applied sciences

Article

Efficient and Secure EMR Storage and Sharing Scheme Based on Hyperledger Fabric and IPFS

Jinxi Guo , Kui Zhao *, Zhiwei Liang and Kai Min

School of Cyber Science and Engineering, Sichuan University, Chengdu 610207, China; guojinxi@stu.scu.edu.cn (J.G.)
* Correspondence: zhaokui@scu.edu.cn

Abstract: This study examines the issues of privacy protection, data security, and query efficiency in blockchain-based electronic medical record (EMR) sharing. It proposes a secure storage and sharing scheme for EMR based on Hyperledger Fabric and the InterPlanetary File System (IPFS). To mitigate the privacy risks of data mining that could reveal patient identities, we establish an attribution channel in Hyperledger Fabric to store EMR ownership information and a data channel to store the storage location, digest, and usage records of medical data. Encrypted medical data are stored in the IPFS. To improve query efficiency in the blockchain, we integrate queryable medical data attributes into a composite key for conditional queries, avoiding complex data filtering processes. Additionally, we use a zero-knowledge proof combined with smart contracts for decentralized identity verification, eliminating reliance on third-party centralized verification services and enhancing system security. We also integrate AES and proxy re-encryption techniques to ensure data security during sharing. This scheme provides a more secure, efficient, and privacy-preserving approach for EMR systems, with significant practical implications and broad application potential.

Keywords: blockchain; electronic medical record sharing; privacy protection; zero-knowledge proof; proxy re-encryption

check for updates

Citation: Guo, J.; Zhao, K.; Liang, Z.; Min, K. Efficient and Secure EMR Storage and Sharing Scheme Based on Hyperledger Fabric and IPFS. *Appl. Sci.* **2024**, *14*, 5005. https://doi.org/10.3390/app14125005

Academic Editor: Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 17 April 2024
Revised: 4 June 2024
Accepted: 5 June 2024
Published: 8 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, with continuous innovation and development, the application research of blockchain technology has gradually expanded to various aspects of socio-economic activities. Blockchain technology plays a crucial role in fields such as supply chain management [1], the Internet of Things [2], identity authentication [3], and healthcare services [4], showing a trend of diversified and cross-industry applications. Despite its vast potential, integrating blockchain into electronic medical record (EMR) data sharing faces significant challenges. Ensuring data privacy, security, and query efficiency is particularly difficult. Protecting sensitive medical information while maintaining efficient data access remains a critical issue.

Data storage is crucial for patient privacy and data usage efficiency. Although research on blockchain-based data storage has made some progress [5,6], there is still room for improvement. On one hand, mainstream storage methods currently fail to effectively isolate the relationship between EMR and patients, making them vulnerable to privacy attacks exploiting this relationship, such as data mining. On the other hand, due to existing performance bottlenecks in blockchain technology, it exhibits inefficiency when facing large-scale data sharing scenarios.

Additionally, current blockchain-based data sharing schemes often use centralized identity authentication to prevent malicious access. However, centralized systems heavily rely on trust in the central authority. If this authority acts dishonestly, maliciously, or suffers a data breach, users' digital identity information is at serious risk. Moreover, data querying is a crucial part of the data usage process. Improving the query speed can significantly enhance data sharing efficiency. However, there is limited research on improving the efficiency of specific data queries on blockchain.

To address these challenges, this study proposes an efficient and secure EMR storage and sharing scheme based on Hyperledger Fabric and the InterPlanetary File System (IPFS), which works as follows:

1. We propose a storage method that combines dual channels with the IPFS. In Hyperledger Fabric, we create an attribution channel to store EMR's attribution information and a data channel to store the storage location, summary, and usage records of medical data. The encrypted medical data are then stored in the IPFS. This ensures that patient privacy is not threatened by attacks such as data mining.
2. We integrate medical data attributes that can be used for a conditional query into a composite key. This composite key, along with the medical record number, is then stored as a key–value pair in the blockchain. This approach aims to enhance the efficiency of data querying in the blockchain.
3. We introduce a zero-knowledge proof and combine it with smart contracts to achieve decentralized identity verification for system users. This eliminates the reliance on central third-party verification services, thereby enhancing system security.
4. We combine AES encryption with proxy re-encryption to ensure data security during the sharing process. We provide the principles behind implementing this technique.

2. Related Work

There are already numerous solutions based on blockchain technology that address the issues of secure storage or the sharing of medical data. Cao et al. [7] proposed an EMR management scheme that combines cloud and blockchain technologies, which ensures the secure storage of EMR. However, it does not address the implementation of secure data sharing. Carter et al. [8] proposed an EMR sharing scheme that integrates Ethereum blockchain and cloud computing networks, but it does not fully address privacy concerns. Huang et al. [9] proposed recording the operations of various stages of EMR on the blockchain to ensure traceability and tamper resistance. Xia et al. [10] proposed ensuring the security of data stored in the cloud through access permissions and effectively tracking and monitoring all operations on the data using smart contracts. Ref. [9,10] suggested that monitoring and recording the process of medical record operations can prevent tampering with the records, but they cannot prevent the leakage of patient privacy. Fu et al. [11] encrypted medical data using interleaved coding techniques, effectively protecting personal privacy. Wang et al. [12] proposed a blockchain-based medical data sharing scheme, integrating searchable encryption and proxy re-encryption to achieve secure sharing. Dagher et al. [13] introduced the Ancile framework, which transfers the ownership and control of EMR to data owners via the Ethereum platform and effectively ensures the information security of EMR through proxy re-encryption technology. Akkaoui et al. [14] proposed a secure and efficient data management framework based on Ethereum, utilizing edge computing and blockchain to ensure data security and privacy. Ref. [15] combined searchable encryption algorithms to construct index expressions for EMR data and stored them on the blockchain, allowing data owners to control access permissions for their data. Azaria et al. [16] introduced MedRec, an EMR sharing scheme, employing three Ethereum smart contracts to implement fine-grained access control for patients' medical records, enabling third-party users to access data upon successful authentication. These approaches [12–16] safeguard data security and protect patient privacy through access control. However, the systems proposed are implemented on Ethereum, where normal transactions require a certain amount of tokens and additional gas expenses, thus increasing system complexity and management costs, which may not be entirely suitable for hospital information systems.

