

MediVault: A Blockchain-Based Secure Electronic Health Record System

Dhivya V¹, Charitha A², Charvi S³, Nidhi Dongre⁴, Prapthi S Nagavi⁵

¹Assistant Professor, Department of Computer Science and Engineering, Sir M Visvesvaraya Institute of Technology, Bengaluru, Karnataka, India

^{2,3,4,5}UG Student, Department of Computer Science and Engineering, Sir M Visvesvaraya Institute of Technology, Bengaluru, Karnataka, India

Abstract - Electronic health record platforms routinely ask patients to extend a trust they have no way to audit: that whoever operates the servers will neither read a record it should not, nor quietly release it to someone who should not have it. MediVault narrows that request rather than restating it. It is a prototype in which a patient's permission to read a record is expressed as a transaction on an Ethereum smart contract and checked before any file leaves the backend, while the clinical files themselves stay off the chain, encrypted with AES-256 in Amazon S3 and described by structured metadata in PostgreSQL. A SHA-256 digest recorded on-chain lets a later reader notice if a stored file has been altered, and doctors sign prescriptions with an Ethereum wallet so a pharmacy can confirm the signing key and detect tampering through ECDSA. The implemented workflows cover registration and login, consent grant and revocation, record upload and retrieval, prescription verification and time-limited emergency access. We are deliberate about two boundaries: a three-guardian key-recovery scheme is described as a proposal and was not built, and because decryption currently depends on a single backend-held master key, the contract governs release but does not yet enforce cryptographic access on its own. Functional behaviour was verified; latency, throughput and transaction-cost measurement remain future work.

Key Words: electronic health records, blockchain, Ethereum smart contracts, consent management, AES-256, SHA-256, ECDSA, off-chain storage.

1. INTRODUCTION

Moving a patient's chart from a paper folder into a database made it searchable, shareable and far easier to lose in bulk. The same consolidation that lets a clinician retrieve a history in seconds also means that a single stolen credential, an over-broad administrator role or one misconfigured server can expose records by the thousand rather than the dozen. Blockchain-backed record sharing has been examined as a way to make particular actions who was granted access, when, and to what independently verifiable, but a ledger entry is an assurance about a transaction, not a lock on a file, and it does not relieve the surrounding application and storage of their own obligations [1], [2].

A second, older difficulty is fragmentation. A person's history is scattered across the clinics and hospitals that happened to treat them, and the clinician at the point of care rarely holds all of it. Any system that sets out to improve this exchange has to hold two goals in tension at once: make a record reachable by the providers who legitimately need it, and keep it unreachable by everyone else [3], [4].

Smart contracts offer a natural way to write access rules as transactions, and a ledger offers a natural place to keep an immutable history of how those rules changed and whether a file still matches its recorded fingerprint. Prior work has applied both ideas to medical-record sharing and to patient-controlled access [5], [6]. What such designs cannot do on their own is close the gap between recording a permission and enforcing decryption: approving access on-chain says nothing about who holds the key to the ciphertext sitting in some bucket elsewhere, and the backend that bridges the two is where the real trust ends up concentrated.

MediVault takes the hybrid position that follows from this observation. Consent decisions and integrity evidence live on the chain; the encrypted clinical files live off it. The contract is consulted before the backend releases anything, so permission is checked rather than assumed, and the parts of the system are kept apart by design ciphertext in object storage, structured metadata in a relational database, and only digests and consent events on Ethereum. We describe the prototype together with the trust assumptions it actually makes, rather than the ones a diagram might imply.

This paper makes three contributions. First, a consent workflow in which grants and revocations are recorded by a smart contract and verified before a record is released, so a revoked permission stops the next retrieval through the system even though it cannot recall a copy already taken. Second, a storage split that keeps AES-256-encrypted files in Amazon S3, their metadata in PostgreSQL, and only record digests and consent events on Ethereum, so the ledger never carries readable clinical content. Third, a working prototype spanning record handling, prescription signing and emergency access, presented alongside an

explicit account of what a backend-held master key does and does not protect.

2. LITERATURE SURVEY

Research on blockchain-based health records has circled three concerns keeping records intact, governing who may read them, and moving them between organisations. The work below is grouped by where its emphasis falls.

