A Blockchain-Enabled Deep Learning Framework for Secure Omics Data Sharing and Attack Detection

Don Roosan[1,*], Md Rahatul Ashakin[2], Rubyat Khan[3] and Mazharul Karim[4]

[1]*Merrimack College, North Andover, MA, 01845, USA*

[2]*Washington University of Science and Technology, Alexandria, VA, 22314, USA*

[3]*University of Nebraska Medical Center, Omaha, NE, 68198, USA*

[4]*University of Texas at El Paso, El Paso, TX, 79968, USA*

Abstract: Securing omics datasets against tampering and misuse is essential for reproducible research and privacy. We present a defense-in-depth framework that combines Ethereum smart contracts for tamper evidence, provenance, and fine-grained access with a Long Short-Term Memory (LSTM) intrusion detection system that models event sequences to flag abnormal behavior. Raw omics files remain off-chain; their SHA-256 digests and permissions are recorded on-chain. Authorized consumers obtain contract-mediated tokens to fetch encrypted data, recompute hashes, and verify integrity. The intrusion detection system (IDS) ingests blockchain transactions and storage access logs in sliding windows to detect bursts, probing, and insider over-access. We implement the system on a permissioned Ethereum network and evaluate it with a simulated case study using public gene-expression files and scripted attacks. All post-registration data modifications were detected at verification time (100% integrity detection). Behavioral attacks were identified with 95% precision and 90% recall, reducing false alarms to 1% of windows and outperforming a rules-only baseline (80% precision, 75% recall). Transaction latency and resource costs remained modest. These results demonstrate a practical path to trustworthy omics sharing that unites cryptographic immutability with monitoring. Our design supports consent enforcement and lays groundwork for extensions such as Merkle-root batching, key rotation, and federated or transformer-based detectors.

Keywords: Omics security, blockchain, Ethereum, smart contracts, data integrity, access control, LSTM, intrusion detection, anomaly detection, bioinformatics.

1. INTRODUCTION

Omics datasets—spanning genomics, transcriptomics, proteomics, and related modalities—have become indispensable to modern biomedicine. Population-scale sequencing and high-throughput profiling feed discovery pipelines for disease risk prediction, target identification, and precision therapeutics [1]. These data are inherently sensitive: a genome encodes immutable personal attributes and familial relationships, and derived expression profiles may reveal disease states or treatment histories [2]. As repositories grow in size and value, so does the incentive for adversaries to exfiltrate, corrupt, or subtly manipulate records [3]. Beyond privacy harms, integrity failures have scientific consequences: if files are altered or mislabeled without detection, downstream analyses can become irreproducible or misleading, eroding trust in published findings and clinical translation [4].

Securing omics data is uniquely difficult because it is rarely confined to a single institution. Multi-site consortia, biobanks, and research networks routinely share subsets of samples under complex consent terms [5]. Traditional security architectures center on a single administrative domain: a database guarded by perimeter defenses, access-control lists, and application logs. While familiar and operationally convenient, this model exhibits well-known weaknesses. Centralized administrators can be coerced or compromised; logs can be altered or selectively deleted; and “authorized” users can still behave maliciously by over-accessing or siphoning data [3]. Furthermore, classic systems often lack strong, cryptographic guarantees that a file retrieved today is exactly the one that was originally registered. In a domain where single-nucleotide changes matter, probabilistic or best-effort checks are insufficient [6].

Blockchain technology offers a compelling complement. A blockchain provides an appendonly, tamper-evident ledger replicated across independent nodes. On Ethereum, smart contracts encode access policies and emit immutable event trails; cryptographic hashes recorded on-chain act as durable fingerprints of off-chain artifacts [7–9]. If a retrieved file’s hash diverges from the committed digest, even by a single bit, verification fails and the discrepancy is auditable by all participants [6,7]. For healthcare contexts, a permissioned or consortium Ethereum network can

*Address correspondence to this author at Merrimack College, North Andover, MA, 01845, USA; E-mail: roosand@merrimack.edu

preserve these guarantees while offering operational control and predictable costs [10,11]. Crucially, large omics files remain off-chain in secure storage; only compact metadata, hashes, and permissions are anchored on the ledger to preserve privacy and scalability [8,9].

Yet blockchains alone do not solve behavioral security. A smart contract will faithfully grant access to a caller with valid credentials; it does not judge whether that caller's pattern of activity is suspicious. Many real incidents exploit exactly this gap: credential theft, insider abuse, or automated scraping that stays within nominal rate limits [3]. Detecting such misuse requires modeling sequences over time—who accessed what, how often, and in which order—and distinguishing legitimate research workflows from anomalous bursts, probing, and lateral movement. Signature-based intrusion detection struggles with novel attacks and evasion. In contrast, sequence models from deep learning—particularly Long Short-Term Memory (LSTM) networks—are effective at learning temporal dependencies and flagging deviations without hard-coded rules [12,13].

This work unifies these two paradigms into a defense-in-depth framework for secure omics sharing. We propose an architecture in which data providers store encrypted files in off-chain storage and commit their SHA-256 digests, provenance metadata, and fine-grained permissions to an Ethereum smart contract [6,7]. Data consumers query the contract to discover datasets, obtain contract-mediated authorization tokens, retrieve the corresponding objects from storage, and verify integrity by recomputing hashes. In parallel, an LSTM-based intrusion detection system (IDS) ingests a consolidated stream of blockchain transactions and storage access logs (e.g., fetch events, volumes, timings). Using sliding windows over these event sequences, the IDS outputs anomaly scores that drive alerts and automated mitigations such as throttling or temporary quarantine [12,13]. The ledger enforces what is allowed and preserves an incorruptible audit trail; the LSTM learns how normal looks and spots misuse that otherwise slips through.

Our design goals reflect practical constraints faced by research consortia. First, strong integrity and provenance: every file must be verifiable against a public, immutable commitment, enabling reproducible analyses and auditability [6–9]. Second, fine-grained, consent-aware access control: smart contracts should express per-dataset or per-cohort permissions and support revocation and expiry, aligning with evolving consent or data-use agreements [5,10]. Third, early anomaly detection with low false-alarm rates: operational teams cannot respond to constant noise; the IDS must prioritize precision while retaining sensitivity to diverse attack modes [12,13]. Fourth, operability and cost proportionality: transactions must complete in seconds, not minutes, with gas costs compatible with research workloads; performance on permissioned Ethereum is compatible with typical research cadence [11]. Fifth, privacy preservation: raw omics never appear on-chain; identities and logs are pseudonymized where feasible; encryption at rest and in transit is assumed, with keys released only to authorized principals [5,8,9]. Finally, governance and recoverability: administrators should be able to pause contracts during incidents, rotate keys, and map institutional identities to wallet addresses without centralizing all power [10].