With the evolution of blockchain technology, there have been research efforts adopting the more lightweight Hyperledger Fabric [17] architecture to implement the blockchain network component of the designed systems. Hyperledger Fabric is one of the widely adopted solutions in the current realm of consortium blockchains [18]. It integrates a member management service mechanism, allowing multiple organizational entities to participate in and manage it jointly. Moreover, this solution supports smart contract development in various common programming languages, exhibiting greater flexibility in practical deployment and application. Tanwar et al. [19] proposed an access control policy algorithm aimed at enhancing data accessibility among healthcare providers, enabling the implementation of an EMR sharing system based on Hyperledger Fabric. However, they did not discuss the data integrity aspect of the solution. Ref. [20] used Hyperledger Fabric to enhance interoperability among healthcare institutions and to solve the fragmentation problem.

Table 1 summarizes the limitations of related work and explains how this study addresses these shortcomings. In addition to the issues mentioned in the table, most related work does not separate ownership information from the specific medical data of patients. This makes it difficult to prevent privacy threats from data mining that could reveal patient identities. Our study mitigates this threat by creating an attribution channel and a data channel to store the relevant information separately.

Table 1. Limitations of related work and solutions in this study.

Literature	Limitation	Solution in This Study
[7]	Lack of shared program	Combining blockchain, the IPFS, and proxy re-encryption
[8–10]	Lack of privacy protection	Using pseudo-identities in the system
[11]	No efficient search mechanism	Creating composite keys for conditional queries
[12–16]	Requires tokens or gas expenses	Based on the Hyperledger Fabric architecture without the incentive mechanism
[19]	Lack of data integrity discussion	Combining blockchain and the IPFS
[20]	The source medical data are not decentralized storage	Storing medical data ciphertexts using the IPFS

3. Framework Components

This section introduces the main components used in this study and explains the reasons for selecting them. This helps in understanding the proposed scheme.

3.1. Hyperledger Fabric

Hyperledger Fabric is an open-source, enterprise-grade, permissioned distributed ledger technology platform developed under the leadership of the Linux Foundation [17]. It features a highly modular architecture that brings innovation, versatility, and optimization to the healthcare industry. As a consortium blockchain application development platform, Hyperledger Fabric allows only authorized organizations and nodes to join designated blockchains for conducting transactions and recording data. Additionally, Hyperledger Fabric does not rely on cryptocurrencies to incentivize mining or drive smart contract execution, thus avoiding the risks or vulnerabilities associated with cryptocurrency use.

In the Hyperledger Fabric network, smart contracts are referred to as chaincodes. Hyperledger Fabric allows for the creation of different channels as needed, with different chaincodes deployed within each channel. A chaincode is embedded and executed within the nodes, requiring all nodes in the channel to install the chaincode.

There is isolation between channels in Hyperledger Fabric, meaning that data are not exchanged between different channels, and one channel does not depend on others. Each channel can be understood as an independent instance of Hyperledger Fabric, and all channels are completely separate. This feature helps us separate patient ownership information from medical data.

3.2. InterPlanetary File System

The InterPlanetary File System(IPFS) is an open-source distributed file storage and content distribution protocol [21]. Designed to create a more robust, secure, and efficient environment for data sharing and transmission, the IPFS is particularly well suited for applications requiring long-term preservation, high redundancy, and resistance to censorship. When accessing medical data, the IPFS does not rely on server addresses but locates data based on its content hash, ensuring the uniqueness of the data and allowing for the replication and storage of identical content across multiple nodes in the network. When users access content stored on the IPFS, they are actually downloading data from the nearest or available nodes, rather than fetching it from a single server. This approach enhances the reliability and efficiency of the system.

3.3. zk-SNARKs

A zero-knowledge proof (ZKP) requires the participation of both a prover and verifier. In employing the ZKP techniques, the prover can convince the verifier of the truthfulness of a statement without revealing any useful information related to the statement beyond its truth. ZKP has been widely utilized in identity authentication [22].

ZKPs can be categorized into interactive and non-interactive types. A non-interactive proof requires at most one authentication interaction during the verification process, thereby reducing the communication overhead caused by multiple interactions in the proof process, especially in blockchain systems. The Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) [23], Zero-Knowledge Scalable Transparent Argument of Knowledge (zk-STARKs) [24], and Bulletproofs [25] are three typical non-interactive ZKP schemes. Table 2 presents a comparison of three typical non-interactive ZKP technologies. zk-SNARKs have the smallest proof size. Verifiers of zk-SNARKs only need to check a fixed-size structure, such as the verification key and proof. This eliminates the need to reproduce or understand the original computation. Consequently, verifiers can complete the verification in a very short time, which is crucial for enhancing the scalability and efficiency of blockchain systems.

Table 2. Comparison of zk-SNARKs, zk-STARKs, and Bulletproofs.

	zk-SNARKs	**zk-STARKs**	**Bulletproofs**
Algorithmic complexity of the prover	O(N·log(N))	O(N·polylog(N))	O(N·log(N))
Algorithmic complexity of the verifier	O(1)	O(polylog(N))	O(N)
Communication complexity	O(1)	O(polylog(N))	O(log(N))
1TX proof size	200 B	45 KB	1.5 KB

Zokrates [26] is a ZKP toolkit that includes a Domain-Specific Language (DSL) processor, compiler, and zk-SNARKs generator. The zk-SNARKs process is facilitated in Zokrates by initializing a zokratesProvider object, with the option to choose Groth16 [27] as the verification scheme during initialization. This process encompasses the generation of proof key PK and verification key VK, the generation of proof pf, and the verification of pf.

Figure 1 illustrates the process of generating proof and verifying it using zk-SNARKs through Zokrates. The steps are as follows:

1. $Setup(f) \rightarrow (PK, VK)$: Use DSL to describe the constraint verification program intended for ZKP implementation, denoted as $sourceCode$. Next, compile it to generate the internal arithmetic circuit representation, denoted as $f = compile(sourceCode)$. Finally, generate the key pair (PK, VK) by constructing $Setup$.

2. $GenProof(f, witness, PK) \rightarrow proof$: Given the public input $input_{pub}$ and the private input $input_{pri}$, compute $witness = compute(f, [input_{pub}, input_{pri}])$. Finally, generate the proof using Equation (1):

$$pf = GenProof(f, witness, PK). \tag{1}$$

3. $Verify(VK, proof) \rightarrow result$: When there is a verification requirement, the correctness of pf can be verified by constructing $Verify$ and providing pf and its corresponding VK as inputs.

