IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 9, September, 2026, pp. 2555-2569
Journal homepage: http://iieta.org/journals/jesa

Constant-Time Risk-Aware Authorization for Blockchain-Secured Electronic Health Record Systems

Hajar Mujeeb[1], Enas F. Aziz[2*], Rana Jumaa Surayh Al-Janabi[3]

[1] Department of Artificial Intelligence Engineering Technology, Technical Engineering College for Computer and Artificial Intelligence, Northern Technical University, Kirkuk 36001, Iraq
[2] Department of Cybersecurity Engineering Technologies, Technical Engineering College for Computer and Artificial Intelligence, Northern Technical University, Kirkuk 36001, Iraq
[3] College of Computer Science & Information Technology, University of Al-Qadisiyah, Al-Diwaniyah 58002, Iraq

Corresponding Author Email: enasfaek@ntu.edu.iq

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/jesa.590911

Received: 21 July 2026
Revised: 14 September 2026
Accepted: 24 September 2026
Available online: 30 September 2026

Keywords:
access control, blockchain, electronic health records, health data security, Hyperledger Fabric, risk-aware authorization

ABSTRACT

Two main challenges to the use of cloud-based electronic health record (EHR) systems lie in the fact that static access control policies cannot be changed depending on different request contexts, and standard integrity verification via hashing cannot differentiate benign changes from any potential malicious tampering. In this research work, risk-aware blockchain access control (RABAC) is proposed based on three-party policy-based access control, with four contextual risk factors taken into account: temporal, behavioral, environmental, and data sensitivity. Integrity metadata and authorization results are stored in Hyperledger Fabric v2.5, and AES-256-GCM-encrypted EHRs are stored off-chain with the help of IPFS Kubo 0.20. The prototype was tested for a documented synthetic access pattern based on 260 selected records from the chronic kidney disease (CKD) dataset. During five experiments, RABAC demonstrated an average risk screening accuracy of 99.38% ± 0.16%, reducing the amount of successful tampering and unauthorized access by 77.7% and 86.3%, respectively, compared to the baseline approach with cloud-only infrastructure. With an offered load of 150-300 transactions per second, the prototype showed an average server-side processing time of 33 ms without considering ordering and block formation delays. An ablation study was conducted to decrease the false-positive rate from 15.1% to 3.7%.

1. INTRODUCTION

Cloud computing is becoming popular among electronic health records (EHRs) because of its scalability and cost-efficient approach, which would be difficult to achieve otherwise for any regional provider [1]. The price for such convenience is that one has to assume that the data has not been modified by the provider without any cryptographic evidence [2]. The regulation has increased the standards–General Data Protection Regulation (GDPR) in Europe, Health Insurance Portability and Accountability Act (HIPAA) in America, and blockchain processing guidelines with auditability as one of the standards [3].

Permissioned ledgers do not fully address the issue because they allow an audit trail, which is impossible in a centralized cloud system [4], and hash trees anchored in ledgers allow effective checking of the integrity of cloud data [5]. That is, however, two problems. First is the authorization process, which will meet all requests that meet pre-set criteria, irrespective of when it is requested, what device is used or the pattern of the user's activity [6]. Secondly, the integrity check process involves digest comparison that does not carry any semantics: a change of a medication dose by a doctor and modification of a diagnosis code by a hacker will generate the same alert, as both of them will modify the data encrypted in the ciphertext. This is a semantic integrity gap defined as one that has a strong relationship with the static policy issue, and the meaning of integrity events depends on the information the authorization engine has. Throughout this paper, the term semantic integrity will be used to mean the interpretation of integrity events, and not necessarily verification of the contents of the record. Another limitation is related to the design since it introduces a challenge of compliance with GDPR and HIPAA privacy and data minimization requirements [7]: any possible design should be based on the storage of ciphertexts off-ledger and digest commitments on-ledger.

We thus propose a joint approach that will solve both problems: risk-aware blockchain access control (RABAC). The three main results are: (i) an authorization paradigm to enhance the current policy-based access control system with a contextual risk score including four dimensions: (ii) temporal, behavioral, environmental, and data-sensitive (Section 3.2), such that any request that satisfies all the policies may be denied if the context is suspicious. (iii) A 5-level architecture that integrates the Secure Hash Algorithm 256 (SHA-256)

commitment function, smart-contracts policy enforcement and off-chain storage in the InterPlanetary File System (IPFS) within the same risk context (Section 3.1). (iv) A comparison of our solution to clinical data, with different workloads for various threats, means, and resource usage for five different runs (Section 4).

The architecture includes a semantic interpretation mechanism (Section 3.1). The standalone contribution of our semantic interpretation mechanism to the reduction of false alarms is estimated by using component ablation in Section 4.6 (3.7% FPR with semantic interpretation, 15.1% without, $p < 0.05$). In the future, there will be additional follow-up evaluations of production alert streams (Section 5).

In its technical definition, it is a computation-experiment fusion: The risk-bounded authorization model, the formal guarantees of computational cost of the risk evaluation, the constant number of application-level operations per request, the explicitly defined model of adversaries, and the experimentally validated methodology of measurement, which involves calibrated workloads, the definition of ground truth by the generator, five end-to-end experimental runs, and t-distribution confidence intervals.

2. RELATED WORK

RABAC brings together four research streams: blockchain integrity verification, on-chain healthcare access control, risk-aware authorization and hybrid off-chain storage. Blockchain-integrity-related research has focused on compactness: four-branch trees and ranked sampling [8, 9] for cloud data integrity, keyword search on similar data sets [10], constant-sized proofs [11], on-chain EHR verification using an oracle [12] and group signatures stored via IPFS [13]. All the methods result in a binary answer which isn't further used for authorization.

Access control has been achieved for the healthcare chain by subject, controller and requester policies in Ethereum smart contracts [6], optimized Fabric chaincode [14], attribute policies with zero-knowledge proofs [15], hybrid hashing [16], privacy-preserving data sharing [17, 18], full on-chain authorization [19] and discretionary-mandatory rule orchestration [20] and surveys indicate the tendency toward a multi-layered RBAC-ABAC hybrid [21, 22]. They all have one thing in common: both the request tonight and in the morning are allowed by the same policy - there is no modulation done in the authorization decision based on the risk.

Although a newer concept, risk-aware authorization has also been utilized in other contexts prior to the blockchain, such as role and attribute policies and multi-layer risk scores [23], environment-aware deep learning access control [24], fuzzy AHP approach for cloud security posture analysis [25] and multilevel risk-aware authentication for mobile devices [26]. From on-chain perspective, Pu et al. have been able to put risk-conditioned authorization in the contract code with physicians' behavior scored in a time decay method [27]; Mohajan and Jahan have incorporated the zero-trust blockchain structure into the security awareness score [28]; Pandit and Chatterjee have obtained authorization signals from patients' physiological conditions by using IoMT devices [29]. None of the approaches are based on 4D context taxonomies to detect misuse on a small scale, and only a few approaches use the chain as an auxiliary tool, while the rest calculate the risk in a single centralized trust box or apply the risk to enhance the accessibility detection.

The fourth stream is related to the location of the bytes: hybrid approaches store the ciphertext in other places and anchor only digest data [7], while the fourth stream concerns the location of the bytes; raw records should not be stored on the ledger. The design space contains two IPFS-based designs for healthcare IoT [30], off-chain IPFS storage to overcome the throughput limit of Fabric [31], measurements of healthcare Fabric networks [32, 33], HealthBlock [34] and consortium exchange architectures [35]. In all of them, the risk context of a request that has stored the data or retrieved it is not known to the storage layer.

Table 1 summarizes a comparison of ten recent systems on four capabilities of an effective design: semantic integrity verification (SIV), multi-stakeholder policy-based access control (MS-PBAC), four-dimensional contextual risk assessment (4D-CRA), and hybrid off-chain storage (HOCS). The surveyed systems do not support all four, but RABAC integrates them into one pipeline, integrity metadata verification based on the risk level of the requester, and risk-based control over static policy predicates. Criteria: Native if this is the ability that the system uses on its decision path for each request; Partial if this is the ability that can be successfully performed using optional or external components; None if this is not mentioned in the source paper.

Table 1. Capability positioning of recent blockchain–cloud healthcare security frameworks against risk-aware blockchain access control (RABAC)

Framework	Year	SIV	MS-PBAC	4D-CRA	HOCS
Alashwal et al. [36]	2026	Partial	Partial	Partial	Partial
Yaqub et al. [6]	2025	None	Native	None	None
Tawfik et al. [14]	2025	Partial	Partial	None	Partial
Atlam and Yang [23]	2025	None	Partial	Partial*	None
Mohajan and Jahan [28]	2025	Partial	Partial	Partial	None
Sunitha and Saravana Kumar [21]	2025	None	Partial	Partial	Partial
Jayabalan and Jeyanthi [12]	2025	Native	None	None	Native
Taloba and Rayan [22]	2025	Partial	Partial	None	Partial
Pandit and Chatterjee [29]	2025	None	None	Partial**	None
Pu et al. [27]	2024	None	Partial	Partial	None
RABAC (proposed)	—	Native	Native	Native	Native

Note: Present, but not native (partial); absent (none). No full 4D taxonomy is used in hybrid RBAC–ABAC scoring. Physiological risk dimension applies only in the ICU setting. The SIV (semantic integrity verification), MS-PBAC (multi-stakeholder policy-based access control), 4D-CRA (four-dimensional contextual risk assessment), HOCS (hybrid off-chain storage), RBAC (role-based access control), and ABAC (attribute-based access control) are all terms used for access control. Access control is referred to as SIV (semantic integrity verification), MS-PBAC (multi-stakeholder policy-based access control), 4D-CRA (four-dimensional contextual risk assessment), HOCS (hybrid off-chain storage), RBAC (role-based access control), and ABAC (attribute-based access control). Testbed evaluations are reported in studies [6, 12, 14], prototypes on clinical datasets are reported in studies [21, 22], simulations are reported in studies [23, 27], and the setting of study [29] is not reported; RABAC is evaluated on a permissioned blockchain testbed using a public dataset (Section 4).

