A Hybrid Ethereum-Based Architecture for Secure Electronic Health Records: Consent, Integrity Anchoring and Auditable Access

Rodica Doina Zmaranda, Attila-Imre Kovacs, Daniela Elena Popescu, Alexandrina Mirela Pater
Department of Computer Science and Information Technology, University of Oradea, Oradea, Romania

***Abstract*—Securing electronic health records (EHR) requires strong guarantees for confidentiality, integrity, access control, and auditability. Traditional centralized architectures rely on database-level protection and internal logging, which remain vulnerable to insider misuse and undetected data modification. This study proposes a practical hybrid architecture in which medical content is stored encrypted off-chain, while blockchain is used selectively as a governance and evidence layer. An Ethereum-based prototype was designed and implemented to support integrity anchoring of medical documents, patient-controlled consent management, and immutable audit trails for critical actions. In the implemented solution, the actual medical content is not stored on-chain. Instead, the blockchain stores only document-related metadata, cryptographic hashes, document references, and access-control information, while the sensitive medical data remains encrypted and stored off-chain. This design supports GDPR-oriented data minimization, since the immutable blockchain layer does not contain raw medical records or directly identifiable medical content. The prototype separates confidentiality from blockchain immutability. Medical document confidentiality is handled at the application and off-chain storage level, while the blockchain is used for integrity verification, consent management, and auditability. Encryption keys are not stored on-chain, which prevents the blockchain layer from becoming a repository of sensitive or directly exploitable medical information. Security mechanisms are integrated directly into application flows, including hash-based tamper detection and on-chain verification of access rights. The prototype is evaluated through realistic operational scenarios, analyzing security properties, performance, and transaction cost implications. Results show that, relative to a DB-only baseline, the hybrid approach provides structurally stronger support for integrity verification, traceability, and accountability without exposing sensitive medical data on-chain. The study also highlights practical limitations related to latency and costs in public blockchain environments, supporting a selective on-chain design focused on high-value operations.**

Keywords*—Electronic Health Records (EHR); blockchain, ethereum; smart contracts; access control; consent management; auditable access

I. Introduction

Currently, the digitalization of the medical field through EHR is no longer just a problem of computerization, but one of data architecture and governance. In this context, the benefits of EHRs are real, but they are conditioned by the ability of the systems to securely manage sensitive data, to control access in a transparent way and to provide traceability of the actions performed on the data.

Blockchain represents a technology that proposes a distributed ledger in which transactions are recorded in a verifiable form, and the history is difficult to alter without the consensus of the network [1]. The concept was initially popularized by Bitcoin as a mechanism for maintaining a ledger resistant to unauthorized changes in the absence of a central authority [2]. Later, platforms such as Ethereum extended this paradigm by introducing smart contracts – programs that run on the blockchain and can implement business rules and access mechanisms in an automated and transparent way [3].

This study aims to analyze how blockchain technology can support the security and management of electronic health records (EHR), in a practical solution in which blockchain is used as a governance, evidence, and audit layer. From an implementation perspective, each document is registered through the smart contract by storing its hash and associated reference, while a corresponding event is emitted to create an auditable proof of registration. Access control is handled through explicit grant and revoke operations, which update the on-chain permission state and emit audit events for traceability. When a document is requested, the application verifies the current access permission through the smart contract before allowing the off-chain encrypted content to be retrieved and displayed. The approach was supported by the implementation of a functional prototype with a hybrid architecture (web application developed in Next.js, off-chain storage via Prisma/DB and on-chain component on Ethereum in a local test environment), used as a case study to practically validate the basic hypothesis: medical data (content) must be stored off-chain, and blockchain must be used selectively for integrity, consent and traceability, without exposing the medical content. Choosing Ethereum allows for the analysis of practical advantages and limitations of a blockchain solution, based on different criteria such as transaction costs, performance, and considerations regarding on-chain vs. off-chain data storage. The main contributions of this work are as follows:

1) We propose a hybrid Ethereum-based architecture for electronic health records (EHR) that separates sensitive medical data storage from the trust and verification layer. In this architecture, medical data is stored off-chain, while Ethereum is used for governance, integrity anchoring, consent management, and auditability.

This work was funded by the University of Oradea, Romania

2) We introduce a consent-driven access control model implemented through smart contracts, enabling patient-centric authorization workflows across multiple stakeholders. The prototype supports explicit access granting and revocation operations, and each relevant action is recorded through blockchain events to provide an auditable history.

3) We provide a practical design for integrity anchoring of medical records using cryptographic hashes stored on-chain. Each medical document can be linked to an on-chain hash, allowing later verification that the off-chain content has not been modified, while avoiding the exposure of sensitive medical data on the blockchain.

4) Unlike prior work that focuses mainly on permissioned blockchain environments such as Hyperledger Fabric, our approach evaluates the feasibility of using Ethereum as a deployment environment. The work includes a cost-oriented analysis based on gas consumption and varying gas price assumptions, highlighting the economic constraints of public blockchain deployment.

5) We present a prototype implementation and evaluate its performance characteristics in a local Ethereum-based testing environment. The evaluation highlights the difference between local testing conditions, where transaction execution is fast and cost-free, and public blockchain deployment, where latency, gas costs, and scalability constraints must be considered.

6) We discuss the architectural implications of key management and off-chain encryption for secure real-world deployment. In the implemented prototype, encryption and sensitive data handling are kept outside the blockchain layer, while the blockchain stores only hashes, references, permissions, and audit-relevant events. This separation reduces the exposure of sensitive medical information and supports a more GDPR-aligned system design.

The study is structured as follows: Section II presents the related work and the implementation gap that motivates the proposed architecture; Section III presents the architecture and security requirements of the Blockchain (Ethereum)-based EHR prototype; Section IV describes the security issues and their integration into the application functional flows; furthermore, a comprehensive analysis and discussions regarding advantages and limitations based on several criteria such as security, performance, scalability and costs, were presented in Section V. Finally, some conclusions were drawn.