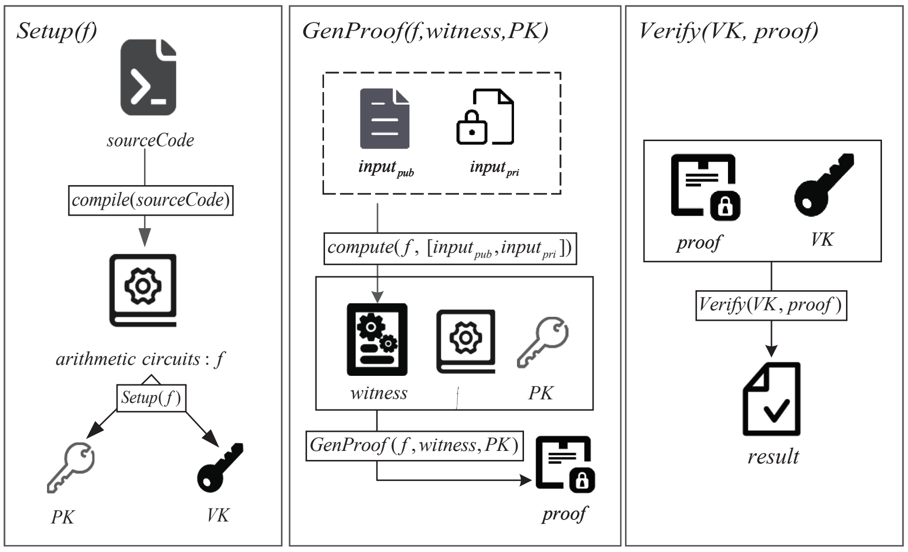

Figure 1. The process of generating zero-knowledge proof and verifying proof.

4. Scheme Model

As shown in Figure 2, the model of the scheme includes six entities: a healthcare management center, healthcare organizations, a data owner, a data user, a blockchain network, and a star file system.

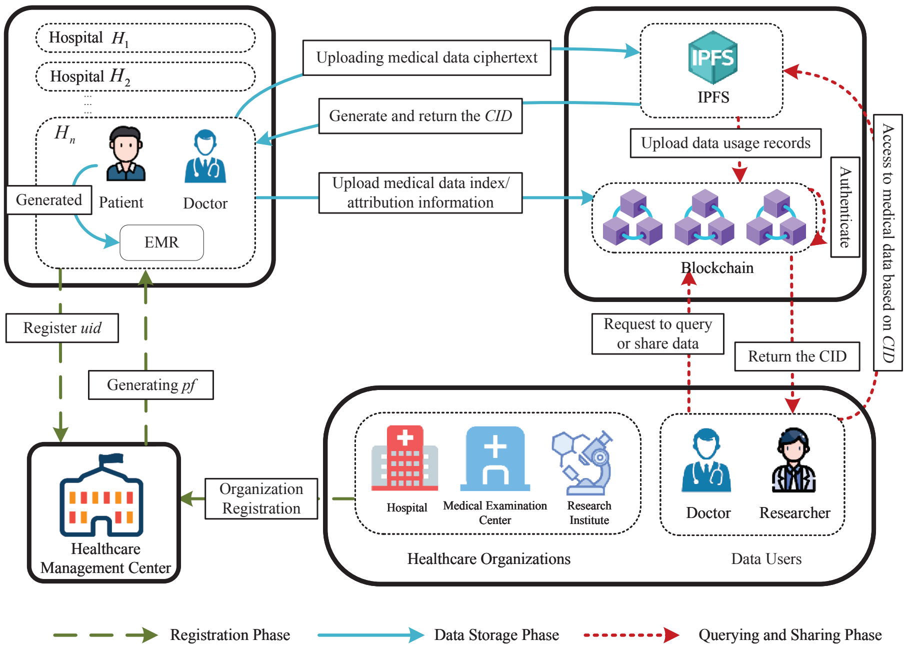
Hospital H_1
Hospital H_2
H_n
Patient
Doctor
Generated
EMR
Uploading medical data ciphertext
Generate and return the CID
Upload medical data index/ attribution information
IPFS
Upload data usage records
Authenticate
Blockchain
Access to medical data based on CID
Request to query or share data
Return the CID
Register uid
Generating pf
Healthcare Management Center
Organization Registration
Hospital
Medical Examination Center
Research Institute
Healthcare Organizations
Doctor
Researcher
Data Users
Registration Phase
Data Storage Phase
Querying and Sharing Phase

Figure 2. Scheme model.

Healthcare Management Center (HMC): The HMC comprises healthcare regulatory bodies responsible for qualifying, registering, and regulating participants, such as healthcare organizations, within the system. Any organization seeking to join the system must undergo registration with the HMC to ensure compliance with regulatory standards and guidelines.

Healthcare Organizations (HOs): The HO include hospitals, medical examination centers, research institutes, and other healthcare-related institutions licensed by the HMC.

Data Owner (DO): The DO is generally the patient. With the consent of the DO, the medical data portion of the EMR can be encrypted and uploaded to a shared network.

Data User (DU): DUs primarily consist of patients, healthcare professionals requiring access to patients' historical medical records, and authorized members of the other organizations necessitating the sharing and utilization of medical data.

Blockchain (BC): The blockchain employed in this study was deployed utilizing the Hyperledger Fabric framework.

IPFS: The IPFS is used for the distributed storage of medical data ciphertext, wherein it generates a CID representing the hash address of the stored data. This CID is then provided back to the client. The client can query the corresponding data on the IPFS using the CID.

5. Scheme Implementation

Our proposed scheme comprises five crucial phases: the system initialization phase, user registration and identity verification phase, data storage phase, data query phase and data sharing phase.

5.1. System Initialization Phase

The initialization phase aims to establish the blockchain network. It involves setting up the Hyperledger Fabric blockchain network and then integrating HOs authorized by the HMC into this network. Hyperledger Fabric's Certificate Authority then allocates keys and certificates to them. Following that, the corresponding chaincode is deployed for each channel in the network.

5.2. Registration and Verification Phase

To ensure secure data communication and prevent unauthorized access to the system, all users U_i who need to join the system must register. Prior to registration, a key pair is generated, where *pk* represents the public key, and *sk* represents the private key. Additionally, users prepare their real identity ID_R for registration. The HMC generates a *uid* and proof.

