BlockHealth: a Blockchain based Framework for Secure and Efficient Healthcare Data Management

Chiara Braghin, Stelvio Cimato, Simone Pesci, Elvinia Riccobene

PII: S2096-7209(26)00030-8
DOI: https://doi.org/10.1016/j.bcra.2026.100468
Reference: BCRA 100468

To appear in: *Blockchain: Research and Applications*

Received date: 28 February 2025
Revised date: 5 December 2025
Accepted date: 17 February 2026

Please cite this article as: Chiara Braghin, Stelvio Cimato, Simone Pesci, Elvinia Riccobene, BlockHealth: a Blockchain based Framework for Secure and Efficient Healthcare Data Management, *Blockchain: Research and Applications* (2026), doi: https://doi.org/10.1016/j.bcra.2026.100468

Highlights

BlockHealth: a Blockchain based Framework for Secure and Efficient Healthcare Data Management

Chiara Braghin, Stelvio Cimato, Simone Pesci, Elvinia Riccobene

- A novel Framework for Secure and Efficient Healthcare Data Management
- A Blockchain-Based Identity Management and Access Control
- Data Integrity Verification with Merkle Trees
- Scalable and Compliant Data Storage
- Secure Data Sharing via Proxy Re-Encryption

BlockHealth: a Blockchain based Framework for Secure and Efficient Healthcare Data Management

Chiara Braghin[a], Stelvio Cimato[a], Simone Pesci[a] and Elvinia Riccobene[a]

[a]*Computer Science Department, Università degli Studi di Milano, Milan, Italy*

ARTICLE INFO

Keywords:
Blockchain
Healthcare Data Management
Identity Management
Non-Fungible Tokens (NFTs)
Merkle Trees
Proxy Re-Encryption
Apache Cassandra
Data Privacy
Ethereum
Smart Contracts

ABSTRACT

The healthcare sector increasingly relies on digital infrastructures to manage large volumes of sensitive medical data. Ensuring integrity, controlled access, interoperability, and auditability remains a fundamental challenge. We propose BlockHealth, a hybrid blockchain-based framework that integrates smart contracts, distributed databases, and proxy re-encryption to support secure and verifiable healthcare data management. The system leverages Ethereum and NFT-based identities for access control, Merkle-tree commitment for tamper-evident integrity verification, and a distributed Cassandra storage layer for scalable and regulation-compliant off-chain data management. Proxy re-encryption enables secure delegation of access without exposing private keys, while a coordinating API service ensures interoperability with existing hospital infrastructures. Our evaluation demonstrates the feasibility and efficiency of core operations âĂŤ including hashing, on-chain commits, and re-encryption âĂŤ indicating that the proposed framework can provide a practical balance among verifiability, performance, and deployability in realistic healthcare environments.

1. Introduction

The healthcare industry is experiencing a profound transformation driven by the digital revolution, which is reshaping economies, industries, and impacting the lives of billions worldwide. Digital technologies are reshaping workflows, improving decision-making, and enabling the integration of heterogeneous sources of medical information, from electronic health records (EHRs) to data generated by wearable and monitoring devices [7]. Ensuring secure, efficient, and interoperable handling of this information is essential for supporting high-quality healthcare delivery.

Traditional EHR systems are predominantly centralized and often suffer from fragmentation, limited interoperability, and vulnerabilities that may expose sensitive medical records to integrity or access-control failures. As health data grows in volume and complexity, there is an increasing need for technical solutions capable of offering robust traceability, verifiability, and secure delegation of access among multiple stakeholders.

Blockchain technologies, originating from the introduction of Bitcoin by Nakamoto in 2008 [23], have emerged as a promising tool to enhance integrity and auditability in healthcare data management. Their tamper-evident logging capabilities and decentralized verification make them suitable for recording identity information, permission updates, and integrity commitments. However, fully decentralized architectures are often impractical in real clinical settings, which must integrate with legacy systems, comply with regulatory requirements, and guarantee high availability. As a result, many deployable solutions adopt hybrid models that combine blockchain-based verifiability with operational components under institutional control.

In this work, we adopt such a hybrid approach and present *BlockHealth*, a framework that integrates blockchain smart contracts, distributed databases, and proxy re-encryption (PRE) to support secure and verifiable medical data management. Rather than aiming for complete decentralization, BlockHealth focuses on providing tamper-evident integrity, fine-grained and verifiable access control, and secure patient-controlled delegation of data access, while remaining compatible with hospital infrastructures and regulatory constraints. BlockHealth employs smart contracts and ERC-1155 Non-Fungible Tokens (NFTs) to represent identities and encode permission hashes, ensuring transparent and auditable access-control rules. Medical records are stored off-chain in a distributed Apache Cassandra database for scalability and compliance with data sovereignty regulations. Their integrity is validated using Merkle Trees whose roots are anchored on the blockchain. Secure sharing of encrypted records is enabled through a proxy re-encryption scheme, allowing patients to delegate decryption capabilities to authorized providers without exposing private keys. We validate our framework through a use case in a hospital setting and evaluate its performance across core cryptographic operations and data access scenarios.

The rest of the paper is organized as follows. In Section 2, we provide background information on blockchain technology, distributed databases, and proxy re-encryption relevant to our framework. Section 3 presents a set of requirements that the BlockHealth framework must satisfy to guarantee data security and governance in compliance with regulations. Section 4 presents the architecture of BlockHealth, while in Section 5, we detail the implementation of our proposed framework, explaining how each component integrates to enhance security and efficiency within our hospital. Section 6 presents a practical use case demonstrating

*This work was supported in part by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

**The third author was supported by the PRIN project SAFEST (G53D23002770006).

*Corresponding author
ORCID(s):

the application of our framework in a real-world scenario. In Section 7, we evaluate the performance of our system, discussing both on-chain and off-chain aspects. Section 8 reviews related work in the application of blockchain in healthcare data management. Finally, Section 9 concludes the paper and suggests directions for future research.

2. Background

This section provides background on the key technologies underlying our framework, including blockchain and NFTs, Merkle trees, and proxy re-encryption. Additionally, we introduce the Fast Healthcare Interoperability Resources (FHIR) standard, whose open-source nature and strong community support drive its growing adoption in healthcare IT worldwide.

2.1. Blockchain Technology

A blockchain is a decentralized and distributed ledger consisting of a chain of cryptographically linked blocks, each containing digitally signed data. Maintained by a network of nodes, it ensures transparency and security by preventing any single entity from tampering with the records. This structure enables peer-to-peer transactions of digital assets without reliance on a central authority or intermediary, making blockchain a trustless and tamper-resistant system.

Ethereum [5, 37] extends the basic blockchain model by supporting smart contracts, which are programs whose execution is triggered by transactions and whose state changes are recorded immutably on-chain. Rather than acting autonomously, smart contracts define the conditions under which specific state transitions may occur, and these conditions are enforced collectively by the network during transaction validation. This model allows developers to encode application-specific logic, such as identity handling or access rules, directly into the blockchainâĂŹs execution environment. In Ethereum, accounts are generally categorized into externally owned accounts (EOAs), which are controlled by private keys, and smart contract accounts, which are governed by on-chain code. Both types are identified by a unique address, with EOAs deriving their address from cryptographic keys that enable secure authentication and transaction signing, while smart contract addresses are deterministically generated at deployment and do not rely on private keys.

Non-Fungible Tokens (NFTs) have emerged as a revolutionary blockchain innovation, enabling unique digital ownership of assets through cryptographically verifiable tokens. The concept originated in 2012 with Colored Coins on the Bitcoin blockchain [28], which introduced the idea of tokenizing real-world assets [9, 35]. NFTs differ from fungible tokens (such as Bitcoin or Ethereum) as they are indivisible and uniquely identifiable due to their distinct metadata and properties. NFTs are powered by blockchain-based smart contract standards that define their properties and behavior. In order to facilitate NFT implementation, several smart contract standards have been developed. ERC-721 is the most widely used, ensuring each token is unique and individually owned, making it ideal for digital collectibles and identity-based assets. ERC-1155 [26] improves upon previous standards by introducing a semi-fungible structure, where a token can represent fungible, non-fungible, or partially fungible assets within a single contract. ERC-1155 provides a more flexible and efficient framework that supports batch operations, reduced gas costs, and improved scalability.

2.2. Merkle Trees

A *Merkle tree* [21], also known as hash tree, is a binary tree where each leaf node contains the cryptographic hash of a data block, and each non-leaf node (internal node) contains the hash of the concatenation of its two child nodes. This hierarchical hashing structure enables efficient and secure verification of data integrity, even for large datasets, without requiring access to the entire dataset.

In particular, Merkle trees support lightweight proofs of data integrity, known as *Merkle proofs*. Given a trusted root hash (Merkle root), a verifier can validate the integrity of a specific data block by inspecting only a small subset of hashes along the path from the leaf node to the root [22]. This property is essential in blockchain networks, where Merkle trees are used to organize transaction data in Bitcoin, Ethereum, and other distributed ledgers, allowing users to verify the integrity of individual transactions without needing to download the entire blockchain [23, 37]. Beyond blockchain, given their robustness in ensuring data integrity and tamper-proof verification, Merkle trees remain a fundamental data structure in modern cryptographic and decentralized applications.

