

Secure Electronic Health Record Management Using an Application-Layer Blockchain with Patient-Mediated Access Control

Puchala Bhanu Vaishnavi*, Dr. D. Latha

Department of Computer Science and Engineering Adikavi Nannaya University College of Engineering Rajamahendravaram, Andhra Pradesh.

ABSTRACT

Electronic health records hold some of the most sensitive information ever collected about a person, yet most institutions still keep them locked inside closed, hospital-controlled systems. Patients have almost no way of finding out who has looked at their record, and no way to confirm that a diagnosis entered years ago was never quietly changed. This paper presents a lightweight, application-layer blockchain framework for EHR management, built on a Flask web tier and a MongoDB Atlas document store, with the goal of closing that gap without the deployment cost of a full permissioned blockchain. Each patient's clinical history is organised as a hash-chained sequence of record blocks: every new block embeds the SHA-256 digest of its canonically serialized predecessor, so any retrospective change to an earlier entry invalidates every digest that follows it, and such tampering can then be exposed through an inexpensive chain-verification pass. Data sharing between patients and clinicians is governed by patient-mediated access contracts, held as auditable documents in dedicated collections. Their status (pending, approved, or revoked) is re-evaluated on every record fetch and mirrored into a persistent access log, so patients retain continuing control over their data along with a full usage trail. Beyond this core mechanism, the framework supports department-wise record capture, appointment scheduling, guardian linking for dependents, organ-donor and insurance modules, QR-based record identifiers, and credential storage hardened with salted, iterated PBKDF2-SHA-256 hashing. The paper describes the system's architecture and methodology and evaluates the integrity, access-control, and auditability properties observed during operation. A clearly bounded future scope, covering Hyperledger Fabric migration, ciphertext-policy attribute-based encryption, homomorphic computation, medical image similarity search, and AI-assisted disease prediction, is kept separate from the implemented system and is not claimed as delivered work.

Keywords: electronic health records, application-level blockchain, SHA-256, hash chain, access control, access contracts, MongoDB, Flask, patient privacy, access logging, healthcare information systems

INTRODUCTION

Healthcare digitisation has made the electronic health record the operational core of clinical practice, yet how these records are stored and shared has lagged well behind the sensitivity of what they contain. Most records still live inside databases that a single hospital administers on its own terms. A patient has no independent way to confirm that a diagnosis logged years ago was never quietly edited, cannot see which clinicians looked at their history, and routinely runs into friction the moment care crosses an institutional boundary. Healthcare data also remains an attractive target for theft, since a single record typically bundles identity, financial, and clinical information together [4].

Blockchain technology, first introduced as the ledger substrate underlying Bitcoin [1], offers a conceptually attractive remedy: an append-only, hash-linked structure in which altering any past entry becomes computationally obvious. A considerable body of work has already explored this idea for healthcare specifically. MedRec was among the first to manage medical-record permissions through Ethereum smart contracts [2]. MeDShare tackled trust-less sharing among cloud providers by auditing every access [3]. Shahnaz et al. proposed a more granular, role-aware blockchain EHR framework [4], and Nguyen et al. paired blockchain with IPFS to support mobile-cloud EHR exchange [5]. More recent efforts lean toward permissioned

Relevant conflicts of interest/financial disclosures: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.



platforms, particularly Hyperledger Fabric [6], [13], [14], combined with heavier cryptography such as ciphertext-policy attribute-based encryption (CP-ABE) [7], [15]–[17], searchable encryption [10], and homomorphic schemes [8], [9], [18].

Two observations shaped the direction of the present work. The first is practical: full-strength systems of this kind bring substantial deployment overhead, including pairing-based cryptography, attribute authorities, ordering services, and chaincode lifecycles, that a resource-constrained clinic, or for that matter an academic prototype, cannot easily take on. The second is that many of the guarantees patients actually care about day to day (tamper evidence, explicit consent before any access, and a complete audit trail) do not strictly require that heavy machinery. They can be obtained at the application layer through disciplined use of ordinary cryptographic primitives, producing a working substrate onto which stronger cryptography could later be grafted without redesigning the clinical workflow from scratch. This paper implements and evaluates precisely that intermediate design point.