3. THE RISK-AWARE BLOCKCHAIN ACCESS CONTROL FRAMEWORK

RABAC arranges integrity, authorization, risk, and storage as sequential steps in one decision pipeline.

3.1 Architecture and integrity pipeline

The architecture is organized in 5 levels that cooperate (see Figure 1). The application interface layer is responsible for authentication and session management of stakeholders. The risk-aware access control layer is an overlaid layer on top of the traditional PBAC layer that performs risk analysis of all requests that comply with the policies. The blockchain verification layer makes use of Hyperledger Fabric to store the integrity metadata and results of authorization on the blockchain. Depending on the content digest of the records on the blockchain, the encrypted payload of the records is stored on the off-chain storage layer either in IPFS or cloud storage.

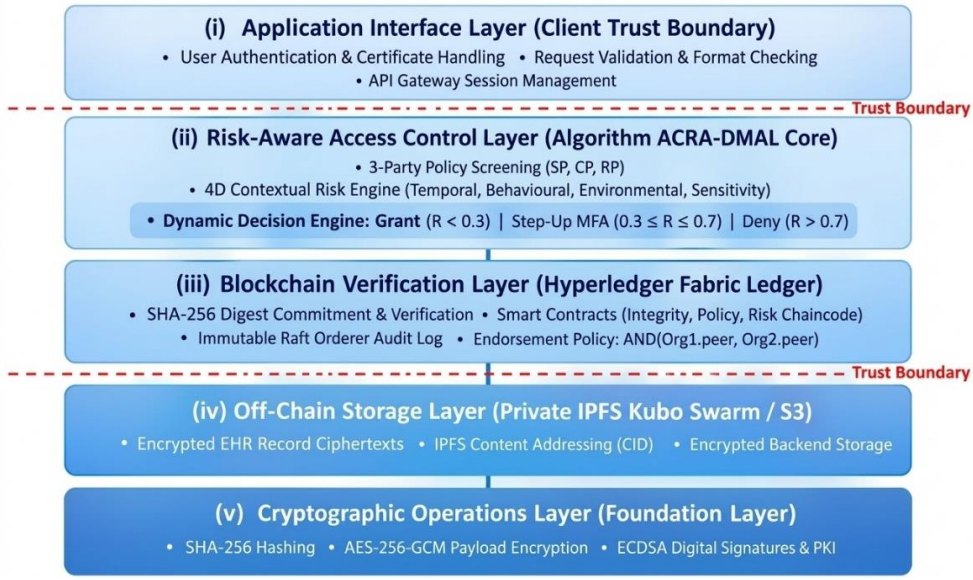


Figure 1. Five-layer architecture of the risk-aware blockchain access control (RABAC) framework (adaptive contextual risk assessment–decision-making algorithmic layer (ACRA-DMAL) core; see Section 3.2)

The commit-and-verify scheme is used for integrity verification. The hash of the commitment, H = SHA-256(C), will be stored in the metadata entry M = (φ, s_i, H, S_i, t_0), where the file ID, owner identity, hash value, storage pointer and commit timestamp will be stored, and the data itself will remain off-chain, and the metadata will be placed into the chain by a dedicated smart contract. In the verification phase, the ciphertext C' is retrieved, H' is computed as SHA-256(C') and the integrity violation event is triggered if it does not match. What makes our design proposal special is the way this event is handled: instead of notifying about the change, the verification layer refers the event to the risk engine for reevaluation in context before doing anything else; changes made at 9:00 am during the ward time and changes done at midnight from an unknown device get different risk scores. This semantic processing is a design decision; the effectiveness of its alert rate reduction is assessed in controlled ablation (Section 4.6), and finally in longitudinal validation with the production alert stream for future work (Section 5).

3.2 Risk-aware access control

The process of authorization follows two stages as in the adaptive contextual risk assessment–decision-making algorithmic layer (ACRA-DMAL) model (Figure 2). Incoming request $\rho = (s_j, d_i, t, \kappa)$, consisting of subject s_j, target object di, time stamp t and context attributes κ, is filtered out using three families of PBAC policies (subject, controller and requester), and if there is denial, the verdict is final and permanently registered in the ledger. Only requests that meet all three policies enter the remaining stage of the authorization process and the risk engine creates the four-dimensional feature vector for the request context.

All four dimensions are static and are consistently applied. They have been selected because they represent the four types of evidence that need to be evaluated to contextually assess the access request context when it occurs (temporal), how the requester behaves (behavioral), where it comes from (environmental), and to which data it targets (data sensitivity), which are used to evaluate the access request by a zero-trust and risk-aware access-control system per request [27, 37]. The temporal dimension RT is computed as the logit of the difference between request time and subject activity pattern, based on the hour-of-day, day-of-week, and frequency features extracted from a set of 144 fixed slots that form a circular buffer.

$$R_T = \sigma\left(\beta_0 + \beta_1 u_{\text{hr}} + \beta_2 u_{\text{dow}} + \beta_3 u_{\text{frq}}\right) \quad (1)$$

The logistic coefficients β_0, β_1, β_2, β_3 were estimated on the calibration partition using regularised logistic regression, with the regularisation setting selected on the validation partition. The calibration on the calibration partition produced $\beta_0 = -2.0$, $\beta_1 = 0.5$, $\beta_2 = 0.3$, and $\beta_3 = 0.8$ for the testbed deployment; the feature definitions and calibration procedure are provided in the replication package to enable reproducibility.

The behavioral dimension RB builds on the Mahalanobis distance dB between the current request's vector v_ρ (request rate, volume of data, object mix) and the subject's behavioral centroid μ_s in a fixed eight-dimensional feature space, with

covariance matrix Σ_s and exponentially-weighted-moving-average smoothing over a 100-observation sliding window:

$$d_B = \sqrt{(v_\rho - \mu_s)^T \Sigma_s^{-1} (v_\rho - \mu_s)} \quad (2)$$

The raw distance $d_B \geq 0$ is unbounded, so it is mapped to the unit interval by $R_B = 1 - \exp(-d_B)$. The covariance matrix Σ_s is estimated on the calibration partition only, regularised as $\Sigma_s + \lambda I$ with $\lambda = 10{-}6$ for numerical stability, and inverted through the Moore–Penrose pseudoinverse whenever it is ill-conditioned.

The environmental dimension R_E consolidates IP reputation, geolocation consistency and device-fingerprint checks into a single normalized score:

$$R_E = \frac{1}{3}(r_{IP} + r_{geo} + r_{dev}) \quad (3)$$

where, r_{IP}, r_{geo} and $r_{dev} \in [0, 1]$ denote the per-check risk contributions, while the data-sensitivity dimension R_S maps the object's classification {public, internal, confidential, restricted} onto an ordered value.

$$R_S = map(d_i) \in \{0.1, 0.3, 0.6, 0.9\} \quad (4)$$

Before aggregation, each attribute is normalized to the range [0, 1]: the aggregate score is the weighted average of the four dimension scores, each of which is the result of normalizing the selected dimension. Eqs. (1)–(4). The contextual risk dimensions are summarized in Table 2. The external signals are passed through an explicitly defined trust boundary; the IP reputation validation and geolocation of the signal sent by the oracle are trusted only if they are within a signed validity period, and the signal is cross-checked from different external sources so that no signal spoofing or outdated signal impacts the score (Section 3.4).

$$R = w_T R_T + w_B R_B + w_E R_E + w_S R_S \quad (5)$$

whose weights satisfy

$$w_T + w_B + w_E + w_S = 1, w_j \geq 0 \quad (6)$$

Table 2. Contextual risk dimensions: Source, range, normalization, history window and anti-tamper safeguard

Property	Temporal R_T	Behav. R_B	Environ. R_E	Sensit. R_S
Source	request time vs activity pattern	rate, volume, object mix	IP reput., geo-location, device fingerprint	object class label
Range	[0, 1]	[0, 1]	[0, 1]	{0.1, 0.3, 0.6, 0.9}
Normalization	sigmoid, Eq. (1)	1 − exp(−d), Eq. (2)	mean of 3 checks, Eq. (3)	ordered map, Eq. (4)
History window	144-slot buffer	100-obs. EWMA	per-request proof	static per object
Anti-tamper	commit time anchored	clipped EWMA; skip if R > 0.7	signed time-limited tokens with nonce	label in anchored entry M

The absence of any prior information favoring any given context dimension for general hospital deployment is consistent with the result of sensitivity analysis in Section 4.4, and needs to be independently tested on other cohorts. If any deployment-dependent priors exist, weights can be re-estimated with local data with no changes to the architecture - the reference implementation includes options for named weight profiles (w = (0.15, 0.50, 0.15, 0.20) for behavior-emphasized, w = (0.10, 0.20, 0.10, 0.60) for sensitivity-emphasized). The threshold of "below" T_{grant} = 0.3 and "above" T_{deny} = 0.7 is configurable per deployment and decisions are made based on those thresholds. The whole decision process is defined by the ACRA-DMAL formulation below (Figure 2).