2.3. Proxy Re-Encryption

Proxy Re-Encryption (PRE) is a cryptographic technique that enables a semi-trusted intermediary, often called proxy, to transform ciphertexts encrypted with one public key into ciphertexts that can be decrypted with the private key associated to a different public key without accessing the underlying plaintext. This mechanism extends traditional public-key encryption by allowing controlled re-encryption, making it particularly useful in scenarios requiring secure data sharing, decentralized systems and cloud storage by enabling users to delegate access to encrypted data without exposing their private keys [3]. In a typical PRE scheme, a data owner encrypts a message with the recipient's public key. When the data needs to be shared with a third party, the owner generates a re-encryption key and provides it to the proxy. The proxy then uses this key to convert the ciphertext into a new ciphertext that the new intended recipient can decrypt with his own private key. Importantly, the proxy never learns the original plaintext, preserving data confidentiality. PRE can be categorized into unidirectional (allowing re-encryption in one direction only) and bidirectional (supporting mutual re-encryption) schemes, as well as single-hop (where re-encryption happens only once) and multi-hop (allowing multiple sequential re-encryptions) [4].

The Umbral Proxy Re-Encryption scheme [25] differs from classic PRE protocols by introducing a decentralized

approach that enhances security and flexibility. Traditional PRE typically relies on a centralized proxy to re-encrypt data between the sender and recipient, with limited support for multi-party re-encryption or conditional access control. In contrast, Umbral supports threshold cryptography, where multiple proxies can cooperate to perform re-encryption and access to the decrypted data requires a minimum number of re-encryption key fragments. In addition, it supports dynamic addition of new recipients without re-encrypting data for each one.

2.4. Fast Healthcare Interoperability Resources

Fast Healthcare Interoperability Resources (FHIR) [13] is a standard developed by Health Level Seven International (HL7) to facilitate the interoperability of healthcare data across different systems, such as electronic health records (EHRs), mobile applications, and cloud-based platforms. It defines a set of resources and APIs for representing and exchanging clinical and administrative data. The standard defines a set of *resources* and application programming interfaces (APIs) for representing and exchanging clinical and administrative data. Resources represent clinical and administrative concepts, such as patients, medications, laboratory results, and appointments. This modularity allows developers to implement FHIR incrementally, ensuring flexibility in integration with existing infrastructures. In FHIR, an EMR is modeled as a structured collection of interrelated resources that represent all relevant aspects of a patient's medical history and care, including demographics, conditions, treatments, and observations.

FHIR builds on previous data format standards from HL7, such as HL7 version 2.x and HL7 version 3.x, and adopts a resource-oriented model with RESTful APIs and standardized representations (JSON, XML, or RDF) to facilitate integration with modern web technologies. For example, in FHIR, the information of a patient named Jane Smith can be represented in JSON as follows:

```
{
  "resourceType": "Patient",
  "id": "12345",
  "name": [{
    "use": "official",
    "family": "Smith",
    "given": ["Jane"]
  }],
  "gender": "female",
  "birthDate": "1980-05-15"
}
```

3. Requirements for Secure Healthcare Data Management

Effective governance of healthcare data presents unique challenges due to the sensitive nature of patient information, the need for interoperability among different stakeholders, and stringent regulatory requirements. Ensuring security, privacy, and compliance while enabling efficient data sharing and access control is critical. Moreover, data sharing must always be conducted in a patient-centric manner, prioritizing privacy and consent management.

To meet these challenges, our framework must satisfy the following security and governance requirements [6]:

[R1] *Secure Identity Management and Access Control*: Ensuring that only authorized entities (e.g., healthcare professionals, institutions, and patients) can access sensitive medical data, with fine-grained permissions and robust authentication mechanisms.

[R2] *Granular Access Control*: Implementing role-based and attribute-based access control (RBAC/ABAC) policies to enable context-aware and fine-grained data access tailored to different user roles and scenarios.

[R3] *Patient-Centric Data Governance* Empowering patients with control over their medical data, including consent management, access logs, and the ability to grant or revoke data-sharing permissions.

[R4] *Privacy-Preserving Data Sharing*: Enabling secure and controlled data sharing among authorized entities through encryption, anonymization, or privacy-enhancing cryptographic techniques while maintaining patient confidentiality.

[R5] *Data Integrity and Tamper Resistance*: Providing a tamper-proof and audit-traceable record of all transactions and modifications to guarantee the authenticity and integrity of medical records.

[R6] *Scalability and Compliance*: Managing large volumes of medical data efficiently while ensuring compliance with regulatory frameworks such as the General Data Protection Regulation (GDPR) [10] and the Health Insurance Portability and Accountability Act (HIPAA) [34].

[R7] *Integration with Healthcare Standards*: Ensuring compatibility with widely adopted healthcare data standards such as FHIR (Fast Healthcare Interoperability Resources) and DICOM (Digital Imaging and Communications in Medicine) to facilitate interoperability across healthcare systems.

By addressing these requirements, our proposed framework ensures a secure, privacy-preserving, and interoperable healthcare data governance system that balances data protection with usability and accessibility.

4. BlockHealth Architecture

To meet the aforementioned requirements, we developed a comprehensive technical framework that integrates blockchain technology, distributed databases, and advanced cryptographic mechanisms to ensure secure, scalable, and interoperable healthcare data management. This framework has been carefully designed by leveraging the following technical solutions:

- *NFT-based identity management and smart contract-driven access control*: user identities, roles, and access permissions are managed through smart contracts, which enforce fine-grained access control policies. This ensures that only authorized entities can retrieve

or modify healthcare data, providing transparency and traceability in access operations.

- *Hybrid on-chain and off-chain data storage*: in order to balance confidentiality, transparency, and performance, we adopt a hybrid storage model. Medical records are stored off-chain in a distributed database, while their cryptographic hashes are organized in a Merkle tree. The Merkle root is stored on-chain, serving as an immutable and tamper-evident reference to off-chain data. This approach guarantees data integrity while maintaining efficient storage and retrieval.
- *Scalable and interoperable data managment*: the system uses a distributed database architecture to support high scalability and reliable access to large volumes of EMRs. Data and access policies are structured according to the FHIR standard, facilitating seamless data exchange across heterogeneous healthcare systems.
- *Privacy-preserving data sharing via PRE*: in order to enable secure, patient-controlled data sharing, we integrate a Proxy Re-Encryption scheme. This allows patients to delegate decryption rights to authorized recipients without exposing plaintext data to intermediaries, ensuring confidentiality and maintaining compliance with privacy regulations.

By combining these technologies, our framework ensures robust security guarantees, efficient data management, and regulatory compliance, making it well-suited for modern healthcare information systems.

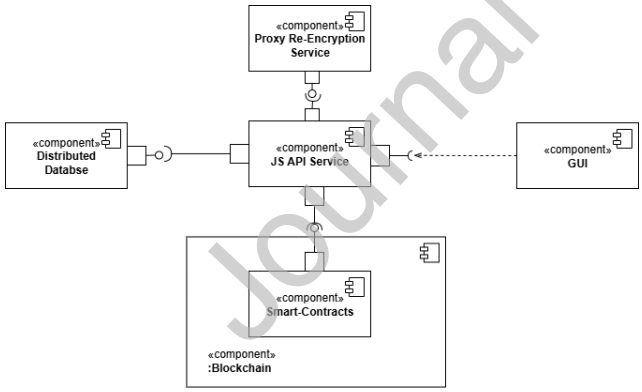

Figure 1: Component Diagram of the Framework Architecture

Figure 1 illustrates the architecture of the proposed blockchain-based framework. The system is composed of several interconnected components, each fulfilling a specific function.

The Graphical User Interface (`GUI`) serves as the front-end through which users - including patients, doctors, nurses, and other hospital staff - interact with the system. It offers role-based access, exposing different features depending on the user's identity and authorization level. For instance, patients can manage their health records and data-sharing permissions, while healthcare professionals can retrieve and process medical information necessary for clinical tasks. The `GUI` communicates directly with the back-end via the `JS API Service`.

The `JS API Service` serves as the central controller of the system. It exposes a comprehensive API that enables user registration, data access, secure sharing operations, and interaction with other back-end services. It mediates communications with the `Distributed Database`, the `Proxy Re-Encryption Service`, and the `Blockchain`. All communication between the service and other components takes place over encrypted channels, ensuring data confidentiality and protection against potential eavesdropping or man-in-the-middle attacks.

The `Proxy Re-Encryption Service` is responsible for enabling secure and privacy-preserving data sharing. Working in collaboration with the `JS API Service`, it manages re-encryption keys that allow patients to delegate decryption rights to authorized healthcare providers. This mechanism ensures that sensitive data can be securely shared without exposing private keys or plaintext to intermediaries, thereby preserving confidentiality.

The `Distributed DB` component serves as the systemâĂŹs primary off-chain storage layer. It is responsible for storing and retrieving all medical records and associated access policies. To enhance interoperability across diverse healthcare platforms, all stored data follows the FHIR standard. This component is queried and updated via the `the JS API Service`, allowing for efficient, scalable data management while maintaining secure access controls.