II. Related Work and Identified Research Gap

Existing research on blockchain-based EHR systems can be grouped into three main directions: conceptual frameworks and literature reviews, prototype-oriented implementations, and studies focused on the trade-offs between on-chain and off-chain storage. The first group, represented by surveys and framework papers [4]-[9], confirms the relevance of blockchain for healthcare interoperability, privacy and auditability, but most contributions remain at an architectural or conceptual level and do not describe an executable application flow that combines encrypted off-chain storage, smart-contract consent and hash-based integrity verification in a single prototype.

A second group contains practical implementations, including solutions based on permissioned environments such as Hyperledger Fabric [10] or more general blockchain-based record-management prototypes [11]. These works are valuable because they show that blockchain can be integrated with healthcare workflows, but they often assume a controlled consortium setting, focus mainly on record registration, or do not explicitly analyze the cost implications of Ethereum-style public deployment. As a result, the specific operational tension between public-chain transparency, transaction fees, medical confidentiality, and GDPR-oriented data minimization remains insufficiently addressed.

The third direction discusses the general storage trade-off between keeping data on-chain and storing only proofs or references off-chain [12]. This literature motivates the hybrid design adopted in this study. However, the identified limitation is that the storage trade-off is frequently discussed independently from concrete EHR consent flows. The proposed prototype addresses this gap by combining three mechanisms in the same implemented case study: encrypted off-chain medical documents, on-chain consent state through grant/revoke operations, and on-chain integrity anchoring through cryptographic hashes and audit events.

Therefore, the purpose of this work is not to replace existing EHR platforms or to claim production-level scalability from a local prototype, but to validate a concrete hybrid security pattern and to analyze its practical implications. The novelty of the study lies in the integrated implementation and evaluation of consent, integrity anchoring, and auditable access in an Ethereum-compatible setting, while explicitly keeping sensitive medical content outside the blockchain layer.

III. Case Study – Blockchain (Ethereum) - Based EHR Prototype

A. Prototype Architecture

To analyze and evaluate a blockchain-based EHR solution, a functional Ethereum-based EHR prototype was developed in a local environment to analyze, in a reproducible way, how security advantages such as integrity, auditability, and consent-based access control can be obtained, as well as investigating the problems that such implementations raise from a practical implementation perspective.

The approach, based on a realistic design, tries to reduce exposure and avoid storing clinical content in raw form on a replicated ledger; for this reason, the prototype uses a hybrid architecture: off-chain for content (document content + operational metadata - local database via Prisma ORM [13]) and on-chain for governance and integrity (access rules, integrity proofs - hashes and audit trails - events). This pattern, frequently recommended in healthcare literature, balances confidentiality with verifiability [14].

The prototype is built around three functional modules, visible in the application interface as separate sections and reflected in the overall architecture presented in Fig. 1, which illustrates the explicit separation between the off-chain and on-chain areas as a main security requirement of the project:

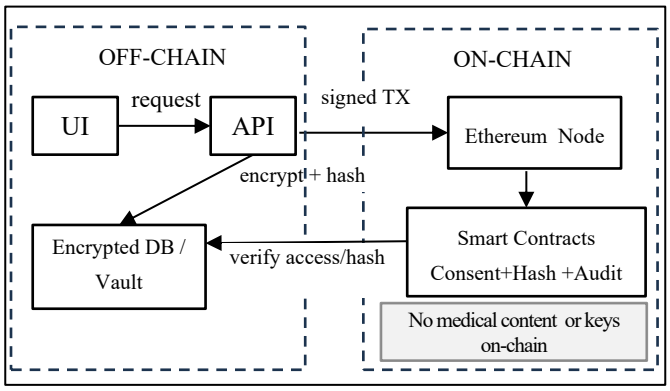

Fig. 1. Overall architecture with off-chain/on-chain storage.

- Document Vault (off-chain): upload, encrypted storage, metadata, indexing, document listing.
- Consent Manager (on-chain + application): grant/revoke access for doctors, with verifiable status on-chain.
- Audit Explorer (on-chain + application aggregation): visualization of events issued by smart contracts (registered document, granted/revoked access), as an "append-only" chronological log.

From the implementation perspective, the prototype is composed of several components. The web client (Next.js UI) is represented by an interface logically separated by roles (patient/doctor). UI offers: uploading documents (patient), listing own documents (patient), granting/revoking access to a doctor (patient), accessing allowed documents (doctor), viewing the audit (both roles, with different levels). Backend (Next.js server: API routes/server actions) performs security logic such as input validations, encryption/decryption, hash calculation, DB interaction (Prisma), calls to smart contracts (local Ethereum). Local blockchain (Ethereum in our test environment) implements contracts for permissions (grant/revoke), recording hashes, issuing audit events. Off-chain storage (DB + Prisma) keeps encrypted content, keeps metadata necessary for searching/listing, and keeps the link between an off-chain document and its on-chain proof (e.g. docId + hash + txHash / blockNumber).

B. Functional and Security Requirements

The functional requirements of the prototype include the following: register users with distinct roles (e.g. patient, doctor); upload medical documents (e.g. results, reports, medical letters) and associate them with the patient; grant access by the patient to a doctor (grant), with the possibility of revocation; consult documents by the doctor only if there is active permission; maintain an audit trail for critical actions (upload, grant, revoke, consult).

Security requirements (what the system must protect) are represented by:

- Confidentiality - medical content must not be exposed publicly; access must be conditioned and controlled.
- Integrity - any unauthorized modification of documents stored off-chain must be detected.
- Auditability / non-repudiation - relevant actions must be traceable (who granted access, when it was revoked, etc.).
- Data minimization - only strictly necessary data is stored on the blockchain (hashes, technical identifiers, events).
- Separation of responsibilities - neither the blockchain nor the database are sufficient on their own; security is achieved end-to-end (application + DB + contracts + keys).