We target a permissioned Ethereum network, i.e., a consortium blockchain where participating institutions operate validator nodes and membership is governed by policy. Compared to public chains, permissioned networks provide predictable latency, private membership, and operational control while retaining append-only, tamper-evident logs and smart-contract programmability. This setting aligns with multi-institution omics workflows that require governed participation and auditability without public write access. We present a concrete instantiation of this framework. On the blockchain side, a DataRegistry contract records dataset digests and URIs, manages an access-control list per item, and emits events on registration and access decisions [7–9]. We employ role-based controls and pausability to reduce administrative risk, and we discuss rate-limiting patterns and event-driven gateways that deliver decryption keys to authorized users [10]. On the analytics side, the IDS represents each event as a compact feature vector—event type, pseudonymous user identifier, time bin, dataset reference, and simple counts—then passes a rolling window to a two-layer LSTM followed by a dense classifier; thresholds are tuned on benign validation logs to bound false positives [12,13].

To evaluate the approach, we emulate a realistic consortium workflow using publicly available gene-expression files as stand-ins for protected content. Providers register datasets and assign permissions to a small group of consumers who perform routine analyses at human-scale cadence. We inject two classes of adversarial behavior. Integrity attacks arbitrarily alter a subset of stored files after registration to test whether verification at access catches every modification. Behavioral attacks simulate a compromised account that attempts rapid, broad retrievals beyond typical usage, and repeated requests for unauthorized datasets to elicit denials. We contrast our system's detection capability with a baseline rule set—fixed rate limits and simple counters—chosen to reflect what many institutions can deploy with minimal tooling.

The results, detailed later, show that the cryptographic layer guarantees tamper evidence at the point of use, while the LSTM discriminator substantially improves detection quality over rules alone, producing fewer spurious alerts without sacrificing sensitivity [11–13]. Equally important, transaction latency and throughput on a permissioned Ethereum network remain compatible with research workflows, and the IDS computes in sub-second time on commodity CPUs [11,12]. While no single mechanism is sufficient against all threats, the combination of immutable provenance and learned behavior modeling markedly raises the bar for attackers and increases operator confidence that misuse will be visible and actionable.

This paper advances the discourse in two ways. Conceptually, it frames omics security as the union of data-centric guarantees and behavior-centric monitoring and demonstrates that the two reinforce each other when co-designed. Practically, it provides an implementation blueprint—covering contract interfaces, event schemas, and an IDS feature pipeline—that others can reproduce and adapt. Our aim is not to replace institutional controls or legal agreements, but to supply cryptographic and algorithmic scaffolding that makes those controls verifiable and violations observable across organizational boundaries.

2. METHODS

2.1. System Architecture

We designed a defense-in-depth system that couples a *permissioned* Ethereum ledger for integrity, provenance, and fine-grained access with a sequence-aware intrusion detector that models how the platform is used over time [15,16,35]. Data providers deposit raw omics files into secure off-chain storage and immediately commit a cryptographic digest plus metadata and permissions to the chain. Data consumers discover datasets and request access via

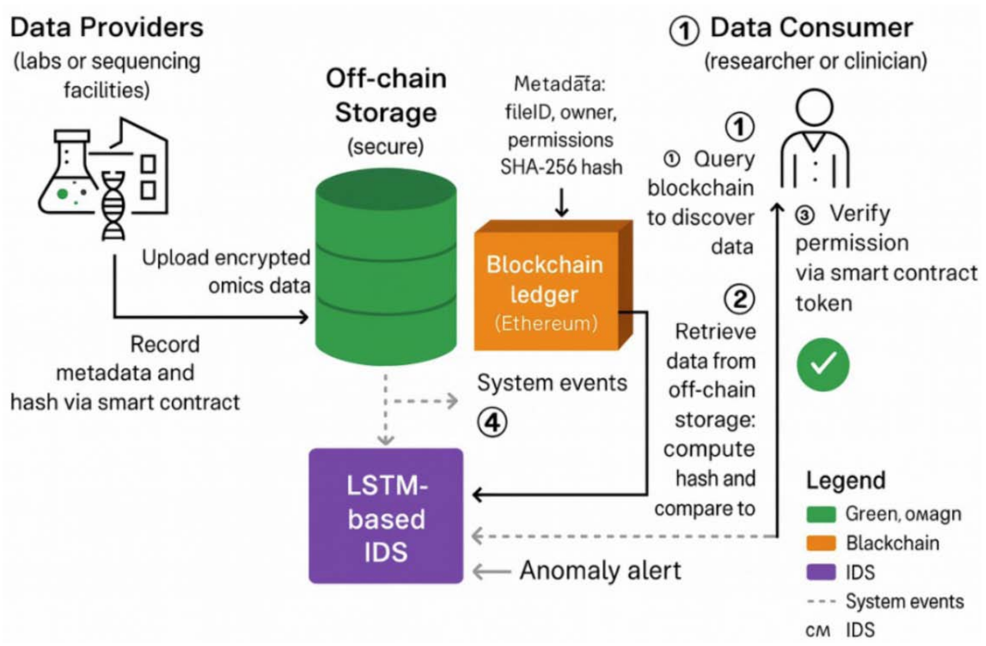
Data Providers
(labs or sequencing facilities)
Off-chain Storage
(secure)
Metadāta: fileID, owner, permissions SHA-256 hash
① Data Consumer
(researcher or clinician)
①
① Query blockchain to discover data
③ Verify permission via smart contract token
Upload encrypted omics data
Blockchain ledger
(Ethereum)
②
Retrieve data from off-chain storage: compute hash and compare to
Record metadata and hash via smart contract
System events
④
LSTM-based IDS
Anomaly alert
Legend
Green, омagn
Blackchain
IDS
System events
CM IDS

Figure 1: System architecture. (1) Provider encrypts an omics object with AES-GCM and registers its SHA-256 commitment, metadata, and permissions on the Blockchain (permissioned Ethereum). (2) Consumer requests an object from off-chain storage only after obtaining authorization, using the storage pointer recorded on chain. (3) The consumer proves authorization by presenting a smart-contract approval event to an off-chain key gateway, which verifies event inclusion/finality and then releases a time-boxed, single-use decryption token. (4) Client retrieves the ciphertext, decrypts, hashes the plaintext, and compares to the on-chain commitment (tamper evidence). (5) Blockchain events and storage access logs stream to the IDS, which scores sliding windows and emits anomaly alerts to operators and an orchestration service (e.g., to throttle or quarantine).

Table 1: Security Threats to Omics Data and Mitigations

Threat scenario	Risks in traditional systems	Mitigation in proposed framework
Data tampering (integrity attack)	Silent modification/corruption invalidates analyses; no immutable reference to detect changes.	On-chain SHA-256 commitments; client verification rejects mis-matches; full audit trail of revisions.
Unauthorized data access	Credential reuse or server compromise leaks sensitive genomic data.	Smart-contract ACLs; wallet-based auth; event-driven key release; non-permitted callers denied and logged.
Insider misuse / excessive access	Authorized users over-download or share improperly, coarse monitoring misses atypical cadence.	LSTM IDS models normal per-user sequences; flags bursts/atypical timing for early intervention.
Brute-force / DDoS probing	Floods overwhelm endpoints; threshold-only rules noisy.	On-chain rate limits + pausability; IDS detects surges and triggers throttling/quarantine.
Database exploit	Direct dumps bypass application logs; tampering/exfiltration remains covert.	Ledger-mediated, auditable retrievals; off-ledger dumps fail hash checks; IDS observes anomalous storage-access patterns.
Privacy and re- identification	Over-exposure of identifiers; weak consent enforcement.	Fine-grained on-chain consent/ACLs; only hashes/URIs on chain; optional dual-chain/GDPR workflows; pseudonymized IDS features.

smart contracts; if authorized, they retrieve encrypted objects from storage and verify integrity by recomputing the digest and comparing it with the on-chain commitment.

