International Journal of Advance and Applied Research
www.ijaar.co.in
ISSN – 2347-7075 Impact Factor – 8.141
Peer Reviewed Bi-Monthly
Vol. 7 No. 28 March - April - 2026

GlitchZero: An AI-Native Middleware Trust Infrastructure for Real-Time Institutional Data Integrity Using Hybrid Anomaly Detection, Blockchain Anchoring, Explainable AI, and Agentic Mitigation

Sanika Sameer Tribhuvan
TYBCA, Center for Advanced Studies in Applied Sciences (CASAS)
Mentored by: Prof. D. V. Jagdale, Asst. Professor, CASAS Department
New Arts, Commerce & Science College, Ahilyanagar (Autonomous), Maharashtra - 414 001
Corresponding Author –Sanika Sameer Tribhuvan
DOI - 10.5281/zenodo.19396409

Abstract:

Modern institutional information systems remain critically vulnerable to unauthorized post-write data mutations - commonly called "Silent Edits" - wherein privileged users alter stored records without triggering any system alert. Traditional countermeasures either impose rigid hard-locks that disrupt legitimate administrative operations or rely on passive database logs that provide no real-time forensic analysis. This paper introduces GlitchZero v5, a novel AI-native middleware trust infrastructure implemented as a production-ready, language-agnostic service that integrates with any existing institutional system via a single RESTful API endpoint, requiring zero modifications to the host system's database schema or middleware.

The system employs a five-stage AI pipeline: (1) an Isolation Forest-inspired triage engine classifies each incoming data mutation into ROUTINE, SUSPICIOUS, or CRITICAL tiers using a four-dimensional weighted feature vector; (2) a Semantic Delta Audit computes SHAP-style Explainable AI factor attribution, decomposing every risk score into normalized percentage contributors; (3) a Merkle Tree blockchain anchoring mechanism batches SHA-256 hashes and commits roots to Polygon Amoy Testnet; (4) an Agentic Mitigation Engine autonomously fires real HTTP POST rollback webhooks to the host system for CRITICAL events; and (5) an in-process performance logger computes live F1-score, Precision, Recall, Accuracy, and latency percentile metrics. Uniquely, GlitchZero v5 uses SQLite persistence with WAL mode, ensuring all audit history survives server restarts - a critical requirement for production deployment.

Experimental evaluation on a synthetic dataset of 200 mutation events demonstrates a 66% ROUTINE triage rate reducing average processing latency by 40%, a weighted average latency of 3.4ms with sub-10ms p95, and classification metrics of Precision 0.94, Recall 0.92, and F1-score 0.93.

Keywords: Data Integrity, Anomaly Detection, Isolation Forest, Explainable AI, SHAP Attribution, Merkle Tree, Blockchain Anchoring, Agentic AI, Middleware Security, Silent Edit Detection, SQLite, Production Deployment

Introduction:

The integrity of data stored in institutional information systems is a critical and largely under addressed challenge. In higher education, student attendance percentages directly influence examination eligibility under university regulations; in administrative systems, unauthorized role escalations can grant unintended access privileges; in financial modules, inflated procurement figures can result in significant

institutional losses. Each of these scenarios represents a "Silent Edit" - a post-write modification executed by a privileged or compromised account that leaves no visible trace in standard application logs.

Existing countermeasures fall into two inadequate categories. The Hard-Lock model prevents certain write operations after a fixed condition but generates administrative friction. The Passive Log model records transactions in an audit table but provides no real-time anomaly analysis, no cryptographic immutability, and no automated response capability.

This paper introduces GlitchZero v5 as a third paradigm - the Open-Gate Trust Layer. GlitchZero allows all write operations to proceed unobstructed while simultaneously anchoring, analyzing, and autonomously responding to every data mutation through a five-stage AI pipeline. Implemented as a Node.js/TypeScript/Express service with SQLite persistence, it integrates with any host application through a single asynchronous POST call placed after a successful database write. The CASAS departmental attendance portal at New Arts, Commerce & Science College, Ahilyanagar serves as the primary reference implementation, with multi-domain validation across payroll, procurement, and access control scenarios. The project is publicly available at https://github.com/SanikaTribhuvan/glitchzero.

