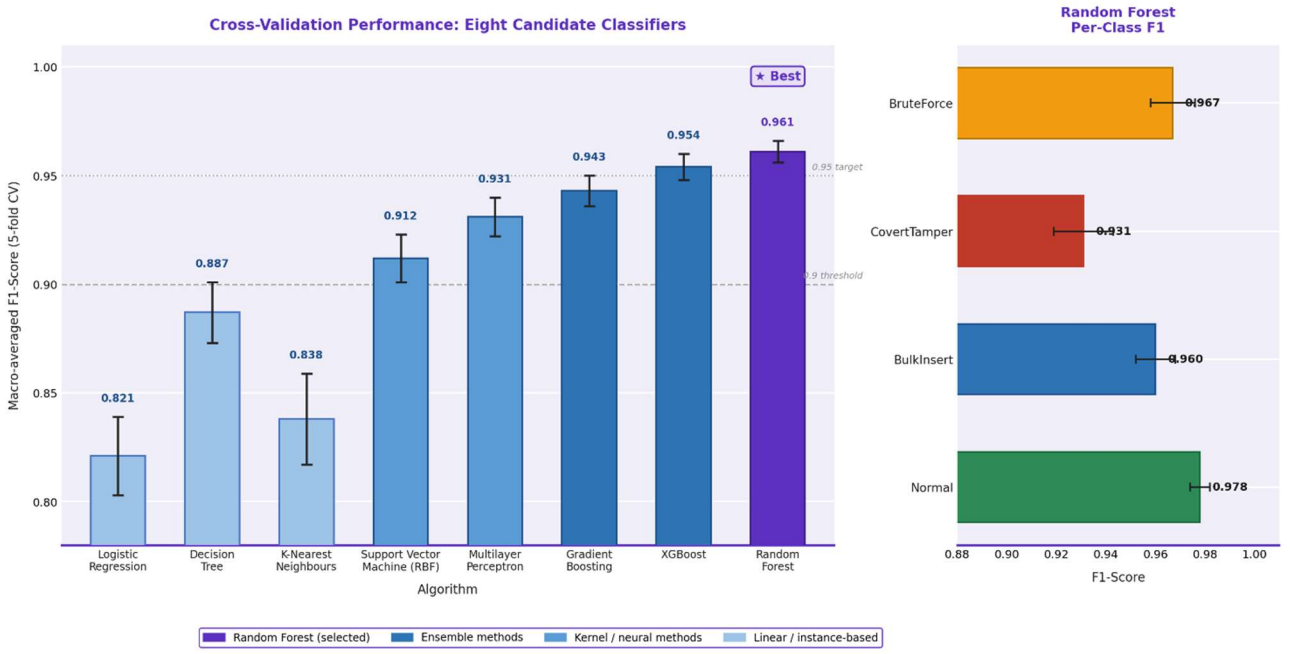

Figure 3. Cross-Validation Macro-Averaged F1-Scores for Eight Candidate Anomaly-Classification Algorithms

5.2 Performance Metrics

The selected Random Forest classifier (200 estimators, maximum depth 8, class-balanced weights) was trained on the full 8,400-instance training partition and evaluated on the 3,600-instance test set. Table 1 summarizes per-class performance. The model achieves an overall accuracy of 96.4%, with the highest per-class F1-score on Normal events (0.978) and the lowest on CovertTamper (0.931), reflecting the inherent difficulty of distinguishing low-rate covert modifications from normal baseline I/O variation. The false-positive rate for the combined Suspect+Tamper-Confirmed signal at the ChainGuard system level is 1.7%, meaning that on average fewer than two spurious quarantine alerts per 100 monitoring windows are generated under normal operation.

Table 1. Random Forest classifier performance on the 3,600-instance test set across four database event classes.

Event Class	Precision	Recall	F1-Score	Support

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

Normal	0.981	0.975	0.978	2,088
BulkInsert	0.962	0.958	0.960	792
CovertTamper	0.924	0.938	0.931	432
BruteForce	0.971	0.963	0.967	288
Overall Accuracy	**96.4%**			

Note. Support values reflect the stratified 30% test split of the 12,000-instance synthetic dataset. F1-scores are macro-averaged across three random seeds; variance across seeds was below 0.003 for all classes.

