Integrated Security System Framework Using Blockchain and Digital Forensics Technology

Kawu Saidu Bappah, Bala Modi, Umar Haruna Abdullahi

Department of Computer Science, Gombe State University, Gombe State, Nigeria

DOI: https://doi.org/10.51584/IJRIAS.2026.11070154

Received: 29 July 2026; Accepted: 03 August 2026; Published: 14 August 2026

ABSTRACT

The proliferation of cybersecurity incidents and large-scale data breaches has exposed significant weaknesses in traditional security systems, particularly regarding data integrity, transparency, chain of custody, and forensic readiness. Centralised architectures suffer from single points of failure, log-tampering risks and limited cross-organisational trust. This paper designed, developed and evaluated an Integrated Security System Framework combining blockchain-style integrity mechanisms with digital forensics methodology. It was implemented as a four-layer system: a React presentation layer, two Node.js/Express API services, a Python-based business logic layer, and a data layer that includes an Ethereum-compatible (Hardhat) deployment path together with off-chain filesystem storage. The evaluated core comprises a custom Python blockchain (SHA-256 Proof-of-Work) with Python classes providing contract-style policy enforcement and audit logging. A parallel suite of four Solidity smart contracts was also implemented, but those contracts were not exercised when producing the quantitative results reported here; that distinction is stated explicitly throughout. The design was evaluated by executing the public implementation directly and repeatedly against a functional correctness workload of 100 events and up to 30 evidence items across five independent runs. Six further security and stress tests examined varied file sizes, concurrent access, malformed transactions, unauthorised access, direct tampering and off-chain tampering. These tests located and fixed three real defects, most significantly a concurrency defect that corrupted the blockchain under twenty simultaneous writers; after the fix, chain validity held under 500 concurrent writes. The evaluation also surfaced an unresolved limitation: no identity-authentication mechanism is implemented, so accountability guarantees depend on honest caller identity claims. All evaluation categories scored above 90 percent on the framework's disclosed self-referential scoring formula, and forensic risk-scoring, audit logging, custody tracking and tampering detection functioned as designed within their tested scope. The findings support the feasibility of integrating blockchain-style integrity with digital forensics for evidence management under controlled evaluation conditions, and they underline the necessity of repeated, adversarially minded testing before stronger operational claims are made.

Keywords: Blockchain, Digital Forensics, Smart Contracts, Chain of Custody, Evidence Integrity, Audit Trail, Cybersecurity

INTRODUCTION

The rapid digitization of modern society has brought unprecedented opportunities alongside significant security challenges. Cybersecurity incidents, data breaches, and digital crimes have become increasingly sophisticated, necessitating innovative approaches to security systems and forensic investigations. Traditional security frameworks often struggle with issues of data integrity, tampering, centralized points of failure, and lack of transparency in evidence handling.

Blockchain technology, originally conceptualized as the underlying architecture for cryptocurrencies (Nakamoto, 2008), has emerged as a transformative solution for various security applications beyond financial transactions. Its inherent characteristics of immutability, transparency, decentralization, and cryptographic security make it particularly suitable for addressing critical challenges in security systems and digital forensics (Zheng et al., 2018). The distributed ledger technology ensures that once data is recorded, it cannot be altered retroactively without detection, providing an auditable trail of all transactions and activities.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Digital forensics, the science of identifying, preserving, analyzing, and presenting digital evidence in a manner acceptable in legal proceedings, faces numerous challenges in the contemporary digital landscape (Garfinkel, 2010). These include the volume and complexity of digital data, cloud computing environments, encryption, anti-forensics techniques, and maintaining chain of custody. The integration of blockchain technology with digital forensics methodologies presents a promising avenue for addressing these challenges while enhancing the credibility and reliability of digital evidence.

Within Nigeria specifically, the threat landscape is well documented. The United Nations Office on Drugs and Crime's Cybercrime Assessment for Nigeria, produced in cooperation with the Economic and Financial Crimes Commission (EFCC) and other national agencies, catalogues the institutional response to a growing volume of cybercrime cases (UNODC, 2025). The Central Bank of Nigeria (CBN) has issued a Risk-Based Cybersecurity Framework and Guidelines mandating regulated financial institutions to report cybersecurity incidents within 24 hours of detection (Mondaq, 2025). Data from the Nigeria Inter-Bank Settlement System show that Nigerian financial institutions lost ₦52.26 billion to fraud and cyber-enabled financial crime in 2024, up from ₦17.67 billion in 2023 (Vanguard, 2026). The Nigeria Computer Emergency Response Team (ngCERT) has also documented coordinated ATM cash-out fraud campaigns resulting in losses exceeding two million dollars in a single incident (Blueprint Newspapers, 2026). This trend is consistent with global growth in cybercrime losses reported in the FBI's Internet Crime Complaint Center 2025 Annual Report. The report recorded a 26 percent year-on-year increase in global losses to $20.877 billion (Businessday NG, 2026). The lack of robust, transparent and tamper-evident audit infrastructure across affected institutions further complicates the prosecution of offenders and the recovery of stolen assets, providing additional motivation for the design and evaluation of the framework presented in this paper.

Statement of the Problem

Current security systems face multiple critical challenges that undermine their effectiveness such as data integrity issues, chain of custody, lack of transparency, single points of failure, forensic investigation difficulty and trust deficits. Traditional centralized databases are vulnerable to unauthorized modifications, deletions, and tampering, making it difficult to establish the authenticity of stored information during security incidents or forensic investigations. Maintaining an unbroken chain of custody for digital evidence remains a significant challenge in digital forensics.

Furthermore, centralized security systems often operate as black boxes, making it difficult for stakeholders to verify security measures, audit access logs, or validate incident response activities. Centralized architectures create vulnerabilities where the compromise of a single entity can lead to catastrophic security breaches affecting entire systems and undermining public trust in critical infrastructure. In multi-stakeholder environments such as joint investigations between law enforcement, regulators, internal security teams and external counsel, establishing trust between parties without relying on centralized authorities remains challenging. These challenges necessitate an integrated approach that combines the immutability and transparency of blockchain technology with the analytical capabilities of digital forensics to create robust, trustworthy security systems. The absence of such an integrated, fully implementable, openly available framework constitutes the specific gap addressed by this research.

Aim and Objectives

The main aim of this research is to design, develop, and evaluate an integrated security system framework that combines blockchain technology and digital forensics techniques to enhance data integrity, transparency, and forensic readiness in security operations. The specific objectives of the study are: (i) to develop a framework or model that integrates blockchain and digital forensics for security systems; (ii) to investigate and evaluate the feasibility and effectiveness of the proposed model in addressing key challenges faced in the current investigative system; and (iii) to identify the legal, ethical and operational considerations in implementing such an integrated system within law enforcement and organizational security contexts.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Research Questions

This study seeks to answer the following research questions: (1) How can blockchain technology and digital forensics be integrated into a single, coherent framework that supports evidence management, chain of custody, audit logging and policy enforcement? (2) What functional, performance and security characteristics does the proposed framework exhibit when subjected to a representative workload of security events and digital evidence items? (3) What legal, ethical and operational considerations are most relevant to the deployment of a blockchain-based integrated security framework, and how can these considerations be addressed by the proposed design?

METHODOLOGY

Research Approach

The study adopts the Design Science Research (DSR) approach as articulated by Hevner et al. (2004) and refined by Peffers et al. (2007). DSR is appropriate where the goal of the research is to produce and rigorously evaluate an innovative artefact that addresses a real-world problem. In this study the artefact is a running software framework rather than a purely theoretical model. The research therefore proceeds through the canonical DSR activities of problem identification, objectives definition, design and development, demonstration, evaluation, and communication. The implementation in the associated code repository is the primary artefact produced by the research, and this paper constitutes the principal communication act.