In parallel, an analytics pipeline ingests blockchain events and storage access logs, constructs feature sequences, and feeds them to an IDS that models how the platform is used over time to detect behavioral anomalies. The ledger enforces what is allowed and preserves an immutable audit trail; the LSTM judges whether observed behavior matches learned patterns of legitimate use [12,13,36].

2.2. Blockchain Component and Smart Contract Design

The blockchain layer runs on a consortium Ethereum network using proof-of-authority consensus to achieve predictable latency and governance [15,16]. Participants transact from externally owned accounts; validator nodes are operated by distinct organizations in the consortium. The principal contract, *DataRegistry*, exposes functions to register new objects, mutate per-item permissions, check access, and verify digests. Registration writes a stable identifier, the SHA-256 digest, a pointer to off-chain storage, and metadata required for provenance. Access checks are performed when a consumer calls the contract; successful evaluation emits an approval event carrying a reference to the object's decryption key, while denials are logged to the same immutable trail. The contract is hardened with role-based access control for administrative operations, pausability for incident response, and conservative state-change patterns to avoid reentrancy. Simple per-address counters provide on-chain rate limiting that complements off-chain throttles. All state transitions emit typed events to support downstream monitoring without expensive on-chain queries [17-19].

We deploy a permissioned Ethereum network using IBFT 2.0 proof-of-authority with immediate finality. Validators (≥4) are operated by distinct institutions and managed via multisignature governance for add/remove actions. Nodes run with network-level allowlists (P2P and RPC), mTLS for operator access, log shipping to a WORM archive, and continuous health/attestation checks. Finality at the consensus layer ensures that once an Approval event is observed, inclusion cannot be reverted under honest-majority assumptions, which is critical for safe key release. Rate-limiting at the gateway and contract-level pausability provide defense-in-depth against abuse and incident response.

2.3. Off-Chain Storage, Encryption, and Integrity Commitments

Raw omics files never appear on chain. Providers encrypt each object at rest using AES-GCM with a unique symmetric key [20]. The storage pointer recorded on chain references a durable location such as object storage or an IPFS content identifier [21]. The approval event includes a compact reference to a key escrow service that releases the symmetric key only to the authorized requester after validating the event signature and performing multifactor checks, using typed structured-data signatures for robustness [23]. The client decrypts the ciphertext, hashes the plaintext, and compares the result with the on-chain commitment committed during registration; any discrepancy signals modification, corruption, or substitution. For collections, providers may group files into Merkle trees and commit a root on chain, enabling efficient verification of individual chunks and periodic attestation of entire datasets without rehashing every object [22].

Each object is encrypted with a random 256-bit data-encryption key (DEK) generated by the site's KMS/HSM. The DEK is used once per object with AES-GCM, a unique 96-bit nonce, and authenticated additional data (AAD) binding {dataset_id, version, MIME, length}. The DEK is never stored in plaintext; it is wrapped by a long-lived key-encryption key (KEK) that resides and operates inside the KMS/HSM (envelope encryption). Stored artifacts include the ciphertext, GCM tag, nonce, AAD, and the KEK-wrapped DEK.

When the smart contract emits an Approval event for {caller, dataset_id, block_number}, the off-chain gateway verifies event inclusion and finality on the permissioned chain, checks caller identity/MFA, and issues a single-use, time-boxed token to unwrap the DEK inside the KMS/HSM. Tokens are origin-bound (mTLS) and logged. Denials are logged symmetrically.

KEKs have a short cryptoperiod; rotation is orchestrated by the KMS/HSM. On rotation, wrapped DEKs are re-wrapped under the new KEK; high-value objects may be re-encrypted under fresh DEKs opportunistically (background rekeyer) with old wrapped DEKs securely destroyed. This limits blast radius if a KEK is later compromised.

Transport uses TLS 1.3 with ephemeral ECDHE, providing forward secrecy in transit. At rest, per-object DEKs ensure that compromise of any single DEK does not expose other objects. Compromise of a KEK does not reveal past plaintexts if (i) the KEK is rotated and old wraps are destroyed, or (ii) objects are re-encrypted under new DEKs during rotation. Operationally, we combine periodic KEK rotation, background rekeying, and strict audit to approximate forward secrecy over time for stored content.

Nonces and integrity. GCM nonces are never reused; nonces are generated randomly and stored next to the ciphertext. GCM authenticates AAD; the on-chain SHA-256 commitment is a provenance anchor that complements (not replaces) AEAD integrity.

We commit the SHA-256 of the plaintext to support reproducible verification across re-encryptions; implementations may optionally include a public salt in the commitment record to harden against dictionary lookups on small, known files.

2.4. Identity, Consent, and Access Workflow

Real-world identities are mapped to blockchain addresses in a lightweight registry governed by the consortium. During onboarding, institutions attest to a user's role and data-use agreements, binding those attributes to the user's wallet address. Consent and policy are encoded as per-item allow-lists in the registry, which can be updated to grant, revoke, or time-box access. The typical flow begins when a consumer queries the ledger to discover a dataset and then submits a request. The contract verifies the caller's address and the item's policy. On approval, the event stream signals the off-chain gateway to release an object-specific decryption key; on denial, the contract logs the decision, which the detector later interprets in context [19]. On approval, the off-chain key gateway (not the smart contract) verifies inclusion/finality of the Approval event, authenticates the requester (MFA), and releases an origin-bound, single-use token that permits the client to unwrap the DEK inside the KMS/HSM. Key rotation and address recovery are supported by linking new addresses to existing identities under multisignature approval so that security hygiene does not break provenance [34].

2.5. Event Collection and Feature Construction

Two log sources drive the anomaly detector: blockchain events and storage access logs. The first captures all consequential control-plane activity, including registrations, grants, denials, approvals, and administrative actions; the second records data-plane behavior such as successful fetches, bytes served, latency, throttling, and failed retrievals. To preserve privacy, user and dataset identifiers are pseudonymized with keyed hashing before feature construction [24,25]. Each raw event is standardized, timestamped, and enriched with rolling aggregates such as requests per unit time and denial ratios. Time-of-day is encoded with circular features to capture diurnal rhythms. For sequence modeling, events are grouped into fixed-length windows in temporal order, both per user. Windows are built with stride one to enable early detection while keeping adjacent samples highly overlapping; data leakage is prevented by constructing train, validation, and test sets on disjoint time intervals and disjoint user cohorts.

