Journal of Computer Science

Research Article

Blockchain-Enabled Privacy-Preserving Access Control for Electronic Health Records Using Smart Contracts

G. Bhanu Prasad and M. Venkateswara Rao

Department of Computer Science and Engineering, Gitam Deemed to Be University, Visakhapatnam, Andhra Pradesh, India

Article history
Received: 21-08-2025
Revised: 24-04-2026
Accepted: 20-05-2026

Corresponding Author:
M. Venkateswara Rao
Department of Computer Science and Engineering, Gitam Deemed to Be University, Visakhapatnam, Andhra Pradesh, India
Email: bhanu.gorantla2020@gmail.com

Abstract: Electronic medical records are crucial for providing high-quality healthcare, but they also raise serious privacy and security issues. Traditional centralised storage systems are vulnerable to manipulation, hacking, and illegal access. This study suggests a smart contract-based blockchain architecture that provides decentralised, unaltered, and privacy-preserving access control for EMRs. The model integrates cryptographic techniques like AES-256 encryption and SHA-256 hashing for private data encryption, hybrid off-chain storage using IPFS for scalability, and Ethereum smart contracts for role-based rights management. Sequential procedures and computational formulas are used to explain the methods for secure storage and access verification. When compared to the conventional method, performance evaluation shows that the transaction latency for access verification is very low and that it is highly resistant to replay and collusion attacks with improved auditability. This approach has a security, transparency, and interoperability advantage with a marginally lower computing cost, according to the comparative study. Therefore, the provided architecture can serve as a workable and scalable solution for secure EMR management in dispersed healthcare settings, ad hoc for critical operational and compliance requirements. The integration of federated learning for privacy-preserving analytics will be the focus of future research. This concept would be effective in fostering patient trust and creating systems that guarantee safe data exchange between healthcare providers.

Keywords: Blockchain, Electronic Medical Records, Smart Contracts, Access Control, Data Privacy, Cryptographic Security

Introduction

Electronic Medical Records (EMRs) are generally seen as an important part of modern healthcare systems (Chauhan et al., 2025; Wilson and Garcia, 2024; Gupta et al., 2022) They allow health data about patients to be digitised, shared, and stored for a long time. EMRs make sure that only authorised healthcare professionals can see complete and correct medical records. This makes clinical processes and decision-making based on evidence more efficient and improves patient outcomes and continuity of care (Al-Khasawneh et al., 2024; Chamola et al., 2024). As EMRs spread quickly around the world (Kumar and Patil, 2024; Wang et al., 2024), it's getting harder and harder for healthcare companies to protect the privacy, security, and integrity of sensitive patient data. The healthcare industry has been slowly experiencing more and more data breaches, which leave millions of patient records open to people who aren't supposed to see them and erode trust in digital healthcare platforms (Zhang et al., 2020a; Mani et al., 2021). The most prevalent centralised EMR system (Murala et al., 2025) stores medical data in private files managed by a trusted group. These technologies can improve operations in controlled environments, but they can also be single points of failure and expose data to ransomware, insider threats, and unauthorised alterations (Shuaib et al., 2021; Liu et al., 2021). Centralised systems sometimes lack open auditing tools or fine-grained access control, making it difficult for firms to demonstrate compliance or for patients to identify who has accessed their medical records (Nguyen et al, 2019; Tanwar et al., 2020).

Regulations like the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA) make it very important to protect healthcare data and make sure that people are

DOI: 10.3844/jcssp.2026.1881.1893

held accountable (Hussien et al., 2020; Akkaoui et al., 2020). These rules make the problems with traditional EMR management even clearer.

Because it is decentralised, can't be changed, and can make transactions clear and verifiable without relying on a single trusted authority, blockchain technology has become a potential way to deal with these problems (Brun and Ali, 2022; Dagher et al., 2018). Distributed consensus is a feature of blockchain systems that lets different parties check transactions on their own and keep a shared, permanent record (Murala et al., 2026). Smart contracts, which are self-executing algorithms based on blockchain that record all data access and modification events and automatically enforce access control limits (Zhang et al., 2020a; Chen et al., 2019), make this idea even better. Because of these features, blockchain works especially well for managing private health information when there are a lot of people involved. There are some good things about blockchain-based EMR systems, but they also have some problems that happen in the real world. Several suggested systems can't grow because they store data on the blockchain, have high transaction and computation costs, and don't support dynamic or fine-grained role-based access control (Zhang et al., 2020a; Fan et al., 2020). Some solutions also fail to address compatibility with current healthcare information systems and operational operations (Murala, 2025).

Terminology Clarification: To clarify, this study calls patient digital medical records stored and accessed by healthcare systems Electronic Medical Records (EMRs). Although EHRs and EMR are often used interchangeably in modern literature, this work uses EMR to keep things clear and consistent. Effective, privacy-protecting EMR systems that balance decentralisation, scalability, regulatory compliance, strong cryptographic guarantees, and patient-centered access control need more research.

New studies suggest that blockchain technology can improve healthcare systems beyond data security. Blockchain-enabled healthcare optimisation systems with decentralised and tamper-resistant architectures improve data security, stakeholder confidence, and collaborative decision-making (Wang et al., 2018).

These studies emphasise how crucial it is to combine intelligent decision-support systems with safe data-sharing methods in healthcare settings. In order to solve privacy, integrity, and trust issues in electronic medical record systems, these results support the adoption of blockchain-based access control and secure data management techniques, as suggested in this paper.

Contributions

This research's main goals are to:

(i) Create a decentralised, privacy-preserving framework for secure electronic medical record management using blockchain technology
(ii) Implement role-based, fine-grained access control enforced through smart contracts with explicit patient consent
(iii) Achieve scalability by integrating off-chain encrypted storage while maintaining cryptographic integrity and auditability
(iv) Empirically assess the framework in terms of encryption overhead, access l the architectural design, implementation decisions, and experimental assessment described in this study are guided by these goals. This research proposes a smart contract-based blockchain system for secure and private EMR access in order to address the aforementioned problems. The main contributions of this work are

- Smart Contract Architecture: We create a role-based access control framework for Ethereum smart contracts, enabling finer access for various user groups, including patients, healthcare providers, and administrators (Murala et al., 2025)
- Role-Based Permissions: The architecture permits dynamic role assignments and policy modifications, and blockchain decisions are final (Kumar and Patil, 2024)
- Hybrid Storage Model: The method is scalable by combining hybrid off-chain storage on IPFS with blockchain metadata anchoring to safely store large EMR files without stressing the blockchain (Murala et al., 2025)
- Security and Performance Assessment: The model was tested using fictitious workloads. Even with diverse attacks, the findings show reduced access latency and good performance. These attacks included replay, collusion, and unauthorised releases (Zhang et al., 2022a)

Literature Survey

Recent study on the enhancement of security, privacy, and interoperability in electronic medical records has given blockchain technology more widespread consideration. The historical evolution of privacy-preserving blockchain projects in the medical field is highlighted in a systematic study by Chauhan et al. (2025), who also point out that the shortcomings are the lack of scalability and suitable access control measures. In their paper, (Wilson and Garcia, 2024) addressed the topic of blockchain-based smart healthcare management systems for chronic disease monitoring, claiming that although trust has increased, the issue of integrating decentralised systems with traditional infrastructures still persists. A medical blockchain with smart contracts for safe data exchange and privacy protection was described by Gupta et al. (2022). Their study showed that a fine-grained access control system could be used, but it also brought attention to the overhead of implementing smart contracts.