The `Blockchain` component acts as the decentralized trust anchor for the framework. It does not store sensitive medical data directly; instead, it maintains an immutable ledger of critical security metadata. This includes NFTs used for identity representation, Merkle tree roots for ensuring the integrity of off-chain medical data, and hashes representing user permissions. All blockchain interactions are mediated by smart contracts that enforce strict rules for identity verification, data integrity, and access control. These smart contracts ensure the transparency and tamper-resistance of the systemâĂŹs core security mechanisms.

Please, notice that our system, although integrating blockchain components to ensure integrity and verifiability, is not fully decentralized. Indeed, hospitals often require controlled API gateways for logging, auditability, operational robustness, integration with regulated databases (e.g., Cassandra clusters deployed in specific jurisdictions), and interoperability with legacy systems. For these reasons, BlockHealth adopts a hybrid architecture: blockchain ensures tamper-evident integrity and verifiable permissions, while a centralized, auditable coordination service mediates interactions without accessing plaintext data. This design strikes a balance between decentralization and operational requirements, enabling realistic deployment in modern healthcare institutions.

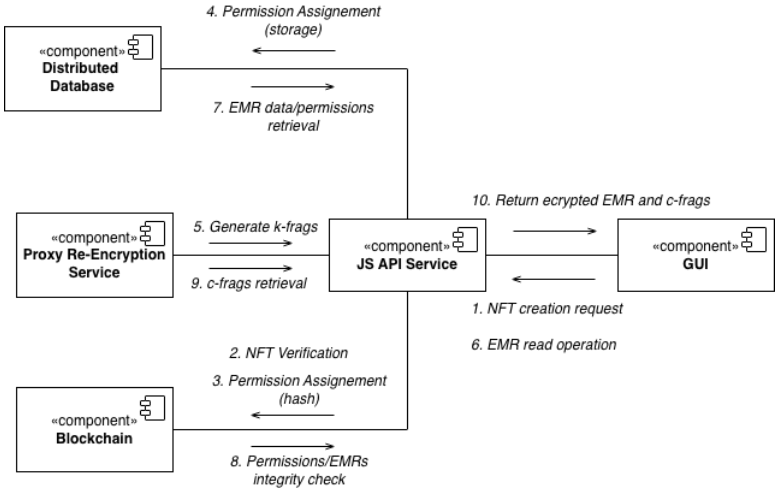

Figure 2: Application workflow

4.1. Workflow Operation

The workflow of the BlockHealth components is illustrated in Figure 2, with the numbered exchange messages representing to key interactions between the framework's components.

A new user registers through the GUI (1), triggering an NFT creation request to the `JS API Service`. The `JS API Service` then communicates with the Smart Contracts to verify and issue the NFT (2). Once the user's identity and permissions are validated, the `JS API Service` assigns the appropriate access rights and updates the corresponding hashes in both the `Blockchain` via Smart Contracts (3) and in the `Distributed DB` (4).

For a doctor to later access a patient's encrypted EMR, the patient must first delegate access. This is achieved via the `Proxy Re-Encryption Service`, which generates re-encryption key fragments (`kfrags`) for a specific doctor (5).

When a user or an authorized doctor requests access to an EMR through the GUI (6), the `JS API Service` first retrieves the encrypted EMR and the associated permissions from the `Distributed DB` (7). It then verifies the integrity of this data by checking it against the hash stored on the `Blockchain` (8). If permissions and integrity are valid, the `JS API Service` requests the re-encrypted capsule fragments from the `Proxy Re-Encryption Service` (9). It then returns only encrypted data and re-encrypted capsules to the doctor (10). The final decryption is performed exclusively client-side using the doctorâĂŹs private key.

This workflow ensures secure identity management, fine-grained access control, and tamper-proof data integrity while enabling efficient and privacy-preserving healthcare data sharing, thereby meeting the security and interoperability requirements (R1-R7) outlined in Section 3.

5. BlockHealth Ethereum-based Implementation

We implemented the proposed framework using the Ethereum blockchain, Solidity smart contracts, the Cassandra distributed database, and Umbral Proxy Re-Encryption. Blockchain interaction is handled via the `web3.js` library over HTTPS, which provides a reliable interface for communicating with smart contracts and accessing blockchain data. This section outlines the key implementation details. The complete source code, along with the dataset used for testing the framework's operations, is available in a public GitHub repository.[1]

5.1. Blockchain-Based Access Control

We use the Ethereum blockchain and NFTs based on the ERC-1155 standard to represent *identities* within the healthcare system, including doctors, patients, and hospital staff. Each NFT uniquely identifies a user and is linked to his access permissions.

To enable this, we developed a Solidity smart contract, `HospitalToken`, which manages NFT creation and enforces access control. The contract assigns NFTs to users based on their role, associating each token with the corresponding permissions defined by the hospitalâĂŹs access control policy. Our current implementation uses Role-Based Access Control (RBAC), with predefined roles for patients and various hospital staff. However, the framework is designed to be extensible, allowing future support for more granular models like Attribute-Based Access Control (ABAC). Access control policies and user permissions are stored securely off-chain in a Cassandra distributed database, ensuring flexibility and scalability.

In order to ensure data integrity and detect unauthorized changes, a hash of each user's permissions list is stored on-chain. Any modification to a user's permissions updates the corresponding on-chain hash, maintaining consistency

[1] `https://github.com/SimonePesci/ResearchUnimi--FHIR/`

between on-chain identity records and off-chain permissions, while ensuring traceability. Permissions cover a wide range of operations, such as accessing medical records in read and write modes (`READ_EMR`, `WRITE_EMR`) or physical access to restricted hospital areas (`ACCESS_RADIOLOGY`). Because permissions are managed off-chain, they can be updated dynamically and efficiently, without necessitating changes to the underlying smart contract.

When a new user joins the system, his/her identity information is collected and verified. An NFT representing his identity and role is minted and assigned to his Ethereum address, establishing his digital identity within the healthcare network. When a user attempts to perform an action, the system retrieves his/her permissions from Cassandra, hashes the list, and compares it with the on-chain hash stored in a suitable state variable of the `HospitalToken` contract. If the hashes match and the required permission is present, the action is authorized, ensuring that only users with valid permissions can access or modify sensitive medical data.

The smart contract handles NFT creation and interactions with blockchain data, while the `JS API Service` component facilitates communication with the Cassandra database. Data exchange between components uses JSON, maintaining compatibility with the FHIR standard.

Algorithm 1 presents the pseudocode for the `mint` function in the `HospitalToken` contract. To ensure that new identity tokens can only be created by a trusted administrative entity, the function is protected with an `onlyOwner` access control modifier, as shown at the beginning of the algorithm at line 3. Once this security check passes, the function proceeds to validate the user's role before invoking the library function responsible for minting the NFT. Subsequently, it initializes the user's permissions as an empty set, which are later updated through a separate smart contract function once the security policy is retrieved from the off-chain database.

In this context, it is important to clarify how privileged blockchain operations are authorized. The private key associated with the contract-owner role is not held by any end user and is never handled through the GUI or the `JS API Service`. Instead, it is managed exclusively within the secure back-end infrastructure of the deploying institution (e.g., via a dedicated signing service or key vault). As a consequence, operations that require elevated privileges, such as updating permission hashes or Merkle roots, cannot be triggered or intercepted by client-side components or by a compromised API. The `JS API Service` may request such updates, but the actual signing process is performed by the protected back-end component, ensuring that administrative authority remains tightly controlled and fully auditable.

Algorithm 1 Pseudocode for generic `mint` function

```
1:  function MINT(role, account)
2:                          ▷ Ensure only the contract owner can mint new tokens
3:      if caller ≠ owner then
4:          raise error "Caller is not the owner"
5:      end if
6:                                                          ▷ Validate role
7:      if role ∉ {Doctor, Patient, Assistant} then
8:          raise error "Invalid role"
9:      end if
10:     if balanceOf(account, role) > 0 then
11:         raise error "This address already owns this role."
12:     end if
13:                 ▷ Determine and increment token counter based on role
14:     tokenId ← 0
15:     if role = Doctor then
16:         tokenIdDoctor ← tokenIdDoctor + 1
17:         tokenId ← tokenIdDoctor
18:     else if role = Patient then
19:         tokenIdPatient ← tokenIdPatient + 1
20:         tokenId ← tokenIdPatient
21:     else if role = Assistant then
22:         tokenIdAssistant ← tokenIdAssistant + 1
23:         tokenId ← tokenIdAssistant
24:     end if
25:                                                      ▷ Mint the token
26:     MINT_TOKEN(account, role, 1)
27:                                              ▷ Record token possession
28:     tokenPossession[account][role] ← tokenId
29:     addressTokenPossession[role][tokenId] ← account
30:     return "Token minted successfully"
31: end function
```

5.2. Distributed Database for Medical Storage

For scalable storage of medical records, we utilize a distributed Apache Cassandra database cluster, ensuring both high availability and fault tolerance. Moreover, it provides adherence to data protection regulations such as GDPR. In order to comply with regional data localization laws, we configure Cassandra with multiple data centers located in different geographic regions. For example, European data is stored within EU data centers, while US data is confined to US-based data centers. This geographic segregation enhances data sovereignty and compliance with local regulations.