2.1 Secure Record Sharing on a Ledger

Yao et al. describe a sharing scheme in which the body of a medical record stays off-chain while the rules that govern access to it are carried by blockchain machinery [1]. Vardhini et al. build a smart-contract framework for electronic medical records, and Mettler's earlier survey set out the general case for applying blockchain to health-record management [2], [3]. Read together, these point to the lesson we return to throughout: the ledger and the services arranged around it have to be examined as one system, because the guarantees of one do not transfer to the other.

2.2 Patient-Centred Access and Reviews

Khemissa and Mourad put the patient at the centre, giving the record's owner a direct role in deciding who may read it [4]. Broader surveys map the surrounding terrain: Haddad et al. review record-management systems that combine artificial intelligence and blockchain and catalogue their privacy, security and scalability concerns [5], while Mamun et al. survey blockchain-based EHR management and note how many proposals stall at the implementation and evaluation stage [6].

2.3 Smart-Contract Workflows and Prototypes

Several named systems differ in architecture while sharing a concern with controlled access. Das et al.'s ReChain and Daraghmi et al.'s MedChain foreground permission and request handling [7], [8]; Alqahtani et al.'s MedEHR is a further ledger-based approach [9]. Alongside these sit a secure-records prototype [10], a documentation framework [11], the IntelliHealth decentralised system [12] and a hospital-focused record-management design [13], with Wang and Zhou and Saratkar et al. providing review coverage of the area [14], [15].

2.4 The Key-Management Gap

One question runs underneath all of this and is seldom answered directly: once access has been approved, who holds the key that actually decrypts the record, and what happens if that holder is compromised? MediVault does not solve this, but it refuses to hide it. Its backend decrypts under a single master key, which means the contract's

permission check gates release without independently enforcing a cryptographic restriction on decryption. We make that boundary explicit because it shapes the security discussion in Section 7.

3. THREAT MODEL AND DESIGN GOALS

We state the trust boundaries plainly, because the value of the prototype lies as much in what it does not claim as in what it does. The blockchain is trusted for ordering and for the immutability of what reaches it, and the Ethereum signatures it carries are assumed unforgeable without the corresponding wallet key. The backend is trusted, and that is the uncomfortable part: it holds the master encryption key, so a compromise of the server or the key exposes more than any single account should. Object storage is assumed to hold only ciphertext, and the relational database is assumed to be protected by its own access controls. Network-level denial of service and traffic analysis are out of scope.

The design pursues four properties.

- 1) Confidentiality: clinical files are unreadable to anyone who obtains only the stored ciphertext, and no readable medical content is written to the ledger.
- 2) Integrity: a stored file altered after upload is detectable, because its recomputed digest will no longer match the value recorded on-chain.
- 3) Accountable access: every grant, revocation and emergency retrieval is recorded as a transaction, so access decisions can be reviewed after the fact rather than taken on trust.
- 4) Attributable prescriptions: a prescription carries a signature that ties it to a particular wallet, so a pharmacy can confirm the signer and notice any change to the signed content.

4. SYSTEM DESIGN

MediVault is organised around four aims keep stored files from disclosure, detect changes to stored content, verify the signer of a prescription, and make access decisions reviewable. The workflow divides into the components below.

4.1 Registration and Authentication

Patients, doctors, pharmacies and administrators each register with email credentials. The backend validates a login and issues a JSON Web Token that authorises the API calls that follow; every account carries a unique identifier and a role that fixes which actions it may take. Providers are not trusted on sign-up alone — a doctor or pharmacy account must be verified by an administrator before it can touch patient records.

4.2 Consent and Access Control

A patient grants or withdraws a provider's permission through the interface, and each change is written to the Solidity contract. The Dynamic Consent Wallet gives the patient a single view of which grants are live and which have lapsed. Before the backend returns any file it consults the contract and proceeds only against a valid permission. Revocation is honest about its reach: it stops the next authorised retrieval through this workflow, but it cannot reach into a provider's own machine and erase a copy already taken.

4.3 Encryption and Off-Chain Storage

Scans, reports and other clinical files never go onto the chain. The backend encrypts each one with AES-256 and uploads the ciphertext to Amazon S3, while the structured fields that describe it stay in PostgreSQL. In the present prototype a single master key, supplied through an environment variable, drives both encryption and decryption. The arrangement is easy to run and easy to reason about, and it concentrates the system's risk squarely in the backend and that one key a trade-off we name here and return to in Section 7.