The project was organised around six concrete objectives, each realised in the delivered software rather than left aspirational. These are: maintaining every patient's clinical history as a tamper-evident, per-patient hash chain using SHA-256 digests with previous-hash linkage; placing

record sharing under explicit patient control through grant/revoke access contracts that are checked at read time rather than only at grant time; logging every access decision and every read so patients end up with a complete usage trail; delivering a full clinical workflow, spanning patient, doctor, and admin roles, department-wise records, appointments, guardian linking, organ-donor and insurance modules, and QR-based record identifiers, on a single Flask/MongoDB stack; protecting authentication credentials with salted, iterated PBKDF2–SHA-256 hashing and role-separated sessions; and stating the system's security boundaries honestly enough to derive a staged enhancement roadmap from them.

Every claim made about the system in this paper describes software that has actually been built and exercised. The technologies discussed under Future Scope are proposed directions only, and they are not part of the implementation. The rest of the paper is

organised as follows: Section II reviews recent literature, Section III describes the system architecture, Section IV presents the methodology, including the ledger design, hashing scheme, access-control mechanism, and the modules built around them, Section V discusses the results observed during evaluation, Section VI concludes the paper, and Section VII lays out the future scope.

II. Literature Survey

Blockchain-based EHR research since 2020 has broadly progressed along two fronts: richer cryptographic protection for record content, and increasingly mature deployments on permissioned ledgers. Antunes et al. [20] carried out a systematic review of federated learning for healthcare and proposed a reference architecture. This is useful here because it frames the privacy-preserving, collaborative-learning setting into which ledger-anchored medical data is increasingly expected to feed.

Wang et al. [10] proposed MedShare, a scheme that combines constant-size attribute-based encryption with non-interactive Boolean search executed on-chain over Ethereum. Their results show that encrypted retrieval with an embedded access policy is achievable on-chain, though the accompanying token and index pipeline is not trivial and inherits Ethereum's gas costs. In the same year, Jain et al.

[12] built an Ethereum/Solidity EHR prototype on ReactJS and Firebase, defining CRUD-style smart contracts for patient records. The work is a reasonable proof that a small team can prototype blockchain EHR systems, but it depends on a testnet and a centralized identity backend rather than anything production-grade. Yadav and Jason [11] described a blockchain–IPFS storage design that pairs an improved CP-ABE primitive with Paillier-based protection for insurance claims. Their paper is one of the clearer articulations of the attribute-plus-homomorphic vision that later shaped the roadmap adopted here, although it reports comparatively little implementation or evaluation detail.

On the permissioned-platform side, Hasnain et al. [13] conducted a systematic review of 57 studies on Hyperledger Fabric in healthcare and found privacy, integrity,

traceability, and availability to be the dominant reasons organisations adopt it, while also flagging persistent gaps in configuration and benchmarking practice. Adanur Dedetürk and Bakir-Güngör [14] implemented AguHyper, a Fabric-based EHR framework combining IPFS storage, CouchDB state, and Raft consensus, and reported throughput and latency figures across different dataset sizes. It stands as a good example of the measured, fully permissioned end-state that the present roadmap points toward.

Cryptographic access control specifically has kept advancing as well. Li et al. [15] proposed a more flexible, policy-driven approach to EHR sharing on blockchain that improves policy expressiveness without escaping ABE-class computational cost. Yang [18] combined blockchain coordination with threshold homomorphic encryption for federated medical learning, preventing gradient leakage while keeping accuracy close to what plaintext training achieves. Qiao et al. [16] presented a lightweight CP-ABE scheme for cloud-hosted EHRs that pairs blockchain with secure multi-party computation specifically to reduce the burden on constrained client devices. Most recently, Worapaluk et al. [17] took on the revocation bottleneck that CP-ABE-based EHR sharing is known for, using lazy re-encryption, proxy assistance, and Bloom-filter queries. Their results confirm that dynamic permission change is still an unresolved problem in this space.