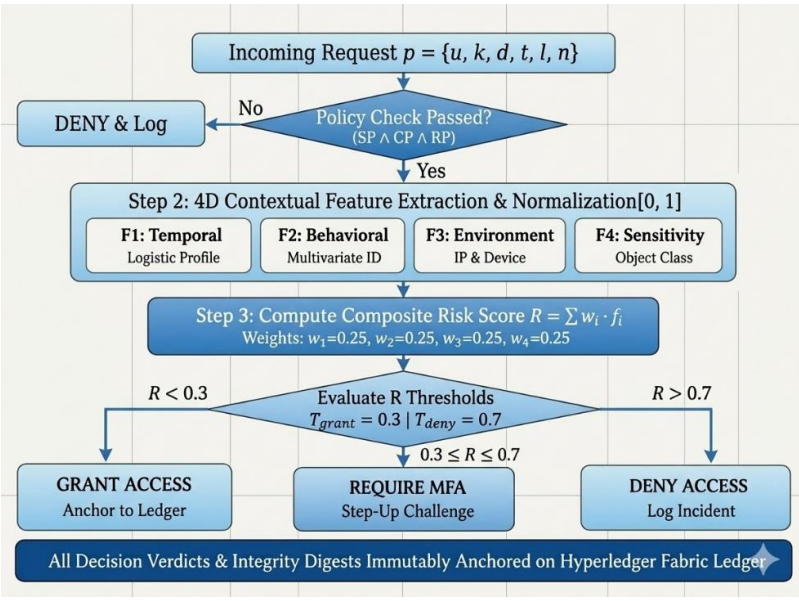


Figure 2. Decision flowchart of the adaptive contextual risk assessment–decision-making algorithmic layer (ACRA-DMAL) algorithm (grant below 0.3, step-up MFA 0.3–0.7, deny above 0.7)

After the blockchain certificate of the requester is verified, the request is filtered according to three policy families, and any denial is permanently recorded in the ledger; in the case when the request satisfies all three policies, its risk R is calculated according to the four normalized dimensions defined by Eq. (5). This evaluation and combination are done using Eqs. (1)–(4), and the three-tier rule is applied to grant, deny, or escalate a request to multi-factor authentication. Each decision is entered into the ledger, along with the integrity metadata, as specified in Section 3.3.

3.3 Evaluation-cost analysis

Latency considerations apply to authorization in healthcare. The following theorem provides an upper bound for the number of application-level operations performed during the risk evaluation phase and with the fixed structural assumptions of the prototype implemented.

Theorem 1. (bounded number of application-level operations in risk evaluation). For a specific access request ρ, the risk evaluation pipeline executes a bounded number of application-level operations for the implemented prototype, which has fixed structural assumptions such as the number of policy decision trees, feature vector dimension, size of the policy base and the number of keyed state access requests per policy decision tree node. The theorem applies to the number of application-level operations, and it's important to note that it doesn't mean that the wall-clock performance of the underlying storage engine is state-size independent.

Proof (sketch). It contains a limited number of application-level operations throughout the implementation for every access request: (i) ledger verification τ_1 – single world-state lookup and endorsement validation; (ii) policy evaluation τ_2 – traversal of a ROBDD of limited size (32 nodes); and (iii) contextual scoring τ_3 – fixed-dimension scoring with bounded-arity checks and a fixed number of feature operations. Hence, the number of application-level operations per request, $T(\rho) = \tau_1 + \tau_2 + \tau_3$, is bounded by a constant specific to the prototype. It is not assumed that any of the aforementioned LevelDB-based underlying operations take constant wall-clock time: network communication, endorsement, ledger commit path, and the underlying LevelDB lookup. The world-state size, hardware state, caching and system load are among the factors that may influence the underlying cost of the state database, in particular. Hence, the outcome is reported as a limited number of application-level operations in prototype set-up and not as a constant time end-to-end complexity. The complete proof is given in Appendix A.

Corollary 1. The average execution time of the risk evaluation stage in the deployed testbed ($\sigma = 0.4$ ms) is 2.3 ms. This is an empirical observation, whereas the in-memory decision-logic benchmark (4.6.2) is a pure in-memory benchmark, being independent of the first. The 2.3 ms time is a limit on the number of operations, and not on the proof of constant wall-clock time for the prototype setup. The 2.3 ms is state access and request-processing overhead from chaincode, whereas the 8.23 μs is the in-memory decision logic benchmarking in Python, which is for a different execution context.

The per-subject state is fairly light: the temporal profile takes up 2.3 KB, the behavioral model occupies 576 bytes, and the device list accounts for 160 bytes, bringing the total to approximately 3 KB, depending on the number U of active users at the time.

3.4 Adversary model and security guarantees

We assume that an adversary can sniff messages passing over the network, transmit messages to others, steal credentials from other peers, and compromise a subset of the peers, but is unable to find collisions in the SHA-256 function, forge an ECDSA signature without the corresponding private key, or calculate preimages of cryptographic hash functions. The following security properties are made about a particular fault model: the prototype has the following properties with regard to a particular fault model: crash faults only, so they hold for crash faults only.

G1 (**Data integrity**). The protocol stores a ciphertext C in the state and uses a digest, H = SHA-256(C), to anchor its storage. The only way that the substitution $C' \neq C$ will be unnoticed is if the adversary finds the second preimage of the digest anchored on the ledger, namely if SHA-256(C') = H. Considering our assumption on second-preimage resistance of SHA-256, it will be impractical to construct such a C'. G1 applies to ledger-anchored content: whenever the commit-and-verify procedure of Section 3.1 is applied, it guarantees that the client will detect if their ciphertext/IPFS payload has been altered, but does not guarantee that the client will detect if their ciphertext/IPFS payload has been replaced by content that is off-channel. The residual successful tampering cases observed in the experiments correspond to the simulated off-channel substitution scenarios described in Section 4.2.

G2 (**Authorization correctness**). As a result, if the three policy predicates $SP(\rho) \wedge CP(\rho) \wedge RP(\rho)$ are true for the request being processed, then a forgery of the policy outcome will require valid signatures from both organizations' peers, but no organization will have the ability to rewrite an authorization record alone. G2 is for policy enforcement when presented with valid credentials, meaning that when granting access against a valid credential, G2 says that no request will be granted if it does not conform to the policy conjunction; however, access will not be denied based on a compromised credential. The risk engine treats attempts involving credential theft or bypass as anomalies. The remaining unauthorized access cases observed in the simulation (3 per 1,000 requests) occurred when compromised credentials satisfied the policy predicates but the resulting requests were not screened successfully by the risk engine.

G3 (Audit immutability). The log-matching property of Raft ensures that any subsequent leader that receives an ordering entry at index n also receives r; the leader-completeness property ensures that no crash-free ordering node can fail to receive r as the ordering entry's latest commit block if a majority of the nodes are crash-free; all blocks contain a hash of the previous block's data, so any tampering with r will corrupt all subsequent blocks' hashes, thus making it easy to detect the tampering.

G4 (Risk freshness). The context of the request and the state of the ledger as known when the proposal is executed. As GetTxTimestamp() returns a client-supplied one in the channel transaction header, it is not an independent source of time. Signed oracle-attestation validity intervals, nonce-based replay prevention, and application-layer request expiry checks with synchronized peer clocks (via NTP) ensure the freshness of the risk assessment. It is planned to have Fabric-level time-windowed timestamping as a future implementation hardening step. Oracle attestations include a signed validity interval (time-to-live) that expires after which the attestation becomes invalid, and include a unique nonce, stored in the blockchain,

and therefore any replayed or outdated attestations are rejected prior to scoring. The other types of possible failure are handled via extra mechanisms: MVCC read conflicts protection guards from concurrent modifications of the integrity contract, multi-sig attestation verification guarantees the integrity of the external data that goes into the risk contract, ordering node majority voting protects from ordering node failures, and triple redundancy and cloud backup guard from IPFS node failures.

Gradual and low-level behavioral poisoning (SLBP). Slow and low behavioural poisoning pretends to make a very small, but concerted set of behaviour changes in order to slowly alter a user's profile. To reduce the effect of big changes per period, a clipped EWMA update:

$$\mu_{t+1} = \mu_t + \gamma \cdot \mathrm{sgn}(\Delta_t) \cdot \min\left(|\Delta_t|, \delta_{\mathrm{cap}}\right) \quad (7)$$

where, $\Delta_t = x_t - \mu_t$, $\gamma = 0.05$, and $\delta_{\mathrm{cap}} = 1.2\sigma$. This constraint limits the effect of the update of one user on the assumed model. It is effective as shown in the bounded-drift simulation in Section 4.2; the testing of its effectiveness in longitudinal testing with adaptive SLBP campaigns needs to be investigated.

3.5 Implementation and deployment setup

The prototype is based on five virtual machines (Ubuntu 22.04 Long-Term Support (LTS), 4 vCPUs and 8 GB RAM each, gigabit LAN) running Hyperledger Fabric v2.5 (LTS release line), IPFS Kubo 0.20, and Go as the chaincode language. Three smart contracts are used to implement the pipeline of Section 3.1: (i) an integrity contract used to store the metadata entry M and implement commit-and-verify; (ii) a policy contract compiling the three PBAC families (30 rules in the testbed policy base, Section 4.1) into the ROBDD of Section 3.3; and (iii) a risk contract used to evaluate the four-dimensional score of Eqs. (1)–(5) and recording the result. No clinical payload is stored in the ledger, and each decision record contains: the record key and six fields written by one RecordAccessDecision transaction, including the subject ID, object ID, risk score, decision, RFC 3339 timestamp and IPFS CID. The network contains two organisations and each transaction that changes the state should be signed by AND (Org1.peer, Org2.peer) (G2, Section 3.4). Two of five machines are running the peer nodes (one peer and one CA node per organisation) and the three are running the Raft ordering-service nodes, which are the default nodes for the Fabric 2.5 LTS release line, with a default block-cutting timeout of 2 s and a maximum of 500 messages per block. The peer nodes' clocks are loosely synchronized to each other using NTP, and the expiration checks of requests are done at the application level (Section 3.4, G4). The communication between Fabric and IPFS is of course also hybridized as in Section 3.1: encrypted by AES-256-GCM payloads are uploaded to IPFS, the CID is anchored on-chain, and the verification procedure downloads the ciphertext by CID, computes the SHA-256 digest. The setup is complete with MVCC read conflict detection, multi-signature verification of the oracle attestations and triple redundancy of IPFS with cloud backup (Section 3.4). The risk engine implements the clipped-EWMA update of Eq. (7) with $\gamma = 0.05$ and a per-feature clipping bound $\delta_{\mathrm{cap}} = 0.1$ (approximately 1.2σ for the normalised behavioural features), and completely ignores behavioural-profile updates for requests with a score of $R > 0.7$. The entire chaincode, network-topology files and workload-generator configuration are provided, as described in the Data Availability statement.