The database schema is designed to support FHIR resources, enabling standardized data exchange and interoperability across different healthcare systems. CassandraâĂŹs schema flexibility and wide-column data model make it well-suited for handling diverse healthcare data formats while ensuring seamless integration with existing healthcare platforms.

As a highly scalable, open-source NoSQL database, Cassandra is optimized for managing large volumes of distributed data with no single point of failure. Its ability to replicate data across multiple nodes and data centers ensures uninterrupted system operation, even in the event of failures or network disruptions, essential for mission-critical healthcare applications. Additionally, its robust security features, including fine-grained access controls, encryption (at rest and in transit), and audit logging, protect sensitive medical data from unauthorized access and breaches.

5.3. Medical Records Integrity with Merkle Trees

To ensure the integrity of medical records without storing sensitive data on-chain, we employ Merkle Trees. This technique allows for efficient and secure verification of large datasets by storing only a single, compact cryptographic summary, the Merkle Root, on the blockchain.

The process begins by hashing each individual resource of each Electronic Medical Record to create a set of leaf nodes. Parent nodes are then formed by iteratively hashing pairs of lower-level nodes until a single Merkle Root is derived. This root represents the entire dataset of EMRs.

When an EMR resource is accessed, a Merkle Proof is generated to verify its integrity. The input to this verification process is the hash of the EMR resource, not the raw data itself. This hash, along with the Merkle Proof, is submitted to the `verify` function of the `MerkleTree` smart contract (see Algorithm 2). The smart contract then reconstructs the Merkle Root and compares it with the version stored on the blockchain.

The decision to perform this final verification step on-chain is deliberate. It provides a trustless integrity guarantee, as the trustworthiness of the operation stems directly from the immutable and transparent nature of blockchain execution. This makes sure an EMR has not been tampered with, without needing to trust any off-chain component for the final, critical verification step.

When an EMR resource is updated, the Merkle Tree is reconstructed off-chain by the `JS API Service`, and the new Merkle Root is stored on the blockchain via the `updateMerkleRoot` function, ensuring the integrity anchor is always current. To prevent unauthorized modifications, this function is protected by an access control modifier. As shown in Algorithm 2, this modifier ensures that only the contract `owner` –in our architecture, the trusted `JS API Service`– can call this function. This call is only made after the system has successfully verified the permissions of the user who initiated the update.

Algorithm 2 Pseudocode for MerkleTree Contract

```
// Property: Hardcoded Merkle root computed from EMRs
merkleRoot ← 0xb750bb...
function VERIFY(leaf, proof)
                ▷ This function is public for anyone to verify integrity
    computedHash ← leaf
    for each proofElement in proof do
        if computedHash ≤ proofElement then
            computedHash ← keccak256(computedHash || proofElement)
        else
            computedHash ← keccak256(proofElement || computedHash)
        end if
    end for
    return (computedHash == merkleRoot)
end function
function UPDATEMERKLEROOT(newRoot)
                ▷ Ensure only the contract owner can update the root
    if caller ≠ owner then
        raise error "Caller is not the owner"
    end if
    merkleRoot ← newRoot
    return true
end function
```

5.4. Proxy Re-Encryption for Secure Data Sharing

To enable secure and confidential data sharing among authorized participants, we integrate proxy re-encryption using the Umbral library.

A patient's EMR resource is first encrypted with the patient's public key and then stored securely in the Cassandra database.

To delegate access, patients generate re-encryption key fragments (`kfrags`) for authorized healthcare providers. These `kfrags` are distributed across multiple proxy nodes. When a doctor requests access, the proxies provide capsule fragments (`cfrags`), which the doctor collects. Once a sufficient number of `cfrags` are obtained, the doctor can reconstruct the re-encrypted capsule and use their private key to decrypt the EMR resource. By leveraging Umbral's capability to use multiple proxies, our system ensures greater fault tolerance, enhancing reliability and security during the re-encryption process.

All of this is handled by the `Proxy Re-Encryption Service` component, making the process transparent to users. We assume that the user has a wallet that helps him/her when the re-encryption keys are needed.

6. A Practical Walkthrough

To illustrate the practical application of our proposed framework, we present a comprehensive use case involving Dr. John Doe, a newly registered doctor at the hospital, and the patient Jane Smith. We describe three scenarios of the use case: (*i*) First, Dr. Doe undergoes the *registration* process; then, (*ii*) he *gets the access* to Jane's EMR; and subsequently, (*iii*) he can perform the usual tasks of *reading* and *updating* the resources of Jane's EMR.

For the last two scenarios, we use sequence diagrams to show the practical interaction of the components to address the relevant steps of the use case scenarios.

Registration of a new user. Currently, we envision that new doctors must be registered by an HR employee on their first day of work. Dr. John Doe authenticates himself using traditional identification, and the HR staff initiates the NFT creation request. This request is processed, and the `HospitalToken` smart contract mints a new NFT representing Dr. Doe's identity, which is linked to his Ethereum address. The NFT contains his professional credentials and is securely stored on the blockchain.

The NFT creation request for Dr. John Doe, formatted according to FHIR, is as follows:

```
{
 "role": "doctor",
 "address": "0x73D38E148032f93c53612f3A6fcd7F48Ff63626B"
}
```

After the NFT is minted, the user is provided with the following permissions:

```
{
```

```
 "role": "doctor",
 "tokenID": "1",
 "hospitalID": "1",
  "permission": ["READ_EMR", "WRITE_EMR", "ACCESS_RADIOLOGY"]
}
```


Secure Data Sharing. Jane grants access to her EMR to Dr. Doe, using proxy re-encryption (see Figure 3). The system generates re-encryption key fragments (`kfrags`) for him:


```
{
    "patient_id": "Jane",
    "doctor_id": "Doe"
}
```


Note that it is possible for a patient to simultaneously grant access to his/her EMR to a sequence of `"doctor_ids": ["Doe", "OtherDoctor"]` by a unique request.

The system responds with a message indicating that Jane has successfully generated re-encryption key fragments (`kfrags`) for Dr. Doe (alternatively, if the operation involves multiple healthcare providers, a separate key fragment is generated for each of them):


```
{
    "kfrags": {
        "Doe": "20 KFrags generated",
    },
    "message": "KFrags generated successfully",
    "patient_id": "Jane"
}
```


The `kfrags` are distributed to multiple proxy nodes in the network. When Dr. Doe requests access to Jane's EMR, the proxies use the `kfrags` to produce capsule fragments (`cfrags`). Dr. Doe collects a sufficient number of `cfrags` to reconstruct the re-encrypted capsule. Using the re-encrypted capsule and his private key, Dr. Doe decrypts Jane's EMR (resources). The decrypted EMR resource appears as follows:


```
{
    "doctor_id": "Doe",
    "message": "EMR decrypted successfully",
    "patient_id": "Jane",
    "plaintext": "{"resourceType":"Patient","id":...
}
```


This response confirms that Dr. John Doe (with ID `Doe`) has successfully decrypted the EMR resource for `Jane`.

Accessing a Patient's EMR. If Dr. John Doe is a registered physician within the healthcare network and has been granted to a patient's EMR, he can access the system to read and update the patient's EMR, Jane Smith, for example. To do so, he authenticates himself in the `GUI` using his unique NFT-based credentials.

The system first verifies whether Dr. John Doe has the appropriate permissions to update Jane's EMR (see Figure 4 for components interaction during the verification phase), specifically the `WRITE_EMR` permission for Jane's hospital, since each set of permission is relative to an hospital (`hospitalID=1` in our use case). These permissions are stored off-chain in the Cassandra database and hashed using `Keccak256`. The system then compares the hashed permissions with the hash stored on-chain. If the hashes match and the required permission is found, Dr. Doe is granted access to proceed with updating the EMR.

Dr. John Doe then updates the given resource(s) of the Jane's EMR, which is subsequently encrypted and stored in the distributed database (see Figure 5 for the process flow). After performing integrity checks using the Merkle root, the updated EMR resource(s) is encrypted with Jane's public key, resulting in ciphertext and an associated capsule required for proxy re-encryption.

The resulting encrypted data is as follows:


```
{
  "EMR": {
     "capsule": "A5pkgnnhMfStY3b9Nqcw3B29NsqRmcmuaLvca...",
     "ciphertext": "v0U0HEN@0y0mY+Ps0Oo2J8SMgnFEndz5C..."
  },
  "message": "EMR encrypted successfully",
  "patient_id": "Jane"
}
```


This JSON response indicates that the EMR resource has been encrypted successfully for `Jane`, with the encrypted data consisting of a `capsule` and a `ciphertext`. Then, the encrypted EMR resource is stored in the Cassandra database.