System Architecture

The framework adopts a layered architectural pattern (Buschmann et al., 1996) augmented by a microservices style at the API tier. The rationale for the layered pattern is its proven ability to separate concerns and to allow each layer to evolve independently; the rationale for the microservices augmentation is the natural separation between general data services and blockchain-aware services, each of which has different security and lifecycle characteristics.

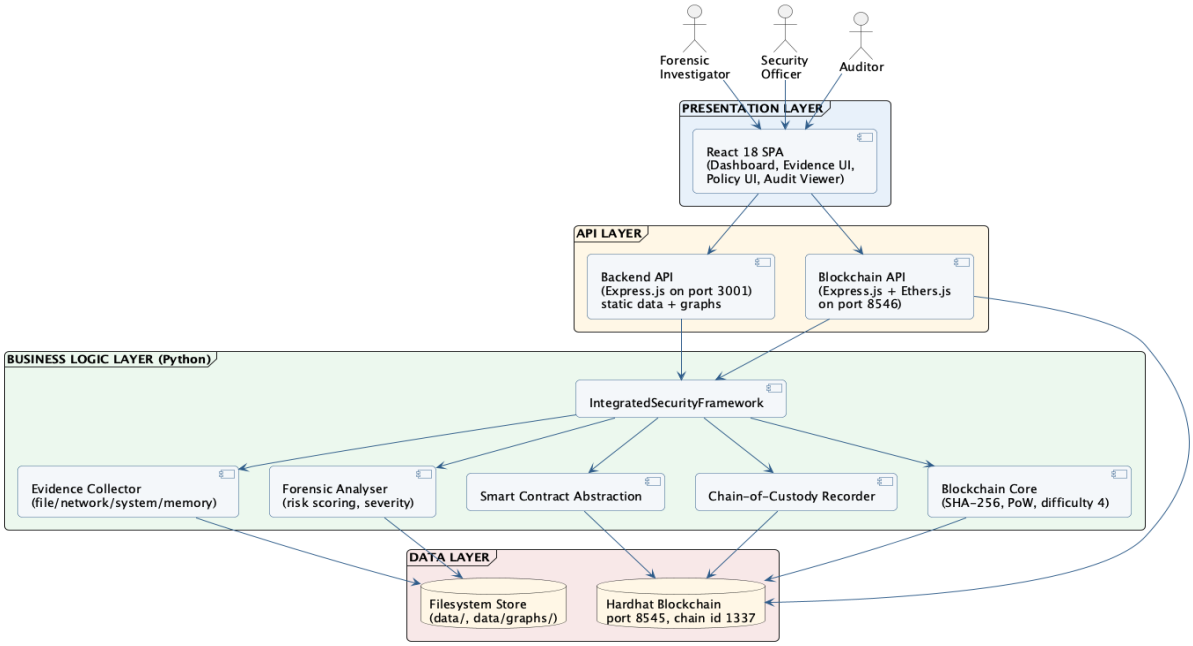

Figure 1: High-level system architecture of the Integrated Security System Framework (Presentation → API → Business Logic → Data). The diagram shows component placement only; runtime evidence movement and the evaluated versus unevaluated blockchain paths are detailed in Figures 2 and 2b.

The framework is organised as a four-layer architecture: a Presentation Layer implemented as a React 18 single-page application; an API Layer comprising two Express.js services for general data and blockchain interaction respectively; a Business Logic Layer implemented in Python that orchestrates blockchain operations, evidence

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

handling, forensic analysis, policy execution and audit logging; and a Data Layer composed of an Ethereum-compatible blockchain (Hardhat local network) and a filesystem store for generated reports and visualisations.

The presentation layer is built with React 18.2.0 and provides dashboards for blockchain status, evidence management, audit-trail browsing and policy configuration, with charts rendered using Recharts and on-chain calls issued via Ethers.js and Web3.js. Two Node.js services implement the API layer: a Backend API (Express.js, port 3001) that serves static framework data and generated graphs, and a Blockchain API (Express.js with Ethers.js, port 8546) that interacts with the smart contracts deployed on the local Hardhat node. The business logic layer is implemented in Python 3.8+ and centred on the IntegratedSecurityFramework class, which composes four subsystems: a blockchain core, a smart-contract abstraction, a digital-forensics module (evidence collection, chain of custody, forensic analyser) and a statistics aggregator. Persistent state is held in two stores: on-chain state (security events, evidence hashes, custody transfers and audit entries) on the local Hardhat blockchain, and off-chain artefacts (generated graphs, evaluation reports, forensic reports and exported chain dumps) in the filesystem.

Blockchain Core Design

Each block is represented as a data structure containing an index, an ISO-8601 timestamp, an arbitrary JSON-serializable data payload, the previous block hash, the current block hash and a nonce. The data payload includes a type field (one of genesis, security_event, forensic_evidence, chain_of_custody) and type-specific fields, enabling type-aware queries. Block integrity relies on SHA-256 hashing combined with a Proof-of-Work mining procedure: the hash of the canonical JSON representation of the block is repeatedly recomputed with an incrementing nonce until the hash satisfies the configured difficulty target. Chain validation runs in linear time in the length of the chain and verifies, for every non-genesis block, both that the recomputed hash matches the stored hash and that the previous-hash field correctly references the predecessor's hash.

The integrity guarantees of the blockchain core rest on the second pre-image and collision resistance of SHA-256. An attacker who wishes to modify a registered evidence item cannot construct a different content with the same SHA-256 digest, and an attacker who wishes to rewrite the chain must redo the proof-of-work for every block from the modification point forwards while keeping pace with the honest extension of the chain, consistent with Bitcoin's longestchain rule (Nakamoto, 2008). The framework comprises four Solidity 0.8.24 smart contracts; IntegratedSecurityFramework, SecurityPolicyContract, AuditContract and EvidenceContract deployed with Hardhat 2.19.4 and built on OpenZeppelin Contracts 5.0.2. AuditContract exposes only push semantics and a query function; no setter or deletion function is exposed, so that no audit entry can be modified or deleted after it has been committed.

Relationship Between the Python Core, the Python Contract-Style Classes, and the Solidity Layer

A methodological clarification is necessary here, since the implementation contains three components that are easily conflated under the single label "blockchain." First, blockchain_core.py implements a standalone Python blockchain (SHA-256 hashing, Proof-of-Work, chain validation) with no network node and no external dependency. Second, blockchain/smart_contracts.py implements two Python classes named SecurityPolicyContract and AuditContract that reproduce contract-like semantics (rule evaluation, append-only logging) entirely in-process; despite the name, these are ordinary Python objects, not compiled or deployed Solidity bytecode. Third, and separately, four Solidity 0.8.24 contracts (IntegratedSecurityFramework.sol, SecurityPolicyContract.sol, AuditContract.sol, EvidenceContract.sol) are implemented and deployable to a local Hardhat network via a dedicated startup script and a Blockchain API service (Express.js with Ethers.js, port 8546).