4. EXPERIMENTAL STUDY

4.1 Dataset and testbed configuration

The deployment is the same as described in Section 3.5: 5 VMs, Hyperledger Fabric 2.5, IPFS Kubo 0.20 and Go chaincode.

We use the chronic kidney disease (CKD) dataset from the UCI Machine Learning Repository [38] (250 records, 150 NotCKD), which was collected at Enam Medical College, Savar, Dhaka, Bangladesh, for experiments. After filtering out 140 records with more than a fifth or 20% of missing attributes, 260 records were used from the dataset to ensure that in each record, more than a fifth or 20% of the attributes were missing. Categorical attributes were filled with the mode and numeric attributes were filled with median, without any data leakage, on the calibration split set. Categorical attributes were encoded and numeric attributes were min-max normalized to [0, 1]. The statistics were computed on the calibration split and normalization and imputation were carried out based on these statistics. Each raw record was encapsulated in a clinical document with synthetic attachments (sizes' generation procedure, uniformly distributed in range of 50 KB to 5 MB, without any patient data in them); stored objects range in size from 50 KB to 5 MB, otherwise, they would not have been able to execute the pipeline of the stored records. The distributed feature table is a set of clinical and laboratory characteristics that does not include any direct identifiers such as name, address, telephone number or national identifier. The data is licensed with CC BY 4.0 [38]. As the present study performed secondary analysis of publicly available datasets and did not involve any new participant recruitment and data collection, no human-subject data were collected here. The replication package (see Data Availability) contains the record-selection procedure and the list of record IDs so that filtering can be ensured to be replicable.

The access-event workload consisted of subsets of calibration, validation, and test workloads that totaled 60%, 20%, and 20%, respectively. Such splitting is done group-aware both at the synthetic requester and at the record levels, i.e., all access events related to the same requester or record ID were assigned to one single subset so as to avoid requester or record leakage between calibration, validation and test (Table 3). Stratification according to the class label (CKD/NotCKD) was kept in these constraints and uniformity of stratification was checked with Kolmogorov–Smirnov tests at the 0.05 significance level. All parameters (weights and thresholds) were chosen on the validation split, and the test split was used only once for each set of parameters to prevent overfitting of test split on the headline metrics. There were 30 rules (12 subject, 8 controller and 10 requester rules) in the policy base.

The following checks were done before each run: Zero-overlap constraints. The distribution of the patients in the CKD/NotCKD class was maintained as close as possible to the group constraints through stratification. All the risk-model parameters were selected based on calibration and validation data only and the test partition was used only at the final configuration reporting stage for synthetic generation of workload and threat scenarios.

4.2 Workload and threat-scenario synthesis

As tabular data contained information about specific patients and not about accesses, it was also necessary to develop a workload model around which the other measures were calculated. Four types of requesters (physician, nurse, administrator and researcher) were instantiated, with each having a corresponding activity template (schedule of working days and hours, average request rate, mix of sensitivities, session volume). Requests were generated through the Markov-modulated Poisson process so that each behaviourally-historical requester is included for the risk engine to learn from. The partition composition and leakage control are summarized in Table 3.

Table 3. Partition composition and leakage control of the access-event workload

Partition	Records	Requesters	Events	Malicious (%)	Record-ID Overlap	Requester-ID Overlap
Calibration	156	300	60,000	12.0	0	0
Validation	52	100	20,000	12.0	0	0
Test	52	100	20,000	12.0	0	0

Note: Records: a unique synthetic identity for the retained records; requesters: a unique synthetic identity for the requesters; event counts are per workload seed. Overlapping columns are those that contain identifiers that are common to another partition, and they are counted where they do overlap; the zero overlap was confirmed during the run by programming.

Four types of malicious activities that were introduced in the workload: (i) tamper attempt – change the ciphertext that is being committed; (ii) unauthorized accesses – policy violating accesses (inter-departmental access and bulk export) that go directly through the policy enforcement path, credential-compromised accesses: stolen credentials satisfy the policy predicates, the responsibility for detection is assumed by the risk engine; (iii) policy compliant but risky accesses (e.g. out of schedule access by authorized requester) – should be flagged by the risk engine; (iv) behavioral drift – modeled as a 1.5σ perturbation for each session. A special simulation analysis dedicated to CLIPPED-EWMA protection against SLBP is carried out, which is mentioned later in this section.

Risk engine weights and decision thresholds were not considered when generating attacks. Threat prior data was taken from the sectoral incident reports [39, 40]. Malicious activities had a fixed value of 12% and were used in all experiments as an experimental hypothesis. The ground truth label (not the risk engine) generated a label for every request, and this was the only available ground truth to use to calculate accuracy, false positives and false negatives. For each experiment, we ran 20,000 access requests, with 500 concurrent requesters, for five random seed workloads.

Metrics definition. If the request is malicious, as the generator ground truth identifies it, it is a positive (also known as a tamper attempt); if it is an access policy violation, a credential compromise, a context violation (that is, it is authorized by the access policy, but the context does not support it), or a behavioral drift, it is a positive; if it is an authorized access policy and the context is supported, then it is a negative (legitimate request). The following is a definition of the terms true positive (TP), false positive (FP), false negative (FN), and true negative (TN): TP is a malicious request that has been denied/challenged. FN is a malicious request that has been granted. FP is a legitimate request that has been denied/challenged. TN is a legitimate request that has been granted. In the case of the three-outcome risk engine decision rule (Section 3.2), we refer to the accuracy as risk-screening accuracy, since the challenge issued against a malicious request is considered as detection at the screening phase, while the challenge issued against a legitimate request is regarded as false alarm, and the challenges and outright denial verdicts are summarized and reported as TPs/FPs in the metrics report, as future reporting will be carried out separately. The false-positive rate is FPR = FP/(FP + TN) and the false-negative rate is FNR = FN/(FN + TP). Successful tampering refers to an instance where a modified ciphertext has been accepted without any integrity warning; on the other hand, unauthorized access implies that a malicious access request was granted because, in cases where the policy enforcement is carried out successfully (G2), requests that violate the policy are denied. Therefore, for RABAC, the unauthorized access instances are the ones where the policy predicate was fulfilled, and the risk was not screened, while in the baselines there is also acceptance of the requests that violate the policy. The two measures are measured on a per 1,000 requests basis: accuracy = $0.88 \times (1 - \text{FPR}) + 0.12 \times (1 - \text{FNR})$; All the accuracy data reported in Tables 4 and 5 satisfy this equation. Protective simulation of SLBP. While the reference algorithm was applied, two poisoning attacks were applied to the safeguarded EWMA algorithm given in Eq. (7) (supplementary materials). In a direct poisoning case, the vector $v = (20, 50, 15)$ was provided for the behavioral profile $\mu = (4, 9, 2)$—the projections of the 8D behavioral space of Eq. (2) on three features, which is illustrated—having the raw deviation of (16, 41, 13), the Mahalanobis distance ≈ 45.9; the request was given the score of $R = 0.85 > 0.7$, thus the profile remained unchanged (drift 0). If this were not for the safeguard mechanism, then the profile shift resulting from an accepted EWMA update would be (0.80, 2.05, 0.65). If the request is not above the escalation threshold, in a stealthy poisoning case, the clipping bound $\delta_{cap} = 0.1$ limits the per-feature drift to at most $\gamma \cdot \delta_{cap} = 0.005$ per accepted update; shifting any profile feature by 0.5 therefore requires at least 100 consecutive accepted poisoned updates (200 updates for a shift of 1.0). They must each be below the escalation level, one by one. This way, one session's poison attack is now a long-term one, and the attack is serious, detectable by an immutable attack decision log as described in Section 3.4, G3. Future work includes a longitudinal study of adaptive SLBP attacks on our audit data (Section 5).

The results will be statistically treated and compared with the baseline.

4.3 Statistical treatment and baseline comparison

Five-fold cross-validation was used only for the calibration and validation sets. System-level statistics (Table 4) were derived from five test partition runs for each of which a different workload seed was used, and in which 20,000 requests were applied per partition (5 × 20,000 for each configuration). The means, standard deviations and 95% confidence interval estimates (t-distribution, four degrees of

freedom) for RABAC and two hardware-similar baselines: cloud-based with no blockchain, using a static PBAC model, and centralized database (DB) system with application-level access control, are reported in Table 4. The difference between the cloud-only baseline and the complete RABAC stack is an ablation study of static PBAC and the full RABAC stack, because the same static PBAC policies are used without the contextual risk layer. Both baseline comparisons to paired t-tests were statistically significant ($p < 0.001$; Figure 3). As for the baseline without cloud, the number of tampering events dropped from 18.0 to 4.0 per 1,000 requests and from 22.0 to 3.0 per 1,000 accesses (without rounding), which is a 77.7% and 86.3% reduction, respectively (see unrounded numbers in Table 4); the rest of the tampering and accesses were due to off-channel substitution and credential compromise (Section 4.2). There was an experimental arm for Fabric-level deployment for the RBAC/ABAC that was not added for two reasons. First is the attribution: this arm would be similar to RABAC in that it contains the risk layer and the decision log that are being anchored at the same time, but it would be different because the cloud-only and centralized-DB baselines would each isolate one of those. Second, metric definability: the metrics are defined based on a per-request risk score and an anchored decision log, which is not generated by the classical RBAC and ABAC framework. Instead, it's the new design of those frameworks that's needed to assess them. Section 4.6, however, reports a simulation-level benchmarking of the decision logic of RBAC, ABAC, PBAC, and RABAC in a common runtime environment (Section 4.6.2) for policy evaluation and the presence of contextual violations, but not for the equivalence of the testbed. The comparison is asymptotic for the latency measure. Theorem 1 proves that the complexity of RABAC evaluation in terms of the number of application-level operations is the same as that of deterministic rule matching.