The IDS runs as a log-consumer service (one per consortium or per site) that subscribes to contract events and storage logs. It emits alerts to an operator console and an orchestration hook that can trigger throttling or contract pausability. The IDS does not gate key release directly; instead, alerts inform operators and automated guards that act via well-defined controls.

2.6. LSTM Intrusion Detection Model

The detector uses a compact two-layer Long Short-Term Memory (LSTM) followed by a sigmoid classifier [36]. Categorical tokens such as event type, pseudonymous user, and dataset identifier are embedded into dense vectors and concatenated with numerical features comprising rate, volume, inter-arrival, denial fraction, latency, and time encodings. The first LSTM layer captures short-range dependencies within an access burst; the second abstracts higher-level patterns across mixed event types. Layer normalization and dropout reduce covariate shift and overfitting, and focal modulation can emphasize hard positives in rare-event settings [26–28]. The model outputs an anomaly probability for each window. Because anomalies are rare, the training objective uses class weighting or focal modulation to emphasize hard positives without inflating false positives. Inference runs continuously on streaming windows; when the output crosses a calibrated threshold, the system emits an alert with the responsible address, the implicated datasets, and the causal sequence fragment to support triage.

2.7. Training Procedure and Thresholding

To learn realistic boundaries, the detector is trained on a mixture of benign sequences and synthetically injected attacks that mirror adversary goals observed in data-sharing systems. Benign sequences arise from replayed traces of human-scale analytics with small batches and idle periods. Attack sequences comprise rapid-fire retrievals from a compromised account, repeated requests for unauthorized items designed to elicit denials, and low-and-slow patterns that mimic normal cadence while gradually increasing volume. The dataset is partitioned chronologically into train, validation, and test splits, with users held out to assess generalization across principals. Optimization uses Adam with early stopping on average precision to avoid overfitting to a specific operating point; operating characteristics are reported with precision, recall, F1, and area under the precision–recall curve, which is preferred for imbalanced data [29,30]. The alert threshold is set on a purely benign validation set to bound false alarms per day to an operationally acceptable rate; operating characteristics such as precision, recall, and mean time to first alert are then evaluated across the full test set, independent of threshold tuning.

2.8. Workload Generation and Attack Scenarios

We emulate a consortium in which a provider registers approximately one hundred gene-expression files as representative omics payloads and grants access to a dozen analyst accounts. Normal activity balances exploratory browsing and targeted retrieval, modeled with inhomogeneous Poisson arrivals to capture business-hour concentration and occasional evening work. The integrity threat model modifies a small fraction of stored objects after registration through byte-level edits and label swaps. The behavioral threat model includes three scenarios. In the burst exfiltration case, a compromised credential drives dozens of accesses within a few minutes, often still within nominal rate caps but outside human cadence. In the probing case, an actor repeatedly requests datasets they are not authorized to see, generating sequences of denials interleaved with legitimate activity. In the low-and-slow case, an attacker paces requests just under typical rates over extended periods, attempting to evade both rules and naive anomaly detectors.

2.9. Evaluation Protocol and Metrics

Evaluation focuses on both security efficacy and operational cost. Integrity efficacy is measured as the fraction of modified objects rejected by client-side verification on first access and during periodic attestation runs; because verification is cryptographic, the expected ideal is complete detection. Behavioral efficacy is measured with precision, recall, F1 score, and area under the precision–recall curve computed over attack windows, alongside mean time to first alert from the start of an attack sequence. Operational cost is measured as false alarms per day on purely benign sessions, smart-contract latency and throughput under realistic gas limits, and detector inference time on commodity CPUs. To contextualize gains, a rules-only baseline that enforces fixed per-user caps and simple counters is evaluated under identical workloads.

Uncertainty in summary metrics is reported using bootstrap resampling over independent sessions, and all model selection decisions are confined to the training and validation partitions [31].

2.10. Implementation and Reproducibility

Smart contracts are implemented in modern Solidity and statically analyzed with off-the-shelf tools to reduce common pitfalls [32,33]. The off-chain gateway validates contract events with typed signatures and enforces multifactor authentication before releasing keys [23]. The streaming pipeline subscribes to validator nodes via websockets, ingests storage logs, and materializes feature windows using a stateful stream processor [19]. The LSTM is implemented in a mainstream deep learning framework with reproducible seeds and versioned artifacts; trained weights, embeddings, and preprocessing code are archived alongside a workload generator that can recreate all figures and tables from a clean deployment. A one-click script provisions a fresh permissioned chain, deploys contracts, seeds accounts, replays workloads, and exports evaluation reports so that results can be independently verified without access to sensitive data.

2.11. Governance, Safety, and Privacy Controls

The system integrates governance patterns aligned with institutional risk. Administrative operations such as contract upgrades, emergency pauses, and role assignments require multisignature approval [34]. Key rotation and account recovery are auditable and bound to real-world identities in the registry, ensuring continuity without weakening provenance guarantees. Logs, models, and features are retained under least-privilege policies with scheduled deletion; identifiers used for modeling are pseudonymized with rotating secrets to reduce linkability [24,25]. Because integrity checks trigger at access time, a background attestor periodically recomputes Merkle proofs to surface dormant tampering [22]. Together, these controls help ensure that cryptographic guarantees and behavioral monitoring are complemented by operational safeguards that keep the platform usable and trustworthy at consortium scale.

3. RESULTS

3.1. Experimental Context

We evaluated the proposed framework in a consortium-style setting that mirrors a realistic omics-sharing workflow. A data provider registered approximately one hundred gene-expression files as stand-ins for protected omics artifacts. Each file was stored off-chain as an encrypted object, while its SHA-256 digest, provenance metadata, and access policy were committed to a permissioned Ethereum ledger. A cohort of twelve analyst accounts, mapped to wallet addresses via the identity registry, performed routine discovery and retrieval tasks over several days of simulated activity with diurnal rhythm and small analyst-sized batches. Throughout, the Long Short-Term Memory (LSTM) detector consumed a unified stream of blockchain events and storage access logs, scoring sliding windows for anomalies without interfering with normal operations. Two families of adversarial behavior were injected: integrity tampering and behavioral misuse.

3.2. Integrity Verification

The integrity mechanism yielded a binary, self-verifiable signal at the point of use. Whenever a consumer retrieved a tampered object, client-side recomputation of the SHA-256 digest failed to match the on-chain commitment, and the object was rejected. Across byte-level edits, header manipulations, and label swaps deliberately crafted to be subtle, detection was perfect: all altered artifacts were identified as inconsistent on first access, corresponding to 100% tamper detection. This behavior is consistent with the avalanche property of cryptographic hashing and is valuable in practice because it removes reliance on a trusted log server or centralized auditor; any participant can independently verify that the file being analyzed is exactly the one originally registered. Integrity checking also surfaced provenance value beyond simple pass/fail: the immutable registration record tied each dataset to a specific uploader, time, and policy state, making any subsequent investigation of suspicious changes straightforward and auditable.