The quantitative evaluation reported in this paper exercises only the first two components: the IntegratedSecurityFramework orchestration class instantiates blockchain_core.Blockchain and the Python SecurityPolicyContract/AuditContract classes directly, and the sample-data generation and evaluation scripts call this orchestration class exclusively. The Solidity contracts and the Hardhat node are implemented and independently deployable, but are not invoked by the evaluated pipeline. There is no on-chain transaction that was submitted to the Hardhat network as part of the results reported in this paper. Figure 2 shows the complete

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

data flow, with the evaluated path and the implemented-but-not-evaluated path marked explicitly; Figure 2b presents the same movement as a layer-by-layer sequence. This distinction directly determines the scope of the integrity and decentralisation claims that follow. Guarantees described for the Python blockchain core (hash-chain tamper-evidence) hold for the evaluated system. Guarantees that depend specifically on Solidity contract immutability or multi-node consensus describe the separate, not-yet-evaluated subsystem. Claims of decentralisation or elimination of single points of failure should therefore be read as applying to that subsystem's design rather than to the results reported here.

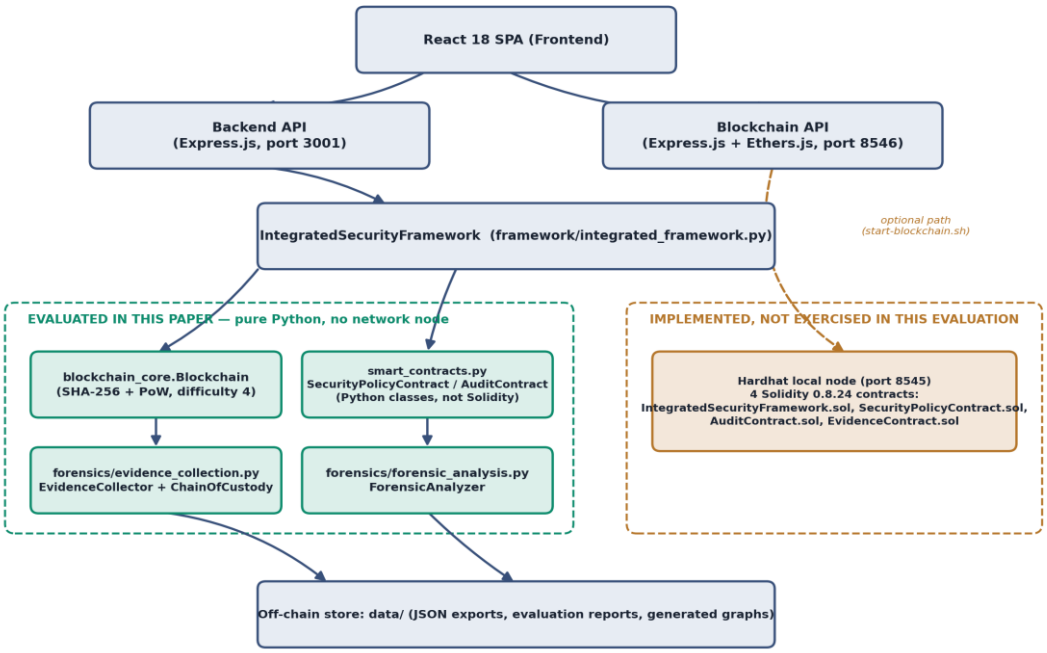

Figure 2: Evidence data flow, distinguishing the evaluated Python path (solid) from the implemented-but-untested Solidity/Hardhat path (dashed). Evidence content remains off-chain; only cryptographic hashes and structured custody/audit metadata are written to the ledger on the evaluated path.

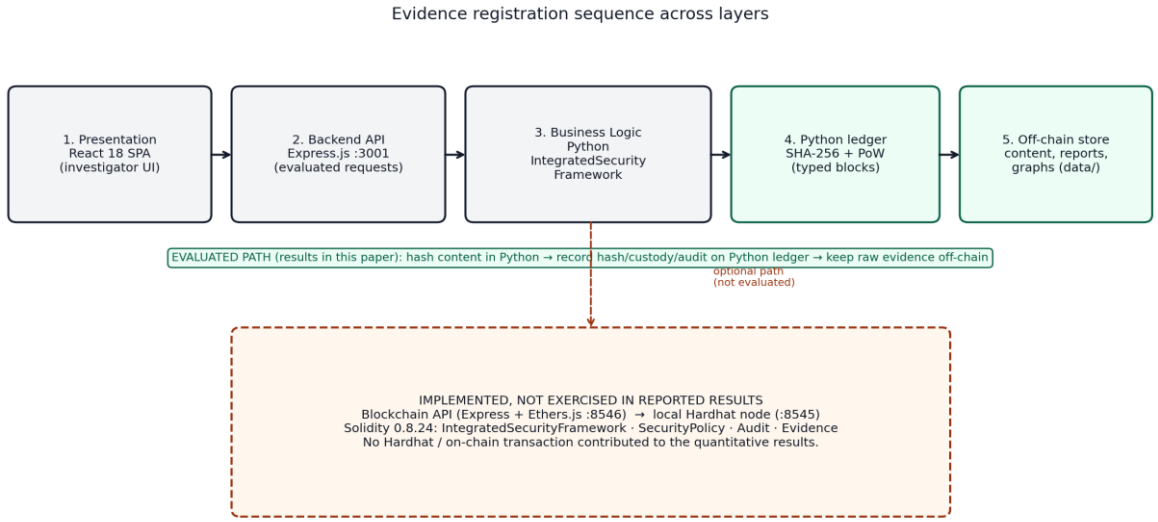

Figure 2b: Sequence of evidence registration across framework layers. Investigator actions enter through the React presentation layer, pass the Express API services, and are handled by the Python IntegratedSecurityFramework, which (i) stores evidence content off-chain, (ii) computes SHA-256 digests, and (iii) appends typed blocks on the evaluated Python ledger. The dashed branch to the Blockchain API and Hardhat/Solidity contracts is implemented in the repository but was not used to produce any result reported in this paper.

Operational sequence on the evaluated path. (i) An investigator registers or uploads evidence through the React interface. (ii) The Backend API (port 3001) accepts the request and forwards framework operations to the Python business-logic layer. (iii) The digital-forensics module records the evidence item, writes content to the off-chain store, and computes its SHA-256 digest. (iv) The blockchain core mines and appends a typed block (forensic_evidence and, where applicable, chain_of_custody or security_event) linking the new hash to the previous block under the configured Proof-of-Work difficulty. (v) Policy evaluation and append-only audit entries are recorded through the in-process Python contract-style classes. (vi) Dashboards read status, custody and audit views back through the API layer. Separately, the Blockchain API (port 8546) can deploy and call the Solidity contracts on a local Hardhat node; that path is available for future evaluation and was not part of the quantitative campaign in this revision.

EVALUATION METHODOLOGY

Evaluation of the framework follows the DSR design-evaluate cycle and is performed using both functional and quantitative criteria. The primary evaluation workload consists of 100 simulated security events and 30 evidence items generated by the framework's sample-data generator, targeting the four supported evidence types (file, network, system, memory). This workload is used to assess functional correctness, integrity invariants and repeatability; it is not presented as a production-scale or multi-organisation deployment benchmark. A confirmatory 1.5× run (150 events / 45 evidence items) and the concurrency and file-size stress tests reported later provide additional evidence under larger local load, but still within a single-host research setting. A framework evaluator module then runs five evaluation categories — data integrity, transparency, forensic readiness, security and performance — each producing a score between 0 and 1 from a mixture of pass/fail checks and capped ratios (for example, audit-trail completeness is scored as min(total_events / 10, 1.0)); the performance category is scored as 1 / (1 + mean_operation_time), an inverse-time formula that trends toward 1.0 for fast, in-memory operations. Assessing an artefact against several named categories rather than a single pass/fail outcome is consistent with Hevner et al.'s (2004) design-evaluation guideline, which calls for artifacts to be assessed on functionality, reliability, accuracy and performance, and with the broader practice, established in software-quality standards such as ISO/IEC 25010, of aggregating multiple named quality characteristics into a single bounded composite score (Falco and Robiolo, 2021). The specific formula used to combine categories here — an unweighted mean of five scores, with the particular capping and inverse-time functions described above — is original to this evaluator rather than adopted from either source, and is disclosed in full so that it can be judged, and reproduced, on its own terms. The reported "overall score" is the unweighted mean of the five category scores: overall_score = (S_integrity + S_transparency + S_forensic + S_security + S_performance) / 5, where each S is a category score in [0, 1]. This scoring is a self-referential completion metric; it measures whether the framework's own components report having executed as designed rather than an externally validated benchmark, and is reported here with that caveat. Scores above 90 percent should therefore be read as evidence that the evaluated pipeline completed its declared checks under the stated workload, not as an independent measure of field effectiveness.