7. Evaluation of BlockHealth

In this section, we present a comprehensive evaluation of the proposed framework from multiple perspectives. Our analysis covers four key dimensions: (*i*) the execution time of both off-chain and on-chain operations, to assess real-time usability; (*ii*) the latency overhead introduced by Proxy Re-encryption to achieve secure data sharing; (*iii*) the costs (both in gas and Euros) incurred by blockchain transactions, to evaluate cost-effectiveness in the Sepolia test network; and (*iv*) the frameworkâĂŹs ability to meet the security and scalability goals outlined in Section 3.

All tests were conducted on a system running Linux Ubuntu with the following specifications: Processor: AMD Ryzen 7 5800 (8-core), RAM: 16 GB.

7.1. Average Database Read and Write Times

We evaluated the performance of both off-chain and on-chain storage systems by measuring average read and write times. Specifically, we compared PostgreSQL (a centralized relational database) and Apache Cassandra (a distributed NoSQL database) to assess their suitability for our framework. Although Cassandra is used in our implementation, PostgreSQL was considered to provide a baseline for performance comparison.

PostgreSQL was selected for its proven efficiency in handling complex queries and transactions with strong consistency guarantees. Cassandra, on the other hand, offers

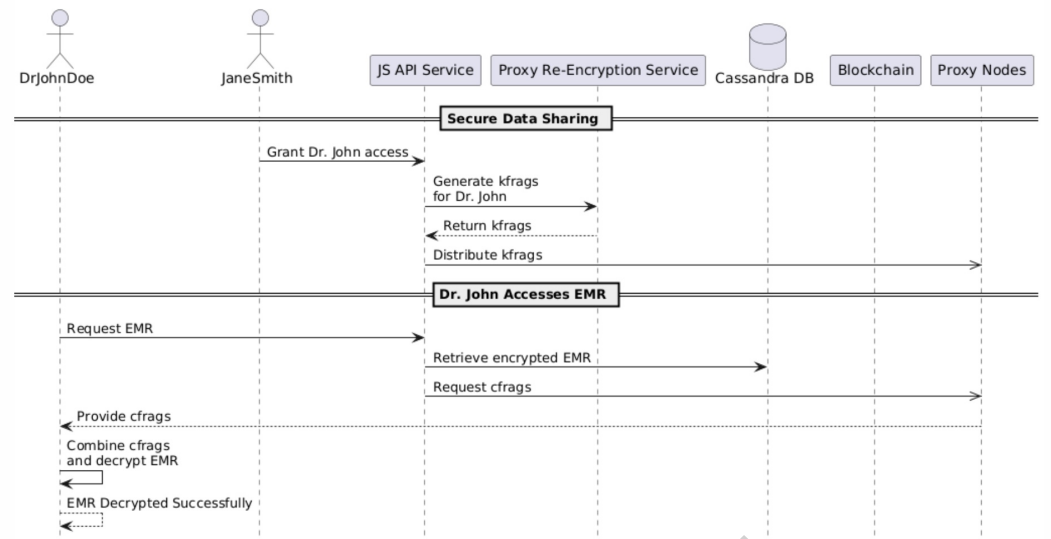

Figure 3: Proxy Re-Encryption flow for a single EMR resource

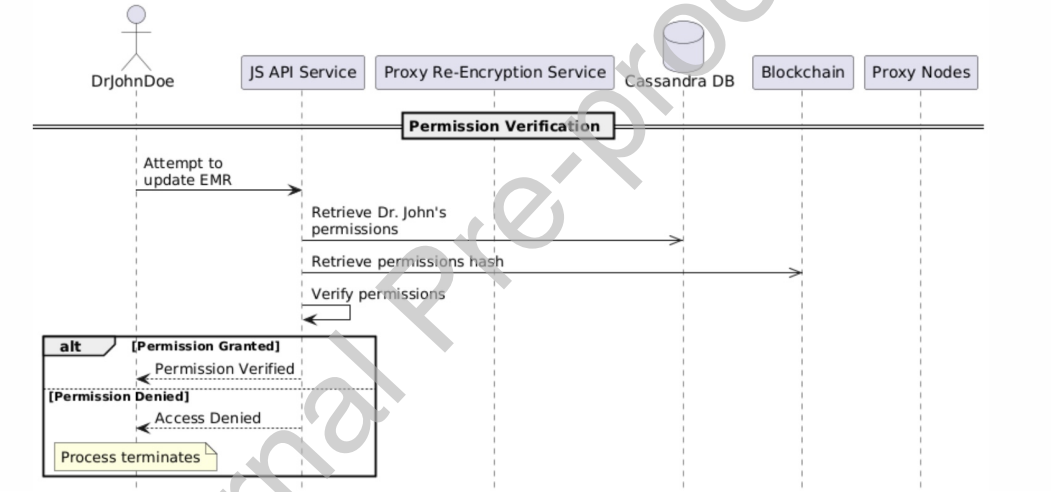

Figure 4: Verifying the request's permissions

high availability, fault tolerance, and horizontal scalability, key features for a distributed healthcare data system.

As summarized in Table 1, PostgreSQL outperformed Cassandra in both read and write operations within a local test environment. This is expected given its single-node architecture and immediate consistency. However, CassandraâĂŹs slightly higher latency (3.81 ms read, 5.33 ms write) remains well within acceptable limits for clinical use and is justified by its distributed nature, which ensures resilience and scalability in real-world deployments.

For on-chain performance, we measured transaction latency on the Sepolia Ethereum test network at different times of day to account for network variability (Table 2). On-chain operations, particularly writes, introduced significant latency (over 13 seconds on average), emphasizing the importance of minimizing on-chain interactions and leveraging off-chain solutions where appropriate

Table 1
Average On-chain Read and Write Times

Database	Average Read Time (ms)	Average Write Time (ms)
PostgreSQL	0.58	3.20
Cassandra	3.81	5.33

7.2. Encryption and Re-encryption Performance

We evaluated the performance of the Umbral proxy re-encryption scheme to measure the latency of its core cryptographic operations: encryption, decryption, and re-encryption. To ensure the reliability of our measurements, we conducted approximately 20 tests for each operation across various data sizes. The selection of these sizes was guided by an analysis of datasets generated by the Synthea

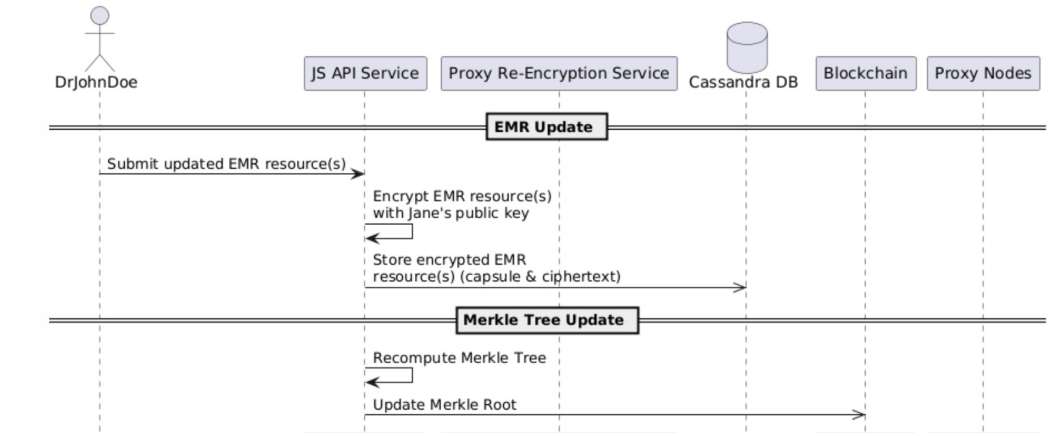

Figure 5: EMR update flow

Table 2
Average Read and Write Times on Sepolia Network

Time of Day	Average Read Time (ms)	Average Write Time (ms)
Morning	140.75	14,045.15
Afternoon	142.80	17,114.05
Evening	142.12	13,391.05

project [2], a well-known synthetic patient record generator. The 800-byte size represents small medical messages or metadata objects (e.g., vital sign updates or authorization tokens); the 800 KB size corresponds to standard EMR resources such as individual encounter summaries or lab results; and the 4.2 MB size approximates a complete EMR containing comprehensive patient histories. These categories effectively capture the most common data sizes encountered in clinical workflows. We assumed that content variability has a negligible impact on cryptographic performance compared to data size; therefore, testing multiple EMRs of the same size would be redundant.

As shown in Figure 6, all operations remained performant across the range. For the largest data size (4.2 MB), encryption and decryption completed in under 100 ms, while re-encryption, a key operation for secure delegation, remained below 50 ms. This low latency ensures responsive access to encrypted records, even in real-time clinical settings.

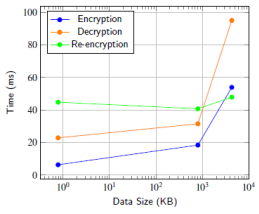

Figure 6: Time vs. Data Size for Core Crypto Operations.