1. User
id (UUID)
role (PATIENT / DOCTOR)
displayName
ethAddress (0x… associated with the wallet account)
createdAt
2. Document
id (UUID – internal docId)
ownerUserId (pacient)
type (ex: analyses, imaging, letter, recipe)
createdAt (moment upload)
ciphertext (blob / encrypted file) or internal pointer if storing in filesystem
cipherAlgo (ex: AES-256-GCM)
hashAlgo (ex: SHA-256)
contentHash (consistently calculated hash)
sizeBytes
anchorTxHash (tx on blockchain)
anchorBlockNo
status (ACTIVE / TAMPERED / REVOKED_BY_OWNER)
3. PermissionCache (optional, for UI performance)
patientUserId
doctorUserId
statusLocal (informational mirror)
lastCheckedAt

Fig. 2. Entity conceptual model.

The off-chain component is represented by a local database accessed through Prisma ORM. The reason for this choice is twofold: from a performance perspective, it ensures efficiency for searches, listings, and file/metadata storage, and from a security perspective it ensures increased confidentiality, with the medical content remaining in a controllable area, where classic policies can be applied (backup, encryption, access control, retention). In the prototype, the off-chain modelling aims to clearly separate the following elements:

- Application identity data: users, roles, mapping to an Ethereum address (for signing transactions).
- Medical documents: content (stored encrypted or in protected form), metadata (type, date, issuer).
- Access bindings: a local representation of permissions (optional), used for UX and caching, but the final decision is made based on the on-chain state.
- Application audit: useful internal logs (e.g., errors, requests), separate from the on-chain audit (which covers critical actions).

The on-chain model uses Ethereum/Ganache for governance, permissions, and audit; the component is represented by a set of smart contracts implemented in Solidity [15] and run on a local Ethereum network (Ganache [16]). Their

role is to be a verifiable registry for: access permissions (grant/revoke – formal decision, signed by the patient through a transaction); integrity proofs (the document hash, anchored in the registry); audit events – critical actions are reflected through events, so that the audit can be rebuilt.

The design of the contracts follows a strict principle: no medical content or explicit clinical information is stored on-chain. The contract only contains elements such as technical identifiers, such as document IDs, hashes, Ethereum addresses, logical timestamps, and events. It can be seen that the key to the approach is minimization: in the DB, only what is necessary for operation and security is kept, and in the blockchain, only the evidence/documents and rules are kept.

IV. Security Issues and Implementation into Functional Flows

Both on-chain and off-chain layers require specific approaches from the security point of view for implementation:

A. Security in the Off-Chain Data Model (Database)

In the prototype, the DB has the role of enforcing a series of security invariants: the medical content is stored encrypted; there is no "direct read" of the content without on-chain verification; the document is associated with a hash and with the reference to the on-chain proof; permissions are doubly conditional: on-chain state + integrity validation.

A conceptual model of tables/entities (compatible with our Next.js + Prisma implementation) is presented in Fig. 2.

The model clearly separates the identity and roles of users, the protected medical content, and the metadata necessary for operation, respectively, the verifiable link to the on-chain proofs. Thus, the medical content remains off-chain and encrypted (confidentiality), the integrity is verifiable through anchored hashes (integrity), and critical actions are linked to on-chain transactions and events (auditability and traceability).

It is noted that the User entity contains only the minimum information necessary for operation (id, role, display name) and a blockchain layer connection field (ethAddress), used for mapping between the application account and the address that signs transactions. Choosing this field allows auditability and non-repudiation for critical operations (grant/revoke/register), while keeping the detailed clinical identity off-chain (data minimization principle).

The Document entity is designed so that the medical content remains off-chain, protected: the ciphertext field (or an internal pointer to the encrypted file) allows the data to be stored in encrypted form, and the cipherAlgo and hashAlgo fields explicitly document the algorithms used (e.g., AES-256-GCM, SHA-256), increasing reproducibility and verifiability. The contentHash field represents the consistently calculated fingerprint of the content and is the key to the integrity mechanism: upon access, the recalculated hash from the DB is compared with the anchored value. The anchorTxHash and anchorBlockNo fields directly link the off-chain document to the on-chain transaction in which the proof was recorded, allowing independent verification of the moment and existence of the proof.

The status field (ACTIVE/TAMPERED/REVOKED etc.) is introduced to reflect the security state observed by the application (e.g. hash mismatch → TAMPERED) and to prevent displaying a suspicious document.

The PermissionCache entity is optional and has a strictly UI performance role: it can locally store the result of the last on-chain check (per patient–doctor–document), together with lastCheckedAt, to reduce repeated calls to the contract in fast navigation scenarios. Importantly, this entity is not the source of truth for permissions; the final access decision remains on-chain (hasAccess), and the cache is only an information mechanism that can be revalidated/invalidated based on events (AccessGranted/AccessRevoked) or upon expiration.

B. Security and On-Chain Layer: Smart Contracts and Executable Rules (consent + Audit)

In the prototype, the blockchain is used strictly as a governance and evidence layer. For clarity and a small attack surface, it is recommended to use a two-contract design.

Contract 1 – ConsentRegistry (used for permissions) has the following responsibilities: stores the state of permissions: patient → doctor → allowed/denied; optional: includes expiration (timestamp) and a minimal "scope" (document types). Typical functions are the following: grantAccess (address doctor, uint64 expiresAt) – called by the patient; revokeAccess(address doctor) – called by the patient; hasAccess(address patient, address doctor) – view (used by the server). Typical audit events: AccessGranted(patient, doctor, expiresAt); AccessRevoked(patient, doctor).

Contract 2 – DocumentAnchor (used for hashes), has the following responsibilities: records the proof of integrity for documents: (patient, docId) → hash; emits a DocumentAnchored event on each record. Typical functions: anchorDocument(bytes32 docId, bytes32 contentHash) – called by the patient (or by an authorized "uploader" on their behalf); getHash(address patient, bytes32 docId) – view. Typical events: DocumentAnchored(patient, docId, contentHash).

A justified design decision in this regard is that the events are intentionally minimal: they do not include diagnoses, results, document names, or other sensitive metadata, containing only technical identifiers and actions (grant/revoke/anchor).

C. Integrating Security Issues into Implementation Flows

One of the challenges of the implementation was to integrate security elements as smoothly as possible into the functional flows of the application.