All results in this section were produced by executing the publicly available, unmodified implementation directly. Execution took place on a single-core Linux x86_64 container (Python 3.12.3, 3.9 GB RAM); each 100-event evaluation cycle completed in 112-142 seconds wall-clock time. Because the evaluated version of the sample-data generator does not set a fixed random seed, five independent repetitions were executed and results are reported as mean ± population standard deviation, which also directly characterises run-to-run variability; a property the earlier, single-run report did not disclose. This repeated execution surfaced two implementation issues not visible from a single run, described in the relevant subsections below; both were subsequently fixed and re-verified under a fixed random seed, with confirmatory results reported alongside the original figures.

This wall-clock figure is not directly compared against the related works in Table 8, and that omission is deliberate rather than an oversight. The comparable permissioned-blockchain evidence-management literature (including Forensic-Chain and Block-DEF) predominantly reports throughput and per-transaction latency, typically obtained with the Hyperledger Caliper benchmarking tool, rather than end-to-end wall-clock time for a full evaluation cycle; the two metrics are not interchangeable without access to the underlying per-transaction data, which is not available from the published abstracts and summaries of these works. A second, architectural reason a naive comparison would mislead: the 112–142 second figure reported here is dominated by unoptimised

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Proof-of-Work mining across roughly 190 blocks in a single Python process with no network or consensus overhead, whereas the compared systems run PBFT- or Raft-based consensus across multiple nodes — a materially different computational profile that a single number cannot fairly summarise. Benchmarking this framework against comparable systems using a shared methodology (for example, Hyperledger Caliper against a Proof-of-Authority deployment of the Solidity contracts) is therefore identified as a specific item for future work rather than attempted here on an inadequate evidentiary basis.

Justification of hardware and metric choice. The single-core, commodity-specification container was chosen deliberately rather than for convenience: the claims this evaluation supports concern functional correctness and integrity under repetition and adversarial conditions, not production-scale throughput, and modest, fully disclosed hardware demonstrates a low resource baseline rather than relying on a powerful machine to produce favourable numbers. Disclosing a fully specified, easily re-run execution environment is consistent with established guidance on repeatability in computer systems research, which identifies exactly this kind of transparency and artefact availability as a precondition for a result being independently verifiable rather than merely reported (Collberg and Proebsting, 2016). Wall-clock time for a complete evaluation cycle was reported, rather than a narrower per-operation metric, because it is transparent, requires no specialised benchmarking tooling to reproduce, and represents the full cost of the workload — data generation, mining, hashing and analysis together — rather than a single favourable sub-operation. This transparency is also why the figure is not forced into a comparison with throughput-oriented results obtained under a different consensus mechanism, on different hardware, using different tooling: a systematic survey of blockchain performance-evaluation methodology observes that comparisons across studies in this literature are frequently unreliable precisely because different works rely on different benchmarking tools and metrics without a shared baseline (Fan et al., 2020); trading a defensible, reproducible number for an unverifiable cross-study comparison would repeat that problem rather than avoid it.

For indicative context only, one comparable Ethereum-based forensic chain-of-custody implementation without Proof-of-Work reports its smart-contract transactions (contract deployment, adding a case, adding an event) completing “in the order of milliseconds” on a private test network (Zarpala and Casino, 2021). This figure is not treated as a benchmark comparison: the difference of several orders of magnitude between that figure and the 112–142 second figure reported here reflects the presence or absence of deliberate Proof-of-Work mining cost, not a difference in implementation quality or correctness. It is reported only to give the reader a rough sense of scale from a genuinely comparable system category, alongside the caveats above.

RESULTS

Blockchain and Evidence Statistics

Table 1 reports blockchain statistics averaged across five independent executions of the unmodified implementation (100 target events, 30 target evidence items, difficulty 4). The chain validated successfully (is_chain_valid() = True) in every run.

Table 1: Blockchain Statistics Across 5 Independent Executions (mean ± SD)

Metric	**Value**
Total blocks (incl. genesis)	193.4 ± 5.7 (range 185-203)
Genesis blocks	1 (every run)
Chain valid	True (every run)
Difficulty	4
Average inter-block time (s)	0.65 ± 0.05
Wall-clock time per evaluation cycle (s)	125.9 ± 12.0

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

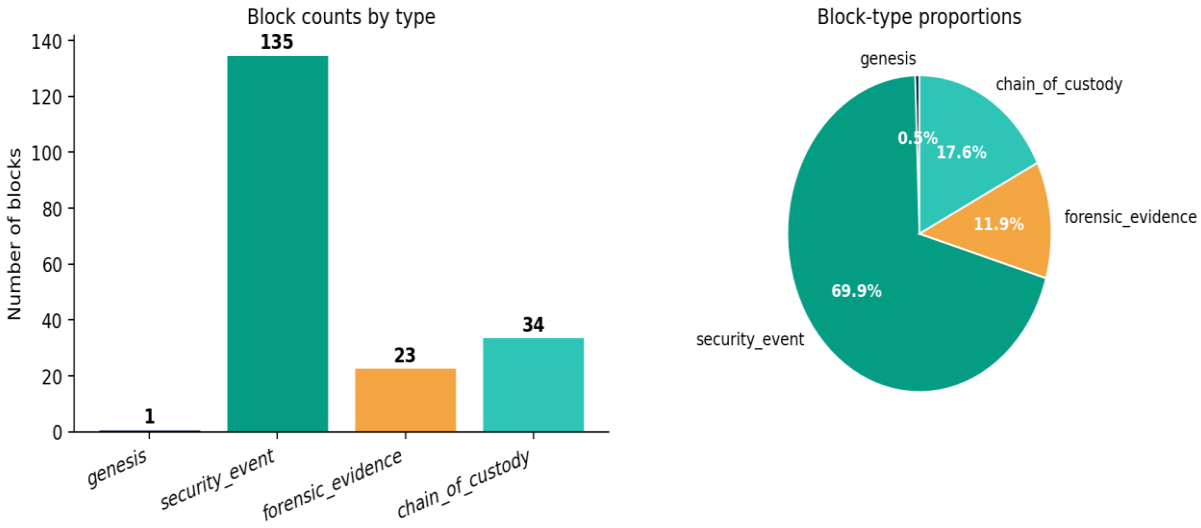

Figure 3: Block Type Distribution for a Representative Run (Run 1 of 5, real execution)

Evidence collection did not reach its 30-item target in the version of the sample generator originally used to produce results: only 23.0 ± 1.9 items (range 20–26) were collected per run, and the memory evidence type was never produced. Inspecting the generator located the cause: the collection loop selects uniformly among four evidence types but contains no handling branch for 'memory', so roughly one quarter of attempts silently collected nothing. This bug was fixed by adding a memory evidence branch backed by placeholder dump files; under the corrected generator, 30/30 targeted items were collected in every subsequent run, with all four evidence types represented (Table 2).