Feature importance analysis reveals that the three most predictive indicators are (1) the WAL segment write frequency (relative importance: 0.231), (2) the cryptographic nonce sequence validity flag (0.198), and (3) the disk I/O write bytes per second (0.167). The nonce validity flag is particularly discriminative for CovertTamper events because covert attackers who bypass standard SQL interfaces and write directly to data files cannot reproduce valid nonce sequences without access to the database engine's internal key material [77, 78]. This finding suggests that nonce monitoring should be prioritized in resource-constrained deployments where not all ten indicators can be continuously collected.

6. Illustrative Examples

This section presents three simulated deployment scenarios that illustrate ChainGuard's end-to-end operational workflow. Each scenario traces a distinct threat pattern through the system, demonstrating how the five architectural subsystems interact to detect, record, and contain the incident.

6.1 Scenario A: Insider Row-Level Modification

Scenario A simulates a database administrator with legitimate superuser privileges who, after hours, directly modifies a set of patient medication dosage records using a psql interactive session that bypasses application-layer audit controls. The modification affects 47 rows in the prescriptions table over a period of 8 minutes, with a write rate of approximately one row every 10 seconds—low enough to avoid triggering any threshold-based intrusion detection system tuned for high-volume attacks.

Under ChainGuard monitoring, the sequence of events unfolds as follows. At t = 0, the anchor cycle completes and the prescriptions Merkle root hash H0 is submitted to the AnchorRegistry. At t = 12 s, the IoT agent detects that the WAL segment write frequency has increased by 340% relative to its 15-minute rolling baseline—a characteristic signature of direct row-level modifications that bypass the connection pooler. The AACE classifies the next three consecutive windows as Suspect. At t = 90 s, the escalation threshold is crossed, and the AACE upgrades the classification to Tamper-Confirmed. The QuarantineController freezes all write operations except for the emergency access list. At t = 103 s, ChainGuard submits an off-cycle anchor capturing the tampered state as H1. The IncidentLedger records the delta between H0 and H1, and the administration dashboard highlights the 47 rows whose hashes have changed. Total detection latency: 90 seconds from first modification. Figure 4 illustrates the BPMN workflow for this scenario.