5.2.1. Generating *uid*

The *uid* serves as a unique identity identifier for recording U_i as a valid user on the blockchain and is used as the unique ID for updating ledger data on the blockchain. During the user registration phase, patients are required to provide basic information and na ID_R for registration with healthcare institutions. Subsequently, healthcare institutions confirm the patient's *uid* with the HMC: if the patient's *uid* does not exist, a new *uid* is generated by the management center upon request, using a hash function calculation, as shown in Equation (2):

$$uid = Hash(Enc_{PK}(ID_R)); \quad (2)$$

5.2.2. Generating pf

ZKP is utilized for the decentralized identity verification of all system users. The HMC generates proofs for all registered users. The specific process involves signing with the user's real identity ID_R, timestamp T, and a random number r, using this signed information as privacy input, computing the proof pf_i utilizing zk-SNARKs technology, and sending the verification key VK_i and pf_i to the user for private storage. When identity verification is required, users need to provide VK_i and pf_i to the system, which then executes the chaincode for decentralized identity verification on the blockchain.

5.2.3. Verification

When a DU wishes to access medical data stored in the system, identity verification and access control are required based on different roles and scenarios. When initiating a data access request, the DU needs to provide pf_{DU}, VK_{DU}, and uid_{DU}.

If $Exist(uid_{DU}) = true$, the chaincode proceeds to authenticate the identity of the DU and verify the correctness of pf_{DU}. If $Verify(VK_{DU}, proof_{DU}) = true$, the verification result is returned to the data client. Then, the system determines the legitimacy of the user's request. Otherwise, the verification fails, and the data usage request is rejected.

5.3. Data Storage Phase

To provide a clearer explanation of the sharing scheme proposed in this study, Figure 3 illustrates the entire process from the data storage stage to the data sharing stage.

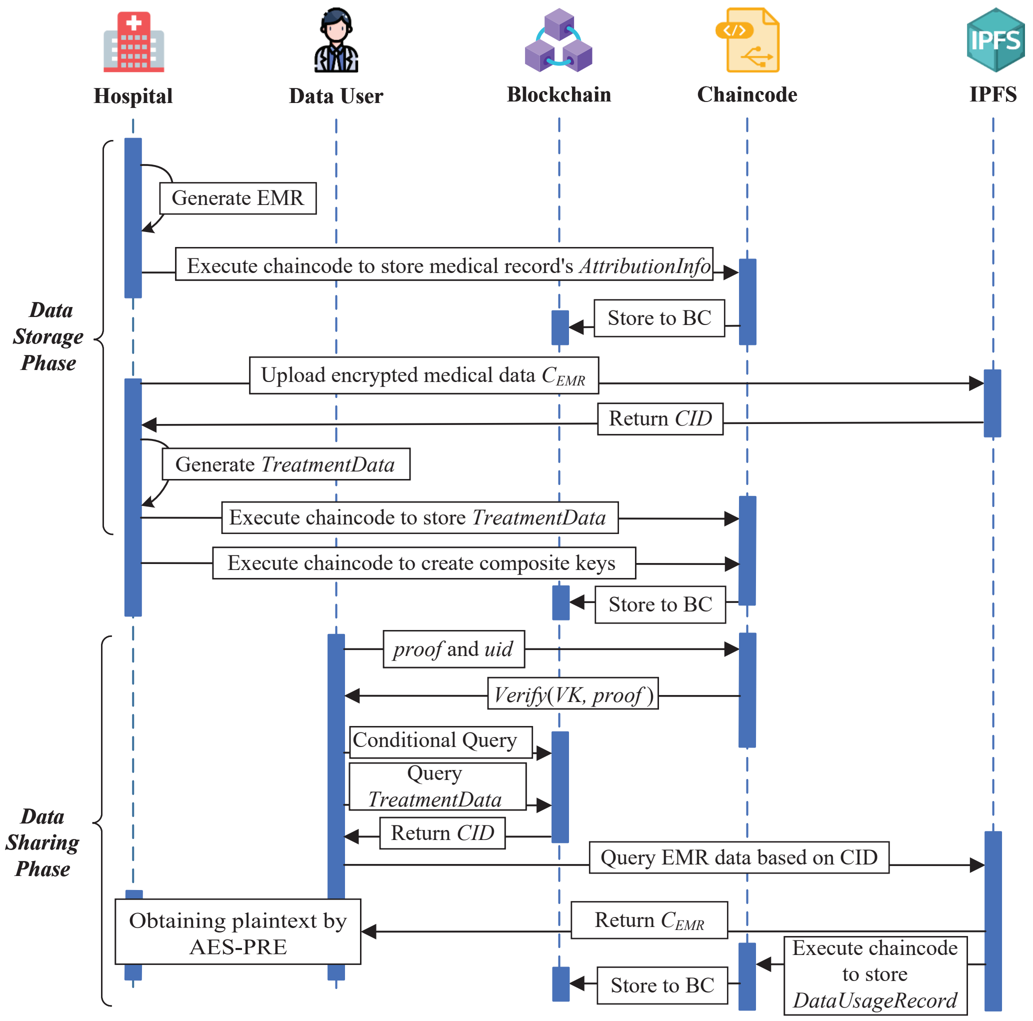
Hospital
Data User
Blockchain
Chaincode
IPFS
Data Storage Phase
Generate EMR
Execute chaincode to store medical record's AttributionInfo
Store to BC
Upload encrypted medical data C_{EMR}
Return CID
Generate TreatmentData
Execute chaincode to store TreatmentData
Execute chaincode to create composite keys
Store to BC
Data Sharing Phase
proof and uid
Verify(VK, proof)
Conditional Query
Query TreatmentData
Return CID
Query EMR data based on CID
Obtaining plaintext by AES-PRE
Return C_{EMR}
Execute chaincode to store DataUsageRecord
Store to BC

Figure 3. Timing diagram for data storage phase and data sharing phase.

The scheme creates an attribution channel and a data channel. The attribution channel is specifically designed to record the attribution information of patient records. The data channel records details such as the digests of medical data, storage locations, and usage records of EMR. Simultaneously, the encrypted original medical data are stored in the IPFS. This design effectively segregates medical data from patient identity information, offering a dependable technical solution for safeguarding patient privacy.

5.3.1. Encrypting Medical Records

Patient U_i undergoes diagnosis and treatment at hospital H, generating medical data plaintext M_{EMR} and its corresponding EMR's number N_{EMR}. After patient confirmation, H encrypts the medical data excluding the patient's personal information to obtain the ciphertext of medical data, $C_{EMR} = En_{pk}(M_{EMR})$, and calculates the message digest of the medical data ciphertext: $MD_{EMR} = Hash(C_{EMR})$.