Reading across this body of work, a fairly clear pattern emerges. At one end, cryptographically rich designs [10],

[15]–[17] deliver strong confidentiality and fine-grained control, but they assume the existence of attribute authorities, pairing libraries, and usually a running permissioned or public chain, infrastructure that small hospitals, clinics in developing regions, and most academic prototypes simply do not have. At the other end, survey-level evidence [13] shows that even organisations with the resources to adopt Hyperledger Fabric still struggle with configuration, benchmarking, and integration. What is comparatively under-explored is the middle ground, namely how much of the patient-facing value of blockchain EHRs (tamper evidence, explicit consent, and auditability) can be delivered using commodity web infrastructure and standard hashing and key-derivation primitives, structured so stronger cryptography can be added later rather than designed in from day one. The present

work addresses part of that gap by implementing and evaluating exactly that intermediate point, and it deliberately stops short of claiming the confidentiality guarantees of attribute-based encryption, the encrypted-computation capability of homomorphic schemes, or the consensus-backed immutability of a permissioned platform.

III. System Architecture

The system takes the form of a web-based EHR management platform in which the database itself is organised as a collection of per-patient ledgers rather than a single shared table. Three design principles govern how it behaves. Tamper evidence over trust means that instead of simply trusting whoever administers the database, every

record carries a verifiable cryptographic link to the one before it. Consent before access means that no clinician can successfully read a record without a currently valid, patient-approved contract in place. Everything auditable means that record creation, contract-status changes, and reads all leave a durable trace. Relative to a conventional EHR portal, this adds integrity verification and consent enforcement that would not otherwise exist. Relative to a platform blockchain, it strips out consensus, mining, and chaincode infrastructure, trading decentralisation for something that is actually deployable on a modest budget.

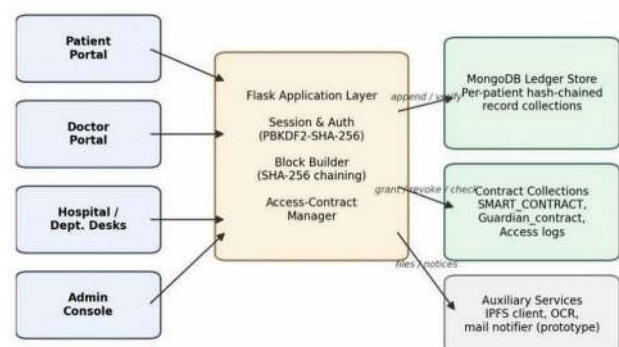


Figure 1: Implemented System Architecture.

Fig. 1 shows the implemented architecture. Four presentation-tier portals (patient, doctor, hospital departments, and admin) all terminate at a single Flask application layer that houses the session and authentication service, the block builder, and the access-contract manager. Persistence sits in MongoDB Atlas, with per-patient ledger collections on one side and contract and audit collections on the other. A handful

of auxiliary utilities, namely an IPFS client, OCR ingestion, and mail notification, exist as stand-alone scripts alongside the main application; these are reported here as engineering scaffolding rather than as integrated, production features.

Each of the four portals is scoped to a distinct role. The patient module is the most feature-rich of the four: it handles registration and login, lets a patient view their own hash-chained history, approve, deny, or revoke access contracts, link or remove guardians together with a delegation level, book appointments, register as an organ donor, apply for insurance, review their personal access log, and generate a QR code that encodes the record identity for quick retrieval at a reception desk. The doctor module is deliberately narrower: it authenticates the clinician, lets them search for patients who have granted access, raise new access requests, create department-specific record blocks after an examination, and view only the chains they currently have permission to see, alongside prior consultation history. Department desks (general medicine, biochemistry, cardiology, and dermatology in the present deployment) each expose a tailored clinical template (glucose, serum, lipid-profile, and pulse fields in the general-medicine template, for instance), but every one of these templates ultimately feeds the same block builder, so the chaining

semantics stay uniform no matter which specialty produced the entry. The admin module is the smallest of the four, covering onboarding of doctor and department accounts, provisioning of new patient ledger collections, and oversight dashboards. A set of supporting modules sits alongside these four: appointment management with request, schedule, and notification views; guardian linking implemented through dedicated contract documents; an organ-donor registry; insurance application and review; and QR-code generation for record identifiers. All of them are built on the same underlying ledger and contract primitives rather than as separate subsystems.