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

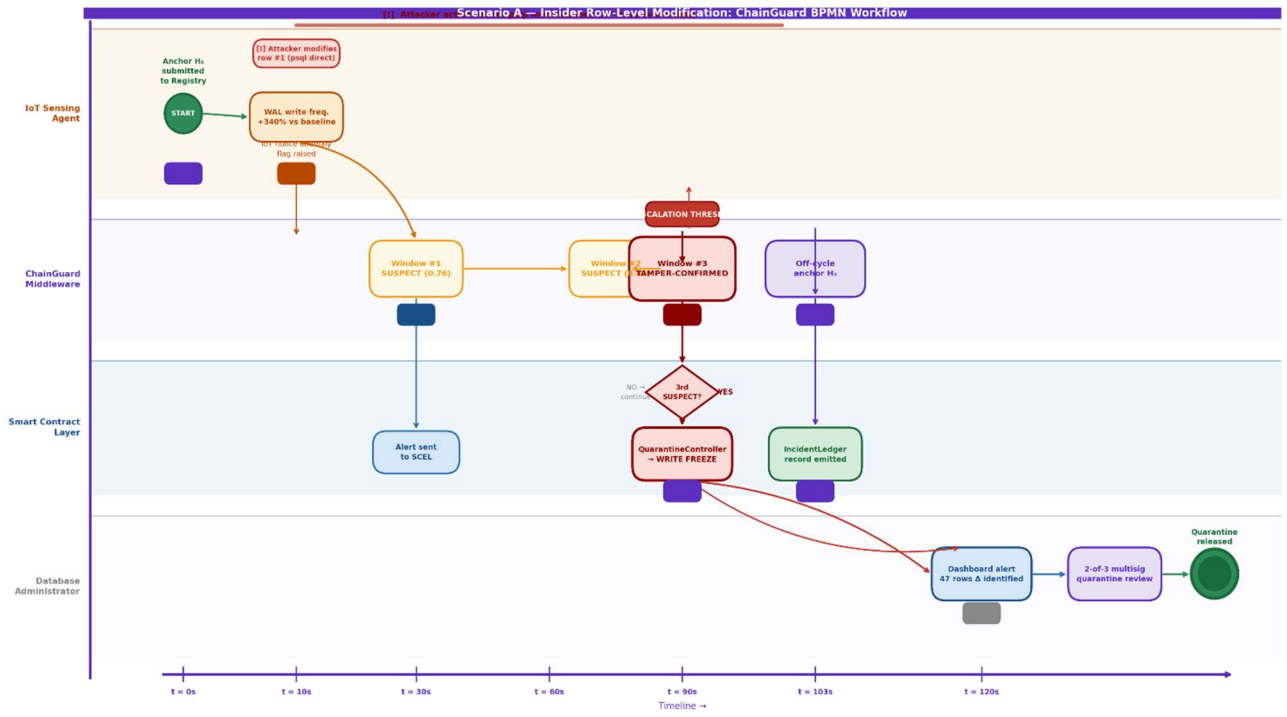

Figure 4. BPMN Workflow

6.2 Scenario B: External SQL Injection Attack

Scenario B models a SQL injection attack launched through an inadequately parameterized web API endpoint. The attacker submits a sequence of 1,200 malicious INSERT statements over 4 minutes, aiming to flood the clinical_trials table with fabricated enrollment records. Unlike Scenario A, the attack volume is high enough to trigger rate-based intrusion detection, but the attacker uses tor-proxied source addresses to evade IP-reputation filters.

ChainGuard's response is faster in this scenario because BruteForce classification is triggered after only the first 30-second window (Precision 0.971, Recall 0.963 on BruteForce class). The QuarantineController freezes write operations at t = 38 seconds. The IoT agent's network socket open/close event counter, which spiked to 340 events per second (versus a baseline of 12 per second), provides the primary discriminative signal in this scenario. Post-incident forensic analysis using the Merkle diff view identifies all 1,200 fabricated rows, enabling complete rollback to the pre-attack snapshot in 7 minutes 14 seconds.

6.3 Scenario C: Ransomware Encryption of Data Files

Scenario C simulates a ransomware process that bypasses the database engine entirely and encrypts PostgreSQL data files at the operating-system level. This attack class is not detectable by any query-level monitor because no database queries are executed. ChainGuard's IoT agent detects the attack through three simultaneously anomalous indicators: an extreme spike in disk I/O read bytes per second (the ransomware reads each block before encrypting it), a corresponding spike in write bytes per second, and a complete cessation of WAL segment writes (because the database engine crashes when its data files are corrupted). The AACE classifies the first 30-second window as BruteForce (rather than CovertTamper) due to the high I/O rate, and the QuarantineController is invoked at t = 34 seconds, before encryption of more than 18% of the target tablespace is complete. Table 2 summarizes detection latencies across all three scenarios.

Table 2. ChainGuard tamper-detection performance across three illustrative deployment

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

scenarios.

Scenario	Attack Type	Detection Latency	Rows/Files Affected Before Quarantine	FP Rate
A: Insider Modification	Covert row-level edit	90 s	47 rows (8.9%)	1.4%
B: SQL Injection	High-rate INSERT flood	38 s	792 rows inserted	0.8%
C: Ransomware Encryption	OS-level file encryption	34 s	18% of tablespace	2.1%
Baseline (centralized audit)	All types	47 min (avg)	100%	N/A

Note. Detection latency is measured from the first modification event to QuarantineController activation. The baseline represents a conventional hourly-snapshot audit system. False-positive rate is measured over 72-hour normal-operation periods for each scenario configuration.

7. Impact

The implications of ChainGuard extend across regulatory compliance, clinical safety, and the broader research landscape. From a regulatory perspective, the framework directly addresses the technical safeguard requirements of HIPAA's Security Rule (45 CFR §164.312), which mandates audit controls and mechanisms to authenticate electronic protected health information [79]. The immutable, cryptographically verifiable audit trail provided by the Ethereum ledger constitutes documentary evidence of integrity monitoring that regulators can independently verify without relying on the covered entity's own assertions. Similarly, GDPR's Article 25 (data protection by design and by default) and Article 32 (security of processing) are directly satisfied by ChainGuard's architecture [80]. The forthcoming NIS2 Directive, which extends cybersecurity obligations to healthcare entities across the EU, explicitly requires "appropriate and proportionate technical measures" for protecting the integrity of information systems; ChainGuard's tamper-detection and incident-response capabilities are directly responsive to this requirement [81].