[2] https://syntheticahealth.github.io/synthea/

Table 3
Costs analysis for Ethereum Transactions

Transaction	Cost 10^{-4}ETH	Cost EUR
HospitalToken.sol Deploy	96.22317	20.58
MerkleTree.sol Deploy	9.37220	2.00
NFT Minting	7.88995	1.69
Permission Update	1.34022	0.29
EMR Update/Add	0.68007	0.15

7.3. Costs Evaluation

This section analyzes the gas consumption and corresponding monetary cost (as of the time of submission) for deploying the smart contracts and performing the most common transactions in our system. Table 3 summarizes the costs obtained from the execution of the use case. The first two entries correspond to the one-time deployment of the smart contracts, which are the most expensive operations due to the need to store contract code on-chain. The usual operations, such as minting an NFT, updating access permissions for authorized doctors, or creating and updating EMRs, incur significantly lower costs. For instance, issuing an authentication token (NFT) costs approximately âĆň 1.69, while updating permissions or EMR content ranges from âĆň 0.29 to âĆň 0.15 per transaction.

7.4. Security Evaluation

In this section, we assess how our architecture meets the security requirements defined in Section 3. We first describe the core security mechanisms exploited by our framework, explicitly linking each requirement to the feature enforcing it. This is followed by a structured threat analysis focused on the systemâĂŹs main components and their interactions.

7.4.1. Security Features

Identity Management and Access Control: We use ERC-1155 NFTs to assign verifiable identities to system actors, such as patients and medical personnel. These identities are linked to off-chain role-based access policies, enforced

Table 4
Identified Threats, Applied Countermeasures, and Corresponding Security Requirements

Threat ID	Description	Applied Countermeasures	Requirements
T1	Unauthorized access to EMRs	On-chain identity via ERC-1155 tokens; off-chain access control (RBAC/ABAC); proxy re-encryption delegation	R1, R2, R3
T2	Replay or MITM attacks on data in transit	TLS-secured API endpoints; Ethereum signature validation on messages	R2, R4
T3	Tampering with off-chain medical data	Merkle tree hashing; on-chain root verification; hash matching during access	R5
T4	Smart contract vulnerabilities	Minimal logic in smart contracts; security-audited libraries; immutability of deployed code	R1, R5
T5	Abuse of re-encryption capabilities	Ephemeral key usage; strict delegation policies; audit logging of key exchanges	R3, R4
T6	Compromise of the off-chain gateway	API authentication; event logging; hash validation before blockchain updates; (plans for distributed oracle or ZKPs)	R5, R6
T7	Availability loss or data silos	Cassandra-based distributed storage; fault tolerance and replication; regional compliance deployment	R6

jointly with the on-chain logic implemented by smart contracts, ensuring that only authorized actions are performed (**R1**, **R2**).

Delegation via Proxy Re-Encryption: The Umbral proxy re-encryption scheme allows patients to delegate access without revealing their private keys or requiring full data re-encryption. This approach ensures confidentiality and flexible sharing (**R3**, **R4**).

Data Integrity via Merkle Trees: Medical records stored off-chain are organized into Merkle trees whose root hashes are stored on-chain. This guarantees tamper-evident storage and enables lightweight integrity checks (**R5**).

Scalable and Compliant Storage: We use Apache Cassandra for distributed, scalable EMR storage, enabling geographic replication and data protection compliance (e.g., GDPR) (**R6**).

FHIR-Based Interoperability: All medical data is structured in compliance with the FHIR standard, enabling interoperability with existing health information systems (**R7**).

7.4.2. Adversarial Model and Threat Analysis

To provide a more rigorous evaluation, we analysed the main components of the framework using a threat-oriented approach [24], focusing on potential adversarial scenarios and the applied countermeasures to mitigate them.

We assume the blockchain infrastructure to be tamper-resistant, with smart contracts correctly enforcing the prescribed logic, and the cryptographic primitives used (proxy re-encryption, hash functions) to be secure against known attacks. In contrast, all off-chain components, including the database and the entity responsible for computing and submitting Merkle roots, are considered potentially vulnerable. Although all communications occur over secure HTTPS channels, we still consider common web-based threats, including replay attacks, rollback attempts, and API forgery. Users in the system may act honestly, but may also behave maliciously by attempting to escalate privileges or access data beyond their authorization.

We identified several potential threats to the system, which are summarized in Table 4, alongside the corresponding countermeasures and their alignment with the defined security requirements. These include:

- *T1 âĂŞ Unauthorized Access*: Adversaries may attempt to gain access to patient records without proper authorization. In our framework, access control is enforced through a combination of ERC-1155-based identity NFTs and off-chain permission records. Additionally, data confidentiality is protected through encryption, with private keys never stored on the server, and the Umbral proxy re-encryption scheme ensuring that only explicitly authorized users can decrypt sensitive medical data. These countermeasures minimize the risk of unauthorized decryption. Moreover, data integrity is guaranteed by the use of Merkle trees, which enable the detection of any unauthorized modifications or access attempts. Since modifying data requires possession of the corresponding cryptographic keys, unauthorized changes are effectively prevented. In the event of tampering, inconsistencies in Merkle root hashes provide traceability and enable prompt detection of breaches.

- *T2 âĂŞ Man-in-the-Middle or Replay Attacks*: Adversaries may attempt to intercept, modify, or replay data transmitted between system components. In our framework, all communication is secured using TLS and HTTPS to ensure confidentiality and integrity. Furthermore, blockchain transactions are cryptographically signed and include timestamps and nonces, effectively preventing replay attacks and unauthorized modifications.

- *T3 âĂŞ Off-chain Data Tampering*: A malicious actor could attempt to alter the content of the off-chain database (e.g., EMRs, permission lists, or stored hashes). In the framework, Merkle tree roots are stored

on-chain, enabling clients to verify the integrity of retrieved data through Merkle proofs. This ensures that any unauthorized modification of off-chain content can be efficiently detected.

- *T4 âĂŞ Smart Contract Exploits*: Adversaries may attempt to exploit vulnerabilities in smart contracts to manipulate identities or escalate permissions. To mitigate this risk, our contracts are minimal, adhere to established security best practices, and are subject to thorough auditing. Furthermore, no sensitive data is stored on-chain, reducing the potential impact of any exploitation.

- *T5 âĂŞ Delegation Misuse*: Malicious proxies may attempt to disrupt the system by refusing to re-encrypt (leading to a DoS), sending malformed ciphertext, or attempting to reconstruct the re-encryption key. Our systems uses Umbral, which remains secure even in the presence of such proxies: denial-of-service is mitigated by switching to alternative proxies; malformed ciphertext is detected and rejected through verifiable re-encryption; and key reconstruction requires collusion among multiple proxies, which is unlikely if they are independently run.

- *T6 âĂŞ Compromise of the* `JS API Service`: The JS API Service constitutes the only coordinating off-chain component and is therefore a potential target for attackers. Its compromise, however, does not enable privilege escalation or unauthorized data access. Although an attacker gaining control of the API could modify the permission records stored in Cassandra, such tampering is ineffective unless it is accompanied by a corresponding update of the permission hash stored on the blockchain. Since the smart contract accepts updates to this hash only when they are signed by a privileged administrative account, an adversary must also compromise the associated private key to alter the on-chain state. Crucially, the administrative private key required to perform such updates is not managed by client wallets nor transmitted through the JS API Service; instead, it is securely stored within the back-end infrastructure of the deploying institution (e.g., in a dedicated key vault or signing service). An adversary compromising the API alone therefore cannot access this key.

  Without this key, the blockchain remains the authoritative source of truth, and any mismatch between on-chain and off-chain permissions is detected during integrity verification. Even in the extreme case where both the API and the administrative private key are compromised, the resulting on-chain update is immutably logged and therefore auditable. Such an attack cannot occur silently and is equivalent to an authenticated insider action rather than a failure of the architecture. This separation of concerns preserves tamper-evidence and ensures that the API cannot unilaterally subvert access-control or integrity guarantees. To reinforce authentication, a Kerberos-like mechanism [32] can be integrated to allow users to prove control of their NFT-based identity before sensitive operations are requested. Although a distributed oracle or ZK-based verification mechanism [1, 11] may further reduce the trust placed in this component, these improvements are part of future work and are not required for the correctness of the current framework.

- *T7 âĂŞ Availability Loss or Data Silos:* System availability may be compromised due to infrastructure failures or fragmented data storage across incompatible systems. Our framework employs Apache Cassandra, a distributed and fault-tolerant database with built-in replication to ensure high availability. Additionally, the system may support regional deployment strategies to comply with local data regulations, while maintaining data accessibility and interoperability (e.g., by means of FHIR standard).

By adopting a threat-oriented perspective, we demonstrate how our framework mitigates key security risks. Overall, our architecture satisfies the security requirements identified in Section 3, while offering a realistic path to deployment in privacy-sensitive environments like healthcare.

8. Related Work

Blockchain technologies have been widely explored in the healthcare domain to strengthen integrity, auditability, and controlled data exchange. Several surveys [39, 41] highlight how decentralized ledgers can support identity governance, delegated access, interoperability, and scalability. However, most existing solutions address these dimensions only in part. BlockHealth combines multiple techniques âĂŤ NFT-based identity management, on-chain anchoring of integrity through Merkle roots and hashed permission sets, proxy re-encryption for delegated access, and scalable off-chain storage âĂŤ into a unified hybrid framework.