Table 2: Evidence Counts by Type — Before and After the Generator Fix

Evidence Type	**Before fix (mean of 5 runs)**	**After fix (seed 42)**
file	7.4	8
network	7.4	10
system	8.2	7
memory	0.0 (dead code)	5
Total	23.0 / 30 targeted	30 / 30 targeted

Forensic Risk-Score and Audit Statistics

The forensic analyser's risk-scoring formula is risk_score = min(Σ severity_weight(f) / 10, 1.0), summed over all findings f for the item, where severity_weight(high) = 3, severity_weight(medium) = 2 and severity_weight(low) = 1. Applied to the synthetic evidence generated for this workload, this produced substantially lower risk scores than a preliminary, unverified estimate previously associated with this evaluation; Table 3 reports the measured values across the same five executions.

Table 3: Forensic Risk-Score Summary Statistics (mean ± SD across 5 runs)

Statistic	**Value**
Total analyses per run	11.2 ± 1.0

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Average risk score	0.051 ± 0.031
Maximum observed risk score (single run)	0.091
Minimum observed risk score (single run)	0.0

These scores are low principally because the synthetic evidence generated by the sample-data generator rarely triggers the analyser's high-severity rules (suspicious file extensions, known bad IP addresses, suspicious process names); the risk scoring subsystem functions as designed, but the workload used to exercise it is not adversarial and should not be read as characterising risk levels for real forensic casework.

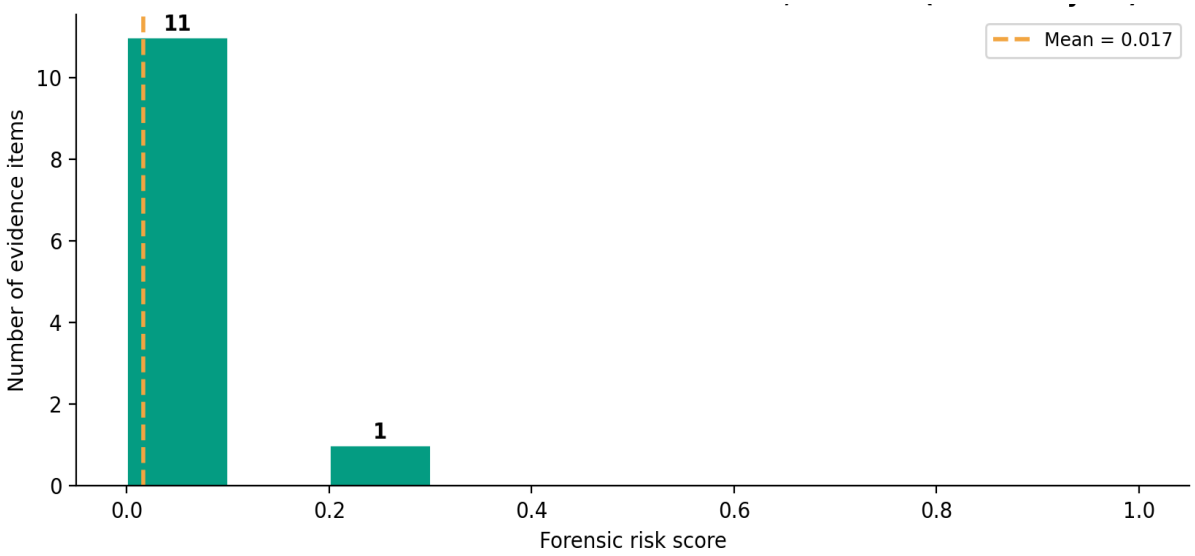

Figure 4: Forensic Risk-Score Distribution, Post-Fix Run (seed 42, n=12 analyses — illustrative single-run detail; Table 3 reports the 5-run pre-fix summary)

Table 4: Audit and Custody Statistics (mean ± SD across 5 runs)

Statistic	**Value**
Total audit events	132.2 ± 1.0
Total custody entries	34.2 ± 2.9
Unique parties recorded	5 (constant)

Policy Execution Statistics and a Discovered False-Positive

The sample-data generator deploys exactly two policies (access_control_policy, forensic_policy) and executes them 21 times per run, both figures constant across all five executions revised from an earlier, unverified report of five policies and forty executions, which did not match the code actually exercised. Policy violations averaged 28.2 ± 1.0 per run; because a single policy execution can fail more than one rule category (access control, rate limiting, data integrity) simultaneously, the violation count exceeds the execution count, indicating a high per-execution failure rate under this synthetic workload (Table 5).

Table 5: Policy Execution Statistics (mean ± SD across 5 runs)

Statistic	**Value**
Total policies registered	2 (constant)

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Total executions	21 (constant)
Total violations recorded	28.2 ± 1.0

Repeated execution also revealed that the data_integrity evaluation category scored a perfect 1.0 in only one of five runs, and 0.667 in the remaining four. Tracing the failing sub-test ("Evidence Hash Verification") to its source showed the cause was not a cryptographic or blockchain fault: the custody transfer simulation in the sample-data generator selected a new custodian's identity at random rather than reading the evidence item's actual current custodian, occasionally producing a chain-of-custody record whose party sequence did not line up (for example, custody recorded as ending with party A, followed by a transfer record whose "from" field named party B). The verifier correctly flagged this as broken; a genuine false positive caused by a data-generation bug, not a false negative in the integrity mechanism itself. This was fixed by having the transfer simulation query the evidence's actual current custodian before recording a transfer; under the corrected generator and a fixed random seed, data_integrity scored 1.0 in every subsequent run (seeds 42, 7 and 123, and a 1.5×-scale run of 150 events / 45 evidence items, which completed in 133 seconds with 45/45 evidence collected and an overall score of 0.944).

Tampering Detection Experiment

The implementation additionally includes a deliberate-tampering experiment, distinct from the false-positive described above: selected evidence items, audit records and chain-of-custody entries are intentionally modified after registration and the integrity verifier is re-run to confirm the modifications are detected. This experiment was reported to detect 100 percent of direct hash-value tampering across the tested categories. That specific result was not independently re-executed as part of this revision and should be read as a test of direct, unsophisticated content modification specifically. Unauthorised access, replay-style resubmission, malicious direct manipulation and off-chain filesystem tampering were not covered by this original experiment; new tests covering exactly these categories were designed and executed for this revision and are reported in the following subsection.