Flow A – Document upload, off-chain protection, and on-chain anchoring of proof of integrity is shown in Fig. 3.

- Step 1 (UI – patient): patient uploads a .pdf file, e.g., analyses
- Step 2 (server): backend validates file type and size; generates a per-document symmetric key; encrypts the content (e.g., AES-256-GCM); calculates a hash (e.g., SHA-256) over encrypted content (or over original content – it is important to be consistent); saves in DB: ciphertext, contentHash, docId, metadata.

- Step 3 (on-chain): application sends transaction anchorDocument(docId, contentHash) to contract; contract records hash and issues DocumentAnchored.
- Step 4 (DB completion): DB is updated with anchorTxHash and anchorBlockNo.

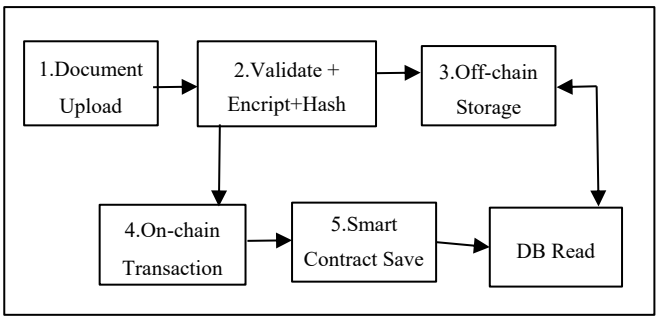

Fig. 3. Document upload, off-chain protection, and on-chain anchoring of proof of integrity.

Flow B – Granting access and document consultation sequence (permission + integrity check) is shown in Fig. 4.

- Step 1 (UI – patient): patient selects a doctor (e.g. 0xB2...) and sets duration (e.g. 7 days).
- Step 2 (on-chain): transaction grantAccess(doctor, expiresAt) is sent; contract updates permission state and issues AccessGranted.
- Step 3 (UI/Audit Explorer): event appears in log: AccessGranted – patient X → doctor Y.

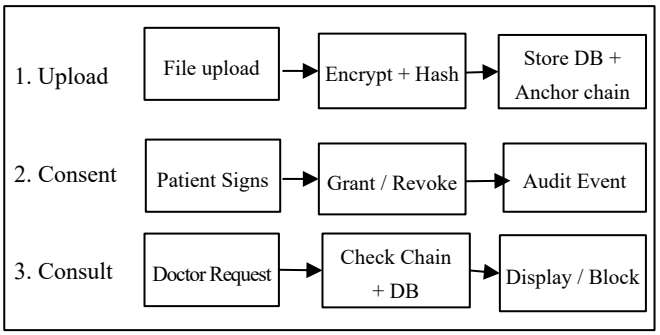

Fig. 4. Granting access and document consultation sequence (permission + integrity check).

Thus, in the prototype, a grant is a cryptographically signed transaction (patient wallet) and produces an on-chain event, which makes consent easier to prove and harder to challenge later (who, to whom, for what and when).

Flow C – doctor's document consultation (access control + integrity), involves the following steps:

- Step 1 (UI – doctor): the doctor requests the document docId = 7b2f.
- Step 2 (server): the application runs: hasAccess(patient, doctor) - view on-chain → if false, stops; retrieves ciphertext from DB; recalculates the hash on the content (same method as for upload); extracts the on-chain hash from getHash(patient, docId); compares hashes: if they do not match → status = TAMPERED, access blocked + alert; if everything is valid → decrypts and displays the content.

As an observation related to security, it should be mentioned here that the blockchain does not "keep" the document, but acts as a log: any subsequent change becomes detectable. At the time of upload, the document's cryptographic fingerprint (hash, e.g. SHA-256) is calculated, and this value is anchored on-chain (via registerDocument and/or the DocumentRegistered event). Subsequently, at each consultation, the application recalculates the hash on the document retrieved from off-chain storage and compares it with the anchored hash; if the file has been modified, replaced, or corrupted in the database, the recalculated hash will be different, and the discrepancy becomes an objective indicator of alteration, causing access to be blocked/decryption denied or the document to be marked as invalid.

Flow D – Access Revocation involves the following steps:

- Step 1 (UI – patient): the patient revokes the doctor's access.
- Step 2 (on-chain): a transaction revokeAccess(doctor) is made; an AccessRevoked event is generated.
- Step 3 (practical effect): any further consultations are blocked at Step 2 of scenario C (permission check).

An inherent limitation in this context is that revocation does not "erase" the fact that the doctor has already seen the document. It only stops future access; therefore, auditing and organizational policies remain necessary.

Flow E – Tamper detection: in this scenario, if an attacker (or operational error) modifies the ciphertext in the DB, then:

- upon access, the recalculated hash no longer matches the hash anchored on-chain;
- the application marks the document as TAMPERED;
- blocks decryption and display.

This is exactly where the prototype offers a structural advantage over a traditional DB-only implementation: there is an external, immutable proof of the "original state" (the hash anchored on-chain) that cannot be rewritten by accessing the off-chain infrastructure. In a traditional system, an attacker or a privileged administrator who compromises the DB can, at the same time, modify the document, update the metadata (including any hash stored internally), and adjust the application logs to reduce the chances of detection.

In the prototype implementation approach, even if the file in the DB is replaced or edited, the hash recalculated upon consultation will no longer coincide with the value recorded on-chain, and the discrepancy becomes an objective technical indicator of the alteration (tamper evidence). To "hide" the alteration, the on-chain evidence (the anchored hash) would also have to be modified, which is not possible retroactively without a new transaction visible in the history and, implicitly, without leaving audit trails (timestamp/block number). Therefore, the blockchain transforms integrity from a trust-based property into a verification-based property, where the document is accepted

only if it passes cryptographic validation reported to the on-chain anchor.

D. Privacy and GDPR Compliance

The proposed system operates in the context of electronic health records (EHR), which are classified as special category data under Article 9 of the General Data Protection Regulation (GDPR) [17]. As such, privacy protection and data minimization are fundamental requirements of the architecture.