Underneath the portals, the MongoDB Atlas deployment (database Blockchain) is organised into six kinds of collections: per-patient ledger collections, one per registered patient, holding the genesis document and the PID-REC_n block documents that follow it; SMART_CONTRACT, holding clinician-access contracts with identifier, owner, accessor, record, status, and timestamp fields; Guardian_contract, holding guardian delegations together with a level

field; account collections for doctors, departments, and administrators, storing PBKDF2-hashed credentials rather than plaintext passwords; access-log collections that record every read attempt and every contract-status transition; and auxiliary collections for appointments, organ-donor declarations, and insurance applications. A separate database stores uploaded imaging artefacts as base64 documents for the stand-alone image utility. Modelling one document per block keeps chain traversal to a simple ordered scan, and it made the consent workflow straightforward to extend to guardians when that feature was added later.

One terminological point is worth making explicit: the sharing agreements in this system are called access contracts rather than smart contracts in the platform sense. They encode agreement state in ordinary database collections and are enforced by the application layer, not by autonomous on-chain execution. Keeping that distinction clear positions the work accurately against chaincode-based systems such as Hyperledger Fabric [6].

METHODOLOGY

A. Per-Patient Hash-Chained Ledgers

Every patient is provisioned a dedicated MongoDB collection that functions as their personal ledger. The collection starts with a genesis document holding the registration profile, a null previous-hash, and a genesis digest. Every clinical entry recorded after that becomes a block document, identified as PID-REC_n, containing the owner's identifier, the authoring doctor, department-specific clinical fields, a timestamp, the predecessor's digest (prevhash), and its own digest (hash). Because each block embeds the digest of its canonically serialized predecessor, the collection behaves as an append-only chain: retroactively modifying block *i* invalidates the digest stored for block *i* and breaks the prevhash reference held by block *i*+1, which localises any tampering to a precise point in the sequence (Fig. 2). This is deliberately an application-level

blockchain. The chaining and its verification are carried out by the application tier over an ordinary document store, without any distributed consensus, and that particular boundary is examined more closely in Section V.

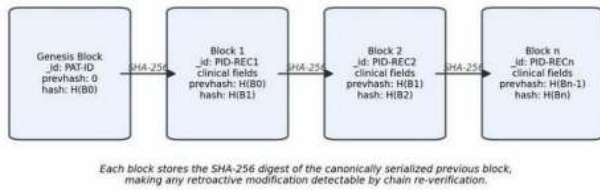


Figure 2: Per-patient Block Structure and SHA-256 Hash Chaining.

Block creation follows a short, deterministic procedure. Given a patient id, a set of clinical fields, and an authoring doctor id, the application locates the patient's ledger collection, reads the hash of its current last block as the new block's prevhash, canonically serializes the proposed block (JSON with lexicographically sorted keys and stable value types), computes its SHA-256 digest, and inserts the completed block, returning its identifier. Verifying a ledger later follows just as directly: the application walks the collection once, recomputing each block's digest over its content (excluding the stored hash field itself) and checking that the recomputed digest matches what is stored and that the prevhash reference matches the actual hash of the preceding block. The first index at which either check fails is reported as the tamper point, and a clean pass through the entire ledger is reported as intact. Canonical serialization is not a cosmetic detail here: any nondeterminism in how a block is serialized, such as unordered keys or inconsistent numeric formatting, would make the digest unreproducible and break verification outright, so the block builder enforces sorted keys and stable value types everywhere a block is written or re-hashed.