4.4 Hashing and Integrity Verification

For every encrypted file the system computes a SHA-256 digest and records it on-chain. On retrieval the digest is computed afresh and compared with the stored value; a mismatch says that the file on disk is no longer the one the ledger entry describes. The check speaks only to whether the stored bytes changed it makes no claim about whether the original clinical content was ever correct.

4.5 On-Chain Layer

The contracts are written in Solidity and compiled and tested with Hardhat against Ethereum. They hold consent changes, file digests, prescription-verification data and emergency-access events. Readable medical content is kept off the ledger entirely, and patients are referenced by identifier rather than by name.

4.6 Prescription Signing and Verification

A doctor signs a prescription with an Ethereum wallet, the signature binding to a hash of the prescription under ECDSA. A pharmacy can then confirm which wallet signed and detect any later change to the signed content. What this establishes is cryptographic provenance, not clinical judgement: it says a particular wallet signed a particular text, not that the prescription was the right one to write.

4.7 Emergency Access

When a patient is in no condition to grant consent, a doctor may invoke a break-glass path. It is restricted to the doctor role and is meant to surface only what an emergency actually needs allergies, blood group, current medication for a bounded interval, with each use written to the chain for later review. The guarantees here rest entirely on correct role enforcement and a correctly configured time limit.

4.8 Guardian-Based Key Recovery (Proposed)

The recovery design we sketch would let a patient nominate three guardians whose joint approval is required before a replacement key is issued. It is a proposal only: it has not been implemented or tested, and nothing in the prototype should be read as providing it.

5. SYSTEM ARCHITECTURE

Fig -1 arranges the system into five layers presentation, backend, security, blockchain and storage and the description below follows a request down through them.

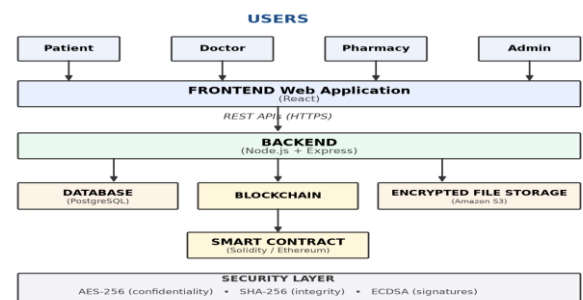


Fig -1: Architecture of the proposed MediVault system.

The presentation layer is a set of React portals, one each for patients, doctors, pharmacies and administrators. Each exposes only the actions its role is entitled to; no portal grants every user the same reach over records.

The backend is a Node.js and Express service that fields API calls, handles authentication and consent checks, performs the cryptographic work, and talks to PostgreSQL, Amazon S3 and the smart contracts. It is the coordinator, and it releases a file only after the consent check has passed.

The security layer is where AES-256, SHA-256 and ECDSA each do one job: encryption limits who can read a file, hashing exposes a file that has changed, and a signature provides evidence that a given wallet signed a prescription. The blockchain layer records consent decisions and audit events through the contracts. PostgreSQL carries structured information metadata, diagnoses, comments

while the encrypted files, X-rays and MRI scans among them, sit in S3.

6. IMPLEMENTATION

The layers speak to one another over REST APIs exposed by the backend. Fig -2 traces an upload from the initial request, through encryption and off-chain storage, to the point where the file's digest is recorded on-chain.

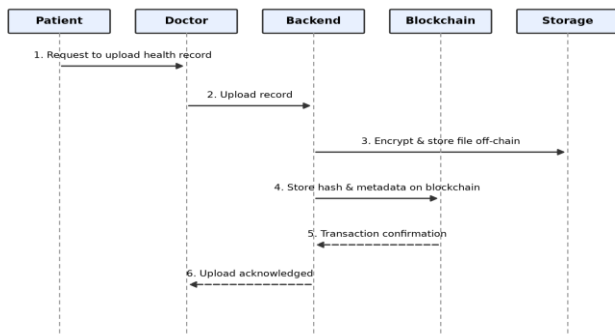


Fig -2: Sequence diagram for health record upload.

The React portals present role-specific views; the Node.js and Express backend wires those portals to PostgreSQL, S3 and the Ethereum contracts and runs the cryptographic and verification steps in between. The Solidity contracts are compiled and tested with Hardhat.