A key challenge arises from the conflict between blockchain immutability and the "right to erasure" under Article 17 GDPR [17]. The proposed architecture mitigates this issue by ensuring that no raw personal data or medical records are stored directly on-chain. Instead, the blockchain stores only cryptographic hashes, document references, access permissions, and audit-relevant events, while the actual medical content is kept encrypted in off-chain storage. As a result, sensitive off-chain data can be deleted, modified, or made inaccessible when required, without needing to alter immutable blockchain records.

The system also supports a consent-driven authorization model. Patient consent is represented through smart contract-based access control operations, such as granting and revoking access to a medical document. These operations create a verifiable audit trail through blockchain transactions and events, allowing the system to demonstrate when access was granted, revoked, or checked.

Data minimization principles under Article 5 GDPR are addressed by limiting the information stored on-chain to technical verification data [17]. No direct identifiers, raw medical records, diagnoses, treatment details, or other sensitive health information are stored on the blockchain. The on-chain hash is used only as an integrity proof, allowing the application to verify whether the corresponding off-chain document has been altered.

Regarding cross-border data transfer under Article 44 GDPR, the use of public Ethereum infrastructure may introduce additional compliance risks, since blockchain nodes can be distributed across multiple jurisdictions [17]. For a real-world deployment, this risk could be mitigated by using controlled access gateways, EU-based node providers, private or consortium Ethereum-compatible networks, or infrastructure operated by trusted healthcare institutions.

Finally, encryption and key management remain critical for confidentiality. In the prototype, confidentiality is handled outside the blockchain layer: medical documents are stored off-chain in encrypted form, while the blockchain is used for integrity verification, access control, and auditability. Encryption keys are not stored on-chain, reducing the risk that immutable blockchain records could expose sensitive or directly exploitable medical information

E. Key Management and Data Protection

The security of the proposed system depends not only on the smart contract logic, but also on the protection of the off-chain medical data and the cryptographic material used to secure it. Since the blockchain is not used to store medical content directly, the confidentiality of the system relies on the encryption and access-control mechanisms implemented at the application and off-chain storage layers.

In the implemented prototype, medical document confidentiality is handled outside the blockchain layer. The blockchain stores only hashes, references, access permissions, and audit events, while the sensitive document content is stored off-chain. This separation ensures that the blockchain can provide integrity verification and auditability without becoming a repository of confidential medical information.

Encryption keys are not stored on-chain. This is an important design decision because any data written to a blockchain should be considered persistent and difficult or impossible to remove. Keeping keys outside the blockchain prevents unauthorized users from deriving access to medical content directly from blockchain data.

For a production-ready implementation, a more advanced key management model would be required. One possible approach is to encrypt each medical document using a document-specific symmetric key and then protect that key using the public keys of authorized users. Under such a model, only users with the corresponding private keys would be able to decrypt the medical data. However, this introduces additional requirements related to secure private key storage, recovery mechanisms, access revocation, and user usability.

Revocation of access is also an important practical consideration. In the prototype, access revocation is represented at the smart contract level by updating the on-chain permission state. This prevents a previously authorized user from passing future access checks through the application. However, in a real-world system, stronger revocation guarantees may require re-encryption of the off-chain data and redistribution of new keys to the remaining authorized users.

Therefore, the architecture assumes that blockchain-based authorization must be combined with secure off-chain key lifecycle management. Future improvements may include integration with hardware-backed keystores, hardware security modules, decentralized identity frameworks, or institutional key management services suitable for healthcare environments.

For a production deployment, the key lifecycle would need to be specified in more detail. Key rotation could be implemented through envelope encryption, where each document key is wrapped by user or institutional public keys and can be re-wrapped without rewriting the encrypted document itself. Multi-device access would require controlled enrollment of additional user keys, while emergency recovery or break-glass access would require explicit institutional policy, strong authentication, and additional audit events. Institutional delegation and insider threats should be handled through role separation, least privilege, hardware-backed key storage, and periodic access reviews. In practice, these controls would normally be implemented through a healthcare-grade KMS/HSM or institutional identity infrastructure rather than directly through the blockchain.

While the proposed architecture improves integrity, auditability, and transparency, several security considerations must be addressed before real-world deployment.

One potential risk is the use of application-level caching mechanisms for access permissions. If permission data is cached, there is a possibility of stale-state inconsistencies, for example when access has been revoked on-chain but the application still relies on an outdated cached permission state. This risk can be reduced through short cache lifetimes, explicit cache invalidation after grant or revoke operations, and periodic revalidation against the smart contract state.

Another important consideration is the reliance on off-chain storage. Since medical documents are not stored directly on-chain, the system depends on the availability and security of the external storage layer. This introduces trust assumptions regarding storage infrastructure, backup policies, and operational security. However, the impact of this risk is reduced by encrypting the off-chain content and verifying document integrity against the hash anchored on-chain.

The correctness of the smart contract logic is also essential. Functions responsible for document registration, access granting, revocation, and permission checking must be carefully designed to prevent unauthorized access or inconsistent authorization states. Since smart contract transactions are immutable once deployed, contract testing and validation are critical before deployment to a production blockchain environment.

Finally, the overall security of the system depends heavily on cryptographic key protection. If a user's private key or application-level encryption material is compromised, unauthorized access to medical documents may become possible. Therefore, secure key storage, recovery mechanisms, user awareness, and institutional security policies are necessary components of any real-world deployment.

V. Analysis and Discussions

A. Security Analysis and Evaluation

The security analysis based on the prototype case study is structured on the following criteria: confidentiality, integrity, availability, complemented by access control and auditability.

1) Threat model and security invariants: To make the security evaluation more explicit, the prototype can be interpreted under the following threat model: an attacker may attempt to read or modify off-chain stored documents, an unauthorized doctor may attempt to access a patient document without consent, a stale application cache may preserve an outdated permission state, or a privileged infrastructure actor may attempt to alter local database records or logs. The model does not claim to solve denial-of-service attacks, theft of a user's private key, or all smart-contract implementation bugs; these remain deployment risks that require additional operational controls.