The doctor uses their private key sk_{doc} to sign MD_{EMR} and uid_{doc}, resulting in a digital signature $DocSign_{EMR}$ as shown in Equation (3):

$$DocSign_{EMR} = Sign_{sk_{doc}}(MD_{EMR}, uid_{doc}). \tag{3}$$

5.3.2. Processing Medical Record Numbers

The medical record number N_{EMR} generated in the above process undergoes encryption and hashing. On one hand, the hash value of N_{EMR} is computed as shown in Equation (4):

$$N_{EMR_{hash}} = Hash(N_{EMR}), \tag{4}$$

where $N_{EMR_{hash}}$ serves as the unique identifier stored on the data channel for medical data indexes $TreatmentData$ and data usage records $DataUsageRecord$. On the other hand, the ciphertext of the EMR's number $C_{N_{EMR}}$ is generated as shown in Equation (5):

$$C_{N_{EMR}} = En_{pk_i}(N_{EMR}), \tag{5}$$

which is used as the unique identifier for the attribution information $AttributionInfo$ stored on the attribution channel.

5.3.3. Attribution Channel

The attribution channel stores the attribution information of EMR generated after a patient's diagnosis, specifically recording the association between the EMR and the patient. Patients can view their own medical record numbers through the ownership relationship established in the attribution channel, thereby accessing their EMR. Since each EMR's number corresponds uniquely to specific medical data, only the relationship between the patient's uid and the ciphertext $C_{N_{EMR}}$ is recorded in the attribution channel, where $C_{N_{EMR}}$ serves as the unique identifier for the record attribution information $AttributionInfo$. Patients compute the signature $PatSign$ for $C_{N_{EMR}}$ as shown in Equation (6);

$$PatSign = Sign_{sk_i}(C_{N_{EMR}}, uid_i). \tag{6}$$

Subsequently, the system invokes the chaincode of the attribution channel to upload $AttributionInfo = \{C_{N_{EMR}}, uid_i, PatSign\}$ for persistent storage in the attribution channel. If the patient's keys are not compromised, it is highly difficult for attackers to obtain the plaintext of the EMR number N_{EMR}.

5.3.4. Data Channel

The data channel is primarily used to record the medical data index and the usage records of EMR that are produced following patient diagnosis and treatment.

The medical data index includes the digest of the medical data, MD_{EMR}, the hash address CID generated from storing the original medical data in the IPFS, the doctor's signature on the medical data, $DocSignEMR$, and hospital information H. Together, these data form the medical data index $TreatmentData = \{N_{EMR_{hash}}, CID, MD_{EMR}, DocSign_{EMR}, H, t\}$.

To ensure that the DU cannot obtain the N_{EMR} through reverse engineering, this scheme adopts the hash value generated from the EMR's number, $N_{EMR_{hash}}$, as the key for the medical data index $TreatmentData$. Differing from the encrypted EMR's number employed in the attribution channel, using $N_{EMR_{hash}}$ in the data channel. Because the inverse hash function $hash^{-1}$ either does not exist or is extremely difficult to solve, the DUs are unable to obtain the plaintext of the EMR's number. This effectively protects the privacy of medical data attribution relationships.

When patients or other authorized DUs access medical data, the system generates corresponding data usage records, $DataUsageRecord$, and uploads these records to the blockchain for persistent storage. This mechanism enables patients and the HMC to effectively track and trace the usage of EMR data.

5.3.5. IPFS Stores Medical Data

The encrypted medical data of patients' EMR are stored in the IPFS, while only the medical data index is saved on the blockchain. This design not only alleviates the burden on the blockchain network but also reduces the risks associated with single points of failure and privacy vulnerabilities caused by centralized storage methods.

After diagnosis and treatment at hospital H, the medical data plaintext M_{EMR}, which does not contain patient privacy, is encrypted by H to generate the corresponding medical data ciphertext C_{EMR}. This ciphertext is then uploaded to an IPFS node for storage. The IPFS calculates a unique data CID identifier from this medical data ciphertext. This CID is subsequently returned to hospital H, which utilizes this identifier to construct the $TreatmentData$ object and uploads it to the data channel for persistent storage. When DUs wish to access the stored data in the IPFS, they must provide the corresponding CID. The IPFS locates and accesses the corresponding medical data based on the CID. If the data are tampered with, the original CID will not match, thereby preventing access to the altered data.

5.4. Data Query Phase

In the scenario of medical data sharing, DUs often need to perform multi-condition queries to obtain corresponding EMR's numbers, and then execute the chaincode in the data channel to obtain the storage information of EMR. Consider a case of a conditional query Q: querying all medical data records generated in hospital H that include a specific diagnostic result DR. After executing the query Q, the query result R should satisfy the following conditions simultaneously:

- The medical data records were generated in hospital H.
- The diagnostic result of the medical data records include DR.

If the conditional query process is not optimized, in a Hyperledger Fabric blockchain-based system, it would require individual queries for each condition, followed by filtering out results that do not satisfy the other conditions one by one. Consequently, the final result set that satisfies all query conditions would exclude the vast majority of data. With an increasing number of query conditions, the system would need to access and parse a corresponding increasing number of blocks, leading to a significant increase in the total amount of data to be processed and resulting in more redundant data. This inevitably increases the time required for conditional queries, thereby reducing query efficiency and making the entire query process appear both time-consuming and inefficient.

To address the above issue, this scheme proposes a method of creating composite key, which can significantly improve the efficiency of conditional queries.

5.4.1. Creating Composite Key

As shown in Algorithm 1, Hyperledger Fabric provides a specific API for constructing composite keys from different attributes: $createCompositeKey(objectType, attributes)$. This method aims to combine the given attributes with the object type to form a composite key, which can then be used to access historical and latest state information.

Algorithm 1 Create composite key

```
Input: a set of attributes $A = \{a_1, \ldots, a_s\}(1 \leq s)$
Output: $k$
1: $k \leftarrow createCompositeKey(String, A)$
2: return $k$
```

During the process of uploading medical data records for storage, for each record $r_i(1 \leq i \leq t)$ in the given collection of medical data records $R = \{r_1, r_2, \ldots, r_t\}(t \geq 1)$, check if there are multiple values corresponding to one attribute. For example, in the “diagnosis" attribute of an EMR, there may be multiple symptoms, such as “hypertension” and “hyperglycemia". If this situation exists, and the attribute corresponds to n values, then split it into n separate records. Then, individually create a composite key for each record, with its corresponding $N_{EMR_{hash_i}}$ as the value for that key. This results in the EMR having n keys associated with it. As shown in Algorithm 2, we generate a key–value pair for each record and then submit all key–value pairs corresponding to these records to the blockchain for storage.