B. SHA-256 Hash Generation and Credential Protection

Block digests are computed with SHA-256 [23], chosen for its preimage and collision resistance: constructing different block content that reproduces a given digest is computationally infeasible with current techniques. Because the stored digest attests that the serialized content has not changed, and each block also embeds its predecessor's digest, the collection as a whole forms a tamper-evident sequence rather than a set of independently verifiable records. The same primitive family is reused to protect user credentials: passwords are never stored in plaintext or as a single

unsalted hash, but as salted PBKDF2–SHA-256 values computed over 30,000 iterations, so a raw database leak does not directly expose usable passwords.

C. Access Control Mechanism

Sharing between patients and clinicians is governed by contract documents held in two dedicated collections: SMART_CONTRACT for clinician access and Guardian_contract for guardian delegation. Each contract

binds an owner (the patient), an accessor (a doctor or guardian), a record or scope, a status field, and a set of timestamps. When a doctor requests access to a patient's records, a pending contract is created and the patient is notified. The patient can then approve, deny, or later revoke that contract from their dashboard, while guardians are linked to dependents with an associated delegation level.

The mechanism that actually enforces this is a contract-gated read. On every access attempt, the application first looks for a matching contract between the requesting accessor and the target record. If none exists, one is created in a pending state, the patient is notified, and the read is denied. If a contract exists but its status is not currently approved, the attempt is logged as a denied read and again refused. Only when a contract is found in the approved state does the read succeed, and that success is itself appended to the access log alongside the record identifier, the accessor, and a timestamp. Because this check is repeated at read time rather than only once at grant time, a revocation issued from the patient dashboard takes effect on the very next access attempt rather than waiting on some separate synchronisation step. Every decision, grant, denial, revocation, and successful read, ends up recorded on the patient's audit page (Fig. 3).

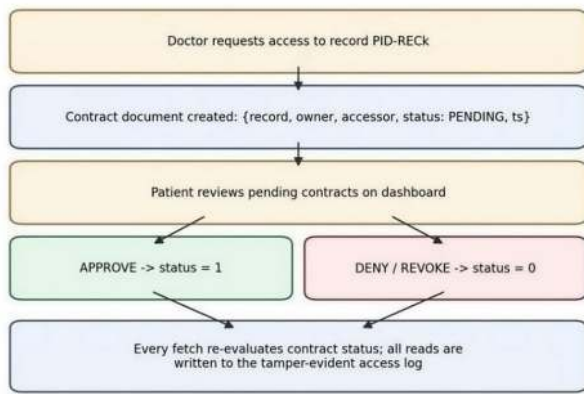


Figure 3: Access-contract Lifecycle with Audit Logging.

D. Development and Deployment Environment

The prototype was developed and exercised in a fairly ordinary environment, and no synthetic performance benchmarks were run or are claimed here. On the software side, the web tier is Python 3 with Flask, pymongo talks to a MongoDB Atlas cluster, hashlib supplies SHA-256 for chaining, passlib provides PBKDF2-SHA-256 at 30,000 rounds for credential hashing, roughly seventy Jinja2/HTML/CSS/JavaScript views make up the four portals, and the qrcode library generates record identifiers. A handful of stand-alone utility scripts exercise ipfshttpclient against a local IPFS daemon, pytesseract for OCR, and flask-mail for notifications, but these remain outside the main application. On the hardware side, development and testing used a commodity x86-64 machine with 8 GB of RAM, with Flask's own development server fronting the application and the database hosted on a free-tier Atlas

cluster. The workload used to exercise the system consisted of manually seeded patient, doctor, and department accounts; end-to-end walkthroughs of registration, record creation across all four departments, contract request/approval/revocation, guardian delegation, and appointment booking; and repeated chain verification after deliberately introduced out-of-band database edits, which the verification procedure correctly flagged every time.