The main security invariants enforced by the prototype are: I1 - raw medical content and encryption keys are never written on-chain; I2 - a document is displayed only if hasAccess(patient, doctor) returns an active permission state; I3 - the hash recalculated from the off-chain content must match the hash anchored on-chain before decryption/display; I4 - PermissionCache is never the source of truth and cannot override the smart contract state; I5 - grant, revoke and anchor operations generate auditable events that can be reconstructed independently from the application database.

Against these invariants, the defense mapping is concrete: database tampering is detected by the mismatch between the recalculated hash and the on-chain anchor; unauthorized access is blocked by the hasAccess check; stale-cache risk is mitigated by short cache lifetime, event-based invalidation and revalidation against the contract; modification of centralized logs does not remove the blockchain event history. Smart contract vulnerabilities are reduced by a small contract surface and testable state transitions, but formal verification and third-party audit would be required before production deployment.

Confidentiality is achieved by: off-chain storage of content; encryption at the application level (before DB); decryption conditional on on-chain verification + integrity. The specific blockchain risk (in public networks) is leakage through metadata and address correlations. In the prototype (local), the risk is low, but for a real system, mitigation measures should be mentioned: minimization of events, pseudonymization, avoiding including the on-chain document type, aggregation, and rate-limiting at the UI/API level. However, blockchain solution privacy remains dependent on encryption and key management (off-chain). Blockchain does not guarantee privacy by itself; it only reinforces it through a design that minimizes what is public.

Integrity is strengthened by on-chain anchored hash and verification at each access. This mechanism is simple, cheap to implement, and very strong as "proof": any difference between the stored content and the anchored proof becomes immediately visible at the time of access, without relying exclusively on centrally managed logs. From an integrity perspective, there is a clear advantage of the blockchain solution: integrity becomes verifiable through an external proof (on-chain hash), not just through trust in the administration.

Regarding access control and consent (non-repudiation and accountability), as shown in the prototype, consent is not a setting in the DB, but a verifiable state on-chain. This produces two effects: access becomes independently verifiable (at least at the system level) and actions become traceable and hard to deny (transaction signature). Grant/revoke actions are cryptographically signed transactions (address/key), and the change of permission state is recorded on-chain. This results in an important practical property: stronger non-repudiation for critical actions (who granted/revoked access and when). In addition, the prototype applies "double checking": even if the DB contains the document, actual access is conditional on on-chain permission verification + hash integrity verification. Thus, from an access control perspective, the blockchain solution reduces the risk of "invisible" abuse at the consent level and makes access decisions easier to prove later.

Audit is built on on-chain events (grant/revoke/anchor). The advantage over centralized logs is the append-only nature and transaction signing. Basically, even if the DB is compromised, the main history of critical actions remains reconstructable from the chain: on-chain events (DocumentAnchored, AccessGranted, AccessRevoked, etc.) are append-only and can be used to reconstruct a verifiable timeline of critical actions. Consequently, from the audit perspective, there is a real

auditability gain, especially for critical actions, in the hybrid blockchain model, as the on-chain log is harder to tamper with without a trace.

Because the content is still off-chain, the availability of documents still depends on the DB and the application. Blockchain does not replace storage and does not solve availability on its own. Thus, the hybrid blockchain approach does not eliminate the off-chain dependency: if the DB goes down, the documents are not accessible, even if the evidence is on the chain. Therefore, in a real architecture, availability remains largely a classic problem and relies on classical approaches (backup, replication, monitoring).

2) Qualitative comparison with a DB-only EHR baseline: The baseline considered for the security comparison is a conventional centralized EHR implementation in which the document, the stored hash, permission flags, and audit logs are all maintained inside the same administrative database or infrastructure domain. In such a baseline, a privileged compromise can modify the medical file and also rewrite the associated hash, permission record, or local audit trail. In the proposed hybrid architecture, the attacker would additionally have to change the previously anchored on-chain hash or remove historical grant/revoke events, which is not possible retroactively without creating a new visible transaction. Therefore, the improvement claimed in this study is a structural and verifiability-oriented improvement, not a quantitative throughput comparison against an independently implemented EHR system.

B. Performance and Scalability Analysis

The performance of a hybrid EHR (off-chain + on-chain) cannot be evaluated as a single "response time", because the flows have two distinct components: 1) off-chain operations, which dominate the user experience (upload, read, list), and 2) on-chain operations, which dominate the governance decision (consent, anchor, audit).

1) Measurement methodology: The measurements were performed in the local environment of the prototype, using the following approach: measured end-to-end time for each operation (from the action in the UI to the response received by the user); separation of times by stages, where relevant: 1) server processing (validation + encryption + hash), 2) DB operation (Prisma), 3) blockchain interaction (submit tx + confirmation), 4) access checks (hasAccess + getHash + compare).

For on-chain operations, two different values were measured: t_submit (how long it takes to send the transaction, until you receive txHash); t_confirm (how long it takes until the transaction is mined/confirmed and the event is available). This separation is important because the UX can be designed so that the user does not "wait in a blocking way" for confirmation in some cases (e.g., anchoring can run asynchronously), but for other cases confirmation is critical (e.g., grant/revoke must be confirmed to be considered active).

2) Example measurements on the prototype case-study: Table I summarizes average values measured on a set of common files and on typical scenarios (upload, grant, consultation).

TABLE I. AVERAGE TIMES PER PROTOTYPE (MS) – END-TO-END MEASUREMENT (LOCAL ENVIRONMENT)

Operation	Measurement results		
	Dominant Components	***Average Time (ms)***	***Observations***
Upload document 1 MB (off-chain)	encryption + DB write	~180–320	does not include on-chain confirmation
Hash calculation (SHA-256)1 MB	CPU (server)	~8–15	subdominant to IO
AES-256-GCM 1 MB encryption	CPU (server)	~25–60	depends on implementation
Save DB (Prisma) 1 MB	IO DB	~60–180	depends on local DB
Submit transaction anchor (txHash)	RPC + signing	~120–250	user sees "sent"
Confirm transaction (Ganache)	local mining	~400–900	stable in controlled environment
GrantAccess (submit)	RPC + signing	~120–250	similar to anchor
GrantAccess (confirm)	local mining	~400–900	becomes active after confirmation
Consult document (verify + display) 1 MB	hasAccess + DB read + compare + decrypt	~220–520	includes on-chain checks (view)

The values are indicative for the local environment and serve to highlight where costs arise and which components dominate. From measurements, we can notice that for off-chain operations, the cost is dominated by I/O (DB) and encryption. Hash is fast and does not represent a significant bottleneck. Also, for on-chain operations, confirmation is the most "visible" component if it is put directly into the clinical flow (the user waits). In the local prototype, confirmation is stable; in real public networks, this latency is variable and can increase significantly. Document consultation remains comparable to a traditional system in terms of access time, with a moderate overhead due to checks (hasAccess + getHash + compare), but with a security gain (integrity + verifiable access control).