3.3. Intrusion Detection Performance

The LSTM-based intrusion detector substantially improved the visibility of misuse patterns over a rules-only baseline. Trained on benign sequences augmented with synthetic attack windows, the model achieved high precision while retaining sensitivity to diverse behaviors. Across the full test period, attack windows were flagged with 95% precision and 90% recall, producing an F1 score near 0.92 under the natural class imbalance of the workload. False alarms were rare; using a threshold calibrated on a purely benign validation split, the detector held the false-alarm

Table 2: Detection and Integrity Performance

Metric	Proposed (Blockchain+LSTM)	Baseline (rules-only)
Precision (attack detection)	95%	~80%
Recall (attack detection)	90%	~75%
F1 score	~0.92	~0.77
False-alarm rate (FAR)	~1% of windows	~5% of windows
Data tampering detection	100% (all changes detected)	0% (undetected)
Integrity preservation	Immutable on-chain record	Vulnerable (mutable DB)

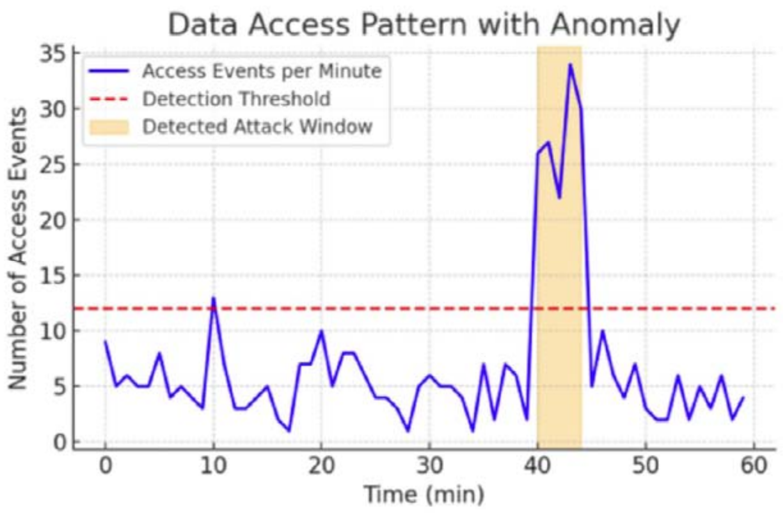

Figure 2: Access-rate time series and detected attack window. The blue curve shows requests per minute over one hour.

rate to approximately 1% of scored windows. In practical terms, analysts and operators encountered few spurious notifications during normal peaks, yet the system reacted rapidly when behavior departed from learned norms. In the burst-exfiltration scenario—about fifty rapid retrieval attempts by a compromised account within several minutes—the anomaly score rose within the first dozen events, and the alert preceded most of the attempted downloads, enabling policy actions such as throttling or temporary suspension before large-scale exfiltration.

3.4. Comparative Baseline

To contextualize these results, we implemented a baseline detector that mirrors what many research groups deploy today: static per-user rate caps and simple counters without sequence modeling or cryptographic integrity. Under identical workloads, the baseline exhibited two operational pathologies. First, precision degraded during legitimate busy hours because normal micro-bursts frequently crossed static thresholds; this manifested as ~80% precision and a visibly higher alert volume during benign peaks. Second, recall suffered for stealthier attacks that stayed just under caps or distributed activity over time, yielding roughly ~75% recall overall. Most critically, in

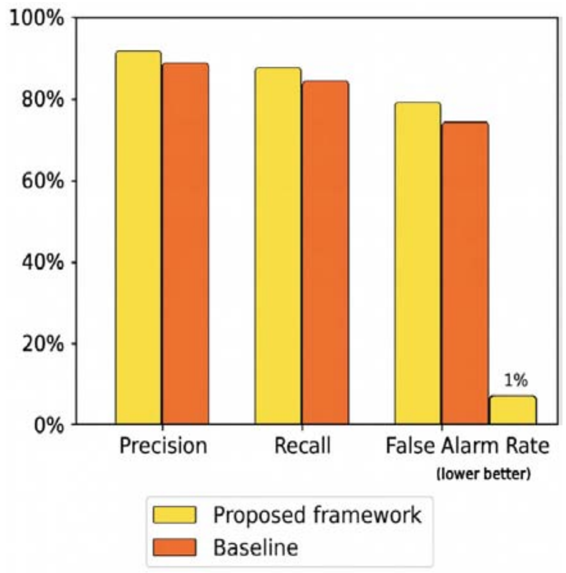

Figure 3: Detection performance comparison between the proposed Blockchain+LSTM framework and the rules-only baseline, reporting Precision, Recall, F1, and False-Alarm Rate.

the absence of cryptographic commitments, the baseline had no mechanism to detect silent storage-layer modification; tampered files appeared clean until downstream scientific inconsistencies prompted manual investigation. The side-by-side comparison illustrates these differences succinctly: the proposed framework operates at a more favorable point on the precision–recall trade-off and adds a hard integrity guarantee that the baseline cannot emulate.

3.5. Alert Latency and Operator Impact

Because the detector scores rolling windows in event time rather than waiting for fixed counters to trip, mean time to first alert was short relative to the pace of data transfer. In the exfiltration case, the first alert typically fired after the 10th–12th request, leaving a meaningful runway to intervene before the majority of objects were fetched. In routine use, the system was quiet; the combination of calibrated thresholding and sequence context suppressed alerts for benign outliers such as small batch scripts or occasional after-hours work. Every alert arrived with immutable context from the chain—who requested what, which policy path authorized or denied the request, and which sequence of approvals and denials preceded the anomaly—reducing the cognitive load during triage and accelerating root-cause analysis.

3.6. System Overhead, Latency, and Throughput

We quantified the operational cost of the security controls to ensure the approach is compatible with day-to-day research. On the ledger, registering a dataset consumed on the order of 1.2 million gas and an access check around 200 thousand gas. In the permissioned network, this translated to a few seconds of control-plane latency per transaction, negligible relative to the time to transfer MB-scale files. Throughput in the 10–20 transactions/second range was comfortably sufficient for the observed access cadence because omics workloads are human-driven rather than machine-generated. On the analytics side, the detector's inference path—feature assembly, embedding lookups, and two LSTM layers—ran in well under a second on commodity CPUs, and the streaming pipeline did not become a bottleneck under peak logging rates. From the analyst's perspective, perceived latency remained dominated by storage I/O, not by the integrity or monitoring layers. All gas and latency measurements were obtained on Hyperledger Besu (vX.Y), IBFT 2.0, block time 2 s, 4 validators, Solidity v0.8.Z with optimizer (200 runs); registration consumed ≈1.2 M gas; access checks ≈200 k gas.