Table 4. Security performance over five independent runs

Metric	Cloud-Only	Central-DB	RABAC (prop.)	p vs. CO	p vs. DB
Risk-screening accuracy (%)	96.10 ± 0.29	97.72 ± 0.24	99.38 ± 0.16 [99.18, 99.58]	$p < 0.001$	$p < 0.001$
Unauthorized access (per 1,000)	22.00 ± 1.58	14.00 ± 1.58	3.00 ± 0.71 [2.12, 3.88]	$p < 0.001$	$p < 0.001$
Successful tampering (per 1,000)	18.00 ± 1.58	10.00 ± 1.58	4.00 ± 0.71 [3.12, 4.88]	$p < 0.001$	$p < 0.001$
Mean detection time (min)	18.32 ± 1.34	12.22 ± 1.09	6.00 ± 0.38 [5.53, 6.47]	$p < 0.001$	$p < 0.001$
False-negative rate (%)	4.50 ± 0.41	2.76 ± 0.36	0.80 ± 0.16 [0.60, 1.00]	$p < 0.001$	$p < 0.001$
False-positive rate (%)	3.82 ± 0.29	2.22 ± 0.29	0.60 ± 0.16 [0.40, 0.80]	$p < 0.001$	$p < 0.001$

Note: Mean ± SD for five independent trials of 20,000 requests each; 95% CI in brackets; p values obtained by applying paired t-test between RABAC and each individual baseline system. CO: cloud-only baseline; DB: centralized database baseline.

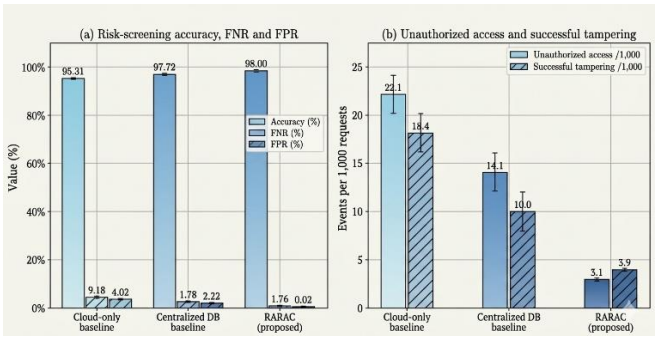


Figure 3. Security performance of risk-aware blockchain access control (RABAC) against the two same-testbed baselines; error bars denote 95% confidence intervals over five runs

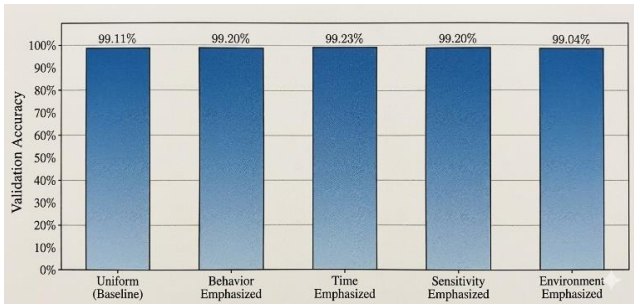


Figure 4. Validation risk-screening accuracy for the five risk-weight configurations in Table 5. The uniform-weight configuration achieved the highest validation accuracy (99.35%), while the maximum decrease across alternative profiles was 0.31 percentage points

4.4 Parameter sensitivity

Weights and threshold values were chosen based solely on the validation partition. The validation risk-screening accuracy for the five weight configurations is illustrated in Figure 4. The sensitivity analysis is described in Table 5 for five weight combinations and two pairs of thresholds, with the nominal being (0.3, 0.7), and the results in the table are from the validation partition and not from the testing partition (no confusion). The parameters chosen were fixed before the test partition was seen, and the test accuracy of the model (99.38%, see Table 4) was only calculated after the parameters were chosen. The worst-case drop in accuracy because of re-weighting is 0.31%, suggesting that the classifier is not on a lucky local maximum. The (0.3, 0.7) version is the best trade-off between false positives and false negatives.

4.5 Throughput, latency and scaling

A synthetic load of 10,000 access transactions was created from a set of 260 records, and the transaction throughput was measured. With a steady-state load of 150–300 TPS sustained by 500 concurrent client sessions, the average processing time on the server side is 33 ms (p50 = 29 ms, p95 = 48 ms, p99 = 63 ms), of which 2.3 ms is spent on the risk evaluation portion, and the remainder is spent on endorsement and peer-side transaction processing. Does not include client-side queuing and orderer block creation delay; this is a measure of endorsement, risk evaluation and peer-side transaction processing time. Thus, it is not to be considered as end-to-end transaction confirmation latency. Orderer queuing and block creation delay is also captured in the client-side end-to-end latency. The creation of a block will start once either condition is met: a batch creation timeout of 2 seconds, or a maximum

batch size of 500 messages (Section 3.5). The observed observations were 312–500 transactions. With the provided 150-300 TPS range, this delay for block creation would be estimated to be approximately 1.67–2.00 seconds before any network transfer, peer validation and client event delivery delay. Therefore, it is expected that the end-to-end client latency will be greater than the reported server-side processing time. The distribution of the measured sustained throughput and the measured server processing time distribution is summarized in Figure 5. The measured operating point of the deployed RABAC prototype is summarized in Table 6.

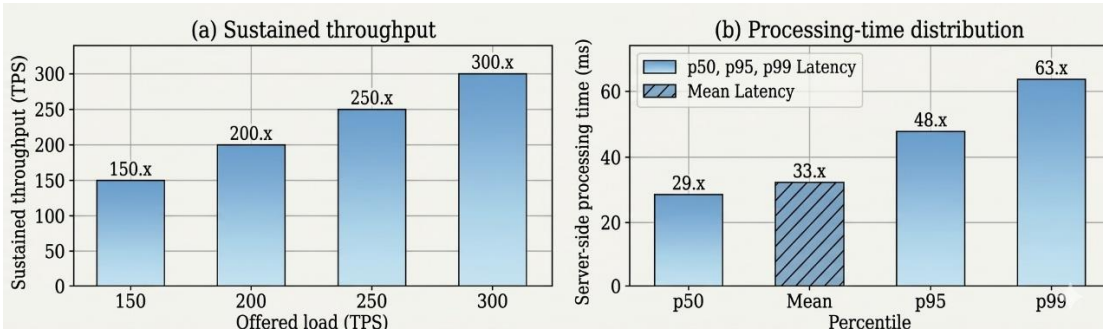


Figure 5. Measured sustained throughput and server-side processing-time distribution of the risk-aware Blockchain Access Control (RABAC) prototype under the 10,000-transaction workload at 150–300 TPS offered load

Table 5. Sensitivity analysis for risk-weight and threshold configurations

Configuration	Parameters	Acc. (%)	FPR (%)	FNR (%)
Uniform weights (baseline)	w = (0.25, 0.25, 0.25, 0.25)	99.35	0.63	0.82
Behavior-emphasized	w = (0.20, 0.40, 0.20, 0.20)	99.29	0.68	0.91
Time-emphasized	w = (0.40, 0.20, 0.20, 0.20)	99.25	0.72	0.96
Sensitivity-emphasized	w = (0.20, 0.20, 0.20, 0.40)	99.20	0.77	0.99
Environment-emphasized	w = (0.20, 0.20, 0.40, 0.20)	99.04	0.94	1.08
Grant threshold loosened	T_grant = 0.4, T_deny = 0.7	99.51	0.35	1.55
Grant threshold tightened	T_grant = 0.2, T_deny = 0.7	98.49	1.65	0.52

Note: All entries represent partition-validation results. FPR: False positive rate; FNR: False negative rate. Deny threshold entry is left out since, due to merged challenge/deny counting as per Section 4.2, Tdeny will have no effect on TP/FP/FN/TN statistics and will be metric-neutral.

Table 6. Measured testbed operating point of the deployed risk-aware blockchain access control (RABAC) prototype (Section 4.5)

VMs/Nodes	Users	Throughput	Server-side Processing Time	Theoretical End-to-end Latency Projection
5 (2 peers, 3 ord.)	500	150–300	33 ms mean (p50 = 29, p95 = 48, p99 = 63 ms)	≈1.7–2.1 s (33 ms + 1.67–2.00 s block-formation delay)

Note. The values reported correspond to the measured operational setting of the prototype implementation on the Section 4.5 testbed (five-node testbed: two organisation peers and three Raft ordering nodes, 500 concurrent users, offered load 150–300 TPS). The server-side processing time comprises endorsement, risk assessment and peer-side transaction processing time, excluding client-side queuing and orderer block creation time (Section 4.5). TPS: transactions per second. The theoretical projection of end-to-end latency is an analytical bound and does not involve any measurements; it represents the sum of the measured server-side processing time and the theoretical block creation time at the offered load (1.67–2.00 seconds, Section 4.5), excluding network latency and client event delivery time.

Table 7. Timing scope of the reported performance metrics

Metric	Included Phases	Excluded Phases	Reported Value
Risk-evaluation stage	Context-feature retrieval, scoring, decision generation, chaincode-side state access	Client submission, ordering, block cutting, peer commit-event delivery	2.3 ms ± 0.4 ms
Decision-logic simulation	In-memory Python implementation of policy and risk decision logic	Fabric, IPFS, state database, network, endorsement, ordering, commit	8.23 µs ± 0.26 µs
Server-side processing time	Endorsement processing, risk evaluation, peer-side transaction preparation and validation handling	Client-side queueing, orderer queueing, block-cut waiting	33 ms mean (p50 = 29, p95 = 48, p99 = 63 ms)
Client-observed end-to-end latency	Client submission to commit-event receipt	None	Not measured in the present study

Note. The measured rows quantify different execution scopes and are not directly comparable; see Sections 3.3 and 4.5. Percentiles are over the five 10,000-transaction workload runs.