Algorithm 2 Generate key–value pairs

```
Input: a set of records $R = \{r_1, r_2, \ldots, r_t\}(t \geq 1)$
Output: a set of key–value pairs $KVP$
1: Declare an empty set $KVP$
2: // $r_i = \{N_{EMR_{hash_i}}, A_i\}$, where $A_i$ is a set of attributes with elements that can be used
   as a query condition
3: for all $r_i \in R$ do
4:     Compute $k_i$ based on Algorithm 1
5:     $kvp_i = (k_i, N_{EMR_{hash_i}})$
6:     Add $kvp_i$ into $KVP$
7: end for
8: return $KVP$
```

5.4.2. Executing Conditional Query

When executing query Q, construct all query conditions into a composite key k. This k contains constraint information for all conditions in query Q. Subsequently, use the key k as a parameter and execute the $getState(k)$ function. $getState$ is an API provided by Hyperledger Fabric for querying the latest state. After the $getState$ function’s execution, the resulting data will contain all the latest records that meet all conditions specified in query Q.

5.4.3. Updating Key–Value Pairs

Using the method of creating composite keys, we can effectively avoid the complex data filtering processes typically involved in conditional queries. When the $getState$ method is executed, it retrieves a result set that meets all the criteria, significantly enhancing the efficiency of data queries. However, there may be challenges associated with updating key values when storing data records using composite keys. Specifically, when new records are added to the ledger, the k_j created for the new record might coincide with one of the existing keys in the ledger, where none of the values corresponding to k_j match any existing records. In such cases, it becomes necessary to update the existing key–value pairs.

In the method proposed in this scheme, the process for updating key–value pairs involves the following steps. First, query the existing value $N_j = \{{N_{EMR_{hash_1}}}^j, {N_{EMR_{hash_2}}}^j, \ldots, {N_{EMR_{hash_t}}}^j\}(t \geq 1)$ associated with the k_j. Next, combine the new ${N_{EMR_{hash_{t+1}}}}^j$ with N_j to form a new set N_{new}, which will serve as the updated value linked with the k_j. Therefore, Algorithm 2 can be enhanced by introducing a key-value pair updating process, as shown in Algorithm 3. Finally, invoke the $putState(k_j, N_{new})$ function to submit the new key–value pair to the blockchain for persistent storage. *putState* is an API provided by Hyperledger Fabric for submitting the latest state. In using the *putState* function, the new record value will overwrite the old value in the state database, thereby achieving the objective of updating the data.

Algorithm 3 Update key–value pairs

```
Input: a set of records $R = \{r_1, r_2, \dots, r_t\}(t \geq 1)$
Output: a set of key–value pairs $KVP$
1:  Declare an empty set $KVP$
2:  // $r_i = \{N_{EMR_{hash_i}}, A_i\}$, where $A_i$ is a set of attributes with elements that can be used
    as a query condition
3:  for all $r_i \in R$ do
4:      Compute $k_i$ based on Algorithm 1
5:      // Define $ls$ as the latest state of $k_i$ in the blockchain
6:      $ls \leftarrow getState(k_i)$
7:      if $ls$ is not null then
8:          // Define $N$ as the old set of $N_{EMR_{hash}}$
9:          Add $N_{EMR_{hash_i}}$ into $N$
10:         $kvp_i = (k_i, N)$
11:     else
12:         $kvp_i = (k_i, N_{EMR_{hash_i}})$
13:     end if
14:     Add $kvp_i$ into $KVP$
15: end for
16: return $KVP$
```

5.5. Data Sharing Phase

During data sharing, the DU must undergo identity verification and validate the legitimacy of the request. The data sharing phase can be roughly divided into two scenarios:

- Patients query their own EMR and then authorize healthcare service providers to view them.
- Other DUs, such as medical staff from healthcare units or researchers from research institutions, query the system for medical data available for sharing based on their specific needs.

In the first scenario, patients can query their encrypted medical record number $C_{N_{EMR}}$ in the attribution channel by decrypting $N_{EMR} = De_{sk_i}(C_{N_{EMR}})$ and then calculating the hash value $N_{EMR_{hash}} = Hash(N_{EMR})$.

Subsequently, they call the chaincode in the data channel to retrieve *TreatmentData*. This enables patients to access all of their medical record information, which they can then authorize others to view. To protect patient privacy from being compromised, this method of query can only be initiated by the patients themselves. Under this storage method, it is impossible for anyone to retrieve $C_{N_{EMR}}$ by querying $N_{EMR_{hash}}$ in the data channel, thus ensuring the protection of patient privacy.

In the second scenario, DUs can execute a conditional query to obtain the $N_{EMR_{hash}}$ associated with the medical data that they wish to access. The system can then invoke the chaincode on the data channel to retrieve *TreatmentData*. Next, they can initiate sharing requests with the medical institutions that hold the data.

5.5.1. Principle of AES-PRE Implementation

To address the potential theft or tampering of shared data during the data sharing process, the system incorporates proxy re-encryption technology [28] to ensure the security of shared data. The nodes in the IPFS only provide proxy services for key transformation and cannot access plaintext data. The proxy re-encryption algorithm, based on asymmetric encryption, is often inefficient for large volumes of data sharing. Therefore, in the proposed method, we introduce the AES encryption algorithm and combine it with proxy re-encryption to improve the efficiency of the encryption process. We assume that Alice and Bob are the data exchangers. The specific process is described as follows:

1. Alice has sk_A and pk_A, and Bob has sk_B and pk_B.
2. Alice generates a symmetric key, k_{AES}, encrypts the plaintext M with k_{AES} to generate ciphertext C, encrypts k_{AES} with pk_A to generate $C_{(AES)}$, and sends C and $C_{(AES)}$ to the proxy.
3. Bob requests data from Alice and sends his public key pk_B to Alice.
4. Alice generates the proxy re-encryption key $ReKey_{A\to B}$ based on sk_A and pk_B and sends $ReKey_{A\to B}$ to the proxy.
5. The proxy transforms $C_{(AES)}$ into $C_{(AES)_{new}}$ based on $ReKey_{A\to B}$.
6. Bob retrieves C and $C_{(AES)_{new}}$ and then decrypts $C_{(AES)_{new}}$ using sk_B to obtain k_{AES}. With k_{AES}, he decrypts C to retrieve the plaintext data.

With AES-PRE, we can achieve higher efficiency while ensuring data security. We next present one way to implement AES-PRE.