The key contributions of this work are:

- A production-ready AI-native trust middleware with SQLite persistence, CORS-aware API, and environment-driven configuration - deployable to Render or Railway with a single command.
- A hybrid Isolation Forest triage engine using a four-dimensional feature vector with weights (f1=0.45, f2=0.35, f3=0.12, f4=0.08) for adaptive audit-skipping.
- A SHAP-inspired XAI attribution layer producing normalized factor breakdowns summing to 100% for every flagged event.
- A real outgoing rollback webhook architecture driven by ROLLBACK_URL in .env, enabling genuine autonomous self-healing of the host system.
- A Merkle Tree anchoring service supporting simulated mode and live Polygon Amoy Testnet via ethers.js.
- A live stress-test generator producing 100 mixed-risk events for immediate Research Tab population at conference demonstrations.

Objectives:

1. To design and implement a hybrid triage architecture simulating Isolation Forest classification using a four-dimensional weighted feature vector, enabling efficient classification into ROUTINE, SUSPICIOUS, and CRITICAL tiers with adaptive audit skipping.
2. To develop a SHAP-inspired Explainable AI layer that decomposes risk scores into normalized factor attributions, providing interpretable forensic evidence for every flagged event.
3. To establish a Merkle Tree blockchain anchoring mechanism using SHA-256 hashing, supporting both simulated anchoring and live Polygon Amoy Testnet transactions via ethers.js.
4. To implement an Agentic Mitigation Engine that fires real HTTP POST webhooks to a configurable ROLLBACK_URL endpoint in the host system for CRITICAL-tier events.
5. To migrate from volatile in-memory storage to persistent SQLite with WAL mode, ensuring all audit records survive server restarts in production deployment.

6. To provide a built-in research telemetry API computing live F1-score, Precision, Recall, Accuracy, and latency percentiles suitable for academic publication data tables.
7. To validate platform-agnostic design across attendance management, HR and payroll, financial procurement, and administrative access control domains.

Methodology:

1. System Architecture:

GlitchZero v5 is implemented as a standalone Node.js/Express/TypeScript service with SQLite persistence via better-sqlite3 in WAL (Write-Ahead Logging) mode. All state is stored in a file-based SQLite database, making the system fully deployable on Render.com or Railway with persistent volumes. Table 1 summarizes the core modules.

Table 1: GlitchZero v5 Module Summary

Module	File	Responsibility
API Server	server/index.ts	Express bootstrap, CORS (ALLOWED_ORIGINS), dotenv, request logger
Pipeline Engine	server/routes.ts	Five-stage AI pipeline, all API endpoints, agentic mitigation
Persistent Storage	server/storage.ts	SQLite + better-sqlite3, WAL mode, survives restarts
Blockchain Service	server/blockchainService.ts	Simulated by default; live Polygon Amoy via ethers.js (BLOCKCHAIN_MODE=live)
Type Definitions	shared/schema.ts	Anchor, ApiKey, MerkleBatch, PerfLog, XAIFactor, TriageClass
DB Schema	server/db/schema.ts	Drizzle ORM definitions: api_keys, anchors, merkle_batches, perf_logs

2. Integration Model:

Any host system integrates with GlitchZero via a single asynchronous HTTP call placed immediately after a successful database write. Because the call is fire-and-forget, the host system's user-facing response latency is not affected:

POST /api/anchor | Headers: { x-api-key: <gz_key> } | Body: { app_id, user_id, entity_id, data_payload, timestamp }

The entity_id field enables GlitchZero to retrieve the entity's previous anchor from SQLite, enabling delta comparison across successive mutations. In CASAS Portal integration, entity_id maps to the student's ID, app_id to "casas-attendance", and user_id to the JWT-authenticated teacher's ID from the Prisma user record.

3. Five-Stage AI Pipeline:

Stage 1 - Isolation Forest Triage: A four-dimensional weighted feature vector is computed for each incoming payload by the runIsolationForestTriage() function in server/routes.ts. Tables 2 and 3 detail the feature definitions and triage thresholds respectively. A deterministic MD5-derived jitter term of maximum +/-0.03 simulates model variance. ROUTINE events bypass Stage 2 entirely, reducing average processing latency by approximately 40%.

Table 2: Feature Vector (f1-f4) with Weights

Feature	Weight	Description	Threshold
f1	0.45 (45%)	Absolute magnitude — any numeric field value	> 90
f2	0.35 (35%)	Delta magnitude — relative change vs. previous anchor for same entity_id	> 50%
f3	0.12 (12%)	Semantic sensitivity — role, designation, access_level, admin, privilege fields	Present
f4	0.08 (8%)	Temporal anomaly — operation outside business hours	Hour < 7 or > 19

Table 3: Triage Classification Thresholds

Triage Class	Anomaly Score	Audit Path	Agentic Action
ROUTINE	< 0.35	Fast-path — semantic audit skipped	None
SUSPICIOUS	0.35 – 0.69	Full semantic audit + XAI	None
CRITICAL	>= 0.70	Full audit + XAI	Rollback webhook if riskScore >= 96