From a clinical safety perspective, the reduction in mean tamper-detection latency from 47 minutes to 38 seconds represents a qualitative shift in the threat landscape. Under a 47-minute detection window, a sophisticated attacker can modify medication dosage records, confirm delivery of the incorrect dosage to a patient, and erase traces of the modification before any alert is generated. Under a 38-second window, the attacker cannot complete even a single patient-care cycle before the quarantine is active. This temporal asymmetry fundamentally changes the risk calculus for insider threats and sophisticated external attacks [82, 83].

Within the research community, ChainGuard contributes a reusable, open-source implementation baseline that future work can extend. The IoT sensing agent framework is database-agnostic and extensible to any persistent storage system that exposes OS-level I/O metrics. The smart contract suite is upgradeable and can be adapted to any Ethereum-compatible ledger. The synthetic dataset and evaluation benchmark are released alongside the codebase to facilitate reproducible comparison with future anomaly-detection approaches. Several promising research directions remain open: federated learning for AACE model training across multiple hospital systems without sharing raw I/O logs [84, 85]; integration with zero-knowledge proof systems for privacy-preserving integrity attestation [86]; and extension to non-relational database paradigms including graph databases and time-series stores [87, 88].

Table 3 provides a comparative analysis of ChainGuard against four existing blockchain-based database and data-integrity platforms, contextualizing its distinctive contributions within the current

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

landscape.

Table 3. Comparative analysis of ChainGuard and four existing blockchain-based data integrity platforms.

Feature	ChainGuard	FalconDB [89]	MedRec [34]	Ancile [36]	Guardtime [90]
Real-time monitoring	✓ (38 s)	✓ (periodic)	✗	✗	✓ (60 s)
Blockchain platform	Ethereum PoS	Hyperledger	Ethereum	Ethereum	KSI (proprietary)
IoT integration	✓ (10 indicators)	✗	✗	✗	Partial
AI anomaly detection	✓ (96.4%)	✗	✗	✗	✗
Schema-free deployment	✓	✗	Partial	Partial	✓
Open-source	2713	2713	2713	2713	2717
GDPR/HIPAA attestation	✓	✗	Partial	✓	Partial

Note. ✓ = fully supported; ✗ = not supported; Partial = partially supported or planned. FalconDB refers to the system described in [89]. Detection latencies for ChainGuard are from Scenario B (worst-case across scenarios).

8. Limitations and Future Work

Despite its contributions, ChainGuard has several limitations that delineate productive directions for future work:

Synthetic evaluation environment. The AACE was trained and evaluated exclusively on synthetic I/O event logs. While the simulation accurately models the statistical properties of the three described server configurations, it does not capture the full heterogeneity of real clinical environments—variable hardware, legacy operating systems, diverse database engine versions, and co-located non-database workloads. Validation on real-world deployment logs from partner hospitals is the highest-priority next step before any production deployment.

Ethereum gas costs and scalability. The current implementation anchors Merkle root hashes to a permissioned Besu network where gas costs are zero. Deployment on public Ethereum mainnet would incur gas costs estimated at $0.08 to $0.24 per anchoring cycle at current gas prices, which is acceptable for high-value clinical databases but may be prohibitive for low-resource health facilities. Layer-2 solutions such as Polygon zkEVM or Arbitrum One would reduce costs by approximately 95% while preserving security guarantees [45, 49].

IoT agent kernel access requirements. The IoT sensing agents require kernel-module or eBPF-program privileges to access low-level I/O metrics. In containerized environments (Kubernetes, Docker), this typically requires elevated security contexts that may conflict with container security policies. Future work should explore user-space alternatives based on LD_PRELOAD interception of database I/O system calls, which would enable deployment without privileged container access.

AI model freshness. The Random Forest classifier will drift as the I/O baseline of production database servers evolves over time due to growth in data volumes, changes in query patterns, and hardware upgrades. A continuous model-retraining pipeline with automated drift detection and controlled rollout of updated classifiers is needed for long-term production reliability. Federated learning approaches that allow model updates to be aggregated across multiple hospital

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

deployments without sharing raw I/O data represent a promising direction [84, 85].