3) Interpretation of local measurements and deployment limits: The measurements in Table I must be interpreted as prototype-level measurements obtained in a local Ganache environment. They validate the correctness of the implemented flows and identify the dominant components inside the prototype, but they do not prove public Ethereum feasibility. Ganache eliminates variable consensus latency, fee volatility,

and network congestion; therefore, the public deployment discussion in this study is limited to gas-based cost estimation and architectural implications. A real deployment would require additional testing on public test networks, layer-2 networks, or a controlled consortium network.

The prototype was not stress-tested with thousands of patients, concurrent doctors, or very large repositories. In a production architecture, high-frequency actions should remain off-chain, while on-chain operations should be limited to high-value state transitions such as document anchoring and consent updates. Operational resilience would also require transaction queues, retry and idempotency logic for pending blockchain operations, redundant blockchain RPC providers or nodes, database replication and backups, monitoring, and a fail-closed access policy when the system cannot verify current permission or integrity state.

Storage overhead introduced by the hybrid model is limited compared with the medical files themselves. For each document, the additional off-chain fields include an internal document identifier, content hash, hash/encryption algorithm identifiers, initialization vector and authentication tag for encryption, transaction hash, block number and status flags. These artifacts are typically below a few kilobytes per document, while the medical document can be several megabytes. The on-chain footprint is dominated by the stored hash/reference and event data, while detailed audit records and application logs should remain off-chain or be periodically anchored to avoid high costs.

The current evaluation uses fixed representative parameters, such as a 1 MB document and a moderate number of annual grant/revoke/access operations. A complete sensitivity analysis should vary document size, document frequency, transaction frequency, cache lifetime, encryption overhead, number of users, and concurrent requests. This limitation is acknowledged because the present study focuses on validating the security pattern and estimating cost order of magnitude, not on exhaustive capacity planning.

C. Costs Analysis

1) Premises for comparison (annual scenario): The cost estimations were made on a moderate scenario, realistic enough to estimate their order of magnitude. The assumptions on which the analysis was based are presented in Table II.

TABLE II. PREMISES FOR COST ASSESSMENT (ANNUAL SCENARIO, PER PATIENT)

Parameter	Value (scenario)	Observations
New documents / patient / year	10	e.g.: tests, imaging, medical letter
Average document size	1 MB	order of magnitude for common files
Stored volume / patient / year	10 MB	10 doc × 1 MB
Document requests / year	20	doctor requests for access
Access grants / year	4	access grants (consent)
Access revokes / year	4	revoke access

In public Ethereum, each on-chain operation has a transaction cost (fee), calculated by:

$$\text{fee (ETH)} = \text{gasUsed} \times \text{gasPrice} \quad (1)$$

where, gasPrice is in gwei (1 gwei = 10^-9 ETH).

In the local prototype (Ganache), this monetary cost is practically 0, but in a real scenario (public Ethereum) the cost becomes relevant. For example, we use a further estimation valid at present of price_eth = 2,212.07 EUR/ETH. Plausible estimates for minimalist contracts (mapping + events) were the following: document anchor (hash + event): 95,000 gas; grant access (grant + event): 60,000 gas; revoke access (revoke + event): 45,000 gas; log access (optional audit event): 35,000 gas. GasPrice scenarios (so as not to be "cosmetic numbers") could be chosen from the following: 10 gwei (low), 30 gwei (medium), 60 gwei (high). Consequently, the direct formula that we used will be:

$$\text{Cost (EUR)} = \text{gasUsed} \times \text{gasPrice(gwei)} \times 10^{-9} \times \text{price\_eth (EUR/ETH)} \quad (2)$$

2) Costs comparison results: It can be seen from Table III that, in public Ethereum, there are significant cost per access, making a model where "everything is logged on-chain" unsustainable. This leads to the correct conclusion: on public networks, blockchain is suitable for rare high-value operations (anchors, critical consent), not for turning every access into a transaction. Therefore, for real adoption, some architectures preserve integrity/auditability, but drastically reduce the cost. One of the options is represented by off-chain audit + periodic on-chain anchoring or networks (predictable operational cost).

TABLE III. HYBRID EHR COST ESTIMATION ON PUBLIC ETHEREUM (10, 30, 60 GWEI VARIANTS)

Parameter	Costs Public Ethereum /year		
	10 gwei	***30 gwei***	***60 gwei***
10 document anchors/year/pacient	21 EUR	63 EUR	126 EUR
20 document requests / year (audit)/pacient	15,40 EUR	46,40 EUR	92,80 EUR
4 Access grants / year	5,32 EUR	15,92 EUR	31,84 EUR
4 Access revokes / year	4,00 EUR	11,96 EUR	23,92 EUR
TOTAL/pacient/year	**45,72 EUR**	**137,28 EUR**	**274,56 EUR**
TOTAL for 10000 pacients	**457.200 EUR**	**1.372.800 EUR**	**2.745.600 EUR**

VI. CONCLUSION

This study demonstrates that a hybrid blockchain architecture provides a robust framework for Electronic Health Records (EHR), where integrity stands out as the most direct and demonstrable benefit. The developed prototype confirms that blockchain serves as an essential tamper-evident mechanism, making any modification, replacement, or corruption of documents within the database objectively detectable. Through this approach, consent and access control transcend being internally modifiable states; they become a transparent set of actions with verifiable traces, documenting precisely who granted access, to whom, for which document, and when.