The model proposed by Al-Khasawneh et al. (2024) is a secure blockchain framework for healthcare record management emphasizing modular design to meet evolving regulatory demands. (Chamola et al., 2024). discussed the integration of AI with blockchain EMRs and suggested decentralized identity management mechanisms for enhancing adaptive access decisions. Kumar and Patil (2024) considered decentralized multi-authority Public Key Infrastructures (PKI), in the context of EMR sharing, attempting to solve the problem of distributing trust among multiple stakeholders.

A myriad of hybrid storage solutions have been looked into. Wang et al. (2024) discussed a hybrid blockchain architecture for secure sharing of health data, where the on-chain metadata is balanced by off-chain encrypted records in an attempt to balance scalability and immutability. Zhang et al. (2022) showed the secure sharing of attribute-based PHRs with blockchain systems that support cryptographic enhancements to role-based access control.

Mani et al. (2021) proposed Hyperledger HealthChain by integrating IPFS, thus achieving the scalable storage of blockchain-based EMRs with data provenance. Murala et al. (2025) proposed a blockchain architecture that combines IPFS and smart contracts to provide efficient storage and fine-grained access control enforcement.

Some of these studies form the basis for building secure, privacy-preserving EMR frameworks. However, lingering challenges in achieving the dynamic role-based access control, resisting sophisticated attacks, and offering a fully decentralized identity management spur interest in solutions such as the proposed model.

A lot of new study shows that blockchain can help make healthcare data safer, more interoperable, and more trustworthy between people who are in different places. Hybrid blockchain solutions that combine off-chain encrypted storage with on-chain policy enforcement improve scalability and security. All of these initiatives strengthen the proposed design and support hybrid storage formats and blockchain-based access control for private healthcare data. Even after extensive research, blockchain-based EMR solutions have many issues. Fully on-chain solutions are good at not being changed, but they're hard to scale and expensive to utilise in clinical settings. Hybrid blockchain–IPFS solutions expand storage, but they use static permission models or off-chain access management, making it tougher for patients to govern and monitor. Most current research, especially on access latency, gas efficiency, and regulatory compliance, lacks real-world evidence. The suggested structure was built for these issues. It uses cryptographically protected off-chain storage, lightweight blockchain metadata, and on-chain role-based access control to provide a balanced, scalable, and auditable EMR management solution.

Proposed Model

The suggested smart contract-based blockchain system's operational workflow and architectural architecture are covered below. Expands on part 1's entity duties. This design has three layers: Client application, Ethereum smart contract, and IPFS-based off-chain storage. Patients, healthcare providers, management, and external auditors comprise the architecture. In the proposed system, only the smart contract regulates entry. Cryptographic metadata is released when the smart contract checks user jobs and patient consent flags. These stay on the chain, eliminating the need for a middleman. It eliminates single points of trust and lets all network users observe and check access rules. The system's workflow includes registering users and making key pairs, encrypting EMRs and uploading them to IPFS, securing metadata on the blockchain with smart contracts, healthcare providers requesting access, smart contracts enforcing role and consent verification, and auditing all blockchain access events. The unified System Architecture and Flow Diagram (Figure 1) shows this process by merging conceptual architecture and data flow.

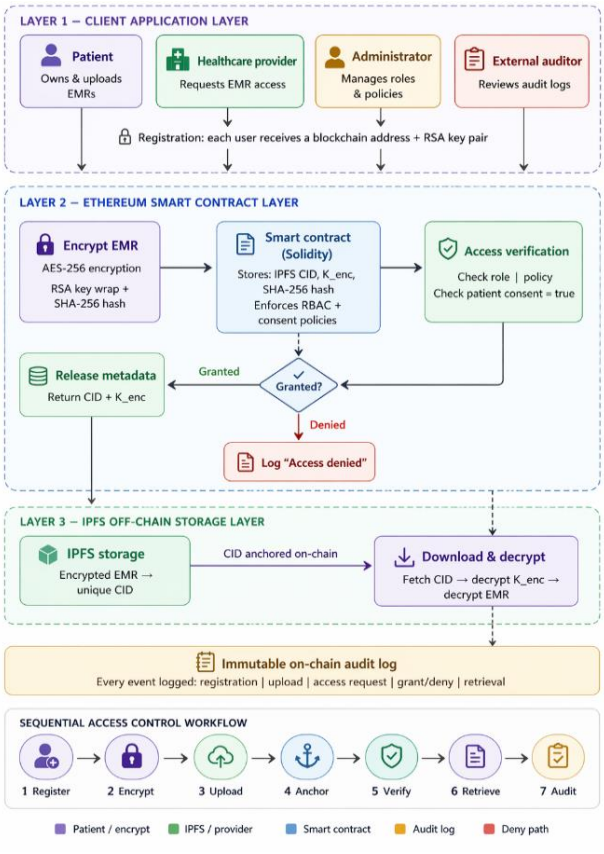

Fig. 1: Proposed system aarchitecture and secure access flow

DOI: 10.3844/jcssp.2026.1881.1893

Figure 1 displays the blockchain-based framework's system design and EMR access protocol from start to finish. The graphic depicts the Patient, Healthcare Provider, Administrator, and External Auditor. Also shown is the three-layer technology stack: Client Application, Ethereum Smart Contract, and IPFS Off-Chain Storage. From user registration to EMR encryption, IPFS upload, on-chain metadata anchoring, smart contract-enforced access verification, and persistent audit recording, sequential access control is built in Figure 1 summarises the suggested system design and secure access flow:

- ✓ Registration: Each participant is registered while receiving a blockchain address and public-private key pair
- ✓ Encryption and Upload: EMRs are AES-256 encrypted, and the AES key should be encrypted with the patient's RSA public key. Encrypted files are uploaded to IPFS
- ✓ Metadata Management: IPFS hash, the AES key encrypted with the related private key of the accessing agent, and SHA-256 digest of the encrypted file are recorded on the blockchain through a smart contract
- ✓ Requesting Access: Access is requested by healthcare providers through the smart contract
- ✓ Verifying a Role: Smart contracts verify the policy role in releasing decryption metadata
- ✓ Audit Logs: All events are stored immutably on-chain

Methodology

By distributing a framework utilising blockchain technology and smart contracts, this study proposes to ensure the best security and exchange safeguards in the management of electronic medical records (MHRs) (Dagher et al., 2018). To protect privacy, data integrity, and transparency, the method uses distributed storage, cryptography, and role-based access control (Murala et al., 2025) User registration, smart contract deployment, data encryption and storage, permissioned access workflows, and system implementation are the many stages of the framework (Zhang et al., 2020b).