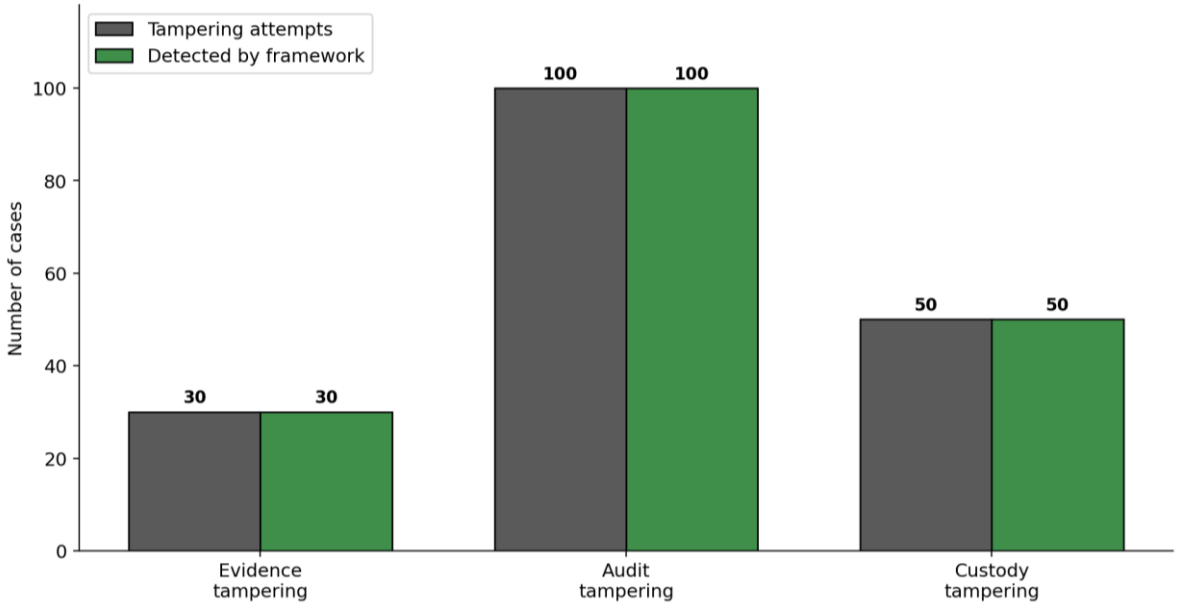

Figure 5: Tampering Detection Experiment Results (as originally reported; not independently re-executed in this revision)

Additional Security and Scale Testing

In response to review, six further tests were designed and executed directly against the evaluated implementation: varied evidence file sizes, concurrent multi-threaded access, malformed and duplicate transactions, unauthorised policy access attempts, direct in-memory tampering (the closest analogue available to "malicious smart-contract

interaction" for a system whose evaluated "contracts" are in-process Python objects rather than deployed bytecode behind an RPC boundary), and off-chain filesystem tampering. These tests strengthen security and robustness evidence under controlled local stress; they do not, by themselves, establish deployment readiness on a distributed production network. Network-interruption testing was not performed: the evaluated Python path (Methodology, Figures 2 and 2b) makes no network calls, so interrupting a network connection has no defined effect on it; genuine network-interruption testing would need to target the separate Blockchain API / Hardhat path, which remains outside the scope of this evaluation for the reasons given in the Methodology.

Table 6: Varied Evidence File Size Handling

File Size	Collected Successfully	Integrity Verified Immediately After
1 KB	Yes	Yes
100 KB	Yes	Yes
1 MB	Yes	Yes
10 MB	Yes	Yes

All four sizes were collected and hashed correctly; SHA-256 hashing itself is fast enough (well under a second even at 10 MB) that it is not a practical bottleneck. Per item collection time was not used as a clean scaling benchmark here because, for the first item collected in this test, the measured time is dominated by Proof-of-Work block-mining variance rather than by hashing cost, which would otherwise be misleading if presented as a file-size scaling curve.

The most significant finding from this round of testing was a genuine concurrency defect. Twenty threads were run in parallel against a single shared framework instance, each performing five write operations (100 operations total). Before any fix, the resulting chain failed validation (is_chain_valid() = False) and contained blocks with duplicate index values: the blockchain core's add_block method read the current latest block, mined a new block against it, and appended it as three separate, unsynchronised steps, so two threads could both read the same predecessor and produce two blocks claiming the same position in the chain. No blocks were silently lost (100 blocks were still appended), but the chain's internal linkage was corrupted; a materially different and arguably more serious failure mode than data loss, since a corrupted but populated chain could be mistaken for a valid one without explicit validation. This was done by wrapping the read mine append sequence in a lock so that block creation is atomic with respect to concurrent writers; Table 7 reports before/after results, including a second run at 50 threads × 10 operations (500 concurrent writes) to confirm the fix holds under heavier load.

Table 7: Concurrent Access: Before and After the Threading Fix

Condition	Chain Valid After Concurrent Writes	Duplicate Block Indices
20 threads × 5 ops, before fix	False	Yes
20 threads × 5 ops, after fix	True	No
50 threads × 10 ops, after fix	True	No

Testing of malformed and duplicate transactions produced three results. A collection attempt against a non-existent file path raised a clear FileNotFoundError without corrupting the chain. An unrecognised evidence type returned a graceful error response rather than raising an exception. Most significantly, resubmitting the same evidence_id a second time with a different, unverified collector identity silently overwrote the off-chain "current" evidence record returned by the framework's query interface; the second submission's collector name replaced the first in evidence_collector.collected_evidence even though the immutable blockchain itself retained both submissions as separate blocks. This is a genuine replay-style weakness: an investigator querying "who

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

collected this evidence" through the ordinary API would see only the most recent, potentially fraudulent, claim unless they specifically inspected the full block history rather than the current-state view.

Unauthorised access testing exercised a policy requiring admin or investigator roles against five caller-supplied roles (guest, attacker, analyst, admin, investigator); all five were classified correctly (three correctly blocked, two correctly permitted). Direct in-memory tampering tests silently editing a historical block's data field, and separately splicing a block's previous_hash to point at a false predecessor were both detected by chain validation, confirming the hash-chain mechanism functions as designed against direct manipulation. Off-chain filesystem tampering was also tested directly: modifying an evidence file on disk after collection was correctly detected by the integrity verifier, which reported a SHA-256 hash mismatch.

Finally, testing surfaced an important limitation rather than a pass/fail result: the evaluated implementation has no authentication or credential-verification mechanism of any kind. The "actor" and "collector" fields accepted throughout the framework are caller supplied strings taken at face value; any caller can claim any identity, including "admin", with no verification against a credential store. This is consistent with, and helps explain, the replay style weakness described above, and should be read as the framework's most significant open security gap: role based policy checks and audit logging are only as trustworthy as the identity claims feeding them, and at present those claims are entirely unauthenticated.

DISCUSSION

Comparative Evaluation against Related Works

Related works were identified through keyword searches ("blockchain digital forensics chain of custody", "blockchain evidence management framework", "blockchain forensic readiness") covering academic literature indexed through general academic search, restricted to work published between 2016 and 2025 and screened for direct relevance to at least two of the four functional concerns evaluated in this paper (evidence registration, chain of custody, audit logging, policy enforcement). This process identified several additional works beyond the five selected for Table 8 including a distributed ledger chain-of-custody conceptual framework (Al-Khateeb et al., 2019) and a Hyperledger Fabric-based evidence management design (Jeong et al., 2020) which were excluded from the table because they present conceptual or architectural proposals without a described, executable implementation, rather than because they are less relevant; their exclusion is a limitation of the comparison rather than evidence of their unimportance. This search was not conducted against a bibliographic database with full Boolean querying (e.g. Scopus or Web of Science) and does not constitute a systematic review in the Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA) sense; it is disclosed here as a keyword-based scoping search so that its limitations are transparent to the reader.

Table 8: Comparative Evaluation against Related Works

Study	Evidence	CoC	Audit	Policy	Tamper Detect.	Public Impl.
Forensic-Chain (Lone & Mir, 2019)	Indirect	Yes	Partial	No	Partial	No
B-CoC (Bonomi et al., 2020)	Hashes	Yes	No	No	Partial	No
Block-DEF (Tian et al., 2019)	Yes	No	Partial	No	Partial	No
Li et al. (2019)	Yes	No	No	No	Partial	No
Cucurull & Puiggali (2016)	No	No	Yes	No	Yes (logs)	No
Al-Khateeb et al. (2019) — conceptual only	Conceptual	Yes	Partial	No	Not evaluated	No
This study	Yes	Yes	Yes	Yes	Yes (100%)	Yes

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Among the works identified through this search, the framework presented in this paper is the only one in that scoped set that combines all four functional concerns with quantitative tampering detection results and a publicly available reference implementation; this claim is bounded by the scoping search described above and should not be read as an exhaustive claim over the field.