In this method, the AES-256 algorithm is used to encrypt the shared data. A finite field $\mathcal{F}_r$ is defined, where the elements are 256 bits. Let A be the data sender and B be the data receiver. The principle of the AES-PRE process is described as follows.

First, define the generators of two groups. Suppose g and h are generators of groups G_1 and G_2, respectively; then, $Z = e(g,h)$, $e : G_1 \times G_2 \to G_T$.

Next, generate asymmetric keys for A and B as shown in Equation (7):

$$\begin{aligned} sk_A \in \mathcal{F}_r, pk_A = g^{sk_A} \in G_1 \\ sk_B \in \mathcal{F}_r, pk_B = h^{sk_B} \in G_2 \end{aligned} \tag{7}$$

A encrypts the AES key K_{AES} to obtain the ciphertext $C_{(AES)} = \left((pk_A)^k, K_{AES}Z^k\right)$.

Let $\alpha = (pk_A)^k$ and $\beta = K_{AES}Z^k = K_{AES}e(g,h)^k$. Decrypting the ciphertext $C_{(AES)}$ using A's private key sk_A can be described as shown in Equation (8):

$$Dec_{sk_A}\left(C_{(AES)}\right) = K_{AES} \tag{8}$$

A generates the proxy re-encryption key $ReKey_{A\to B}$ using sk_A and pk_B, as shown in Equation (9):

$$ReKey_{A\to B} = (pk_B)^{\frac{1}{sk_A}} \in G_2 \tag{9}$$

According to $C_{(AES)} = (\alpha, \beta)$, then, the Proxy can utilize the re-encryption key $ReKey_{A\to B}$ to compute $\alpha' = e(\alpha, ReKey_{A\to B})$ to obtain the re-encryption ciphertext $C_{(AES)new} = (\alpha', \beta)$.

Finally, data recipient B can decrypt $C_{(AES)new}$ using their own private key to obtain $K_{AES} = ReDec_{sk_B}\left(C_{(AES)new}\right)$. Using K_{AES} decrypts the ciphertext of medical data to obtain its plaintext. For the correctness of the decryption process, a proof of the re-encryption–decryption process can be constructed as follows:

$$\frac{\beta}{(\alpha')^{\frac{1}{sk_B}}} = \frac{K_{AES}e(g,h)^k}{e(\alpha, ReKey_{A\to B})^{\frac{1}{sk_B}}} = \frac{K_{AES}e(g,h)^k}{e\left((pk_A)^k, (pk_B)^{\frac{1}{sk_A}}\right)^{\frac{1}{sk_B}}} = \frac{K_{AES}e(g,h)^k}{e\left((g^{sk_A})^k, (h^{sk_B})^{\frac{1}{sk_A}}\right)^{\frac{1}{sk_B}}} = K_{AES} \tag{10}$$

As shown in Equation (10), AES-PRE satisfies correctness.

5.5.2. Sharing Process

The sharing process described in this subsection focuses on the second scenario described above.

Assume that a user U_{rec} in a healthcare organization HO_j as a DU needs to use the medical data in the system. After the authentication and role verification of U_{rec} is passed, the data index $TreatmentData$ on the blockchain is accessed based on the request to obtain the corresponding $\{N_{EMR_{hash}}, CID, H_i, MD_{EMR}\}$. Send the shared request with $\{N_{EMR_{hash}}, CID, MD_{EMR}\}$ to hospital H_i. When H_i receives the sharing request, H_i queries the IPFS to obtain the medical data ciphertext C_{EMR} based on the CID. Calculate the digest of C_{EMR}: MD'_{EMR} and compare it with MD_{EMR} to verify whether the data have been tampered with and to ensure the data integrity.

After verifying integrity, H_i decrypts C_{EMR} to obtain plaintext M_{EMR}. H_i randomly generates a shared symmetric key $KeyGen(H_i, U_{rec}) \overset{AES}{\rightarrow} k_{AES}$ based on the AES algorithm. In using this key, H_i encrypts the medical data plaintext M_{EMR} and the main information in the $TreatmentData$ record to obtain the shared ciphertext $C_{share(EMR)} = Enc_{k_{AES}}(M_{EMR}, H_i, N_{EMR_{hash}})$. Data sharing security is protected through AES-PRE.

After receiving the data forwarded by the IPFS, U_{rec} decrypts it using private key sk_{rec} to obtain the AES key k_{AES}. With k_{AES}, U_{rec} decrypts $C_{share(EMR)}$ to obtain the plaintext medical data M_{EMR}, along with other information such as $N_{EMR_{hash}}$ and H_i. Upon obtaining the plaintext medical data, U_{rec} can encrypt them using the public key of H_i, calculate the message digest MD'_{EMR}, and sign it as $sign = Sign(MD'_{EMR}, U_{rec})$. In querying the blockchain using $N_{EMR_{hash}}$ to retrieve the corresponding $TreatmentData$, U_{rec} verifies the integrity of the medical data by comparing the stored MD_{EMR} in the index. This allows U_{rec} to determine whether the data have been tampered with by the sender. Upon successful verification, U_{rec} sends $N_{EMR_{hash}}$, MD'_{EMR}, and $sign$ to the IPFS.

The IPFS uploads the constructed data structure for U_{rec}'s medical data usage record to the blockchain network for persistent storage as follows: $DataUsageRecord = \{N_{EMR_{hash}}, UID_{rec}, MD_{EMR}', sign, HO_j, H_i, t\}$.

6. Scheme Analysis

6.1. Security Analysis

6.1.1. Medical Data Security

All users requiring access to the data must be authenticated. Data access control is enforced based on the identity of the DU, preventing unauthorized access. The encrypted medical data generated by the system are stored in the IPFS distributed storage servers. Data sharing is protected through proxy re-encryption, ensuring that IPFS nodes only provide computational power for ciphertext transformation and cannot access the plaintext medical data.

For data integrity, on one hand, the IPFS utilizes content addressing, ensuring the integrity of the data. On the other hand, the cryptographic hash of the medical data ciphertext is persistently stored on the blockchain ledger. When users access the data, they are required to sign and verify the integrity of the medical data. Additionally, usage records are uploaded to the blockchain for storage, ensuring that encrypted data are not altered and that usage is traceable.

6.1.2. User Anonymity

All users operate with pseudo-identities and use ZKP technology for identity verification, which can ensure the correctness and validity of user identities without exposing any real private information.The security of the ZKP depends on the specific algorithm implementation. In our scheme, the generation and verification of proof are based on the Groth16 algorithm-implemented zk-SNARKs, which satisfies the completeness and soundness properties of a ZKP [27]. Therefore, user anonymity can be guaranteed during the interaction process.