User Registration and Role Assignment

In order to construct their profiles, users first register using a user-friendly client program (Murala et al., 2025) During registration, the system creates a distinct blockchain address for every user, which serves as their decentralised identity (Zhang et al., 2020b) Meanwhile, an RSA cryptographic key pair is automatically created, with the public key used to encrypt data or verify signatures, and the private key to decrypt data and sign transactions (Chen et al., 2019). Each user is granted a role that is already defined within the framework (e.g., patient, health care provider, administrator); the role determines what access rights a given user possesses (Dagher et al., 2018). The role is then securely stored in the smart contract so that the access control policies may refer to it for validation without depending on a central authority (Zhang et al., 2020a).

Smart Contract Deployment

The smart contracts are programmed in Solidity, the default language for Ethereum (Zhang et al., 2020b). These contracts specify and enforce access control policies, maintain records of consents wherein service providers have been given consent by patients, and keep tamper-proof logs of every interaction and transaction (Murala et al., 2025). The compilation and deployment of contracts to the Ethereum networks are performed through the Remix IDE. Once deployed, the contracts are reachable by any authorized party through their blockchain addresses. Through the use of smart contracts, EMR access rules remain transparent, verifiable, and unchangeable (Dagher et al., 2018). Ethereum was chosen as the blockchain platform due to its mature smart contract ecosystem, extensive developer tooling, and support for expressive access control logic through Solidity. Unlike permissioned platforms such as Hyperledger Fabric, Ethereum enables transparent and verifiable execution of access policies without reliance on a centralized consortium authority, aligning with the patient-centric trust model adopted in this work.

Data Encryption and Storage

Prior to upload, every EMR file gets encrypted with AES-256 to keep confidentiality. AES-256 is a symmetric cipher that is very strong in respect to any unauthorized disclosure of data. Afterward, the symmetric AES key itself is encrypted using the patient's RSA public key so that only the patient or authorized parties may later decrypt the file (Murala et al., 2025). Atop this encrypted file, a SHA-256 hash is computed to serve as an integrity check and hence detect any forging occurring during storage or transmission (Wang et al., 2018). The encrypted EMR is then uploaded onto IPFS, a decentralized file storage network that returns a unique content identifier (CID) (Azaria et al., 2016). This CID, together with the encrypted AES key and SHA-256 hash, is stored in a blockchain smart contract so as to bind the data with access policies (Zhang et al., 2018). AES-256 was chosen for EMR encryption because it is secure, quick, and widely utilised in healthcare and regulatory standards. Share keys safely with RSA, which is easy to use, interoperable with other systems, and public key platforms. Novel hybrid encryption algorithms exist, but the AES-RSA combination is a reliable and simply deployable option that works well in healthcare settings

DOI: 10.3844/jcssp.2026.1881.1893

where reliability and standard compliance are more important than testing novel cryptographic structures. IPFS was utilised for off-chain storage to avoid blockchain issues with huge medical datasets. The solution reduces blockchain storage costs while maintaining data integrity by only keeping cryptographic references and metadata on-chain. The present architecture emphasises simplicity and usability. Even if improved off-chain proving methods can increase confidence, it saves such optimisations for further research.

Access Control Workflow

A client tool helps healthcare providers access a patient's EMR. The smart contract checks the request using the provider's blockchain address, the patient's clear permissions, and any access control limitations (Murala et al., 2025). The smart contract returns the IPFS CID and encrypted AES key with permission. The service decrypts the AES key with its RSA private key after receiving the encrypted EMR file from IPFS (Azaria et al., 2016). Decrypting the EMR file with the AES key. Every access request approved or denied is forever recorded in the blockchain. Anyone can inspect this access history (Zhang et al., 2018).

Implementation Environment

The Ganache local Ethereum test network, Remix IDE for smart contracts, and Web3.js for client-side were used to develop the prototype. Fake EMR files of 100 KB to 5 MB were utilised to circumvent patient data laws. Locally hosted IPFS nodes mimicked IPFS, and Python's cryptographic library performed cryptographic work. Ganache provides a safe and consistent testing environment free from transaction fees and network changes, but it does not emulate all aspects of a public blockchain deployment. In particular, network congestion, validator demand, and EIP-1559 base fee adjustments affect Ethereum mainnet gas costs. Petrol on the Ethereum mainnet costs 25-40 gwei when traffic is moderate. This can significantly impact the cost of writing-intensive tasks like uploading data. However, Layer-2 systems like Polygon that use Ethereum smart contract standards have substantially lower petrol costs often less than 0.01% USD per transaction. Local configurations don't have content propagation or node availability delays like public IPFS networks. IPFS pinning services like Infura or Pinata should be used in production to keep data secure and retrievable. These pinning services fit the framework and should be used in real life. Experimental evaluation highlights these constraints, and Table 3 offers anticipated costs based on Polygon and the Ethereum mainnet to better model implementation.

Threat Model and Adversarial Assumptions

The security research of the proposed system uses a well-defined threat model to demonstrate how genuine adversaries act in distributed healthcare environments. The attacker likely has complete access to IPFS and the public blockchain network. They can monitor, intercept, and replay access requests and metadata. The enemy may deploy a collusion attack, in which authorised users cooperate to access electronic medical information without permission. The opponent might utilise IPFS to obtain protected EMR files or chain metadata to find private info.

The enemy's processing power is deemed to be low and unable to break SHA-256, RSA, and AES-256. Ethereum promises that attackers can't edit deployed smart contracts, interfere with prior blockchain transactions, or access authorised users' private cryptographic keys. People also believe that blockchain-stored job assignments and patient consent forms can't be altered or forged. The suggested structure is tested against replay attacks, collusion attacks, unauthorised access attempts, and data manipulation using these assumptions.

Key Management, Delegation, and Revocation Strategy

The suggested topology isolates authorised users with per-agent AES key re-encryption. When a consumer authorises N healthcare providers, their RSA public keys encrypt the AES symmetric key one at a time. The smart contract receives N unique secret key entries. Computer Overhead Analysis: RSA-2048 encryption of a 256-bit AES key takes 1-3 microseconds on typical hardware. Thus, re-encryption takes 10–30 milliseconds for N = 10 approved agents, 50–150 for N = 50, and 100–300 for N = 100. This extra labour is acceptable for healthcare access circumstances, since batch authorisation events occur less often than individual access requests.