We structure related work across three main themes: (i) blockchain-based data management, identity and access control; (ii) hybrid storage and integrity verification; (iii) delegation-based data sharing. To complement this thematic analysis, Table 5 compares the most representative healthcare systems discussed in this section and shows how our approach integrates established techniques within a coherent hybrid architecture in which encrypted EMRs are stored off-chain while integrity is anchored on-chain, and secure delegation is possible by means of proxy re-encryption.

A separate paragraph discusses architectural patterns for blockchain-based data placement, given their relevance for understanding how blockchain components are integrated into broader systems and how trade-offs among security, scalability, and data governance are achieved.

Blockchain-based data management, identity and access control

Numerous blockchain-based platforms address data management, identity, and access control. Approaches range from private-chain authorization mechanisms [12] and permissioned identity management for EHRs [27] to token-based authentication using NFTs [38]. In HSPBCI [12], access to patient records is enforced through a private blockchain integrated with cloud IoT services, resulting in faster authorization and encryption operations. Rao and Naganjaneyulu [27] similarly use a permissioned blockchain to ensure that only authorized entities can retrieve EHR data.

Authentication schemes for IoMT devices have also been explored. For instance, Jia et al. [15] propose a blockchain-assisted protocol that strengthens device-level identity verification. The Ancile framework [8] leverages smart contracts to enforce access rules and improve EHR interoperability across institutions.

Hierarchical architectures have also been explored to balance security and scalability. HierChain [2] routes highly sensitive data through a more secure, but less scalable, blockchain while delegating less sensitive information to a faster one, supported by fog nodes for preprocessing. Hyperledger-based platforms such as the solution presented in [30] adopt a dual-network model combining public and private ledgers to separate medical activity logging from internal process management.

Token-based solutions also play a role in identity and access control. Yang et al. [38] integrate FHIR-based EMR structures with Ethereum smart contracts, using ERC-1155 NFTs for authentication and anchoring EMR integrity on-chain. FHIRChain [40] follows a related approach, adopting FHIR for standardized data exchange and storing only reference pointers and permission metadata on-chain to reduce storage overhead while maintaining auditability.

Hybrid Storage and Integrity Verification

Hybrid storage architectures are commonly adopted to address scalability and performance constraints in blockchain-based healthcare systems. Theodouli et al. [33] discuss blockchain-supported healthcare data exchange with a focus on interoperability, although the integration of highly scalable distributed databases (e.g., Cassandra) is still underexplored in existing work.

Ensuring the integrity of medical records is essential when data remain stored off-chain. Merkle trees are widely used for this purpose. Yang et al. [38] anchor EMR hashes on Ethereum, enabling the detection of unauthorized modifications without exposing the underlying content. Ancile [8] implements similar cryptographic checks but keeps identity and access management logically separate from integrity verification.

Merkle-tree techniques are also employed in decentralized identity systems. For example, BDIMS [17] fragments identity attributes into Merkle leaves, allowing verification through Merkle proofs without revealing sensitive information. Similarly, Health-ID [14] enables patients and healthcare providers to authenticate across different eHealth domains using blockchain-backed identifiers.

In contrast to these approaches, BlockHealth more tightly integrates identity, access control, and integrity verification by binding permission hashes and EMR commitments directly to NFT-based identities, thereby supporting verifiable and dynamic permission updates.

Patient-centric Delegation-Based Data Sharing

Delegation mechanisms allow controlled sharing of medical data. Ancile [8] uses proxy re-encryption (PRE) to enable patients to authorize other providers without exposing private keys. Wang et al. [36] propose a hybrid-chain scheme (HSHB) that separates local and inter-institution data flows, although it does not provide patient-centric delegation.

More advanced cryptographic approaches combine ABE or PRE with off-chain IPFS storage. Liu et al. [19] use attribute-based signature encryption and PRE to allow fine-grained access to encrypted EMRs. Conditional PRE (CPRE) [18] enables ciphertext transformation only when embedded conditions are satisfied; a smart contract verifies the correctness of the condition before releasing results. Chaotic encryption combined with blockchain has also been explored for securing medical images during cloud transmission [29].

BlockHealth adopts the Umbral PRE scheme, which supports efficient delegation and key separation, and integrates it with NFT-based identity and permission anchoring.

Architectural Patterns: Hybrid Models, Off-Chain Coordination, and Distributed Oracles

Beyond healthcare-specific solutions, recent studies examine architectural patterns for blockchain-based applications, emphasizing the need to balance decentralization, coordination, and scalability. The systematic review by Six et al. [31] identifies a set of recurring design patterns, such as the Oracle, Reverse Oracle, and Off-chain Data Storage patterns, that explicitly model interactions between blockchain and external services. These patterns show that practical systems often rely on auxiliary off-chain components to mediate data access, provide external computation, or manage operations that are unsuitable for smart contracts alone. Notably, these off-chain components may be centralized or partially distributed, depending on performance and governance constraints.

Recent confidential-data exchange frameworks reinforce this observation. CAKE [20] adopts a hybrid architecture in which a Data Manager and a Key Manager operate off-chain to handle encrypted slices of documents, while the blockchain stores only integrity-relevant metadata. Although designed for confidentiality, CAKE's architecture clearly illustrates that central coordination services remain essential even when blockchain is used as a trust anchor. Its successor, MARTSIA [16], moves towards a more distributed model by adopting multi-authority ABE for key management, thereby reducing reliance on a single coordinator. However, the authors explicitly acknowledge that fully

Table 5
Comparison of representative blockchain-based healthcare systems.

Work	Identity & Access Control	Integrity Mechanism	Storage Strategy	Data Sharing / Delegation
HierChain [2]	No explicit identity model	On-chain hashing	Multi-tier blockchain hierarchy (public/consortium/private)	No delegation
HSPBCI [12]	Private-chain permissions	On-chain hashing	Cloud IoT private chain	No delegation
Ancile [8]	Smart-contract RBAC	Cryptographic integrity checks	Encrypted off-chain storage	PRE
FHIRChain [40]	Token-based identity	Hash pointers	Off-chain FHIR resources	No delegation
HSHB [36]	Permissioned blockchain	On-chain validation	Private + consortium blockchains	Potential for PRE
Yang et al. [38]	NFT-based identity (ERC-1155)	Merkle-tree commitment	External DB for EMRs	No delegation
Lin et al. [18]	Attribute-based access control	Blockchain-logged conditions	IPFS encrypted resources	Conditional PRE
BDIMS [17]	Decentralized identity fragments	Merkle-tree commitment	Off-chain personal data fragments	No delegation
BlockHealth	NFT identity + permission hashing	Merkle-tree commitment + on-chain anchors	Distributed Cassandra	Umbral PRE delegation

distributed key-management introduces non-trivial overhead and is not always preferable in domains with established institutions or regulated data flows.

BlockHealth aligns with these architectural principles by separating identity and integrity anchoring (handled on-chain) from data storage and access mediation (handled off-chain), while also remaining compatible with the introduction of distributed coordination components in future versions of the framework.

9. Conclusion

In this paper, we have proposed BlockHealth, a comprehensive framework that integrates blockchain technology, distributed databases, and advanced cryptographic techniques to create a secure, scalable, and efficient healthcare data management system. By leveraging the Ethereum blockchain and Non-Fungible Tokens (NFTs) for identity management and access control, we ensure secure authentication and fine-grained permissions for participants within the healthcare network. Data integrity is maintained using Merkle Trees, with the Merkle Root stored on-chain to verify the integrity of medical records stored off-chain. To address scalability and compliance with data protection regulations such as GDPR, we implement a distributed database architecture using Apache Cassandra, providing high availability and data replication across multiple data centers. Secure and confidential data sharing among authorized participants is enabled through proxy re-encryption schemes, allowing patients to delegate decryption rights to healthcare providers without revealing their private keys.

Our evaluation demonstrates that the BlockHealth framework effectively balances security and performance by minimizing on-chain interactions and optimizing off-chain data management. The use of distributed databases ensures efficient handling of large volumes of data, while the blockchain provides immutable records and robust security features. Compared to existing solutions, BlockHealth advances the state of the art by providing enhanced security, scalability, and compliance in healthcare data management.

9.1. Future Developments

While our framework addresses many challenges in healthcare data management, there are areas for future enhancement and exploration:

Handling Transaction Failures: In blockchain networks, transactions may fail due to network issues, gas price fluctuations, or smart contract errors. Implementing robust error-handling mechanisms is crucial to ensure system reliability. Future work could focus on developing retry policies, transaction monitoring tools, and failover strategies to handle transaction failures gracefully, ensuring data consistency and integrity.

Optimizing Merkle Tree Updates: As medical records are frequently updated, continually recalculating and storing new Merkle Roots on-chain after each update may become

inefficient. Implementing batch update mechanisms or designing incremental Merkle Tree update strategies could improve performance. Exploring methods to update the Merkle Root after multiple record modifications, while maintaining data integrity, would enhance the system's scalability.