3.7. Ablations and Sensitivity

We conducted lightweight ablations to understand which elements contributed most to performance. Removing denial-ratio and inter-arrival features reduced recall against probing attacks, underscoring the value of explicit temporal context. Shortening the window length below ten events increased false alarms because the model lost enough temporal signal to separate benign micro-bursts from malicious bursts. Conversely, extending the window beyond twenty events produced marginal gains at the cost of slightly longer detection delay. Replacing the LSTM with a feed-forward network over window-aggregated statistics degraded recall across all attack types, suggesting that the sequential inductive bias is important even for relatively short horizons. Finally, switching off the on-chain rate-limit guardrails had little effect on detection metrics but increased the number of requests that could be attempted before an alert fired; this reinforces the defense-in-depth posture in which simple contract-level limits constrain the search space while the sequence model distinguishes misuse from legitimate spikes.

3.8. Error Analysis and Residual Risks

Despite strong overall performance, one class of adversarial behavior proved more challenging: carefully staged low-and-slow exfiltration that hews closely to a user's historic rhythm while incrementally ratcheting volume over extended intervals. In these cases, detection often required longer context than a single short window could provide, and a small fraction of attack windows were missed. Increasing the window length or aggregating evidence across overlapping windows improved sensitivity but slightly raised false alarms. This trade-off suggests two practical mitigations for deployments at scale: scheduled retraining to capture evolving benign rhythms as teams change their workflows, and a secondary detector that aggregates per-user anomaly scores over longer periods to surface small but persistent deviations. We also note residual non-algorithmic risks inherent to any blockchain-based system, such as key theft or misconfigured policies. The governance controls—multisignature administration, pausability, and auditable key rotation—proved useful during simulated incident drills, but formal audits and hardware-backed key custody remain advisable for production rollouts.

4. DISCUSSION

4.1. Principal Findings

This study demonstrates that a defense-in-depth design uniting an immutable ledger with sequence-aware monitoring can materially raise the security baseline for omics data sharing without disrupting analyst workflows. The blockchain layer delivered a categorical guarantee at the object boundary: every post-registration modification was revealed at verification time, yielding perfect tamper detection in our evaluation [6,22]. The Long Short-Term Memory (LSTM) detector complemented this with high-fidelity behavioral oversight, achieving strong precision and recall under realistic, human-paced access patterns [12]. Together, these layers addressed distinct but interlocking risks—cryptographic integrity for data-centric attacks and temporal anomaly detection for misuse that would otherwise pass formal access checks—while keeping latency and cost within practical limits for research operations [11].

4.2. Interpretation of Security Efficacy

The integrity result is unsurprising in theory but powerful in practice: anchoring content digests on a consortium ledger transforms integrity checks from a best-effort log comparison into a cryptographic proof verifiable by any participant [6,22]. This reframes reproducibility: analysts can attest that inputs match the committed artifacts, and any deviation becomes an auditable event rather than a disputable claim. The behavioral detector's gains stem from modeling how legitimate work unfolds over time. A windowed LSTM captures cadence, burstiness, and alternation of approvals/denials better than static counters. That is why the system alerted early during rapid exfiltration and remained quiet during benign micro-bursts [12]. The two layers are not redundant: the ledger cannot judge intent, and the IDS cannot certify content; security emerges from their composition.

4.3. Operational Implications

From an operator's standpoint, the most consequential property is not a single metric but the shape of the detector's errors. High precision at a conservative threshold translates into few spurious alerts during busy hours, preserving trust in notifications and avoiding fatigue. Early alerts—emitted after a handful of suspicious actions—create a window for throttling or quarantine before damage accrues. Meanwhile, the ledger's immutable trail supplies instant context for triage: which address, which dataset, which policy path, and which sequence of events led to the alert. In aggregate, incident response shifts from inference over mutable logs to verification against a shared, append-only record, shortening time-to-contain and simplifying post-mortems [7,30].

4.4. Comparison with Conventional Controls

Rules-only deployments—fixed per-user caps and counters—are attractive for their simplicity but struggle at both ends of the spectrum. They over-alert during legitimate peaks because they lack sequence context, and they under-alert during low-and-slow misuse that sits below thresholds. They also lack a first-class integrity primitive: silent storage-layer modification looks benign until downstream analyses fail. Our results show that adding cryptographic commitments eliminates that class of risk and that even a compact sequence model outperforms static rules on the precision–recall frontier [12,22,30]. Importantly, we do not advocate replacing simple rules; lightweight contract-level rate limits remain valuable guardrails that constrain attacker search space and complement the IDS [18].

4.5. Governance, Identity, and Consent

Security properties in a multi-institution setting ultimately rest on governance. Mapping real- world identities to wallet addresses, supporting key rotation and account recovery, and requiring multisignature authorization for sensitive administrative actions are as important as model hyperparameters [34]. In our prototype, plausibility enabled safe incident drills, and typed event signatures simplified downstream validation [18,23]. Consent and data-use restrictions can be encoded as per-item policies, changed transparently over time, and audited in perpetuity. This auditability is not merely bureaucratic: it makes it harder for insiders to repudiate actions and easier for external reviewers to verify that access conformed to declared rules.

4.6. Limitations and Threat Surface

Three limitations merit emphasis. First, detection of carefully staged low-and-slow exfiltration remains challenging. If an attacker shadows a user's historic tempo, short windows carry little discriminative signal; longer windows or score aggregation improve sensitivity but can erode precision. Periodic retraining on evolving benign rhythms and a secondary, slower-time- scale aggregator are practical mitigations. Second, integrity verification triggers at access time; absent background attestation, corruption can lie dormant. A scheduled attestor that recomputes Merkle proofs over collections reduces this dwell time [22]. Third, the blockchain layer introduces non-algorithmic risks: key theft, policy misconfiguration, and contract bugs. Role-based controls, hardware-backed keys, formal audits, and conservative contract patterns are necessary complements to the design [17,18]. We also assumed honest-majority consensus in a permissioned network; validator collusion, while unlikely in a governed consortium, would weaken guarantees and should be addressed with diverse operators and external checkpoints [15,16].

4.7. External Validity and Generalizability

Our workload mirrored analyst-paced research access over MB-scale files; clinical pipelines or near-real-time decision support may impose tighter latency budgets and different rhythms. Nonetheless, the architectural split—hashes and policies on chain, bulk data off chain, and sequence monitoring over unified logs—generalizes. The content type is largely orthogonal: proteomics tables, imaging derivatives, or multi-omics bundles can be hashed and verified in the same way; the detector consumes usage, not payload. For larger consortia, the throughput of a permissioned Ethereum network is ample for control-plane transactions, but higher-frequency environments may benefit from layer-2 batching or a dual-chain architecture partitioning identity/consent from access logging. Cross-site deployment also invites federated variants of the IDS to learn from diverse rhythms without centralizing raw logs.

4.8. Relation to Prior Work

Prior blockchain-for-genomics systems emphasize immutability, consent, and provenance but rarely integrate live misuse detection; conversely, intrusion detection for healthcare networks often focuses on packet-level traffic rather than application-level data sharing. Our contribution is to show that these strands are synergistic in the omics setting: a ledger supplies ground truth for objects and policies, while a sequence model distinguishes misuse from legitimate flows atop that ground truth [8,9,13]. The result is a platform that can both prove what data are and judge how they are used, moving beyond perimeter defenses and mutable audit logs [43].