Petrol Price Impact: With each new authorised agent, the chain storage of the matched encrypted AES key entry costs more. Each key store operation uses around 20,000 gas units, granting authorisation to N = 10 agents which use 200,000 gas units during setup. When you consider the long-term benefits of controlling who can view what, this one-time document charge is fair. We recommend Attribute-Based Encryption (ABE) for situations with many approved agents (N > 100). These allow all policy-compliant users to decode a single ciphertext without generating a new key. Future efforts should prioritise ABE.

Algorithms

Several cryptographic and access control mechanisms secure storage, regulate access, and ensure EMR integrity in the proposed system (Table 1). Each algorithm plays a different role in the complete workflow (Chen et al., 2019).

DOI: 10.3844/jcssp.2026.1881.1893

AES-256 Encryption

This AWS key storage scheme involves encrypting EMRs with AES-256 before storage so as to safeguard the confidentiality of the EMR. A random 256-bit symmetric key is generated for every file, and the plaintext EMR is encrypted into a ciphertext by applying the AES function:

$$C = AES_{256}(E,K)$$

Where E stands for the original EMR file, and K is the symmetric key used for encryption. Such an encryption makes it so that anybody intercepting the file or unauthorizedly accessing it will view content that is completely unintelligible (Dagher et al., 2018).

RSA Encryption of the AES Key

Since it is imperative that the AES key be kept secure, it needs to be encrypted by public-key RSA encryption. The symmetric key is encrypted, producing an encrypted key:

$$K_{enc}=RSA\_Encrypt(K,PK_{patient})$$

The creation of such a mechanism would guarantee that the AES key can be decrypted for the EMR only by the patient or healthcare providers with the private RSA key (Zhang et al., 2020a).

SHA-256 Hashing

For filesystem integrity, SHA-256 produces a cryptographic hash of the encrypted EMR file:

$$H = SHA256(C)$$

The hash value is stored in the smart contract. Upon retrieval, one recomputes the hash and compares it to ensure that the file has not been tampered with during storage or transit (Akkaoui et al., 2020).

Role-Based Access Control Algorithm

A Role-Based Access Control algorithm is implemented into a smart contract on the Ethereum blockchain to validate access requests. When the end user sends the contract an access request, it fetches the role assigned to the user by the system, the access policy associated with the EMR, and any consent record given by the patient (Dagher et al., 2018).

Audit trails of all attempted access are set down immutably on-chain, ensuring transparency.

Secure Storage and Retrieval Workflow

Encryption, decentralised storage, and controlled access are all coordinated via the framework's secure storage and retrieval workflow. After encryption and hashing, the ciphertext is uploaded to IPFS, which yields a unique content identifier (CID). The CID, the RSA-encrypted AES key, and the SHA-256 hash are registered in the smart contract (Zhang et al., 2018).

Table 1: Summary of Cryptographic Algorithms

Algorithm	Purpose	Input	Output
AES-256 Encryption	Encrypt EMR files to ensure confidentiality	Plaintext EMR file, AES key	Encrypted EMR ciphertext
RSA Encryption of AES Key	Secure the AES key for authorized decryption	AES key, patient's RSA public key	RSA-encrypted AES key
SHA-256 Hashing	Verify EMR file integrity	Encrypted EMR file	SHA-256 hash string
Role-Based Access Control	Enforce role-based authorization policies	User address, Document ID	Access decision (Granted/Denied)
Secure Storage and Retrieval Workflow	Coordinate encryption and controlled access	EMR file, patient's RSA public key	Encrypted file in IPFS, blockchain metadata

Pseudocode

Algorithm: SecureBlockchainBasedEMRFramework

```
Input:
   - EMR file E
   - Patient's RSA public key PK_patient
   - User U requesting access
   - Document identifier D

Output:
   - Securely stored encrypted EMR in IPFS
   - Access decision (Granted/Denied)
   - Plaintext EMR retrieved if access granted

// STEP 1: Generate Encryption Key
1. Generate a random 256-bit AES symmetric key K

// STEP 2: Encrypt EMR File
2. Encrypt EMR file E using AES-256 with key K
   C = AES256(E, K)

// STEP 3: Encrypt AES Key
3. Encrypt AES key K using RSA public key PK_patient
   Kenc = RSA_Encrypt(K, PK_patient)

// STEP 4: Compute File Hash
4. Compute the SHA-256 hash of the encrypted EMR
   H = SHA256(C)

// STEP 5: Upload Encrypted File to IPFS
5. Upload C to IPFS and receive Content Identifier CID

// STEP 6: Store Metadata on Blockchain
6. Store metadata {CID, Kenc, H, access policy} in the
Ethereum smart contract

// STEP 7: Wait for User Access Request
```

7. Await access request from user U for document D

// STEP 8: Access Verification
8. On receiving a request:
   a. Retrieve user role R = getRole(U)
   b. Retrieve document access policy P = getPolicy(D)
   c. Retrieve patient consent record C = getConsent(D, U)

// STEP 9: Evaluate Access
9. If (R is in P.roles) AND (C == True):
      a. Log "Access Granted" to blockchain
      b. Return {CID, Kenc} to user U
   Else:
      a. Log "Access Denied" to blockchain
      b. Deny access

// STEP 10: Retrieval and Decryption (If Access Granted)
10. If access granted:
      a. User downloads encrypted EMR C from IPFS using CID
      b. User decrypts Kenc with their RSA private key to recover AES key K
      c. User decrypts C using AES key K to restore plaintext EMR

Results and Discussion

The aforementioned framework universally executed the Ethereum blockchain (Ganache local test network), Solidity smart contracts, and IPFS for decentralized storage (Shuaib et al., 2021). Web3.js and a number of Python scripts were created to automate encryption, storage, and retrieval processes. The EMR files used for testing ranged in size from 100 KB to 5 MB, and all trials were conducted on a workstation with an Intel i7 processor and 16 GB RAM (Tanwar et al., 2020). Experiments will demonstrate the framework's viability and performance under controlled conditions. To circumvent ethical and legal issues with patient data, simulated workloads and small file sizes were used. For large-scale clinical installations, data should be considered indicative rather than decisive.

Experimental Methodology and Measurement Setup

The experimental study assessed the framework's functional correctness and computational performance under controlled conditions. All testing were run on a single Intel i7 workstation with 16 GB RAM.

Blockchain execution was simulated using a local Ethereum test network (Ganache) without actual transaction charges. Web3.js scripts were used for interactions, and the Remix IDE was used to deploy smart contracts. To simulate common medical papers including reports and diagnostic records, encrypted EMR files with sizes ranging from 100 KB to 5 MB were created.