The scope of each reported performance metric, including the processing stages measured and those excluded, is summarized in Table 7. The measured amount of state per active user is ≈3 KB, and for reproducing an instance on 500 users, the total state of the risk engine is ≈150 MB for 50,000 users (≈300 MB for 100,000 users). This is a theoretical estimation of state size extrapolated per user, not considering the overhead of replication or runtime and not measured as

node consumption. The scaling of the risk engine is tied to the peers' scaling, and not to the ledger's scaling separately: a linear increase of world state size with the number of active users, and a linear increase of evaluation capacity with the number of peers. The operating point measured is shown in Table 6. There are no cluster-wide projections: analytical projections were not possible based on measurements made with the five-node testbed. For large deployments, latency growth should be queueing-related, as evaluation does a fixed number of application-level operations per request (Theorem 1). The storage requirements and maximum power usage for the prototype are shown in Table 8. EHR – electronic health record; AES-256-GCM – Advanced Encryption Standard with 256-bit key and GCM mode; CID – Content Identifier. Data is provided by the Section 4.5 testbed experiments. The CPU and memory data are collected through the Docker container statistics API (cgroup accounting) at 5-s intervals during the steady-state period of each 20,000-request test with the 150–300 TPS load and 500 concurrent clients; CPU usage is reported as the percentage of total VM CPU resources (4 vCPUs), while the memory numbers refer to peak resident size. The ≈150 MB estimate for 50,000 users is an estimated theoretical extrapolation of the state size based on the individual user data, not an observed node consumption; the 500 concurrent user testbed needs ≈1.5 MB.

4.6 Component ablation, policy-model benchmark and SOTA comparison

4.6.1 Component ablation: semantic integrity verification

In order to measure the impact of SIV in an experimental way, the component ablation was repeated between two variants: RABAC-Full, which evaluates the integrity events within the context of the trigger request and only issues an alarm if the accumulated risk exceeds the threshold ($R \geq 0.30$); RABAC-No-SIV, which issues a raw alarm for every integrity event, without context evaluation. If the context of the integrity event is evaluated to have a risk greater than the deny threshold, then the write is rolled back (the new CID is unanchored, and the old CID is the valid pointer). The differences between the FPR of RABAC-Full and RABAC-No-SIV manifest only in the presence of SIV, because both make identical policy evaluations and risk assessment for all other types of requests. There were semantic anomalies in the workload: structurally and policy-compliant requests that, however, contain semantic anomalies (e.g., an off-hour change by a clinician from a different department of a restricted record of a pediatrics psychiatric service). Five runs were made with 5 different seeds. Absolute FPR values of the SIV are not directly comparable to the FPR in Table 4 because the workload has a number of modifications to test the SIV component.

All authorized changes to the ciphertext cause a false alarm, resulting in the FPR rising from 3.66% ± 0.14% to 15.06% ± 0.26% when SIV is not present, as shown in Table 9 below. The difference between the two FPRs is statistically significant ($p < 0.05$) using a paired t-test analysis. However, even when using a No-SIV variant, it is able to detect 98.60% ± 0.42% of all tampering attempts, which means that the FPR reduction is not a trade-off for not detecting tampering attempts.

Table 8. Storage allocation and peak-load resource consumption of the deployed prototype

Storage Tier/Component	Data Type	Size/Utilisation	Access Profile
On-chain (Hyperledger Fabric ledger)	SHA-256 digest, integrity metadata, authorization verdict	32-byte digest within metadata record M	Commit–verify anchored; keyed lookup
Off-chain (IPFS Kubo 0.20)	Encrypted EHR payloads (AES-256-GCM)	50 KB–5 MB per stored object	Content-addressed by digest (CID)
Per-subject risk-engine state	Temporal profile, behavioral model, device list	≈3 KB per subject (2.3 KB + 576 B + 160 B)	Aggregate O(U); ≈1.5 MB at the tested 500 users; ≈150 MB at 50,000 users
Risk engine (peak load)	Four-dimensional scoring service	≈25% CPU, ≈256 MB RAM	2.3 ms mean per-request evaluation
Fabric peer (peak load)	Endorsement and ledger commitment	≈22% CPU, ≈768 MB RAM	Measured during the stated workload; not a general capacity claim

Table 9. Component ablation with and without semantic integrity verification (SIV)

Run (Seed)	FPR – RABAC-Full (%)	FPR – RABAC-No-SIV (%)	Tampering Detection – RABAC-Full (%)
1 (101)	3.87	15.40	98.25
2 (202)	3.51	14.85	98.62
3 (303)	3.60	14.76	99.12
4 (404)	3.59	15.10	98.88
5 (505)	3.70	15.17	98.12
Mean ± SD	3.66 ± 0.14	15.06 ± 0.26	98.60 ± 0.42

Note: FPR = FP/(FP + TN) is calculated per run and is based on all valid requests, so the No-SIV variant would raise an alert for any integrity event, thus trivially detecting 100% of all tampering attempts. Paired t-test between the per-run FPR values: $t = -165.37$, $p = 8.02 \times 10^{-9}$ (< 0.05). These numbers are obtained from simulations on the synthetic workload described above, and are not testbed numbers. Average of 5 separate simulation runs of 20,000 requests per run.

4.6.2 Policy-model benchmark

In addition to the same-testbed baselines in Table 4, simplified decision models for RBAC, ABAC, and PBAC have been created and tested against the RABAC decision model using a common environment (Python 3.12, one thread, same host), each evaluating a stream of 20,000 requests that comply with policies over five tests. This is a simulation-based benchmarking of the decision logic where we isolate the cost of evaluating policies and detecting violations; the latencies are not from any testbed of Hyperledger Fabric. The simulation-level performance comparison of the RBAC, ABAC, PBAC, and RABAC decision models is presented in Table 10. Table 11 compares the proposed RABAC framework with representative recent frameworks in terms of platform, storage architecture, risk scoring, permission granularity, and privacy-related considerations.

Table 10. Simulation-level policy-model benchmark in a common runtime

Model	Mean Decision Latency (μs)	Decision Throughput (Decisions/s)	Contextual-violation Detection (%)
RBAC	0.20 ± 0.03	≈4,900,000	0
ABAC	1.17 ± 0.08	≈853,000	0
PBAC	1.68 ± 0.03	≈596,000	0
RABAC (proposed)	8.23 ± 0.26	≈122,000	97.0

Note: Contextual-violation detection is evaluated against 2,000 replayed attacks that are policy-compliant but contextually anomalous (behavioral drift, off-hours bursts, unusual location); the context-free RBAC, ABAC and PBAC implementations considered here have no means to protect against such attacks; the 0% values are indicative of these implementations within the defined scenario, not risk-aware extensions of the same models. RABAC: risk-aware blockchain access control, RBAC: role-based access control, ABAC: attribute-based access control, PBAC: policy-based access control. The mean ± SD was obtained from five repetitions of 20,000 requests per repetition.

Table 11. Design-space comparison with representative recent frameworks

Aspect	RABAC (Proposed)	ACHealthChain [14]	MRAAC [26]	Atlam and Yang [23]
Platform	Hyperledger Fabric 2.5 + IPFS	Hyperledger Fabric + IPFS	None (Android framework)	None (centralized prototype)
Consensus	Raft	Raft (Fabric)	N/A (non-blockchain)	N/A (non-blockchain)
Storage mode	On-chain digests + IPFS ciphertexts	On-chain metadata + IPFS	Local device storage	Centralized database
Risk scoring	Four-dimensional contextual; bounded application-level operation count under prototype assumptions (Theorem 1)	No	Yes (multi-stage)	Yes (unified RBAC–ABAC–RiBAC)
Permission granularity	Three-party S–C–R policies + per-request risk	Role-based	Multi-stage context/sensitivity	Role + attribute + risk
GDPR/HIPAA discussion	Yes (Section 4.8)	Partial (privacy preservation)	Not reported	Not reported

Note. The descriptions are taken from their respective sources: [14, 23, 26] are two non-blockchain risk-aware approaches for the purposes of benchmarking on the risk taxonomy and the term not reported means the feature is not described in the source paper. The hardware, datasets, and workloads differ, so the table provides the comparisons among the designs in the design space; while Table 4 contains the controlled head-to-head comparison. The acronyms used include S-C-R (subject-controller-requester) and RiBAC (risk-based access control).

To give perspective on the cost of constant operation, the cost of constant operation is compared to the decision logic performance in Table 10. The cost of risk assessment in RABAC is 8.23 μs/decision, which is approximately 4.9 times the cost of scanning rules in PBAC and 41 times the cost of the pure RBAC check. However, it is about 4,000 times (≈3.6 orders of magnitude) less expensive than processing a request on the server-side, using the ledger path (Table 6). The computational cost of risk assessment by RABAC is not likely to dominate in this prototype in comparison to the cost of server-side processing. The difference in functionality is that the 2,000 replayed policy-compliant contextual attacks are identified by RABAC with 97.0% accuracy and are not identified by passing through RBAC or ABAC and passing through PBAC.

4.6.3 Comparison with recent frameworks

Under the above assumed prototype conditions, including GDPR/HIPAA compliance discussion (Section 4.8), RABAC offers a hybrid on-chain/IPFS storage scheme and provides a four-dimensional contextual risk mechanism at the application level out of the systems surveyed. In none of the papers compared are quantitative complexity and compliance postures reported, so as a result, estimation is not given, but instead “Not reported.”

4.7 Discussion and limitations

The experiment demonstrates that static policy rules provide an acceptable baseline, while a contextual risk score reduces how easy it is to make authorization decisions and immutable logging makes it difficult to perform and difficult to hide. A measured 0.8% false negative rate contains attacks that are not much different from normal user behavior. Mostly, the 33 ms time spent in the server-side processing is required for peer-side endorsement and transaction processing. The metric does not account for orderer queueing and block forming delays, so it cannot be used as a client-observable end-to-end access latency metric and should be considered in conjunction with ordering parameters and service-level latency requirements.