RESULTS AND DISCUSSION

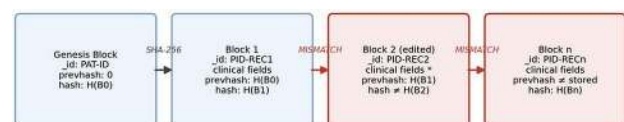
The evaluation reported here is qualitative, matching the scope of a prototype at this stage. What follows is a discussion of the properties actually observed during operation, together with their limits.

Integrity. Fig. 4 shows two consecutive ledger blocks, PAT001 and PAT002, captured directly from the deployed system. The previous-hash field stored in Block PAT002 is identical to the hash field stored in Block PAT001, confirming that the chaining relationship described in Section IV-A works correctly on live data and not just on paper. Building on that baseline, tamper evidence was verified directly: when a stored clinical field was deliberately altered through the database console, the recomputed digest for that block diverged from the digest stored alongside it, and that mismatch propagated through every subsequent block's prevhash check, so tampering was both detected and localised to a specific point in the chain (Fig. 5). To be precise about what this demonstrates: it is tamper evidence, not tamper prevention. An adversary with full database write access could, in principle, rewrite an entire suffix of a chain and recompute consistent digests for it. External digest anchoring would mitigate this in the near term, and consensus-backed replication is the longer-term answer captured in the roadmap.

BLOCK PAT001	
INDEX	VALUE
FIRST NAME	Gita
LAST NAME	reddy
TIMESTAMP OF CREATION	2025-07-04 21:24:38
PREVIOUS HASH	08a035a4d3f733a2a484a4073a2517d048a5a1033eac1541c3402cc5d8f
Hash	2960ea32c17b1a02d2c4d433b3a14111c2a9b6dcffaf343d8f9e45db7022b

BLOCK PAT002	
INDEX	VALUE
FIRST NAME	Lavini
LAST NAME	B
TIMESTAMP OF CREATION	2026-07-09 17:35:14
PREVIOUS HASH	2960ea32c17b1a02d2c4d433b3a14111c2a9b6dcffaf343d8f9e45db7022b
Hash	a2c1a87b39055eacdc65a670a955e4913db1a0004d1a15a1733e4d4247

Figure 4: Sample Ledger Blocks Retrieved from the Deployed System: the Previous-Hash Field of Block PAT002 Matches the Stored Hash Field of Block PAT001, Confirming Correct Chain Linkage on Live Data.



Block 2 is modified after creation: its recomputed digest no longer matches the stored hash, and the break propagates through every downstream prevhash check.

Figure 5: Effect of an Out-Of-Band Edit to Block 2: the Recomputed Digest No Longer Matches the Stored Hash, and the Break Propagates Through Every Downstream Prehash Check.

Access control. No clinician was able to view a patient's data without a contract in the approved state, and revoking a contract from the patient dashboard

denied the very next fetch attempt, confirming that the read-time check behaves as intended. That said, enforcement here is entirely application-mediated: a compromised application server, or a database operator acting outside the application, could bypass it altogether. Closing that particular gap is exactly what ciphertext-bound access policies are meant to address in future work.

Privacy. Patient credentials are protected with salted, iterated PBKDF2-SHA-256, and role separation keeps a doctor's session from browsing patient-portal views or vice versa. Clinical payloads themselves, however, sit unencrypted at rest in MongoDB, so confidentiality currently rests on database-level access controls rather than on cryptography. This is honestly the single highest-priority item in the future-work list.

Auditability. Every read attempt, along with every grant, denial, and revocation, is appended to the access log and surfaced back to the patient. Table entries carry a timestamp, the requesting accessor, the record or contract involved, the action, and its outcome (Fig. 6), which gives patients a degree of usage transparency that centralised EHR systems typically do not offer and that is broadly consistent with what systems like MedShare pursue at cloud scale [3].