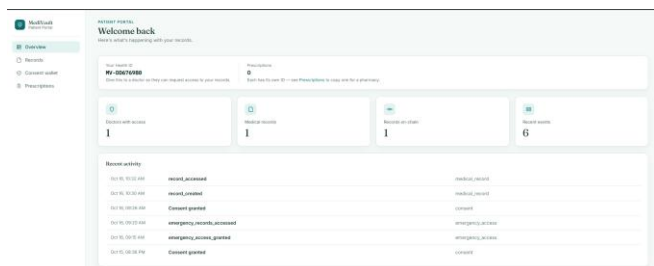


Fig -3: Patient portal of the MediVault prototype.

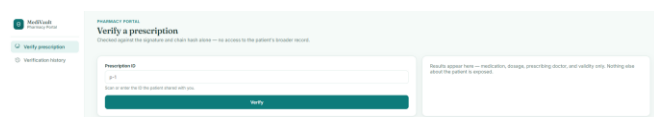


Fig -4: Pharmacy portal of the MediVault prototype.

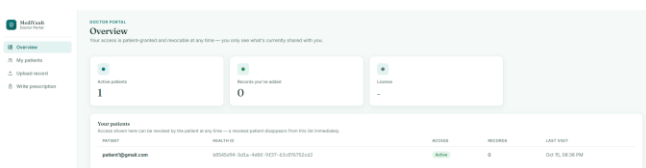


Fig -5: Doctor portal of the MediVault prototype.

Every patient is issued a Unique Health ID at registration. A doctor who wants a record must hold a valid consent for it; grants and revocations are recorded by the contract and shown back to the patient, and the backend checks the contract before it fetches anything.

On upload, the backend encrypts the file, stores the ciphertext in S3 and records its SHA-256 digest on-chain. On retrieval it checks consent, pulls the stored ciphertext and recomputes the digest; if that digest matches the on-chain value it decrypts and returns the file. Because the master key lives in the backend, the whole path still rests on trust in that one service.

Doctors sign prescriptions with their wallets. With the patient's permission a pharmacy locates the relevant prescription by Unique Health ID and verifies its signature before dispensing. Administrators verify provider accounts and respond to reports of suspicious activity, and emergency access follows the restricted, logged path of Section 4.7.

7. RESULTS AND DISCUSSION

We exercised the implemented workflows and recorded their functional status rather than their performance. Table -1 separates the functions that are built and tested from guardian-based recovery, which remains a design. No measurements of response time, throughput or transaction cost were taken, so what follows describes function, not performance or scalability.

Table -1: Functional Status of the MediVault Prototype

Function	Status	Remarks
Registration and login (JWT)	Implemented, tested	Roles: patient, doctor, pharmacy, admin
Record upload (AES-256, S3)	Implemented, tested	Metadata kept in PostgreSQL
Record retrieval after consent check	Implemented, tested	Contract decides, backend releases
Consent grant and revoke	Implemented, tested	Events written to the contract
Dynamic Consent Wallet	Implemented, tested	Patient-side view of all grants
Prescription signing and verification	Implemented, tested	Ethereum wallet signature (ECDSA)
Emergency access (doctor only)	Implemented, tested	Time-limited, essential data, logged
Guardian-based key recovery	Proposed design	Three guardians; not yet built

7.1 Security Analysis

Keeping readable clinical content off the ledger limits what is exposed on-chain, and the recorded digest gives a later reader something to check against. An attacker who comes away with only the S3 ciphertext holds an encrypted file; a file that has been altered should fail the digest comparison, and a tampered prescription should fail signature verification. None of this neutralises the harder cases: a compromised backend, an exposed master key, or an authorised account that is simply misused.

7.2 Limitations

The prototype's limits are real and we state them without softening. Decryption runs on a single master key, so a compromise of the server or the key can expose many records at once. Per-record keys wrapped for individual doctors, and decryption on the client, are not implemented, which is why contract-based consent governs release but does not independently enforce a decryption right. Revocation cannot unsee what has already been viewed or copied. The data in PostgreSQL is only as protected as the database's own access controls make it. Guardian recovery is unbuilt, and no performance evaluation has been carried out.

8. CONCLUSION

MediVault pairs Ethereum smart contracts, which carry consent and audit events, with encrypted files in S3 and structured metadata in PostgreSQL. The working prototype covers registration and login, record upload and retrieval, consent grant and revocation, the Dynamic Consent Wallet, prescription signing and verification, and doctor-only emergency access. Its central limitation is the server-held master key: the ledger's permission checks do not yet give independent cryptographic control over decryption. The right way to read the system is as a functional prototype, not a production-ready or performance-validated platform.