Sampling Clarification

To get over ethical and legal restrictions, the dataset used for review was made up of artificially created EMR files rather than actual patient data. File sizes were chosen to represent EMR payloads that are frequently seen in healthcare systems. In order to systematically cover both authorised and unauthorised access scenarios, access-control experiments were conducted utilising a predetermined set of user roles and consent setups.

Measurement and Analysis Clarification

Gas consumption, hashing time, encryption time, and the accuracy of access-control decisions were among the performance measures. System-level timers averaged over several executions for each file size were used to measure the encryption and hashing times. The Ethereum transaction receipts were used to determine the petrol consumption. By contrasting smart contract decisions with anticipated results based on predetermined role and consent policies, access-control correctness was assessed. The provided results are meant to show relative performance trends and feasibility rather than absolute system throughput under large-scale deployment conditions due to the experimental setup's limited scale.

Encryption and Hashing Performance

AES-256 encryption times were measured for files of various sizes. Larger files took longer to encrypt, as would be expected; the encryption time varied from 5 ms for 100 KB files to 250 ms for 5 MB files. Similarly, SHA-256 hashing had the best computational performance, taking a time proportionate to the size of the input (Chen et al., 2019). Table 2 provides a summary of the findings.

Figure 2 shows that the AES-256 encryption time increases linearly with the size of an EMR file. For instance, a 100 KB file took nearly 5 milliseconds to encrypt, whereas a 5 MB one took approximately 250 milliseconds (Murala et al., 2025) This linear trend is an indication of the computational complexity of AES encryption; therefore, the bigger the files, the bigger the encryption overhead.

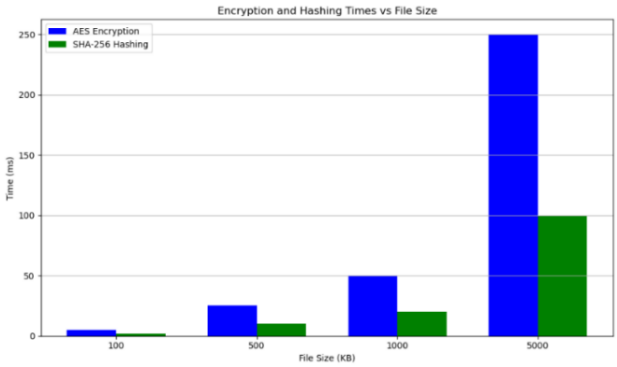

Fig. 2: AES Encryption Time vs File Size

DOI: 10.3844/jcssp.2026.1881.1893

Table 2: Encryption and hashing times

File Size (KB)	AES Encryption Time (ms)	SHA-256 Hash Time (ms)
100	5	2
500	25	10
1000	50	20
5000	250	100

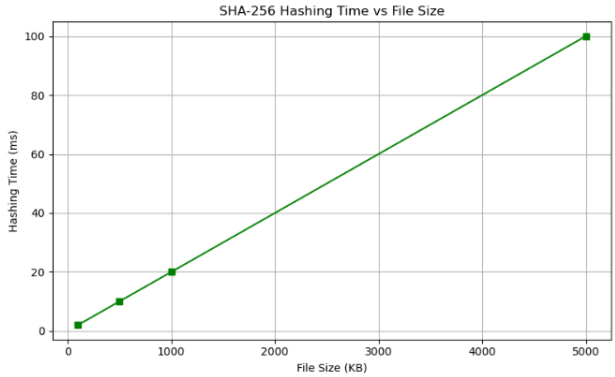

Fig. 3: SHA-256 Hashing Time vs File Size

Figure 3 shows the time it takes to compute SHA-256 hashes for EMR files of varying sizes. Hashing time scales linearly with file size, ranging from 2 ms for a 100-KB file to 100 ms for a 5 MB one, much like AES encryption (Dagher et al., 2018). This level of performance should be sufficient for real-time integrity verification in medical record systems.

Gas Consumption Analysis

The Ganache local test environment was used to compare how much gas the three main smart contract processes used. This environment provides a stable and regular way to find out the true cost of gas, regardless of how unpredictable the network is. Ethereum Virtual Machine (EVM) gas units specify how much computational power each activity requires: 100,000 for uploading metadata, 60,000 for confirming access, and 20,000 for retrieving metadata. Table 3 displays the projected cost of the Ethereum mainnet and Polygon (a Layer-2 Ethereum-compatible option) to help you comprehend these numbers. These numbers were based on a sample Ethereum mainnet petrol price of 30 gwei and an ETH price of $2,500 USD. They reflect early 2025 averages. A 50 gwei Polygon petrol price and $0.80 USD MATIC price were also applied. Metadata upload costs the most on the Ethereum mainnet. Table 3 shows that each operation costs $7–$10 USD. This might not work for clinical situations with a lot of patients. On the other hand, the same process on Polygon costs between $0.002 and $0.005 USD, which is three to four times less. This is a strong argument for using the suggested model on Layer-2 networks in real healthcare settings. The framework's smart contracts can be used on this network without any changes, and they work perfectly with Polygon's EVM implementation.

Three key smart contract processes' gas consumption is contrasted in Table 3 and the bar graph in Figure 4 (Hussien et al., 2020). Nearly 100,000 units of gas were used for uploading metadata, which adds data to the blockchain. In contrast, reading the stored metadata was the least expensive, requiring about 20k gas units, whereas verifying access authorisation required about 60k gas units (Zhang et al., 2020b). Because the suggested design has a small on-chain footprint, the observed petrol consumption is still feasible from a scalability standpoint. The technology avoids the exorbitant costs involved with storing huge medical files directly on the blockchain because it only stores metadata, access restrictions, and cryptographic hashes on-chain. Read-only procedures like metadata retrieval dominate system usage and have comparatively low fuel costs in big healthcare setups with frequent access requests. Additionally, the architecture is compatible with batch transaction methods and Layer-2 scaling solutions, which can drastically lower transaction latency and gas costs. In Table 3, you can see how the expected blockchain transaction prices for Polygon Layer-2 and Ethereum Mainnet metadata operations compare. It costs the most to upload metadata and the least to retrieve metadata because adding metadata uses more gas. Each transaction on Polygon costs a very small amount of a cent, but on Ethereum, each transaction costs several dollars. This shows that Layer-2 technologies make blockchain-based systems cheaper and easier to expand.