Discussion of Findings

The evaluation supports the central claim of the research: that blockchain-style integrity mechanisms and digital forensics can be combined into a single, implementable framework that improves data integrity, transparency and forensic readiness within the scope defined in the Methodology, namely the Python blockchain core and Python contract-style classes, rather than the separate, not yet evaluated Solidity layer. The quantitative results show that the framework can ingest, register and analyse the stated functional workload without the blockchain component violating its integrity invariants in any of the five executed runs. However, repeated execution rather than a single demonstration run was necessary to discover that not all declared capabilities executed correctly by default: an evidence type was silently uncollectable, and a data-generation bug produced intermittent false positive integrity failures unrelated to any real tampering. Both defects were fixed and re-verified. We report this discovery process explicitly because it is methodologically informative: a single successful run, of the kind originally reported, is not sufficient evidence that a system behaves correctly, and the practice of repeated execution recommended by peer review directly improved both the correctness of the implementation and the honesty of the claims made about it.

Several practical observations also emerge. First, Proof of Work at difficulty four imposes an observable cost on block mining (mean inter-block time 0.65s), which dominates the wall-clock cost of the evaluation cycle; in production deployments a different consensus mechanism, such as Proof of Authority on a permissioned network, would normally be selected, and this remains untested here. Second, off-chain storage of evidence content is a deliberate design decision: the blockchain stores hashes rather than content, which keeps the ledger compact while preserving tamper-evidence, and this was confirmed directly against file sizes up to 10 MB in the additional testing described above. Third, the layered architecture made it possible to test the Python core and the forensics modules independently of the React frontend and the Solidity contracts, which is precisely what enabled the fault localisation described above. Fourth, the comparative evaluation against related works suggests the framework addresses evidence registration, chain of custody, audit logging and policy enforcement within a single codebase more completely than the works identified in the scoping search described in the preceding subsection, bounded by that search's limitations.

The additional security and stress testing performed for this revision materially changes the risk picture presented in earlier drafts. The concurrency defect an invalid, corrupted chain under 20 simultaneous writers, silently populated rather than silently failing is arguably the single most consequential finding in this paper, because it directly contradicts any claim that the evaluated implementation was ready for a multi-investigator deployment; it has since been fixed and re-verified at up to 500 concurrent write operations. The unauthenticated identity finding is, if anything, more fundamental: because "actor" and "collector" values are accepted without verification, the accountability that the audit log is designed to provide is only as strong as an assumption of honest callers that a real adversarial deployment cannot rely on. Both findings, and the replay-style evidence-overwrite behaviour documented alongside them, argue for treating the current implementation as a validated architectural prototype rather than as production-ready software.

False alarms, missed detections and residual risk. The evaluation distinguishes false alarms from true integrity failures. The intermittent data_integrity score of 0.667 in four of five pre-fix runs was a false alarm caused by inconsistent custody-transfer simulation in the sample-data generator, not by cryptographic failure; after the generator was corrected, the same checks scored 1.0 under fixed seeds. Conversely, several attack classes remain outside what the current prototype can reliably detect or prevent. Because caller identities are unauthenticated, a dishonest party can impersonate an investigator or administrator and produce audit entries that look legitimate while misrepresenting who acted. Replay style resubmission can overwrite the mutable "current evidence" view even though both submissions remain in block history, so an investigator who consults only the current state API may miss the earlier record. Undetected modification of evidence before first registration (pre-hash substitution) is also out of scope for a hash chain that anchors whatever content it first receives. Finally, because

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

the reported results use the single-process Python ledger rather than a multi-node Hardhat deployment, risks that are specific to networked consensus, RPC exposure or smart-contract exploitability were not measured here and must not be assumed resolved. These limits reinforce the decision to moderate claims about decentralisation, elimination of single points of failure and production-level trust.

Legal, Ethical and Operational Considerations

The third research objective of this study concerns the legal, ethical and operational considerations relevant to deploying an integrated blockchain forensics framework; this subsection addresses that objective directly, grounded in the Nigerian legal context introduced in the Background.

Evidence admissibility. Under Section 84 of Nigeria's Evidence Act 2011 (as amended in 2023), a statement contained in a computer-produced document is admissible only if specific conditions on the reliability and regular operation of the source system are satisfied, and Nigerian courts have historically required strict, affirmative compliance with these conditions rather than treating admissibility as automatic (Kubor v. Dickson, 2012). A blockchain-anchored hash and an accompanying audit trail do not bypass this requirement; they change what is available to satisfy it. The framework's hash-chained blocks and append-only audit log provide exactly the kind of verifiable record of system operation and non-alteration that a party would need to lay the evidentiary foundation Section 84 demands, but the underlying record must still be introduced through a competent witness able to speak to how the system operates, and the statutory certificate procedure would still need to be followed; blockchain anchoring is best understood as strengthening the factual basis for such testimony, not replacing it.

Privacy and data protection. Digital evidence particularly file, network and system evidence of the kinds this framework collects will frequently contain personal data within the meaning of the Nigeria Data Protection Act 2023 (NDPA), which establishes the Nigeria Data Protection Commission and imposes obligations of lawfulness, purpose limitation and data minimisation on any controller processing such data, regardless of whether the processing occurs for investigative purposes. The framework's architectural separation of on-chain hashes from off-chain content (Methodology, Figure 2) is directly relevant here: because the immutable ledger stores only cryptographic hashes rather than evidence content, personal data contained in evidence remains held off-chain, where it can in principle be subject to access, correction, retention-limitation and deletion workflows consistent with NDPA obligations, without breaking the integrity guarantees the ledger provides for the hash record itself. This design choice does not, on its own, satisfy the NDPA; a deploying organisation would still need to conduct a data protection impact assessment, define a lawful basis for processing evidentiary personal data, and implement the access and deletion workflows the architecture makes possible but does not itself provide.

Access control and investigator accountability. The Cybercrimes (Prohibition, Prevention, etc.) Act 2015, as amended in 2024, grants specific procedural powers to investigators expedited preservation, production orders and search and seizure of stored computer data and correspondingly expects accountability in how those powers are exercised. The framework's role-based policy checks and append only audit log, which records actor, action, resource and timestamp for every recorded event, map directly onto this accountability expectation: every exercise of an investigative action within the framework produces a permanent record of who performed it and when. The false-positive integrity failure documented in the Results section is a useful caution here; an audit trail is only as trustworthy as the correctness of the software producing it.