ACKNOWLEDGEMENT

The authors thank the Department of Computer Science and Engineering, Sir M. Visvesvaraya Institute of Technology, Bengaluru, for its facilities and academic support, and the faculty mentors and colleagues whose feedback shaped the work.

REFERENCES

- 1) Y. Yao, X. Cen, Y. Liu, J. Yuan, and Y. Ye, "Blockchain-based electronic medical record security sharing scheme," in Proc. 45th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), 2023, pp. 1–4, doi: 10.1109/EMBC40787.2023.10340218.
- 2) B. Vardhini, S. N. Dass, R. Sahana, and R. Chinnaiyan, "A blockchain-based electronic medical health records framework using smart contracts," in Proc. Int. Conf. Comput. Commun. Informatics (ICCCI), 2021, pp. 1–4, doi: 10.1109/ICCCI50826.2021.9402689.
- 3) M. Mettler, "Blockchain technology in healthcare: the revolution starts here," in Proc. IEEE Int. Conf. Biomed. Health Informatics (BHI), 2016, pp. 25–28.
- 4) S. S. Khemissa and H. T. Mourad, "Blockchain-based user-centric electronic health record management system," in Proc. IEEE Int. Conf. Blockchain Distrib. Syst. Secur. (ICBDS), 2022, doi: 10.1109/ICBDS53701.2022.9935834.
- 5) A. Haddad, M. H. Habaebi, M. R. Islam, N. F. Hasbullah, and S. A. Zabidi, "Systematic review on AI-blockchain based e-healthcare records management systems," IEEE Access, vol. 10, pp. 94583–94615, 2022, doi: 10.1109/ACCESS.2022.3201878.
- 6) A. A. Mamun, S. Azam, and C. Gritti, "Blockchain-based electronic health records management: a comprehensive review and future research direction," IEEE Access, vol. 10, pp. 5768–5789, 2022, doi: 10.1109/ACCESS.2022.3141079.
- 7) A. K. Das, M. Wazid, and N. Kumar, "ReChain: a secured blockchain-based digital medical health record management system," in Proc. IEEE Int. Conf. Consum. Electron. (ICCE), 2023.
- 8) E. Y. Daraghmi, Y. A. Daraghmi, and S. M. Yuan, "MedChain: a design of blockchain-based system for medical records access and permissions management," IEEE Access, vol. 7, 2019.
- 9) A. S. Alqahtani, M. A. AlSalman, and A. A. Alshammari, "MedEHR: electronic health record using blockchain," in Proc. 3rd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS), 2022, pp. 1–6.
- 10) A. Shaout, R. R. Honnalli, K. Trivedi, and V. R. Pulakannti, "Health care blockchain: a prototype for secure medical records using blockchain," in Proc. 2nd Int. Conf. Comput. Intell. Approaches Appl. (ICCIAA), 2026.
- 11) S. Agnal, V. Raj, C. K. Dinesh Kumar, and G. Ashwin Ulagappan, "Blockchain based electronic medical health records framework," in Proc. Int. Conf. Comput. Commun. Technol. (ICCCT), 2025, pp. 1075–1079, doi: 10.1109/ICCCT63501.2025.11019651.
- 12) S. Parshionkar, A. Verulkar, and A. Katkar, "IntelliHealth: a secured decentralized electronic health record system using blockchain," in Proc. IEEE Int. Conf. Blockchain Distrib. Syst. Secur. (ICBDS), 2024.

- 13) A. Vernekar, A. Sakhare, P. Bhapkar, S. Jadhav, and R. B. Adhao, "Blockchain based record management system in hospitals," in Proc. 4th Int. Conf. Innovative Trends Inf. Technol. (ICITIIT), 2023.
- 14) M. Han Wang and R. Zhou, "The application of blockchain to electronic health record systems: a review," in Proc. Int. Conf. Inf. Technol. Biomed. Eng. (ICITBE), 2021, doi: 10.1109/ICITBE54178.2021.00092.
- 15) S. Saratkar, T. Thute, M. Langote, R. Raut, A. Chaudhari, and G. Thakre, "Enhancing healthcare: incorporating blockchain into electronic health records," in Proc. 7th Int. Conf. Inventive Comput. Technol. (ICICT), 2024.