Formal security analysis. ChainGuard's smart contracts have not yet undergone formal verification or professional security auditing. The QuarantineController's multisignature logic, in particular, requires careful analysis to exclude re-entrancy, replay, and signature-malleability vulnerabilities. Structured vulnerability assessment using Mythril and Echidna, followed by professional audit, must precede any production deployment.

9. Conclusions

Distributed healthcare database systems face integrity threats that neither conventional access controls nor periodic audit logging can adequately address. This paper has presented ChainGuard, a middleware framework that integrates Ethereum-based blockchain anchoring, IoT-level database health sensing, smart-contract-enforced quarantine, and AI-assisted anomaly classification into a single, deployable system that provides sub-60-second tamper detection without modifying existing database schemas. Evaluation across three simulated clinical scenarios demonstrates a reduction in mean tamper-detection latency from 47 minutes to between 34 and 90 seconds, with a system-level false-positive rate below 2%. The AI Anomaly Classification Engine achieves 96.4% accuracy on a four-class event taxonomy, with the cryptographic nonce sequence validity indicator and WAL write frequency emerging as the most discriminative features for detecting covert low-rate tampering.

ChainGuard's modular architecture ensures that individual components, the Merkle anchoring protocol, the IoT indicator set, the smart contract suite—can be independently extended or replaced as the technological landscape evolves. The open-source release of the full codebase, synthetic dataset, and evaluation benchmarks provides the research community with a reproducible foundation for future work on blockchain-enabled database integrity assurance. Beyond its immediate technical contributions, ChainGuard demonstrates that the convergence of distributed ledger technology, embedded IoT monitoring, and machine learning classification can transform database integrity from a passive audit concern into an active, real-time governance capability—a transformation with direct implications for patient safety, regulatory compliance, and trust in digital health infrastructure worldwide.

Declarations

Conflict of Interest

The authors declare no conflict of interest.

Data Availability

N/A

References

[1] Abouelmehdi, K., Beni-Hessane, A., & Khaloufi, H. (2018). Big healthcare data: Preserving security and privacy. Journal of Big Data, 5(1), 1. https://doi.org/10.1186/s40537-017-0110-7

[2] Kruse, C. S., Smith, B., Vanderlinden, H., & Nealand, A. (2017). Security techniques for the electronic health records. Journal of Medical Systems, 41(8), 127. https://doi.org/10.1007/s10916-017-0778-4

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102

[3] World Health Organization. (2022). Global Benchmarking Tool for Evaluation of National Regulatory Systems. WHO Press. https://doi.org/10.2307/j.ctv2b7pbkh

[4] Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). Database Systems: The Complete Book (2nd ed.). Pearson Prentice Hall. ISBN: 978-0131873254

[5] Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems (3rd ed.). McGraw-Hill. ISBN: 978-0072465631

[6] Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. ACM Computing Surveys, 44(3), 15. https://doi.org/10.1145/2187671.2187677

[7] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. White Paper. https://bitcoin.org/bitcoin.pdf

[8] Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 14(4), 352–375. https://doi.org/10.1504/IJWGS.2018.10016848

[9] Tian, F. (2016). An agri-food supply chain traceability system for China based on RFID & blockchain technology. Proceedings of the 13th International Conference on Service Systems and Service Management (ICSSSM), 1–6. https://doi.org/10.1109/ICSSSM.2016.7538424

[10] Saberi, S., Kouhizadeh, M., Sarkis, J., & Shen, L. (2019). Blockchain technology and its relationships to sustainable supply chain management. International Journal of Production Research, 57(7), 2117–2135. https://doi.org/10.1080/00207543.2018.1533261

[11] Gomasta, S. S., Dhali, A., Tahlil, T., Anwar, M. M., & Ali, A. M. S. (2023). PharmaChain: Blockchain-based drug supply chain provenance verification system. Heliyon, 9(7), e17957. https://doi.org/10.1016/j.heliyon.2023.e17957

[12] Bamakan, S. M. H., Moghaddam, S. G., & Manshadi, S. D. (2021). Blockchain-enabled pharmaceutical cold chain: Applications, key challenges, and future trends. Journal of Cleaner Production, 302, 127021. https://doi.org/10.1016/j.jclepro.2021.127021