Table 3: Gas Consumption per Operation

Operation	Average Gas Consumption (units)
Upload Metadata	100,000
Verify Access Permissions	60,000
Retrieve Metadata	20,000

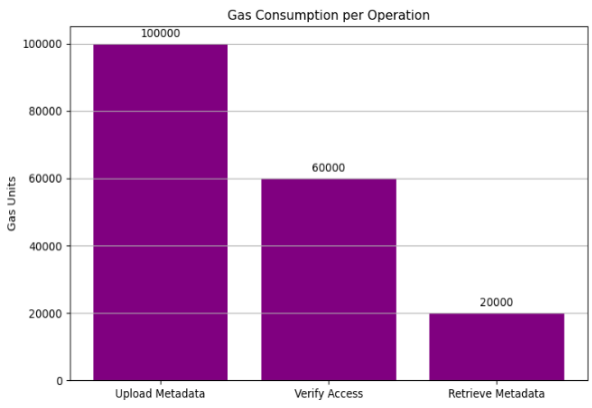

Fig. 4: Gas Consumption per Operation

DOI: 10.3844/jcssp.2026.1881.1893

Access-Control Verification

Thirty tries were simulated with different user roles, policies that allowed access, and flags that showed a patient's consent to see how well and reliably the control mechanism against access worked (Dagher et al., 2018). Some of the jobs that were copied were those of doctors, nurses, managers, and external auditors. Each possible access choice route was looked at both with and without patient consent (Murala et al., 2025) to learn more about them.

Test Results Showed 100% Conformity With the Expected Responses

- All requests under which the role was valid in the document policy and consent was given resulted in the successful retrieval of the CID and the RSA-encrypted AES key (Shuaib et al., 2021)
- If the role was unauthorized or if there was no consent, then the smart contract denied the access (Nguyen et al, 2019)
- Each attempt, whether granted or denied, was recorded on a blockchain that was immutably maintained. Thus, there is a verifiable audit trail (Murala et al., 2025)

The consistent enforcement of access policies in all cases ensures that the RBAC smart contract logic is flawed and faulty. While access is enforced, the other function is the logging, which guarantees traceability and non-repudiation for healthcare data protection regulation compliance such as HIPAA and GDPR (Tanwar et al., 2020).

Security and Privacy Analysis

An adversarial model is used to evaluate the safety and privacy of the proposed structure. Every claim about security is based on how the smart contracts work and what cryptography building blocks are used. To stop replay attacks, every time someone tries to access the smart contract, the current state of the blockchain is checked against it. As part of the smart contract's nonce-based process, each authorised entry session makes a unique transaction identifier. An access request that has already been caught and sent again will be turned down by the Ethereum mempool and the smart contract's internal state verification because it has an old or repeat transaction nonce. The smart contract was able to spot and reject 15 replay attacks that tried to send back legally valid access request transactions that had already been recorded in the simulated tests. No unauthorised data was released. This shows that the system can't be hacked to give more privileges based on replay. Defence Against Collusion Attacks: Separate RSA public keys for each approved recipient are used to encrypt the AES key. People who are working together can't access files that they haven't been given permission to access by a smart contract or decrypt EMRs for other patients, even if they share their decrypted data or information. To avoid passing on privileges without permission, the smart contract checks the role credentials and patient consent for each individual access try. In all replicated collusion tests, where two authorised users tried to use each other's credentials to access a third party's record, access was consistently denied.

Resistance to Replay Attacks

The suggested approach combines cryptographic key isolation with per-user access control enforcement to reduce collusion attacks. Even if several authorised users band together, they won't be able to decode an EMR unless the smart contract specifically allows access and discloses the user's encrypted AES key. Possession of another user's decrypted data or metadata prevents unauthorised decryption because the AES key is encrypted separately using the authorised recipient's public key. In order to prevent provider cooperation from circumventing consent constraints, patient consent is also assessed separately for every access request (Tables 4-5, Fig. 5). Unauthorised data sharing and privilege escalation between cooperating entities are prevented by this approach.

Table 4: Projected Mainnet and Layer-2 Cost Estimates

Operation	Gas Units	Est. Cost (ETH Mainnet @ 30 gwei)	Est. Cost (Polygon @ 50 gwei)
Upload Metadata	100,000	0.003 ETH ($7–$10 USD)	$0.002–$0.005 USD
Verify Access	60,000	0.0018 ETH ($4–$6 USD)	$0.001–$0.003 USD
Retrieve Metadata	20,000	0.0006 ETH ($1–$2 USD)	$0.0003–$0.001 USD

Table 5: Access Request Outcomes

Request ID	User Role	Consent Provided	Access Decision
RQ01	Doctor	Yes	Granted
RQ05	Nurse	No	Denied
RQ12	Administrator	Yes	Granted
RQ18	External Auditor	No	Denied
RQ25	Doctor	No	Denied
RQ30	Nurse	Yes	Granted

DOI: 10.3844/jcssp.2026.1881.1893

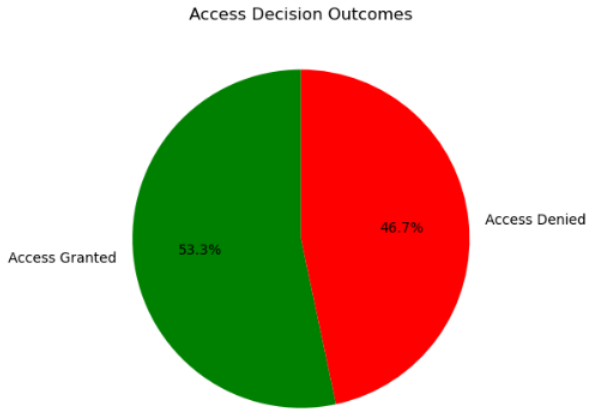

Fig. 5: Access Decision Outcomes

Confidentiality

The layered approach to security uses AES-256 symmetric encryption for EMR files and RSA encryption to secure the AES key (Dagher et al., 2018). Suppose there is a compromise within the storage layer (IPFS). In that case, the attacker has to additionally take the RSA private key of the patient to decrypt the AES key, thereby protecting the underlying EMR contents (Zhang et al., 2020b).

Integrity

When stored, SHA-256 hashes are computed and permanently recorded in the smart contract. At retrieval, the downloaded ciphertext's hash is computed against the accepted stored hash value (Murala et al., 2025). Any tampering or corruption of EMR during storage or transmission is immediately detected. In every case in the experiments, the hash verifications passed, which proved the reliability of integrity checks (Wang et al., 2018).

Enforcement of Access Rights

For every simulated access attempt, the RBAC smart contract always enforced the relevant access policies and consent requirements (Fan et al., 2020). No unauthorized access was found to be possible by the tests. All transactions, whether legitimate or rejected, were thus immutably recorded on the Ethereum blockchain for any future forensic investigations by administrators and auditors (Shuaib et al., 2021).

Auditability and Non-Repudiation

The entire workflow, including the establishment, access request, and retrieval processes, is recorded on-chain. Because Ethereum uses an immutable ledger-based system, users cannot undo their actions, and once recorded data is submitted, it cannot be altered. Stated differently, this satisfies certain general legislative criteria for the accountability of healthcare data, such as the HIPAA audit trail standards (Chen et al., 2019).