There are six restrictions that need to be mentioned with the findings. The first is that the performance evaluation was conducted based on analysis of 260 records with a single institution in a single country; additional testing in other institutions has yet to be done before deployment claims. Secondly, the attack traffic comprises of documented threat priors rather than traffic collected from the production hospital and the ground truth labels were manually assigned to synthetic access events that were conducted on top of real medical records; therefore, the accuracy measured is the accuracy of both the documentation and the detector. Finally, the equal-weight configuration is only a "fact" from this data set. Fourthly, semantic discrimination is only validated for the crash faults (Section 3.4) on the production alert streams longitudinally. Fourthly, the bounded drift validation in Section 4.2 is the only way to validate the validity of the clipped-EWMA mechanism. Finally, although the generating process for the threats is independent of the weights or thresholds in the risk engine, the separation of these two processes does not preclude any dependence between the synthetic workload and the feature space of the detector; the results of the detection metrics reported should be viewed accordingly. Seventhly, the reported 33 ms metric is a server-side processing metric, rather than the client-observable end-to-end confirmation latency. Although the ordering delay is analytically bounded by the block-cut configuration under the tested load, there is no end-to-end latency distribution

collected, and this should be the focus of future work.

4.8 Privacy and compliance considerations

Since the design works with electronic health records, its positioning concerning the GDPR and HIPAA is explained below. The data minimization design is architectural: only SHA-256 digests, integrity metadata, and authorization verdicts are anchored on-chain (Section 3.1), the payload remains encrypted off-chain using AES-256-GCM (Table 8); and the schema of the anchored record contains only pseudonyms, risk score, the decision, a timestamp, and the IPFS CID. Residual metadata linkability risks remain and need to be mitigated during deployment. Immutability and append-only make the decision log accountable by delivering the necessary auditing trail for accessing health data, as required by GDPR and HIPAA [3]. The anonymous publicly available dataset (Section 4.1) was used for the evaluation; no additional human subject data was collected (see Ethical Approval). The deployment-related risks include other risks such as consent management, conflict between erasure rights and immutable ledgers, and risk of re-identification in risk profiles (Section 5).

5. CONCLUSIONS

The proposed RABAC design is an authorization architecture for cloud EHRs and makes the following assumption: Integrity events must be evaluated in light of the request that caused the event, and the authorization decision is made based on that evaluation. The design combines three-party policy-based access control and a permissioned Hyperledger Fabric ledger for the immutable logging of all the decisions, and the four-dimensional risk engine from the ACRA (ACRA-DMAL), where the risk evaluation phase assumes the prototype size of the state but does not guarantee the end-to-end execution time in terms of state size. The prototype was shown to be the better system in terms of risk-screening accuracy, with tampering and access violation reduced by more than three-quarters, and to support 150-300 transactions per second over the measured 33 ms of ordering and block formation delays on the server-side, when compared with the cloud-only and centralized-database baselines in the synthesized workload over publicly available clinical data. Results of the components' removal were shown to be due to SIV in particular, and the policy model benchmark illustrated that the context-free RBAC, ABAC, and PBAC implementations are unable to block a policy-compliant attack that RABAC is able to block.

The evidence gathered is a prototype: One institution's dataset, a synthesized threat workload, and a crash-fault (Raft) adversary model; multi-institutional validation on production audit streams is required before deployment claims. Post-quantum signature and key agreement upgrade, comprehensive multi-arm ablation, Byzantine fault tolerance, comprehensive evaluation (MIMIC-IV), and adaptive SLBP campaigns for production audit logs are planned as future work. Future work is planned to include: adaptive SLBP campaigns for production audit logs, comprehensive evaluation (MIMIC-IV), comprehensive multi-arm ablation, Byzantine fault tolerance, and post-quantum upgrade of the signature and key agreement layers.

AUTHOR CONTRIBUTIONS

Conceptualization, H.M.; methodology, H.M., E.F.A. and R.J.A.; software, H.M.; formal analysis, H.M.; writing—original draft preparation, H.M., E.F.A., and R.J.A.; supervision, E.F.A.; validation, E.F.A. and R.J.A.; resources, E.F.A.; writing—review and editing, E.F.A. and R.J.A. All authors have read and agreed to the published version of the manuscript.

DATA AVAILABILITY

The CKD dataset is publicly available from the UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease, accessed on 12 September 2026) [38] under a CC BY 4.0 license; a Kaggle mirror was used for download (https://www.kaggle.com/datasets/mansoordaku/ckdisease, accessed on 12 September 2026). The supplementary package accompanying this submission contains the workload-generator configuration with the five workload seeds, the Go chaincode, the network-topology files (channel and docker-compose configurations), the experiment scripts, the record-selection script together with the list of retained record IDs, a README with software versions and run commands, per-seed checksums, the per-run confusion matrices and the raw latency logs underlying Section 4.5; a permanent Zenodo DOI will be assigned upon acceptance of this article.

REFERENCES

[1] Mell, P., Grance, T. (2011). The NIST definition of cloud computing. NIST Special Publication 800-145. https://doi.org/10.6028/NIST.SP.800-145

[2] Stallings, W. (2020). Cryptography and Network Security: Principles and Practice, 8th ed. Pearson Education, Harlow, UK. https://bushrabook.com/Books/Cryptography%20&%20Network%20Security%208th%20Ed.pdf.pdf.

[3] European Data Protection Board. (2025). Guidelines 02/2025 on the processing of personal data through blockchain technologies. https://www.edpb.europa.eu/system/files/2026-07/edpb_guidelines_202502_blockchain_v2_en.pdf, accessed on 2 August 2026.

[4] Androulaki, E., Barger, A., Bortnikov, V., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the 13th EuroSys Conference, Porto, Portugal, pp. 1-15. https://doi.org/10.1145/3190508.3190538

[5] Wei, P.C., Wang, D.H., Zhao, Y., Tyagi, S.K.S., Kumar, N. (2020). Blockchain data-based cloud data integrity protection mechanism. Future Generation Computer Systems, 102: 902-911. https://doi.org/10.1016/j.future.2019.09.028

[6] Yaqub, N., Zhang, J.B., Khalid, M.I., et al. (2025). Blockchain-enabled policy-based access control mechanism to restrict unauthorized access to electronic health records. PeerJ Computer Science, 11: e2647. https://doi.org/10.7717/peerj-cs.2647

[7] Eren, H., Karaduman, O., Gencoglu, M.T. (2025). Security challenges and performance trade-offs in on-chain and off-chain blockchain storage: A

comprehensive review. Applied Sciences, 15(6): 3225. https://doi.org/10.3390/app15063225

[8] Liu, Z.P., Ren, L.L., Feng, Y.J., Wang, S., Wei, J.H. (2023). Data integrity audit scheme based on quad Merkle tree and blockchain. IEEE Access, 11: 59263-59273. https://doi.org/10.1109/ACCESS.2023.3240066

[9] Wang, C.X., Sun, Y.F., Liu, B.Y., Xue, L., Guan, X.H. (2024). Blockchain-based dynamic cloud data integrity auditing via non-leaf node sampling of rank-based Merkle hash tree. IEEE Transactions on Network Science and Engineering, 11(5): 3931-3942. https://doi.org/10.1109/TNSE.2024.3393978

[10] Miao, Y., Gai, K.K., Yu, J., Tan, Y.A., Zhu, L.H., Meng, W.Z. (2025). Blockchain-empowered keyword searchable provable data possession for large similar data. IEEE Transactions on Information Forensics and Security, 20: 1374-1389. https://doi.org/10.1109/TIFS.2024.3516563

[11] Li, Y.N., Yu, Y., Chen, R.N., Du, X.J., Guizani, M. (2020). IntegrityChain: Provable data possession for decentralized storage. IEEE Journal on Selected Areas in Communications, 38(6): 1205-1217. https://doi.org/10.1109/JSAC.2020.2986664

[12] Jayabalan, J., Jeyanthi, N. (2025). Blockchain and Chainlink oracle integration in EHR: Ensuring real-time data integrity and interoperability with Ethereum, Hyperledger Fabric and IPFS storage. IEEE Access, 13: 158044-158056. https://doi.org/10.1109/ACCESS.2025.3606194

[13] Sayed, S., Famous, M.S., Mazumder, R., et al. (2025). Blockchain and InterPlanetary framework for decentralized and secure electronic health record management. Blockchains, 3(4): 12. https://doi.org/10.3390/blockchains3040012

[14] Tawfik, A.M., Al-Ahwal, A., Eldien, A.S.T., Zayed, H.H. (2025). ACHealthChain blockchain framework for access control and privacy preservation in healthcare. Scientific Reports, 15: 16696. https://doi.org/10.1038/s41598-025-00757-1

[15] Punia, A., Gulia, P., Gill, N.S., et al. (2025). An access control system for cloud-based healthcare systems driven by blockchain. Journal of Cloud Computing, 14: 78. https://doi.org/10.1186/s13677-025-00831-z

[16] Chakravarthy, D.G., Gopi, R., Murugan, S., Joseph, E.R. (2025). Enhancing confidentiality and access control in electronic health record systems using a hybrid hashing blockchain framework. Scientific Reports, 15: 30379. https://doi.org/10.1038/s41598-025-13831-5

[17] Chen, Y.W., Meng, L.H., Zhou, H., Xue, G.T. (2021). A blockchain-based medical data sharing mechanism with attribute-based access control and privacy protection. Wireless Communications and Mobile Computing, 2021(1): 6685762. https://doi.org/10.1155/2021/6685762

[18] Tawfik, A.M., Al-Ahwal, A., Eldien, A.S.T., Zayed, H.H. (2025). Blockchain-based access control and privacy preservation in healthcare: A comprehensive survey. Cluster Computing, 28: 529. https://doi.org/10.1007/s10586-025-05308-x

[19] De Oliveira, M.T., Reis, L.H.A., Verginadis, Y., Mattos, D.M.F., Olabarriaga, S.D. (2022). SmartAccess: Attribute-based access control system for medical records based on smart contracts. IEEE Access, 10: 117836-117854. https://doi.org/10.1109/ACCESS.2022.3217201

[20] Saini, A., Zhu, Q.Y., Singh, N., Xiang, Y., Gao, L.X., Zhang, Y.S. (2021). A smart-contract-based access control framework for cloud smart healthcare system. IEEE Internet of Things Journal, 8(7): 5914-5925. https://doi.org/10.1109/JIOT.2020.3032997