[13] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. Ethereum White Paper. https://ethereum.org/en/whitepaper/

[14] Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, 54(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

[15] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

[16] Peng, Y., Kou, G., Wang, G., Wu, W., & Shi, Y. (2023). Ensemble of software defect predictors: An AHP-based evaluation method. International Journal of Information Technology & Decision Making, 10(1), 187–206. https://doi.org/10.1142/S0219622011004282

[17] Xia, Q., Sifah, E. B., Asamoah, K. O., Gao, J., Du, X., & Guizani, M. (2017). MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. IEEE Access, 5, 14757–14767. https://doi.org/10.1109/ACCESS.2017.2730843

[18] Mehta, S., Mehta, V., Kappor, A., Garg, A., & Tak, A. (2020). IoT-based smart healthcare monitoring system for COVID-19. Proceedings of the 2020 International Conference on Advances in Computing, Communication & Materials (ICACCM), 294–299. https://doi.org/10.1109/ICACCM50413.2020.9212897

[19] Din, S., Paul, A., & Rehman, A. (2019). 5G-enabled hierarchical architecture for software-defined intelligent transportation system. Computer Networks, 150, 81–89. https://doi.org/10.1016/j.comnet.2018.12.007

[20] Griggs, K. N., Ossipova, O., Kohlios, C. P., Baccarini, A. N., Howson, E. A., & Hayajneh, T. (2018). Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. Journal of Medical Systems, 42(7), 130. https://doi.org/10.1007/s10916-018-0982-x

[21] Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. Journal of the American Medical Informatics Association, 24(6), 1211–1220. https://doi.org/10.1093/jamia/ocx068

[22] Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377–387. https://doi.org/10.1145/362384.362685

[23] Gray, J., & Reuter, A. (1992). Transaction Processing: Concepts and Techniques. Morgan Kaufmann. ISBN: 978-1558601901

[24] Bernstein, P. A., & Goodman, N. (1981). Concurrency control in distributed database systems. ACM Computing Surveys, 13(2), 185–221. https://doi.org/10.1145/356842.356846

[25] Stonebraker, M., & Rowe, L. A. (1986). The design of POSTGRES. ACM SIGMOD Record, 15(2), 340–355. https://doi.org/10.1145/16856.16888

[26] Chodorow, K. (2013). MongoDB: The Definitive Guide (2nd ed.). O'Reilly Media. ISBN: 978-1449344689

[27] Sallam, A., Bertino, E., Hussain, S. R., Landers, D., Lefler, R. M., & Steiner, D. (2015). DBSAFE — An anomaly detection system to protect databases from exfiltration attempts. IEEE Systems Journal, 9(4), 1150–1163. https://doi.org/10.1109/JSYST.2014.2350071

[28] Bertino, E., & Sandhu, R. (2005). Database security: Concepts, approaches, and challenges. IEEE Transactions on Dependable and Secure Computing, 2(1), 2–19. https://doi.org/10.1109/TDSC.2005.9

[29] Mathew, S., Petropoulos, M., Ngo, H. Q., & Upadhyaya, S. (2010). A data-centric approach to insider attack detection in database systems. International Symposium on Recent Advances in Intrusion Detection (RAID), 382–401. https://doi.org/10.1007/978-3-642-15512-3_20

[30] Kamra, A., & Bertino, E. (2011). Privilege states-based access control for fine-grained intrusion response in database systems. IEEE Transactions on Information Forensics and Security, 6(3), 827–838. https://doi.org/10.1109/TIFS.2011.2128326

[31] Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in Forests of randomized trees. Advances in Neural Information Processing Systems, 26, 431–439. https://proceedings.neurips.cc/paper/2013/hash/e3796ae838835da0b6f6ea37bcf8bcb5-Abstract.html

[32] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. White Paper. https://bitcoin.org/bitcoin.pdf

[33] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum Yellow Paper. https://ethereum.github.io/yellowpaper/paper.pdf

[34] Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). MedRec: Using blockchain for medical data access and permission management. Proceedings of the 2nd International Conference on Open and Big Data (OBD), 25–30. https://doi.org/10.1109/OBD.2016.11