Timestamp	Accessor	Record	Action	Result
2026-06-14 09:12:03	Dr. R. Menon (Cardiology)	PID-REC014	READ	Approved
2026-06-14 09:15:41	Dr. R. Menon (Cardiology)	PID-REC017	REQUEST	Pending
2026-06-14 09:20:08	Patient (self)	Contract #C:2291	REVOKE	Access closed
2026-06-14 09:20:55	Dr. R. Menon (Cardiology)	PID-REC017	READ	Denied
2026-06-15 11:02:17	Guardian: S. Rao (level 2)	PID-REC003	READ	Approved

Figure 6: Illustrative Excerpt of the Access-Log Schema, Showing How Grant, Revocation, and Read Events Are Recorded for Patient Review.

Taken together, the framework delivers a tamper-evident history, deployable on ordinary infrastructure without consensus nodes, wallets, or gas fees; patient agency through explicit consent, immediate revocation, and a personal audit trail; consistent chaining semantics across otherwise heterogeneous department templates; hardened credential storage; and a workflow substrate deliberately structured so that a permissioned ledger and stronger cryptography could be layered onto it later without redesigning the clinical processes built around it. That

combination of properties is what makes the design a reasonable fit for small hospitals, clinics, and academic settings in particular.

CONCLUSION

This paper has presented a lightweight, application-layer blockchain framework for electronic health records, implemented end to end on a Flask and MongoDB Atlas stack. Per-patient SHA-256 hash chains give every patient a tamper-evident history of their own records, and patient-mediated access contracts, evaluated at read time and backed by a persistent audit log, give patients real control over who sees that history along with a record of who actually looked. PBKDF2-hardened credentials and role-separated portals round out a complete clinical workflow that spans patient, doctor, department, and admin roles, together with appointments, guardian delegation, organ-donor registration, insurance handling, and QR-based identifiers. The design deliberately sits at an intermediate point in the literature: it delivers the day-to-day guarantees that matter most to patients while staying deployable on commodity infrastructure, and its security boundaries are stated clearly enough to define a realistic, staged path toward stronger cryptographic and platform guarantees rather than leaving that path implicit.

VII. Future Scope

None of what follows is implemented in the present system; each item below is a proposed direction for future work, kept deliberately separate from the description of what has actually been built.

Hyperledger Fabric migration. Re-host the ledger on a permissioned Fabric network [6], [14], with individual hospitals modelled as organisations, chaincode taking over from the application-mediated contract checks used today, and Raft ordering providing consensus-backed immutability that the current design does not have.

Ciphertext-policy attribute-based encryption. Encrypt record payloads under attribute policies [7], [16], [17] so that decryption capability itself, rather than application logic, determines who can actually read a record, with efficient revocation handled through re-encryption techniques.

Homomorphic computation. Apply the Paillier cryptosystem [8], [9] to insurance-claim and billing aggregation so that financial workflows can operate on encrypted amounts without ever observing plaintext clinical figures.

Medical image similarity search. Store imaging studies off-chain with ledger-anchored digests, index them using CNN or Vision Transformer embeddings, and support approximate-nearest-neighbour retrieval of visually similar prior cases [21].

AI-assisted disease prediction with explainability.

Train diagnostic models across institutions through federated learning [19], [20], protect model updates with homomorphic aggregation [18], and expose clinician-facing

explanations through techniques such as Grad-CAM and SHAP [22].

IoT healthcare and national health-information

exchange. Ingest device telemetry directly into patient chains and align record schemas with national health-exchange standards to support interoperability across institutions.