[21] Sunitha, B.J., Saravana Kumar, S. (2025). A novel hybrid blockchain-ABAC framework for multi-layered access control in cloud-based healthcare systems: Performance optimization and regulatory compliance. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 16(3): 178-197. https://doi.org/10.58346/JOWUA.2025.I3.011

[22] Taloba, A.I., Rayan, A. (2025). A privacy preserving medical data management framework using blockchain enabled encrypted role based access control. Scientific Reports, 15: 43864. https://doi.org/10.1038/s41598-025-30916-3

[23] Atlam, H.F., Yang, Y.F. (2025). Enhancing healthcare security: A unified RBAC and ABAC risk-aware access control approach. Future Internet, 17(6): 262. https://doi.org/10.3390/fi17060262

[24] Chhetri, P., Bhatt, S., Bhatt, P., Nobi, M.N., Benson, J., Krishnan, R. (2024). Environment aware deep learning based access control model. In Proceedings of the 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, Porto, Portugal, pp. 81-89. https://doi.org/10.1145/3643650.3659105

[25] Alharbe, N., Aljohani, A., Rakrouki, M.A., Khayyat, M. (2023). An access control model based on system security risk for dynamic sensitive data storage in the cloud. Applied Sciences, 13(5): 3187. https://doi.org/10.3390/app13053187

[26] Chen, J.Y., Hengartner, U., Khan, H. (2024). MRAAC: A multi-stage risk-aware adaptive authentication and access control framework for Android. ACM Transactions on Privacy and Security, 27(2): 1-30. https://doi.org/10.1145/3648372

[27] Pu, X.T., Jiang, R., Song, Z.M., Liang, Z.H., Yang, L. (2024). A medical big data access control model based on smart contracts and risk in the blockchain environment. Frontiers in Public Health, 12: 1358184. https://doi.org/10.3389/fpubh.2024.1358184

[28] Mohajan, A., Jahan, S. (2025). Embedding security awareness into a blockchain-based dynamic access control framework for the zero trust model in distributed systems. Electronics, 14(6): 1095. https://doi.org/10.3390/electronics14061095

[29] Pandit, A.K., Chatterjee, K. (2025). Adaptive access control in healthcare: Integrating health conditions for enhanced security and privacy. International Journal of Information Technology, 17: 3921-3933. https://doi.org/10.1007/s41870-025-02553-6

[30] Mallick, S.R., Lenka, R.K., Sobhanayak, S. (2025). Secure and scalable dual blockchain and IPFS driven IoT ecosystem for next gen healthcare systems. Scientific Reports, 15: 41064. https://doi.org/10.1038/s41598-025-25006-3

[31] Jayabalan, J., Jeyanthi, N. (2022). Scalable blockchain model using off-chain IPFS storage for healthcare data security and privacy. Journal of Parallel and Distributed Computing, 164: 152-167. https://doi.org/10.1016/j.jpdc.2022.03.009

[32] Stamatellis, C., Papadopoulos, P., Pitropakis, N.,

Katsikas, S., Buchanan, W.J. (2020). A privacy-preserving healthcare framework using Hyperledger Fabric. Sensors, 20(22): 6587. https://doi.org/10.3390/s20226587

[33] Al-Sumaidaee, G., Alkhudary, R., Zilic, Z., Swidan, A. (2023). Performance analysis of a private blockchain network built on Hyperledger Fabric for healthcare. Information Processing & Management, 60(2): 103160. https://doi.org/10.1016/j.ipm.2022.103160

[34] Zaabar, B., Cheikhrouhou, O., Jamil, F., Ammi, M., Abid, M. (2021). HealthBlock: A secure blockchain-based healthcare data management system. Computer Networks, 200: 108500. https://doi.org/10.1016/j.comnet.2021.108500

[35] Gupta, A.K., Raj, D., Sharma, Y.K., Sharma, A., Singh, M.K., Agrawal, A.K. (2025). Electronic healthcare data sharing application based on Hyperledger consortium blockchain network. In 2025 International Conference on Networks and Cryptology (NETCRYPT), New Delhi, India, pp. 1405-1410. https://doi.org/10.1109/NETCRYPT65877.2025.11102758

[36] Alashwal, M., Waheed, U., Algarni, M., et al. (2026). Trust aware and explainable access control for Internet of Medical Things: A lightweight hybrid blockchain approach. IEEE Internet of Things Journal, 13(2): 27843-27861. https://doi.org/10.1109/JIOT.2026.3680091

[37] Rose, S., Borchert, O., Mitchell, S., Connelly, S. (2020). Zero trust architecture. NIST Special Publication 800-207. https://doi.org/10.6028/NIST.SP.800-207

[38] Rubini, L., Soundarapandian, P., Eswaran, P. (2015). Chronic Kidney Disease. UCI Machine Learning Repository. https://doi.org/10.24432/C5G020

[39] Verizon. (2023). 2023 data breach investigations report. https://www.verizon.com/business/resources/reports/dbir/, accessed on 2 August 2026.

[40] IBM Security. (2026). Cost of a data breach report 2026. IBM Corporation. https://www.ibm.com/reports/data-breach, accessed on 10 August 2026.

NOMENCLATURE

Latin symbols

C	Stored ciphertext, bytes
C'	Retrieved (candidate) ciphertext, bytes
C	Data controller
c_{max}, c_{min}	Hardware bounds on per-class cost, time
d_B	Raw Mahalanobis distance, dimensionless
Di	Data object (record) identifier in request ρ
H	Anchored SHA-256 digest of C
H'	Recomputed digest of retrieved ciphertext C′
M	On-chain metadata entry (φ, s_i, H, S_i, t_0)
R	Composite risk score, dimensionless
R_B	Behavioral risk component, dimensionless
R_E	Environmental risk component, dimensionless
R_S	Data-sensitivity risk component, dimensionless
R_T	Temporal risk component, dimensionless
r_{dev}	Device-fingerprint risk contribution, dimensionless
r_{geo}	Geolocation-consistency risk contribution, dimensionless
r_{IP}	IP-reputation risk contribution, dimensionless
S	Requesting subject
s_i	Owner (subject) identifier in metadata entry M
S_i	Storage pointer (IPFS CID) in metadata entry M
t_0	Commit timestamp in metadata entry M
T_{deny}	Deny threshold on R, dimensionless
T_{grant}	Grant threshold on R, dimensionless
TPS	Transactions per second, transactions/s
U	Number of active users, dimensionless
u_{dow}	Day-of-week feature, dimensionless
u_{frq}	Request-frequency ratio feature, dimensionless
u_{hr}	Hour-of-day feature, dimensionless
v_ρ	Behavioral request vector (8-dimensional)
w_i	Weight of risk dimension i, dimensionless
X_t	Current behavioral observation at time t

Greek symbols

β_0, β_1, β_2, β_3	Logistic coefficients, temporal model, dimensionless
Γ	Profile-update forgetting factor, dimensionless
δ_{cap}	Per-update magnitude bound, dimensionless
Δt	Per-session poisoning perturbation, dimensionless
κ	Request context component of ρ
μ_s	Behavioral centroid of subject s
ρ	Access request tuple (s_i, d_i, t, κ)
Σ_s	Covariance of s's behavioral profile
$\sigma(x)$	Logistic sigmoid function
τ_1, τ_2, τ_3	Cost classes: ledger verification, policy evaluation, contextual scoring, time
φ	File identifier in metadata entry M

APPENDIX

APPENDIX A

Detailed proof of Theorem 1

Proof. Partition the fixed set of primitive operations into three cost classes, examined from the ledger inward, with costs τ_1, τ_2, τ_3.

(i) Ledger verification, τ_1: one bounded world-state lookup—GetState against the peer's LevelDB world-state store, a keyed lookup whose cost does not depend on the chain length, since the hash-linked block log is not traversed per request—plus one ECDSA endorsement verification: a fixed number of application-level operations. The lookup is keyed and independent of ledger height; its internal LevelDB cost grows only logarithmically with world-state size, so it is treated as bounded within the prototype's bounded-state regime.

(ii) Policy evaluation, τ_2: the three PBAC families compile into a 32-node-capped ROBDD, so traversal cost is bounded regardless of the request. In deployments whose policy bases exceed the 32-node cap (e.g., large multi-department consortia with hundreds of composite rules), τ_2 would grow with rule count; we conjecture logarithmic growth for structured rule sets, but this remains an unproven conjecture, and the bound would need re-derivation under a relaxed cap—a direction left for future validation with larger policy bases than the 30 rules used here (Section 4.1).

(iii) Contextual scoring, τ_3: the remaining operations are hash-indexed lookups (the 144-slot temporal profile buffer; the IP-reputation hash table; the sensitivity label, read from a cached in-memory map), fixed-arity tests (a Bloom-filter

whitelist probe; a point-in-polygon test against six vertices), and fixed-dimension numerics (the logistic score of Eq. (1) over three scalars; the Mahalanobis distance of Eq. (2) in $d = 8$ dimensions, an $O(d^2)$ cost of exactly 64 multiply-adds; the four-term dot product of Eq. (5)). Each executes a constant number of operations, hence so does τ_3.

Then $T(\rho) = \tau_1 + \tau_2 + \tau_3 \leq 3 \cdot c_{max}$ with $c_{max} = \max\{\tau_1, \tau_2, \tau_3\}$ a hardware-dependent constant independent of the request, the ledger height and the user population within this bounded configuration, so $T(\rho) = O(1)$; and as every request executes all three classes at least once, $T(\rho) \geq 3 \cdot c_{min}$ for $c_{min} > 0$, so $T(\rho) = \Omega(1)$. Together, the per-request operation count is $\Theta(1)$ in application-level operations. This is a bound on the number of application-level operations, not on the internal complexity of the storage engine, and it applies only to the bounded prototype configuration, including a bounded number of key lookups per request; deployments that relax that bound would reintroduce a state-size-dependent verification term rather than a constant one.