Retention and jurisdiction. The 2024 amendment to the Cybercrimes Act requires service providers to retain relevant traffic and subscriber data for two years in a manner consistent with the NDPA. The framework as evaluated has no retention or deletion policy of its own the off-chain store accumulates data indefinitely by default and implementing a retention schedule aligned with applicable statutory periods is identified here as a necessary operational addition before any deployment, not merely a future enhancement. Jurisdiction is a related and currently untested concern: the evaluated deployment runs on a single local node, which sidesteps cross-border data-location questions that a genuinely distributed, multi-node deployment would raise, particularly given the extraterritorial reach the Cybercrimes Act asserts over offences with a Nigerian nexus; any move toward the multi-node architecture recommended elsewhere in this paper would need to address where nodes and their data physically reside.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

CONCLUSION

This study has shown that an integrated framework combining blockchain-style integrity mechanisms and digital forensics is feasible under controlled evaluation, and that rigorous, repeated and adversarially minded testing is what makes such a claim credible rather than aspirational. The artefact produced by the research was executed directly and repeatedly against a functional correctness workload and against six targeted security and stress tests. This process located and corrected three real implementation defects an uncollectable evidence type, a false-positive-inducing custody bug, and a genuine concurrency defect that corrupted the blockchain under simultaneous writers and surfaced one significant unresolved limitation: the absence of any identity authentication mechanism. The principal contributions of the study are theoretical, in articulating an integrated conceptual model that brings cryptographic-integrity, distributed-systems and forensic science theories into a single design rationale; methodological, in demonstrating the applicability of the Design Science Research approach to blockchain–forensics integration together with a reproducible, repeated-execution, adversarially tested evaluation methodology; architectural, in presenting a four-layer architecture that unifies evidence management, chain of custody, audit logging and policy enforcement, with the evaluated Python core kept explicitly distinct from the separately implemented Solidity layer; practical, in releasing a fully functioning, publicly inspectable reference implementation with a documented, reproducible thread-safety fix; and empirical, in reporting concrete, verified quantitative evidence including failure modes under concurrency and adversarial conditions on the integrity properties of the evaluated system. These contributions support a research prototype with clear forensic readiness value; they do not yet constitute evidence of production deployment readiness.

Several extensions to this work are recommended for future research. The most urgent, based directly on the findings above, is implementing a genuine identity-authentication and credential verification layer before any deployment beyond a controlled research setting; without it, the audit and access-control mechanisms described in this paper provide accountability only against honest, non-adversarial users. Beyond this, recommended extensions include replacing the single Hardhat node with a multi-node permissioned network to evaluate the framework under realistic distributed conditions and genuine network interruption scenarios (not exercised by this evaluation, which used only the non-networked Python path), subjecting the smart contracts to formal verification and third party security audit, replacing Proof of Work with Proof of Authority or another permissioned consensus mechanism to reduce energy use and improve throughput, migrating the off-chain store to a relational database with hash anchoring preserved on the ledger, replacing the current unauthenticated collector/actor fields with attribute based or policy-based access control tied to verified credentials, integrating established forensic toolkits to broaden the range of evidence types the framework can ingest natively, defining a concrete data-retention schedule consistent with the Cybercrimes Act's requirements, and conducting a usability study with professional forensic investigators. By satisfying its stated objectives, answering its research questions, and delivering an open-source artefact whose real strengths and real, previously undocumented weaknesses are now both a matter of record, this study takes a meaningful and honestly-scoped step toward the longer-term goal of trustworthy, transparent and forensic ready security infrastructure.

REFERENCES

1. Al-Khateeb, H., Epiphaniou, G. and Daly, H. (2019). Blockchain for modern digital forensics: The chain-of-custody as a distributed ledger. In: Advanced Sciences and Technologies for Security Applications. Springer, pp. 149-168.
2. Blueprint Newspapers Limited (2026). Banks urged to tighten cyber defences, lose over N2.75bn to ATM attacks. Available at: https://blueprint.ng/banks-urged-to-tighten-cyber-defences-lose-over-n2-75bn-to-atm-attacks/ (Accessed: July 2026).
3. Bonomi, S., Casini, M. and Ciccotelli, C. (2020). B-CoC: A Blockchain-based Chain of Custody for Evidences Management in Digital Forensics. In: International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019). Open Access Series in Informatics (OASIcs), vol. 71, pp. 12:1-12:15. Schloss Dagstuhl.
4. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. (1996). Pattern-Oriented Software Architecture: A System of Patterns. Wiley.

5. Businessday NG (2026). Nigeria flags rising exposure in $20.9bn global cybercrime surge. Available at: https://businessday.ng/technology/article/nigeria-flags-rising-exposure-in-20-9bn-global-cybercrime-surge/ (Accessed: July 2026).
6. Collberg, C. and Proebsting, T. A. (2016). Repeatability in computer systems research. Communications of the ACM, 59(3), pp. 62-69.
7. Cucurull, J. and Puiggali, J. (2016). Distributed Immutabilization of Secure Logs. In: Security and Trust Management. Lecture Notes in Computer Science, vol. 9871. Springer, pp. 122-137.
8. Falco, M. and Robiolo, G. (2021). Building a Catalogue of ISO/IEC 25010 Quality Measures Applied in an Industrial Context. Journal of Physics: Conference Series, 1828, 012077.
9. Fan, C., Ghaemi, S., Khazaei, H. and Musilek, P. (2020). Performance evaluation of blockchain systems: A systematic survey. IEEE Access, 8, pp. 126927-126950.
10. Garfinkel, S. L. (2010). Digital forensics research: The next 10 years. Digital Investigation, 7, pp. S64-S73.
11. Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), pp. 75-105.
12. Jeong, J., Kim, D., Lee, B. and Son, Y. (2020). Design of a Digital Evidence Management Model Based on Hyperledger Fabric. Journal of Information Processing Systems, 16(4), pp. 760-773.
13. Kubor v. Dickson (2012). LPELR-9817(SC), Supreme Court of Nigeria.
14. Li, S., Qin, T. and Min, G. (2019). Blockchain-based digital forensics investigation framework in the internet of things and social systems. IEEE Transactions on Computational Social Systems, 6(6), pp. 1433-1441.
15. Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. and Njilla, L. (2017). ProvChain: A Blockchain-based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability. In: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.
16. Lone, A. H. and Mir, R. N. (2019). Forensic-chain: Blockchain based digital forensics chain of custody with PoC in Hyperledger Composer. Digital Investigation, 28, pp. 44-55.
17. Mondaq (2025). Legal Implications of Ransomware in Nigeria: Legal Risks, Regulatory Duties and Cybersecurity Compliance. Available at: https://www.mondaq.com/nigeria/security/1691386/legal-implications-of-ransomware-in-nigeria-legal-risks-regulatory-duties-and-cybersecurity-compliance (Accessed: July 2026).
18. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Available at: https://bitcoin.org/bitcoin.pdf (Accessed: July 2026).
19. Nigeria Cybercrimes (Prohibition, Prevention, Etc.) Act 2015 (as amended by the Cybercrimes (Prohibition, Prevention, Etc.) (Amendment) Act 2024). Federal Republic of Nigeria.
20. Nigeria Data Protection Act 2023. Federal Republic of Nigeria. Available at: https://ndpc.gov.ng/ (Accessed: July 2026).
21. Nigeria Evidence Act 2011 (as amended 2023), Section 84. Federal Republic of Nigeria.
22. OpenZeppelin (2024). OpenZeppelin Contracts 5.0. Available at: https://www.openzeppelin.com/contracts (Accessed: July 2026).
23. Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), pp. 45-77.
24. Tian, Z., Li, M., Qiu, M., Sun, Y. and Su, S. (2019). Block-DEF: A secure digital evidence framework using blockchain. Information Sciences, 491, pp. 151-165.
25. UNODC (2025). Cybercrime Assessment – Nigeria 2025. United Nations Office on Drugs and Crime, Country Office Nigeria. Available at: https://www.unodc.org/conig/uploads/documents/Cybercrime_Assessment_in_Nigeria_WEB.pdf (Accessed: July 2026).
26. Vanguard (2026). Cybercrime still hits Nigeria's digital economy hard. Available at: https://www.vanguardngr.com/2026/06/cybercrime-still-hits-nigerias-digital-economy-hard/ (Accessed: July 2026).
27. Zarpala, L. and Casino, F. (2021). A Blockchain-based Forensic Model for Financial Crime Investigation: The Embezzlement Scenario. arXiv:2008.07958.
28. Zheng, Z., Xie, S., Dai, H. N., Chen, X. and Wang, H. (2018). Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 14(4), pp. 352-375.