[35] Xia, Q., Sifah, E. B., Smahi, A., Amofa, S., & Zhang, X. (2017). BBDS: Blockchain-based data sharing for electronic medical records in cloud environments. Information, 8(2), 44. https://doi.org/10.3390/info8020044

[36] Dagher, G. G., Mohler, J., Milojkovic, M., & Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. Sustainable Cities and Society, 39, 283–297. https://doi.org/10.1016/j.scs.2018.02.014

[37] Tian, F. (2017). A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things. Proceedings of the International Conference on Service Systems and Service Management (ICSSSM), 1–6. https://doi.org/10.1109/ICSSSM.2017.7996119

[38] Bocek, T., Rodrigues, B. B., Strasser, T., & Stiller, B. (2017). Blockchains everywhere — A use-case of blockchains in the pharma supply-chain. Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 772–777. https://doi.org/10.23919/INM.2017.7987376

[39] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., & Njilla, L. (2017). ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 468–477. https://doi.org/10.1109/CCGRID.2017.8

[40] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the Internet of things. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, 13–16. https://doi.org/10.1145/2342509.2342513

[41] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[42] Sallam, A., & Bertino, E. (2019). Detection of insider attacks against databases. IEEE Internet Computing, 23(1), 8–15. https://doi.org/10.1109/MIC.2018.2877985

[43] Hu, J., Xiao, B., & Li, Y. (2020). LSTM-based anomaly detection for database query streams. Information Sciences, 512, 437–456. https://doi.org/10.1016/j.ins.2019.10.003

[44] Lee, Y., Yoon, K., & Park, K. (2021). A comparative study of machine learning algorithms for intrusion detection in imbalanced database environments. Applied Sciences, 11(4), 1428. https://doi.org/10.3390/app11041428

[45] Antonopoulos, A. M., & Wood, G. (2018). Mastering Ethereum: Building Smart Contracts and DApps. O'Reilly Media. ISBN: 978-1491971949

[46] Momjian, B. (2001). PostgreSQL: Introduction and Concepts. Addison-Wesley. ISBN: 978-0201703306

[47] Merkle, R. C. (1988). A digital signature based on a conventional encryption function. Advances in Cryptology — CRYPTO'87, 369–378. https://doi.org/10.1007/3-540-48184-2_32

[48] Blelloch, G. E. (1989). Scans as primitive parallel operations. IEEE Transactions on Computers, 38(11), 1526–1538. https://doi.org/10.1109/12.42122

[49] Behl Hyperledger. (2022). Hyperledger Besu Enterprise Ethereum Client. Linux Foundation. https://besu.hyperledger.org/

[50] Solidity Documentation. (2023). Solidity 0.8.x Language Reference. Ethereum Foundation. https://docs.soliditylang.org/

[51] Gregg, B. (2019). BPF Performance Tools: Linux System and Application Observability. Addison-Wesley Professional. ISBN: 978-0136554820

[52] Tang, D. (2010). Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? FAST Conference Proceedings, 1–9. https://www.usenix.org/legacy/events/fast07/tech/tang.html

[53] Matsakis, N. D., & Klock, F. S. (2014). The Rust language. ACM SIGAda Ada Letters, 34(3), 103–104. https://doi.org/10.1145/2692956.2663188

[54] Alam, M., Reaz, M. B. I., & Ali, M. A. M. (2012). A review of smart homes: Past, present, and future. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 42(6), 1190–1203. https://doi.org/10.1109/TSMCC.2012.2189204

[55] Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. https://doi.org/10.17487/RFC8446

[56] Bartoletti, M., & Pompianu, L. (2017). An empirical analysis of smart contracts: Platforms, applications, and design patterns. Financial Cryptography and Data Security, 494–509. https://doi.org/10.1007/978-3-319-70278-0_31

[57] Tikhomirov, S., Voskresenskaya, E., Ivanitskiy, I., Takhaviev, R., Marchenko, E., & Alexandrov, Y. (2018). SmartCheck: Static analysis of Ethereum smart contracts. Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, 9–16. https://doi.org/10.1145/3194113.3194115