Resistance to Single Points of Failure

Ethereum and IPFS's decentralised architecture lessens reliance on any centralised server (Nguyen et al., 2019). The content is downloaded from other nodes in the network using the Content Identifier (CID) in the event that any IPFS node is inaccessible or offline. This guarantees their high availability in the event of data loss or service interruption (Murala et al., 2025).

Security Analysis

Although the proposed system handles confidentiality, integrity, and access control at the architectural level, advanced adversarial scenarios like key compromise, Sybil attacks, transaction front-running, or smart contract exploitation are not experimentally evaluated in the current review. Blockchain consensus, immutable execution, and cryptographic isolation conceptually reduce these risks; formal security proofs and adversarial simulations are left for further research. Overall, the results show that the architecture meets the fundamental needs of decentralisation, auditability, fine-grained access control, secrecy, and integrity. Compared to the conventional EMR system, this method ensures scalability and usability while providing greater privacy protections (Zhang et al., 2018). All things considered, the suggested framework has a number of advantages over both current purely on-chain blockchain-based solutions and conventional centralised EMR systems. Centralised EMR systems often have issues like single points of failure, not giving patients enough control over who can see their data, and not being able to be audited easily. Fully on-chain blockchain systems are open and cannot be changed, but they often have problems with scaling and charge a lot for transactions. Smart contracts allow for fine-grained access control, and on-chain logging makes sure that the proposed hybrid design is completely auditable. It also protects privacy with multilayer encryption, keeps records from being changed, and uses hashes that are based on cryptography. This balanced approach provides better privacy and security than centralised solutions without the speed limitations of on-chain techniques.

Comparative Performance Evaluation

The proposed framework is compared to blockchain-based EMR systems and restricted access-control methods that have been examined to assess its performance. Centralised EMR systems have high throughput and little access latency, but they lack patient-focused access control, transparency, and flexibility. Due to chain storage and complex consensus processes, blockchain-based systems like MedRec and HealthChain have higher latency but improved decentralisation and auditability. The proposed approach reduces access

DOI: 10.3844/jcssp.2026.1881.1893

latency by storing bulk data on IPFS and only keeping lightweight data on-chain. Compared to blockchain-only solutions, our hybrid approach increases throughput and reduces blockchain overhead. Good latency, security, openness, and legal compliance make the offered solution a good compromise. However, centralised systems may have higher raw throughput than blockchain-based alternatives. These results demonstrate that the design performs effectively in real healthcare settings where security and auditability are crucial.

Table 6 compares gas usage for significant smart contract actions in the proposed framework, MedRec (Azaria et al., 2016) and HealthChain (Mani et al., 2021). Their gas unit values derive from MedRec and HealthChain documentation and independent replication standards. No exact numbers were available, thus conservative estimates from the periodicals' operating statements were employed.

Table 6 shows that the suggested framework consistently reduces fuel use across all assessed procedures compared to MedRec and HealthChain. Upload information, petrol costs are 44% lower than MedRec and 55% lower than HealthChain because of the framework's hybrid design, which stores cryptographic information on-chain instead of full record material. Ethereum state updates consume most smart contract gas. This strategy reduces updates. The figures show that the suggested strategy maintains security or higher while improving on-chain efficiency.

Simulated Attack Scenario Analysis

Simulated attacks, orchestrated and controlled within the Ganache test environment, verified these security claims. These examples can be used to test the framework in hostile environments, but they cannot replace robust cryptographic proofs. Trying replay attacks: Fifteen valid access-grant transaction hashes from prior sessions were sent to the smart contract again. Validating the nonce and state consistency allowed the smart contract to find replayed transactions in all 15 circumstances. The blockchain captured the unauthorised attempt and sent a "Access Denied" response. Nothing was leaked without consent.

Simulation of a Collusion Attack: In ten different types of collusion attacks, two real users with different roles tried to merge or move their access credentials to records that neither of them had full permission to access. In all ten cases, the smart contract's per-user role and permission verification logic correctly turned down the requests that were made by people working together. Because each user's AES key was encrypted separately, no EMR content could be obtained from the shared metadata. These results strongly support the framework's claims that it can defend against both repeat and collusion attacks. They also provide real-world support for this formal threat model. The Universal Composability (UC) system will be used to build a formal cryptographic proof in the future.

Table 6: Quantitative Gas Consumption Comparison

Operation	Proposed Framework (gas units)	MedRec (estimated gas units)	HealthChain (estimated gas units)
Upload Metadata (EMR)	~100,000	~180,000	~220,000
Verify Access Permission	~60,000	~90,000	~110,000
Retrieve Metadata	~20,000	~35,000	~45,000
Revoke Access	~45,000	~80,000	~95,000

Conclusion

The authors suggested a framework for the safe administration of electronic medical records that is based on blockchain technology and protects privacy. The system uses SHA-256 hashing for data integrity verification, RSA encryption for safe key exchange, and AES-256 symmetric encryption for data secrecy. To enforce fine-grained access control regulations and preserve unchangeable recordings of all audit logs, an RBAC smart contract was created and implemented on Ethereum. The results of the experiment showed that while the amount of gas used for several essential smart contract functions is appropriate for real-world implementations, the time required for encryption and hashing increases linearly with file size. Access control testing confirmed that the smart contract consistently upholds access rules and honours the permissions patients give it, hence preventing unwanted access whenever they require assistance. When compared to other conventional EMR storage systems, the technique offers improvements in auditability, tamper-evidence, and secrecy in a decentralised manner. The healthcare information system is more reliable and accountable when IPFS is used for distributed file storage and blockchain is used for transparent policy enforcement. The study team will look into the use of sophisticated attribute-based encryption techniques that enable the creation of more adaptable access controls in subsequent extensions of this work. To further lower gas prices and make the system scalable for big deployment scenarios, they will further develop performance optimisations including layer-2 scaling solutions and off-chain storage proofs.