Stage 2 - Semantic Delta Audit with XAI Attribution: Non-ROUTINE anchors proceed to runSemanticAudit(), which evaluates three risk rules: (A) Absolute Threshold - raw weight 45 when any numeric field exceeds 90; (B) Change Magnitude - raw weight 35 when relative change between consecutive anchors exceeds 50%; (C) Designation/Role Change - raw weight 25 when privilege-sensitive field names are detected. A fourth factor, Anomaly Score (IF), contributes proportionally to the Isolation Forest score. XAI weights are normalized so all factors sum to exactly 100%, producing a sorted array of human-readable forensic evidence for every flagged event.

Stage 3 - Merkle Tree Blockchain Anchoring: Each anchor's payload is canonically serialized with keys sorted alphabetically and hashed via Node.js native SHA-256. After every MERKLE_BATCH_SIZE anchors (default: 10, configurable via .env), the buildMerkleRoot() function in shared/schema.ts constructs a binary Merkle tree using Bitcoin's SPV convention: sorted hash pairs are concatenated and re-hashed at each level, with odd-leaf duplication as needed. The blockchainService.ts module routes to either

SIMULATED mode (deterministic SHA-256 pseudo-hash with Polygon Amoy explorer URL) or LIVE mode (real on-chain transaction to Polygon Amoy Testnet via ethers.js JsonRpcProvider, requiring POLYGON_PRIVATE_KEY and MERKLE_CONTRACT_ADDRESS in .env). Dashboard Merkle tab displays clickable PolygonScan explorer links in both modes.

Stage 4 - Agentic Mitigation Engine: When a CRITICAL-class anchor attains a risk score of 96 or higher (AGENTIC_THRESHOLD constant in routes.ts), the triggerAgenticMitigation() function fires a real outgoing HTTP POST request to the URL in the ROLLBACK_URL environment variable. The webhook payload contains: event type (GLITCHZERO_AUTO_ROLLBACK), anchorId, appId, entityId, riskScore, flagReason, action (REVERT_TO_PREVIOUS_SAFE_STATE), and issuedAt timestamp. A 5-second AbortController timeout prevents blocking. The anchor is then updated in SQLite via updateAnchorPostAgentic() to persist selfHealed status and webhookResult across restarts - a key improvement over the in-memory implementation.

Stage 5 - Performance Logging and Research Telemetry: Every processed anchor generates a perf_logs row in SQLite recording latencyMs, triageClass, isFlagged, and riskScore. The GET /api/dashboard/research-data endpoint aggregates these rows to compute live TP/FP/TN/FN counts, Precision, Recall, F1-Score, Accuracy, latency percentiles, triage distribution, and blockchain batch statistics. The stress-test.ts script generates 100 mixed-risk events (65% ROUTINE, 20% SUSPICIOUS, 15% CRITICAL) using realistic CASAS-domain payloads to immediately populate these graphs for demonstration.

4. Production Architecture Highlights:

GlitchZero v5 introduces several production-grade features beyond prior implementations. SQLite with WAL mode provides concurrent read access and crash-safe writes suitable for hosted environments. CORS middleware reads ALLOWED_ORIGINS from .env as a comma-separated allowlist, enabling secure cross-origin calls from the React CASAS Portal frontend without browser blocks. The MASTER_API_KEY environment variable pre-seeds an API key on first boot for zero-click deployment. The configurable ROLLBACK_URL makes self-healing host-system-agnostic: any backend exposing a POST endpoint can receive and act on rollback signals.

5. Multi-Domain Validation:

Synthetic mutation payloads were submitted across four institutional domains. In attendance management, override of a student from ABSENT to PRESENT triggered Delta Magnitude and Semantic Sensitivity XAI factors. In payroll and HR, an unauthorized salary update and designation change to super_admin were classified CRITICAL via Absolute Threshold (access_level: 99) and Role Change rules. In financial procurement, a bill amount inflated by 75% post-approval was detected via Delta Magnitude. In administrative access control, a role field changed from student to admin triggered CRITICAL triage and immediate rollback webhook dispatch.

Results:

Evaluation was performed on a synthetic dataset of 200 data mutation events distributed across ROUTINE (132), SUSPICIOUS (42), and CRITICAL (26) ground truth labels. Flagging ground truth was defined as riskScore >= 50. Tables 4-7 summarize the results.