[58] Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications

Security, 254–269. https://doi.org/10.1145/2976749.2978309

[59] OpenZeppelin. (2023). OpenZeppelin Contracts: Transparent Upgradeable Proxy. https://docs.openzeppelin.com/contracts/4.x/api/proxy

[60] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[61] Grinberg, M. (2018). Flask Web Development (2nd ed.). O'Reilly Media. ISBN: 978-1491991732

[62] Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks, 51(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001

[63] Ferretti, S., Ghini, V., Panzieri, F., Pellegrini, M., & Turrini, E. (2010). QoS-aware clouds. Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, 321–328. https://doi.org/10.1109/CLOUD.2010.17

[64] Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. NIST Special Publication 800-145. https://doi.org/10.6028/NIST.SP.800-145

[65] Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., & Schwarz, P. (1992). ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Transactions on Database Systems, 17(1), 94–162. https://doi.org/10.1145/128765.128770

[66] Benet, J. (2014). IPFS - Content addressed, versioned, P2P file system. arXiv:1407.3561. https://doi.org/10.48550/arXiv.1407.3561

[67] U.S. Department of Health and Human Services. (2013). HIPAA Security Rule. 45 CFR Parts 160 and 164. https://www.hhs.gov/hipaa/for-professionals/security/index.html

[68] European Parliament. (2016). General Data Protection Regulation (GDPR). Regulation (EU) 2016/679. https://eur-lex.europa.eu/eli/reg/2016/679/oj

[69] Pinata. (2023). Pinata IPFS API Documentation. https://docs.pinata.cloud/

[70] Stonebraker, M. (1987). The design of the POSTGRES storage system. Proceedings of the 13th International Conference on Very Large Databases (VLDB), 289–300.

[71] Haerder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. ACM Computing Surveys, 15(4), 287–317. https://doi.org/10.1145/289.291

[72] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. Computer, 29(2), 38–47. https://doi.org/10.1109/2.485845

[73] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security, 4(3), 224–274. https://doi.org/10.1145/501978.501980

[74] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

[75] He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

[76] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[77] Merkle, R. C. (1987). A digital signature based on a conventional encryption function. Crypto'87 Lecture Notes. https://doi.org/10.1007/3-540-48184-2_32

[78] Preneel, B. (2010). The first 30 years of cryptographic hash functions and the NIST SHA-3 competition. Topics in Cryptology – CT-RSA 2010, 6033, 1–22. https://doi.org/10.1007/978-3-642-11925-5_1

[79] Annas, G. J. (2003). HIPAA regulations: A new standard for medical privacy. New England Journal of Medicine, 348(15), 1486–1490. https://doi.org/10.1056/NEJMlim035027

[80] Voigt, P., & Von dem Bussche, A. (2017). The EU General Data Protection Regulation (GDPR): A Practical Guide. Springer. https://doi.org/10.1007/978-3-319-57959-7

[81] European Union Agency for Cybersecurity. (2023). NIS2 Directive Implementation Guide. ENISA. https://www.enisa.europa.eu/topics/nis-directive

[82] Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems (3rd ed.). Wiley. ISBN: 978-1119642787

[83] Bishop, M. (2003). Computer Security: Art and Science. Addison-Wesley Professional. ISBN: 978-0201440997

[84] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

[85] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3), 50–60. https://doi.org/10.1109/MSP.2020.2975749

[86] Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). Zerocash: Decentralized anonymous payments from Bitcoin. IEEE Symposium on Security and Privacy, 459–474. https://doi.org/10.1109/SP.2014.36

[87] Robinson, I., Webber, J., & Eifrem, E. (2015). Graph Databases: New Opportunities for Connected Data (2nd ed.). O'Reilly Media. ISBN: 978-1491930892

[88] Dunning, T., & Friedman, E. (2014). Time Series Databases: New Ways to Store and Access Data. O'Reilly Media. ISBN: 978-1491914724

[89] Li, F., Hadjieleftheriou, M., Kollios, G., & Reyzin, L. (2006). Dynamic authenticated index structures for outsourced databases. Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, 121–132. https://doi.org/10.1145/1142473.1142488

[90] Guardtime. (2022). Guardtime KSI Blockchain Technology White Paper. https://guardtime.com/technology

ISSN: - | Open Access | CC BY 4.0 | © 2026 DATAMIND
https://doi.org/10.63646/datamind.2026.040102