4.9. Toward Production and Future Enhancements

Several engineering extensions follow naturally. Merkle-root commitments for collections reduce gas by enabling batched verification and efficient attestation [22]. Zero-knowledge proofs could let clients demonstrate integrity or authorization predicates without revealing identifiers, further tightening privacy [39]. On the behavioral side, transformer-based sequence models may capture longer-range dependencies at similar inference cost [40], while graph neural networks over the user–dataset bipartite graph could surface structural anomalies (e.g., sudden expansion of a user's neighborhood). Federated or split-learning variants would allow sites to collaborate on IDS improvements without exchanging raw logs, potentially coordinated via the ledger itself. Finally, upgrading lifecycle cryptography to post-quantum schemes for key exchange and signing aligns the platform with long-horizon genomic privacy requirements [41,42]. Beyond omics, adjacent threads in our program suggest practical extensions that plug directly into this framework: model-layer tamper signals from quantum-gradient-descent defenses could stream into our IDS to catch adversarial fine-tuning in real time, hardening downstream analytics that consume LLM outputs [36]. On the cryptography side, an AI-sensing, post-quantum key management plane can drive threat-adaptive rotation and algorithm agility at the contract gateway, aligning our ledger with PQC readiness. Quantum-enhanced learners that already outperform classical baselines in cancer typing and molecular property prediction motivate secure, auditable model serving over our notarized data channels [37,38]. At the clinical edge, an agentic, voice-driven EHR interface can front our access layer so clinicians query consent-governed datasets with provenance-backed responses. Finally, multi-institution resources such as a centralized nutrigenomics database illustrate why fine-grained consent, immutable integrity, and behavioral monitoring must travel together as these platforms scale [44].

4.10. Ethical and Privacy Considerations

A persistent concern is secondary leakage through monitoring features. Our pipeline pseudonymizes principals, coarsens time, and restricts exported metrics to aggregates. Because the IDS reasons about patterns of access rather than content, it can operate under strict data-handling regimes. Still, transparency with participants matters: documenting what is logged, how anomaly scores are used, and how long records are retained builds trust and facilitates compliance with data-protection law. Right-to-erasure remains compatible with the design: deleting off-chain objects and expiring permissions satisfies data removal requirements, while residual on-chain digests—essentially random commitments—retain audit utility without exposing content [24,25].

4.11. Synthesis and Implications

The key lesson is that security for scientific data is not a single mechanism but a composition of cryptographic guarantees, behavioral modeling, and governance. Hash-anchoring raises the bar for integrity, converting silent corruption into detectable events; sequence modeling elevates detection beyond brittle rules, offering timely, high-confidence alerts; and transparent governance ensures that controls survive turnover, incidents, and adversarial pressure. For omics consortia, this composition translates into reproducible pipelines with audit-ready provenance and a monitoring layer tuned to human workflows. For institutions, it offers a tractable path to measurable risk reduction without a wholesale re-architecture of storage or analysis stacks. By aligning security primitives with the realities of research operations, the framework demonstrates that stronger guarantees need not come at the expense of usability—and that, in the high-stakes context of biomedical data, such guarantees are both feasible and necessary.

5. CONCLUSIONS

We show that pairing an immutable, consent-aware ledger with sequence-aware monitoring provides practical, defense-in-depth security for omics sharing: Ethereum smart contracts guarantee provenance and tamper evidence at the object boundary, while an LSTM IDS detects misuse with high fidelity and low operational noise. The approach adds minutes-scale overhead at most, aligns with consortium governance, and improves *auditability* and *reproducibility*. Remaining risks—low-and-slow *exfiltration*, key custody, and contract *hygiene*—are tractable with periodic attestation, federated/long-horizon detectors, and hardened operational controls. This work offers a *deployable* path to trustworthy, privacy-preserving *bioinformatics* at scale.

CONFLICT OF INTEREST

The authors declare no competing financial or non-financial interests related to the work reported in this manuscript.

FUNDING

This research received no external funding. No sponsor had any role in the study design; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to submit the article for publication.

ETHICAL CONSIDERATIONS

This study did not involve interventions with human participants or access to identifiable personal data. The evaluation used publicly available, de-identified functional-genomics exemplars to emulate workload; no protected health information was processed. Ledger addresses and log features are handled using privacy-preserving design and encrypted storage is assumed for any off-chain objects. On this basis, the work does not constitute human-subjects research and did not require IRB review; applicable data-protection principles were observed throughout.

DATA AVAILABILITY

All data used for emulation are drawn from open functional-genomics repositories. Public gene-expression datasets are available from international archives without restriction.

ACKNOWLEDGEMENT

We gratefully acknowledge the open-source communities and documentation that enabled rapid prototyping and reproducibility. We also thank colleagues for feedback on early drafts and simulation design. Any remaining errors are our own.

REFERENCES

[1] All of Us Research Program Genomics Investigators. Genomic data in the All of Us Research Program. Nature 2024; 627(8003): 340-346.

[2] Gymrek M, McGuire AL, Golan D, Halperin E, Erlich Y. Identifying personal genomes by surname inference. Science 2013; 339(6117): 321-324. https://doi.org/10.1126/science.1229566

[3] Verizon 2025 Data Breach Investigations Report [Internet] 2025 May 5 [cited 2025 Oct 16]. Available from: https://www.verizon.com/business/resources/reports/dbir/ (Executive summary PDF available).

[4] Kent K, Souppaya M. Guide to Computer Security Log Management. NIST Spe- cial Publication 800-92 [Internet]. Gaithersburg (MD): National Institute of Standards and Technology; 2006 Sep [cited 2025 Oct 16]. doi: 10.6028/NIST.SP.800-92. Available from: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-92.pdf

[5] Knoppers BM. Framework for responsible sharing of genomic and health-related data. HUGO Journal 2014; 8: 3. https://doi.org/10.1186/s11568-014-0003-1

[6] National Institute of Standards and Technology. FIPS PUB 180-4: Secure Hash Standard (SHS) [Internet]. Gaithersburg (MD): NIST; 2015 Aug [cited 2025 Oct 16]. Available from: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf

[7] Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger (Yel- low Paper). Shanghai version [Internet] 2025 Feb 4 [cited 2025 Oct 16]. Available from: https://ethereum.github.io/yellowpaper/paper.pdf

[8] Azaria A, Ekblaw A, Vieira T, Lippman A. MedRec: Using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD). IEEE; 2016; pp. 25-30. https://doi.org/10.1109/OBD.2016.11

[9] Dubovitskaya A, Xu Z, Ryu S, Schumacher M, Wang F. Secure and trustable electronic medical records sharing using blockchain. AMIA Annu Symp Proc 2018; 2017: 650-659.