Table 4: Classification Metrics (N = 200)

Metric	Value
Precision	0.94
Recall	0.92
F1-Score	0.93
Accuracy	0.935

Table 5: Triage Distribution

Triage Class	Events	Percentage	Audit Path
ROUTINE	132	66%	Fast-path — semantic audit skipped
SUSPICIOUS	42	21%	Full semantic audit + XAI attribution
CRITICAL	26	13%	Full audit + XAI + agentic rollback webhook

Table 6: Latency Metrics by Pipeline Path (milliseconds)

Pipeline Path	Avg (ms)	P50 (ms)	P95 (ms)	P99 (ms)
ROUTINE (fast-path)	2.1	1.8	4.2	6.1
SUSPICIOUS (full audit)	4.8	4.3	8.9	12.4
CRITICAL (audit + rollback)	7.2	6.8	14.1	18.6
Overall weighted average	3.4	2.9	9.2	16.1

Table 7: XAI Attribution Distribution (68 flagged events)

XAI Factor	Avg Weight (%)
Absolute Threshold (> 90)	44.8%
Change Magnitude (Delta > 50%)	32.1%
Role / Designation Change	14.3%
Anomaly Score (IF contribution)	8.8%

Discussion And Conclusions:

The experimental results validate the core claims of this research across four dimensions. First, the hybrid triage architecture achieves meaningful computational efficiency: routing 66% of events through the fast-path reduces average latency by approximately 40%, while maintaining a Recall of 0.92 for critical

events. The p95 latency of 9.2ms confirms GlitchZero imposes negligible overhead on the host system's write throughput even at full audit depth.

Second, the SHAP-inspired XAI attribution layer provides genuine interpretive value beyond a binary flag. By surfacing the specific field and rule responsible for each risk score - with Absolute Threshold contributing 44.8% and Change Magnitude contributing 32.1% of average weight across flagged events - system administrators can understand why a record was flagged without any machine learning expertise. This is critical for institutional adoption where flagged records may carry legal or academic consequences.

Third, the Agentic Mitigation Engine demonstrates that autonomous rollback behavior is achievable within a production-ready middleware framework via real outgoing HTTP webhooks, without requiring a large language model or cloud dependency. The configurable ROLLBACK_URL architecture makes GlitchZero's self-healing capability host-system-agnostic.

Fourth, the SQLite persistence upgrade fundamentally changes GlitchZero's deployment posture. Unlike prior in-memory implementations where audit history was lost on server restart, v5 maintains a permanent, crash-safe record of all anchors, Merkle batches, and performance logs - enabling cumulative research metric computation and longitudinal institutional audit trails.

GlitchZero's field-agnostic risk engine - which derives risk signals from statistical properties of the data itself rather than domain-specific rules - enables true plug-and-play deployment. The open-gate security model, which permits all writes while cryptographically anchoring and auditing each one, is demonstrably superior to hard-lock approaches for real-world institutional deployments where administrative flexibility and data integrity must coexist.

In conclusion, GlitchZero v5 achieves an F1-score of 0.93 with sub-10ms p95 latency, SQLite-persisted audit history, real webhook-based agentic mitigation, and optional live blockchain anchoring on Polygon Amoy Testnet - establishing it as a viable, production-deployable universal trust layer for institutional data integrity. Future work will focus on training the triage model on real CASAS Portal data, extending agentic response to automated alert dispatch, and integrating with examination management and library systems.

Acknowledgements:

The author sincerely thanks Prof. D. V. Jagdale, CASAS Department, New Arts, Commerce & Science College, Ahilyanagar, for expert mentorship and guidance. Thanks are also due to Prof. Arun Gangarde, Head CASAS, and the NCRT-AIML 2026 organizing committee for this opportunity.

References:

1. F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation Forest," in Proc. IEEE ICDM, 2008, pp. 413-422.
2. S. M. Lundberg and S. I. Lee, "A Unified Approach to Interpreting Model Predictions," in Proc. NeurIPS, 2017, pp. 4765-4774.
3. R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in Proc. CRYPTO, 1987, pp. 369-378.
4. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

5. B. Scholkopf et al., "Support Vector Method for Novelty Detection," in Proc. NeurIPS, 1999, pp. 582-588.
6. Y. Wu et al., "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," arXiv:2308.08155, 2023.
7. A. Azaria et al., "MedRec: Using Blockchain for Medical Data Access and Permission Management," in Proc. IEEE Open Blockchain, 2016.
8. M. Rashid et al., "Security Issues in Institutional Database Systems," Int. J. Adv. Res. Comput. Sci., vol. 9, no. 3, 2018.
9. G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Yellow Paper, 2014.
10. C. Dwork, "Differential Privacy," in Proc. ICALP, vol. 4052, 2006, pp. 1-12.