Regarding regulatory compliance and data protection, the architecture was designed to support GDPR-oriented requirements by enforcing a strict separation between sensitive medical data and immutable blockchain records. The implemented prototype does not store raw personal information or medical content on-chain. Instead, the blockchain layer stores only cryptographic hashes, document references, access permissions, and audit-relevant events, while the actual medical documents remain encrypted and stored off-chain. This design supports the principle of data minimization and reduces the risk of exposing sensitive health information through immutable blockchain records.

Privacy is therefore not treated as an inherent blockchain feature, but as an end-to-end architectural responsibility. The blockchain is used for integrity verification, consent management, and auditability, while confidentiality is handled at the application and off-chain storage layers. During document retrieval, the application verifies the current on-chain access permission and the integrity of the requested document before allowing the encrypted off-chain content to be accessed and displayed. Encryption keys are not stored on-chain, which prevents the blockchain layer from becoming a source of sensitive or directly exploitable medical information.

The feasibility of this model depends on a selective on-chain design. To mitigate the high costs and variable fees associated with public Ethereum transactions, only the critical minimum—cryptographic evidence, consent updates, and essential events—is recorded on-chain. Detailed auditing and intensive data operations remain off-chain to maintain a controllable cost profile and competitive user experience (UX) performance.

While the current evaluation was conducted in a local Ganache environment, the prototype provides a functional proof of concept for the proposed architecture, including document registration, hash-based integrity anchoring, access granting, access revocation, and permission verification. The local environment was suitable for validating the correctness of the smart contract logic and the interaction between the application layer and the blockchain layer, but it does not fully capture the operational characteristics of a public Ethereum network.

Consequently, the results should be interpreted as a functional proof of concept and as an order-of-magnitude cost and security analysis. They should not be read as evidence that the same performance would be obtained on public Ethereum or under large healthcare workloads without further load testing, operational hardening, and production-grade key management.

Further research will also explore the adoption of permissioned or consortium networks for increased identity control and the integration with medical standards such as FHIR/HL7. Ultimately, this hybrid approach proves that blockchain can effectively serve as a specialized consent and audit layer, seamlessly integrating with existing healthcare infrastructures while ensuring security, accountability, and legal compliance.

REFERENCES

[1] S. Khezr, M. Moniruzzaman, A. Yassine and R. Benlamri, “Blockchain Technology in Healthcare: A Comprehensive Review and Directions for Future Research”, Appl. Sci. 2019, 9(9), 1736; https://doi.org/10.3390/app9091736

[2] Nakamoto, S. “Bitcoin: A Peer-to-Peer Electronic Cash System”. 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] Buterin, V. “Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform”. 2014. [Online]. Available: https://ethereum.org/en/whitepaper/

[4] Y. Han , Y. Zhang and S. H. Vermund, “Blockchain Technology for Electronic Health Records”, Int J Environ Res Public Health. 2022 Nov 24;19(23):15577. doi: 10.3390/ijerph192315577

[5] N. Ettaloui, S. Arezki and T. Gadi, “Blockchain-Based Electronic Health Record: Systematic Literature Review”, 2024, https://doi.org/10.1155/hbe2/4734288

[6] Blockchain for Healthcare, IEEE Xplore Digital Library, November 20, 2025 [Online]. Available: https://innovate.ieee.org/innovation-spotlight/blockchain-for-healthcare/

[7] A. Althaf Ali, M.A. Gunavathie, V. Srinivasan, M. Aruna, R. Chennappan, M. Matheena Securing electronic health records using blockchain-enabled federated learning for IoT-based smart healthcare, Clinical eHealth, Volume 8, December 2025, Pages 125-133

[8] N. Ul. A. Tahir, U. Rashid, H. J. Hadi, N. Ahmad, Y. Cao, M. A. Alshara and Y. Javed, “Blockchain-Based Healthcare Records Management Framework: Enhancing Security, Privacy, and Interoperability”, Technologies 2024, 12(9), 168; https://doi.org/10.3390/technologies12090168

[9] K. Pampattiwar and P. Chavan, “A secure and scalable blockchain-based model for electronic health record management”. Sci Rep 15, 11612 (2025). https://doi.org/10.1038/s41598-025-94339-w

[10] O. A. Oki, A. O. Agbeyangi and A. Mgidi, “Blockchain in Healthcare: Implementing Hyperledger Fabric for Electronic Health Records at Frere Provincial Hospital” [Online]. Available: https://arxiv.org/html/2407.15876v1

[11] S. Gajarlewar, A. Patle, A. Bhosale, B. Khamkar and A. Sase (2026). “HealthVault: Trustworthy Medical Records on the Blockchain”. In: I. Bhiradi, J. Machado, (eds) Intelligent Control, Robotics, and Industrial Automation. RCAAI 2024. Lecture Notes in Electrical Engineering, vol 1492. Springer, Singapore.

[12] H. Eren, Ö. Karaduman and M. T. Gençoğlu, “Security Challenges and Performance Trade-Offs in On-Chain and Off-Chain Blockchain Storage: A Comprehensive Review”, Appl. Sci. 2025, 15(6), 3225; https://doi.org/10.3390/app150632

[13] Prisma ORM. Next-generation Node.js and TypeScript ORM [Online]. Available: https://www.prisma.io/orm

[14] T. T. Kuo, H.E. Kim, L. Ohno-Machado, “Blockchain distributed ledger technologies for biomedical and health care applications”. Journal of the American Medical Informatics Association, Nov 1;24(6):1211-1220. doi: 10.1093/jamia/ocx068, 2017

[15] The Solidity Authors, “Introduction to Smart Contracts”. [Online]. Available: https://docs.soliditylang.org/en/latest/introduction-to-smart-contracts.html

[16] Truffle Suite. Ganache – Documentation. [Online]. Available: https://archive.trufflesuite.com/docs/ganache/

[17] General Data Protection Regulation GDPR. [Online]. Available: https://gdpr-info.eu/