REFERENCES

1. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper, 2008.
2. A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in Proc. 2nd Int. Conf. Open Big Data (OBD), 2016, pp. 25–30, doi: 10.1109/OBD.2016.11.
3. Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," IEEE Access, vol. 5, pp. 14757–14767, 2017, doi: 10.1109/ACCESS.2017.2730843.
4. A. Shahnaz, U. Qamar, and A. Khalid, "Using blockchain for electronic health records," IEEE Access, vol. 7, pp. 147782–147795, 2019, doi: 10.1109/ACCESS.2019.2946373.
5. D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for secure EHRs sharing of mobile cloud based e-health systems," IEEE Access, vol. 7, pp. 66792–66806, 2019, doi: 10.1109/ACCESS.2019.2917555.
6. E. Androulaki et al., "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in Proc. 13th EuroSys Conf., 2018, pp. 1–15, doi: 10.1145/3190508.3190538.
7. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in Proc. IEEE Symp. Security and Privacy (S&P), 2007, pp. 321–334, doi: 10.1109/SP.2007.11.
8. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Advances in Cryptology—EUROCRYPT, 1999, pp. 223–238, doi: 10.1007/3-540-48910-X_16.
9. A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," ACM Computing Surveys, vol. 51, no. 4, pp. 1–35, 2018, doi: 10.1145/3214303.
10. M. Wang, Y. Guo, C. Zhang, C. Wang, H. Huang, and X. Jia, "MedShare: A privacy-preserving medical data sharing system by using blockchain," IEEE Trans. Services Computing, vol. 16, no. 1, pp. 438–451, Jan./Feb. 2023, doi: 10.1109/TSC.2021.3114719.
11. R. M. Yadav and F. Jason, "EHRChain: A blockchain based EHR system using attribute based and homomorphic cryptosystem," Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE), vol. 12, no. 3, pp. 74–79, Mar. 2023, doi: 10.17148/IJARCCE.2023.12314.
12. M. Jain, D. Pandey, and N. P. Singh, "EHR: Patient electronic health records using blockchain security framework," in Proc. Int. Conf. Innovative Data Communication Technologies and Application (ICIDCA), 2023, pp. 710–715, doi: 10.1109/ICIDCA56705.2023.10099789.
13. M. Hasnain, F. R. Albogamy, S. S. Alamri, I. Ghani, and B. Mehboob, "The Hyperledger Fabric as a blockchain framework preserves the security of electronic health records," Frontiers in Public Health, vol. 11, art. 1272787, 2023, doi: 10.3389/fpubh.2023.1272787.
14. B. Adanur Dedetürk and B. Bakir-Gungor, "AguHyper: A Hyperledger-based electronic health record management framework," PeerJ Computer Science, vol. 10, e2060, 2024, doi: 10.7717/peerj-cs.2060.
15. P. Li, D. Zhou, H. Ma, and J. Lai, "Flexible and secure access control for EHR sharing based on blockchain," J. Systems Architecture, vol. 146, art. 103033, 2024, doi: 10.1016/j.sysarc.2023.103033.
16. J. Qiao et al., "A lightweight CP-ABE scheme for EHR over cloud based on blockchain and secure

- multi-party computation," *Trans. Emerging Telecommunications Technologies*, vol. 36, e70053, 2025.
17. K. Worapaluk et al., "Blockchain-enabled privacy-preserving access control for EHRs sharing with optimized user and attribute revocation," *IET Information Security*, 2026, doi: 10.1049/ise2/3917525.
 18. Y. Yang, "Federated medical learning framework based on blockchain and homomorphic encryption," *Wireless Communications and Mobile Computing*, vol. 2024, art. 8138644, 2024, doi: 10.1155/2024/8138644.
 19. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.
 20. R. S. Antunes, C. A. da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Trans. Intelligent Systems and Technology*, vol. 13, no. 4, pp. 1–23, 2022, doi: 10.1145/3501813.
 21. G. Litjens et al., "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017, doi: 10.1016/j.media.2017.07.005.
 22. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, pp. 618–626, doi: 10.1109/ICCV.2017.74.
 23. National Institute of Standards and Technology, "Secure Hash Standard (SHS)," *FIPS PUB 180-4*, Aug. 2015, doi: 10.6028/NIST.FIPS.180-4.

HOW TO CITE: Puchala Bhanu Vaishnavi, Dr. D. Latha, Secure Electronic Health Record Management Using an Application-Layer Blockchain with Patient-Mediated Access Control, *Int. J. Sci. R. Tech.*, 2026, 3 (10), 635-643. <https://doi.org/10.5281/zenodo.23261794>