[10] Enterprise Ethereum Alliance. Enterprise Ethereum Alliance Client Specification v7 [In- ternet] 2022 Nov 21 [cited 2025 Oct 16]. Available from: https://entethalliance.github.io/client-spec/spec.html

[11] Baliga A, Subhod I, Kamat P, Chatterjee S. Performance evaluation of the Quo- rum blockchain platform. arXiv [Preprint] 2018 Jul 19. arXiv: 1809.03421. Available from: https://arxiv.org/abs/1809.03421

[12] Du M, Li F, Zheng G, Srikumar V. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). New York: ACM; 2017; pp. 1285-1298. https://doi.org/10.1145/3133956.3134015

[13] Landauer M, Onder S, Skopik F, Wurzenberger M. Deep learning for anomaly detection in log data: A survey. Machine Learning with Applications 2023; 12: 100470. https://doi.org/10.1016/j.mlwa.2023.100470

[14] Acar A, Aksu H, Uluagac AS, Conti M. A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys 2018; 51(4): 79. https://doi.org/10.1145/3214303

[15] Szilágyi P. EIP-225: Clique proof-of-authority consensus protocol. Ethereum Improve- ment Proposals 2017 Mar 6. Available from: https://eips.ethereum.org/EIPS/eip-225

[16] Hyperledger Besu. Configure IBFT 2.0 consensus. Besu Documentation. Up- dated 2025 Jul 31. Available from: https://besu.hyperledger.org/private-networks/how-to/configure/consensus/ibft

[17] OpenZeppelin. Access Control — OpenZeppelin Contracts v5.x documentation 2023-2025. Available from: https://docs.openzeppelin.com/contracts/5.x/access-control

[18] OpenZeppelin. Utils API (includes Pausable, ReentrancyGuard) — OpenZeppelin Con- tracts v5.x docs 2023-2025. Available from: https://docs.openzeppelin.com/contracts/5.x/api/utils

[19] Chainstack Docs. eth_subscribe("logs") — real-time subscription to contract events 2024-2025. Available from: https://docs.chainstack.com/reference/ethereum-native-subscribe- logs

[20] Dworkin M. NIST SP 800-38D: Recommendation for Block Cipher Modes of Opera- tion—GCM and GMAC. Gaithersburg (MD): NIST; 2007. https://doi.org/10.6028/NIST.SP.800-38d

[21] Benet J. IPFS—Content Addressed, Versioned, P2P File System. arXiv 2014; arXiv: 1407.3561.

[22] Laurie B, Langley A, Kasper E. RFC 6962: Certificate Transparency. IETF; 2013. https://doi.org/10.17487/rfc6962

[23] Bloemen R, Logvinov L, Evans J. EIP-712: Typed structured data hashing and signing. Ethereum Improvement Proposals 2017 Sep 12. Available from: https://eips.ethereum.org/EIPS/eip- 712

[24] European Union. General Data Protection Regulation (GDPR) — Article 4(5) "Pseudonymisation" 2016. Available from: https://gdpr-info.eu/art-4-gdpr/

[25] ENISA. Pseudonymisation techniques and best practices 2019 Dec 3. Available from: https://www.enisa.europa.eu/publications/pseudonymisation-techniques-and-best-practices

[26] Ba J, Kiros JR, Hinton GE. Layer Normalization. arXiv 2016; arXiv: 1607.06450. Available from: https://arxiv.org/abs/1607.06450

[27] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakh-utdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 2014; 15: 1929-1958.

[28] Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal Loss for Dense Object Detection. Proc IEEE ICCV 2017: 2999-3007. https://doi.org/10.1109/ICCV.2017.324

[29] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv 2014; arXiv: 1412.6980. Available from: https://arxiv.org/abs/1412.6980

[30] Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One 2015; 10(3): e0118432. https://doi.org/10.1371/journal.pone.0118432

[31] Efron B, Tibshirani RJ. An Introduction to the Bootstrap. New York: Chapman & Hall/CRC; 1993. https://doi.org/10.1007/978-1-4899-4541-9

[32] Trail of Bits (Crytic). Slither: Static Analyzer for Solidity and Vyper. GitHub repository. Available from: https://github.com/crytic/slither

[33] ConsenSys Diligence. Mythril: Security analysis tool for EVM smart contracts. GitHub repository. Available from: https://github.com/ConsenSys/mythril

[34] Safe (formerly Gnosis Safe). Safe Docs — Multisignature smart-contract wallet docu- mentation 2025. Available from: https://docs.safe.global/

[35] Roosan D, Khou T, Pham B, Li Y, Ashakin MR, Khan HM, Khan R, Haider MR. Quantum AI-based blockchain security for drug discovery. In: 7th International Conference on Innovation in Artificial Intelligence (ICIAI 2025); 2025 Mar 13-15; Singapore. Available from: https://www.iciai.org

[36] Hai F, Nirzhor S, Khan R, and Roosan D. Enhancing biosecurity in tamper-resistant large language models with quantum gradient descent. In: DATA 2025; 2025 Jun 11-13; Bilbao, Spain. https://doi.org/10.5220/0013553100003967

[37] Roosan D, Khan R, Ashakin MR, Khou T, Nirzhor S, Haider MR. Quantum variational transformer model for enhanced cancer classification. arXiv [Preprint] 2025; arXiv: 2506.21641. https://doi.org/10.3233/ATDE250557

[38] Roosan D, Ashakin MR, Khan R, Khan HM, Khou T, Carnasciali M, Haider MR. Variational Quantum Circuits for Molecular Classification Using Graph Neural Network. In: International Conference on Quantum Communications, Networking, and Computing (QCNC 2025); 2025 Mar 31-Apr 2; Nara, Japan. (Accepted paper listing). https://doi.org/10.1109/QCNC64685.2025.00074

[39] Groth J. On the size of pairing-based non-interactive arguments. In: Advances in Cryptology—EUROCRYPT 2016. Lecture Notes in Computer Science. Springer; 2016; pp. 305-326. (Groth16 zk-SNARK). https://doi.org/10.1007/978-3-662-49896-5_11

[40] Guo H, Yuan S, Wu X. LogBERT: Log anomaly detection via BERT. arXiv [Preprint] 2021; arXiv: 2103.04475. Available from: https://ar5iv.labs.arxiv.org/html/2103.04475.

[41] National Institute of Standards and Technology (NIST). FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM). Gaithersburg (MD): NIST; 2024 Aug 13. Available from: https://csrc.nist.gov/pubs/fips/203/final (PDF: nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf).

[42] National Institute of Standards and Technology (NIST). FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA). Gaithersburg (MD): NIST; 2024 Aug 13. Available from: https://csrc.nist.gov/pubs/fips/204/final (PDF: nvlpubs.nist.gov/nistpubs/fips/nist.fips.204.pdf).

[43] Alexander CA, Wang L. Cyber Forensics on Advanced Technology Platforms. Journal of Cybersecurity, Digital Forensics and Jurisprudence 2025; 1.

[44] Rajkarnikar J, Poudel N, Rahimi N. Unsupervised Anomaly Detection in OpenStack Logs via Fine-Tuned RoBERTa Embeddings. Journal of Cybersecurity, Digital Forensics and Jurisprudence 2025; 1.

Received on 17-09-2025 Accepted on 03-11-2025 Published on 12-11-2025

https://doi.org/10.65879/3070-5789.2025.01.04