Biometric Identification Integration: Enhancing security measures by integrating biometric identification linked to the minted NFTs could provide stronger authentication. By incorporating biometric data, such as fingerprint or iris scans, into the identity verification process, the framework can prevent unauthorized access even if NFT credentials are compromised. Future research could investigate secure and privacy-preserving methods to integrate biometric authentication with blockchain-based identity management, ensuring compliance with data protection regulations.

Distributed oracle integration: A further line of future development concerns the possible introduction of a distributed oracle to validate and submit integrity-related updates to the blockchain. While such an approach could reduce reliance on a single institutional component, its adoption in healthcare environments requires careful evaluation. Hospitals differ significantly in their IT infrastructures, governance models, and regulatory constraints, and a distributed protocol would need to operate reliably across these heterogeneous settings. In addition, the use of multiple validating nodes may introduce coordination overhead and impact system latency, requiring a dedicated performance assessment. For these reasons, the integration of a distributed oracle is identified as a promising but long-term enhancement, whose feasibility and benefits must be weighed against the additional organizational and operational complexity it entails.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit Author Statement

Chiara Braghin: Methodology, Writing- Reviewing and Editing, Formal analysis. Stelvio Cimato: Conceptualization, Writing- Reviewing and Editing. Simone Pesci: Conceptualization, Software, Investigation, Writing- Original draft preparation, Writing- Reviewing and Editing. Elvinia Riccobene: Supervision, Methodology, Writing- Reviewing and Editing, Project administration, Funding acquisition.

References

[1] Imad Aad. *Zero-Knowledge Proof*, pages 25–30. Springer Nature Switzerland, Cham, 2023.

[2] Vidushi Agarwal and Sujata Pal. Hierchain: A hierarchical-blockchain-based data management system for smart healthcare. *IEEE Internet of Things Journal*, 11(2):2924–2934, 2024.

[3] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1âĂŞ30, February 2006.

[4] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[5] Vitalik Buterin. A Next-Generation Smart Contract and Decentralized Application Platform. White Paper, 2014.

[6] Omar Cheikhrouhou, Khaleel Mershad, Maryline Laurent, and Anis Koubaa. Blockchain and Emerging Technologies for Next Generation Secure Healthcare: A Comprehensive Survey of Applications, Challenges, and Future Directions. *Blockchain: Research and Applications*, page 100305, 2025.

[7] European Commission, Directorate-General for Health, and Food Safety. *Opinion on assessing the impact of digital transformation of health services*. Publications Office, 2019.

[8] Gaby G. Dagher, Jordan Mohler, Matea Milojkovic, and Praneeth Babu Marella. Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, 39:283–297, 2018.

[9] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. ERC-721: Non-Fungible Token Standard. Ethereum Improvement Proposal, 2017.

[10] European Parliament and Council. Regulation (EU) 2016/679: General Data Protection Regulation. Official Journal of the European Union, 2016.

[11] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[12] Sangeeta Gupta, Premkumar Chithaluru, Thompson Stephan, Shaik Nafisa, and Sandeep Kumar. Hspbci: a robust framework for secure healthcare data management in blockchain-based iot systems. *Multimedia Tools and Applications*, pages 1–25, 2024.

[13] Health Level Seven International (HL7). Fhir: Fast healthcare interoperability resources. Standard Specification, 2014.

[14] Ibrahim Tariq Javed, Fares Alharbi, Badr Bellaj, Tiziana Margaria, Noel Crespi, and Kashif Naseer Qureshi. Health-ID: A Blockchain-Based Decentralized Identity Management for Remote Healthcare. *Healthcare*, 9(6), 2021.

[15] Xiaoying Jia, Min Luo, Huaqun Wang, Jian Shen, and Debiao He. A blockchain-assisted privacy-aware authentication scheme for internet of medical things. *IEEE Internet of Things Journal*, 9(21):21838–21850, 2022.

[16] Michele Kryston, Edoardo Marangone, Claudio Di Ciccio, Daniele Friolo, Eugenio Nerio Nemmi, Mattia Samory, Michele Spina, Daniele Venturi, and Ingo Weber. Martsia: A tool for confidential data exchange via public blockchain. In *Advanced Information Systems Engineering Workshops*, volume 557 of *Lecture Notes in Business Information Processing*, pages 173–180. Springer, 2025.

[17] Hoang Viet Anh Le, Quoc Duy Nam Nguyen, Nakano Tadashi, and Thi Hong Tran. Blockchain-Based Decentralized Identity Management System with AI and Merkle Trees. *Computers*, 14(7), 2025.

[18] Gaofan Lin, Haijiang Wang, Jian Wan, Lei Zhang, and Jie Huang. A blockchain-based fine-grained data sharing scheme for e-healthcare system. *Journal of Systems Architecture*, 132:102731, 2022.

[19] Guijiang Liu, Haibo Xie, Wenming Wang, and Haiping Huang. A secure and efficient electronic medical record data sharing scheme based on blockchain and proxy re-encryption. *J. Cloud Comput.*, 13(1), February 2024.

[20] Edoardo Marangone, Claudio Di Ciccio, Mattia Samory, Daniele Friolo, Michele Kryston, and Daniele Venturi. Cake: A tool for fine-grained confidential data sharing on blockchain. In *Business Process Management Forum*, volume 499 of *Lecture Notes in Business Information Processing*, pages 247–264. Springer, 2024.

[21] Ralph C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[22] Avi Mizrahi, Noam Koren, and Ori Rottenstreich. Optimizing Merkle Proof Size for Blockchain Transactions. In *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 299–307, 2021.

[23] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. White Paper, 2008.

[24] National Institute of Standards and Technology. Guide for Conducting Risk Assessments. Technical Report NIST Special Publication 800-30 Revision 1, U.S. Department of Commerce, September 2012. Available at: https://doi.org/10.6028/NIST.SP.800-30r1.

[25] David NuÃśez. Umbral: A Threshold Proxy Re-Encryption Scheme. NuCypher Technical Paper, 2018.

[26] Witek Radomski, Andrew Cooke, Philippe Castonguay, James Therien, Eric Binet, and Ronan Sandford. ERC-1155: Ethereum Multi Token Standard. Ethereum Improvement Proposal, 2018.

[27] Rama Rao Katru and Satuluri Naganjaneyulu. Permissioned healthcare blockchain system for securing the ehrs with privacy preservation. *IngÃľnierie des SystÃĺmes d'Information*, 26(4):365–372, 2021.

[28] Meni Rosenfeld. Overview of Colored Coins, 2012.

[29] Usman Shahid, Shamsa Kanwal, Mahwish Bano, Saba Inam, Manal Elzain Mohamed Abdalla, and Zaffar Ahmed Shaikh. Blockchain driven medical image encryption employing chaotic tent map in cloud computing. *Scientific Reports*, 15(1):6236, 2025.

[30] Zaffar Ahmed Shaikh, Abdullah Ayub Khan, Lin Teng, Asif Ali Wagan, and Asif Ali Laghari. Biomt modular infrastructure: the recent challenges, issues, and limitations in blockchain hyperledger-enabled e-healthcare application. *Wireless Communications and Mobile Computing*, 2022(1):3813841, 2022.

[31] Nicolas Six, Nicolas Herbaut, and Camille Salinesi. Blockchain software patterns for the design of decentralized applications: A systematic literature review. *Blockchain: Research and Applications*, 3(1):100061, 2022.

[32] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter*, volume 1988, pages 191–202, 1988.

[33] Anastasia Theodouli, Stelios Arakliotis, Konstantinos Moschou, Konstantinos Votis, and Dimitrios Tzovaras. On the design of a blockchain-based system to facilitate healthcare data sharing. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1374–1379, 2018.

[34] U.S. Congress. Health Insurance Portability and Accountability Act of 1996 (HIPAA). Public Law 104-191, 1996.

[35] Gang Wang and Mark Nixon. SoK: tokenization on blockchain. In *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '21, New York, NY, USA, 2022. Association for Computing Machinery.

[36] Taochun Wang, Qingshan Wu, Jian Chen, Fulong Chen, Dong Xie, and Huimin Shen. Health data security sharing method based on hybrid blockchain. *Future Gener. Comput. Syst.*, 153(C):251âĂŞ261, April 2024.

[37] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Yellow Paper, 2014.

[38] Ching-Nung Yang, Hsin-Chuan Kuo, and Hsiang-Han Cheng. Ensuring fhir authentication and data integrity by smart contract and blockchain enabled nft. In *Proceedings of the 2023 7th International Conference on Medical and Health Informatics*, ICMHI '23, page 123âĂŞ128, New York, NY, USA, 2023. Association for Computing Machinery.

[39] Ibrar Yaqoob, Khaled Salah, Raja Jayaraman, and Yousof Al-Hammadi. Blockchain for healthcare data management: opportunities, challenges, and future recommendations. *Neural Computing and Applications*, pages 1–16, 2022.

[40] Peng Zhang, Jules White, Douglas C. Schmidt, Gunther Lenz, and S. Trent Rosenbloom. Fhirchain: Applying blockchain to securely and scalably share clinical data. *Computational and Structural Biotechnology Journal*, 16:267–278, 2018.

[41] Wei Zhang. Blockchain-based solutions for clinical trial data management: a systematic review. *Metaverse basic and applied research*, (1):8, 2022.