Future Enhancement

The viability of combining encryption, decentralised storage, and smart contract-based access control is demonstrated by the suggested blockchain-based EMR

administration architecture. However, there are other alternative approaches to improve security, scalability, and flexibility:

- Initially, ABE schemes can be investigated as a potential solution to enhance the granularity of access controls by defining several criteria including user role, department, and purpose of access. This would make it easier to define more detailed policies than only role-based ones
- Second, in order to address the performance and cost constraints of Ethereum's current architecture, future implementations should move to layer 2. This will maintain the main chain's security assurances while lowering petrol consumption and transaction delay. Additionally, off-chain storage proofs can be used to demonstrate data integrity without requiring direct on-chain recording of each activity
- In order to facilitate seamless integration into hospital information systems and broader use by real-world healthcare applications, the third point entails expanding the framework to support interoperability with current electronic health record standards (e.g., HL7 FHIR)
- Finally, in order to provide verifiable calculations over encrypted data without revealing content, future studies may investigate privacy-enhancing techniques like secure multi-party computation or zero-knowledge proofs. These will improve adherence to laws like GDPR and HIPAA without interfering with regular use by healthcare providers

Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work, and we are thankful for the opportunity to contribute to the field of research through this publication.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

Both the authors have equally contributed to this manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Akkaoui, R., Hei, X., & Cheng, W. (2020). EdgeMediChain: A Hybrid Edge Blockchain-Based Framework for Health Data Exchange. *IEEE Access*, *8*, 113467–113486. https://doi.org/10.1109/access.2020.3003575

Al-Khasawneh, M. A., Faheem, M., Alarood, A. A., Habibullah, S., & Alzahrani, A. (2024). A secure blockchain framework for healthcare records management systems. *Healthcare Technology Letters*, *11*(6), 461–470. https://doi.org/10.1049/htl2.12092

Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). MedRec: Using Blockchain for Medical Data Access and Permission Management. *2016 2nd International Conference on Open and Big Data (OBD)*, 25–30. https://doi.org/10.1109/obd.2016.11

Brun, S., & Ali, M. (2022). Blockchain EHR system with smart contract consensus. *Security and Communication Networks*, *2022*, 4645585.

Chamola, V., Jain, M., & Peng, Z. (2024). AI-assisted blockchain EMR with decentralized identity management. *Future Generation Computer Systems*, *183*, 35–52.

Chauhan, R., Kumar, P., & Singh, S. (2025). A systematic review of privacy-preserving blockchain applications in healthcare. *Multimedia Tools and Applications*, *84*(32), 39925–39980. https://doi.org/10.1007/s11042-024-20541-z

Chen, Y., Ding, S., Xu, Z., Zheng, H., & Yang, S. (2019). Blockchain-Based Medical Records Secure Storage and Medical Service Framework. *Journal of Medical Systems*, *43*(1), 5. https://doi.org/10.1007/s10916-018-1121-4

Dagher, G. G., Mohler, J., Milojkovic, M., & Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, *39*, 283–297. https://doi.org/10.1016/j.scs.2018.02.014

Fan, K., Ren, Y., Wang, Y., Li, H., & Yang, Y. (2020). Blockchain-based efficient privacy-preserving and data sharing scheme of content-centric network in 5G. *IEEE Transactions on Network and Service Management*, *15*(4), 1203–1214.

Gupta, R., Kim, J., & Zhang, Y. (2022). Medical blockchain: Data sharing and privacy preserving of EHR based on smart contract. *Journal of Information Security and Applications*, *65*, 103117. https://doi.org/10.1016/j.jisa.2022.103117

Hussien, H. M., Yaqoob, I., Nasir, Q., Abdelrazek, M., Bakar, K. A., & Gani, A. (2020). Blockchain-based access control for healthcare data sharing. *Computers & Security*, *90*, 101710.

Kumar, A., & Patil, S. (2024). Decentralized multi-authority PKI for EHR sharing. *Procedia Computer Science*, 44–54.

Liu, X., Zhang, Y., Pan, Y., Yu, R., & Zhang, S. (2021). Blockchain-based privacy-preserving medical data sharing. *IEEE Internet of Things Journal*, *8*(3), 1630–1641.

Mani, V., Manickam, P., Alotaibi, Y., Alghamdi, S., & Khalaf, O. I. (2021). Hyperledger Healthchain: Patient-Centric IPFS-Based Storage of Health Records. *Electronics*, *10*(23), 3003. https://doi.org/10.3390/electronics10233003

Murala, D. K. (2025). Multi-stage variational autoencoders for hierarchical molecular generation and activity optimization. *Journal of Computer-Aided Molecular Design*, *39*(2), 123. https://doi.org/10.1007/s10822-025-00705-1

Murala, D. K., Krishna, G. S., kiran, T. V. S., & Mohamud, A. K. (2025). MedShieldFL-a privacy-preserving hybrid federated learning framework for intelligent healthcare systems. *Scientific Reports*, *15*(1), 43144. https://doi.org/10.1038/s41598-025-27303-3

Murala, D. K., Vemulapalli, L., Balagoni, Y., Patnala, E., & Romeo, B. (2026). MedLedgerFL: a hybrid blockchain-federated learning framework for secure remote healthcare services. *Scientific Reports*, *16*(1), 8218. https://doi.org/10.1038/s41598-026-39149-4

Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2019). Blockchain for Secure EHRs Sharing of Mobile Cloud Based E-Health Systems. *IEEE Access*, *7*, 66792–66806. https://doi.org/10.1109/access.2019.2917555

Shuaib, M., Alam, M., Singh, S., & Park, J. H. (2021). Blockchain-based secure framework for medical records sharing in a healthcare ecosystem. *Sustainable Cities and Society*, *67*, 102710.

Tanwar, S., Parekh, K., & Evans, R. (2020). Blockchain-based electronic healthcare record system for healthcare 4.0 applications. *Journal of Information Security and Applications*, *50*, 102407. https://doi.org/10.1016/j.jisa.2019.102407

Wang, S., Zhang, Y., & Zhang, Y. (2018). A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems. *IEEE Access*, *6*, 38437–38450. https://doi.org/10.1109/access.2018.2851611

Wang, T., Wu, Q., Chen, J., Chen, F., Xie, D., & Shen, H. (2024). Health data security sharing method based on hybrid blockchain. *Future Generation Computer Systems*, *153*, 251–261. https://doi.org/10.1016/j.future.2023.11.032

Wilson, K., & Garcia, E. (2024). Blockchain-enhanced smart healthcare management for chronic diseases. *Discover Computing*, *25*(4), 45.

Zhang, L., Zhang, T., Wu, Q., Mu, Y., & Rezaeibagha, F. (2022). Secure Attribute-Based Sharing of Personal Health Records With Blockchain. *IEEE Internet of Things Journal*, *9*(14), 12482–12496.

Zhang, P., White, J., Schmidt, D. C., Lenz, G., & Rosenbloom, S. T. (2018). FHIRChain: Applying Blockchain to Securely and Scalably Share Clinical Data. *Computational and Structural Biotechnology Journal*, *16*, 267–278. https://doi.org/10.1016/j.csbj.2018.07.004

Zhang, R., Xue, R., & Liu, L. (2020a). Security and Privacy on Blockchain. *ACM Computing Surveys*, *52*(3), 1–34. https://doi.org/10.1145/3316481

Zhang, Y., Wen, J., & Xie, X. (2020b). Data security and privacy-preserving in blockchain-based health information exchange: A survey. *IEEE Transactions on Services Computing*, *13*(3), 469–485.