6.1.3. Patient Privacy and Security

Our proposed scheme stores $C_{N_{EMR}}$ and uid as key–value pairs in the attribution channel, while in the data channel, it saves key–value pairs formed by $N_{EMR_{hash}}$ and

$TreatmentData$. The confidentiality of the relationship between $C_{N_{EMR}}$ and $N_{EMR_{hash}}$ is crucial for patient privacy security. Once the relationship is exposed, attackers can obtain the uid of patients associated with $TreatmentData$, thus compromising patient privacy. Based on this dual-channel mechanism, the threat of data leakage in the system mainly includes two scenarios: one of the channels was attacked or both of the channels were attacked.

(A) One of the Channels Was Attacked

Assume that one of the channels is compromised by a malicious node that obtains all the ledger data in that channel. The design of dual channels provides a high degree of isolation, ensuring that attackers still cannot obtain the association between medical data and the corresponding patients. This feature provides additional assurance for patient privacy and security.

(B) Both of the Channels Were Attacked

Assume that both of the channels have malicious nodes and have access to all the ledger data in the system. We proceed to analyze the probability of exposure of the relationship between $C_{N_{EMR}}$ and $N_{EMR_{hash}}$.

Hash functions are one-way, meaning that for any given original medical record number N_{EMR}, one can compute the corresponding $N_{EMR_{hash}} = hash(N_{EMR})$. However, if an attacker only obtains $N_{EMR_{hash}}$, it is computationally infeasible to calculate the original medical record number $NEMR$ using $hash^{-1}(N_{EMR_{hash}})$, as the inverse function $hash^{-1}$ either does not exist or is extremely difficult to compute. With the assurance of patient key security, attackers cannot directly obtain the mapping between $C_{N_{EMR}}$ and $N_{EMR_{hash}}$.

If $\{C_{N_{EMR}}, uid\}_{i,1\leq i\leq n}$ in the attribution channel and $\{N_{EMR_{hash}}, TreatmentData\}_{j,1\leq j\leq n}$ in the data channel are obtained, then the probability of obtaining $N_{EMR_{hash\,i}}$ corresponding to $C_{N_{EMR\,i}}$ is

$$P_i(n) = 1/n \tag{11}$$

where n in Equation (11) is the number of EMRs stored in the system, and as the number of stored EMRs increases, the probability of obtaining $N_{EMR_{hash\,i}}$ corresponding to $C_{N_{EMR\,i}}$ decreases gradually.

If $\{C_{N_{EMR}}, uid\}_{i,1\leq i\leq n}$ in the attribution channel and $\{N_{EMR_{hash}}, TreatmentData\}_{j,1\leq j\leq n}$ in the data channel are obtained, then patient U is assumed to have z EMRs, and the probability of obtaining the attribution relationship of z EMR of U is

$$P(z,n) = \prod_{i=1}^{z} P_i(n+1-i) \tag{12}$$

The analysis of the monotonicity of Equation (12) reveals that as the n or the z gradually increases, $P(z,n)$ will decrease significantly. Considering real-world scenarios, as the system continues to be used, both the total number of stored EMRs in the system and the number of EMRs owned by patients will gradually increase. Therefore, the above probability will decrease gradually, indicating that the privacy of patients will be increasingly secure.

Even in the scenario where collusion exists among nodes in both channels, and attackers can access all ledger data in the two channels, they still cannot obtain the plaintext N_{EMR}, as long as patient keys remain secure. When confronted with extremely large EMRs, it is extremely difficult for an attacker to obtain a correspondence between $C_{N_{EMR}}$ and $N_{EMR_{hash}}$. Since $N_{EMR_{hash}} \neq C_{N_{EMR}}$, it follows that $TreatmentData \neq uid$, meaning that medical data cannot be associated with patient information.

In summary, with patient key security assured, it can be considered that patients are anonymous in the blockchain network of this scheme, and their personal privacy will not be compromised due to data mining or other attack methods. However, this study assumes that patient keys remain undisclosed in discussing security issues. The improper storage of patient keys could pose significant privacy risks. Therefore, in future work, further research should focus on enhancing key management and addressing data update issues resulting from key changes.

6.2. Performance Analysis

This section evaluates the proposed scheme in terms of feature comparison, time overhead, storage overhead, and throughput. The experiments were conducted based on the Hyperledger Fabric v2.4 framework. The experimental platform hardware consisted of an Intel® Core™ i5-10400 CPU @ 2.9GHz and 16 GB of RAM, running on a 64-bit operating system. The Hyperledger Fabric network was deployed using Docker on a virtual machine running Ubuntu 20.04.

6.2.1. Feature Comparison

To analyze the functional advantages of the proposed sharing scheme, this section compares it with existing medical data sharing schemes. As shown in Table 3, we compare our proposed scheme with the schemes in [29–32] in terms of original data storage location, anonymity, resistance to data mining threats, decentralized identity authentication, and conditional query functionality. Resistance to data mining threats indicates the ability to withstand threats to patient privacy exposure due to data mining.

Table 3. Feature comparison.

Scheme	Original Data Storage Location	Anonymity	Resisting Data Mining Threats	Decentralized Identity Authentication	Conditional Query
[29]	Cloud Server	✓	×	×	×
[30]	IPFS	✓	×	×	×
[31]	Cloud Server	×	×	✓	×
[32]	Centralized Database	✓	×	×	×
Our scheme	IPFS	✓	✓	✓	✓

Refs. [29,30,32] and our proposed scheme all use pseudo-identity information for patients to protect patients' privacy. In contrast, ref. [31] chose to enter patients' real identifying information into the system and relied on access control policies to determine whether different users are authorized to access patients' personal information. This scheme provides security but shows limitations in maintaining user anonymity.

Compared to the other literature, ref. [29] and the approach proposed in this paper are more closely aligned in terms of methodology, as both adopt a dual-channel mechanism for managing medical data, thus enhancing data security. However, refs. [29,30,32] still directly associates the pseudo-identity information used on the blockchain with the medical record number, and the medical record number has not undergone appropriate privacy protection. These schemes fail to fully defend against the risk of data mining attacks. The scheme proposed in this paper has better protection measures for privacy.

In terms of identity verification, both [31] and the scheme proposed in this paper conduct decentralized user authentication on the blockchain, providing stronger privacy protection. Simultaneously, our scheme optimizes the efficiency of blockchain conditional queries, significantly reducing the time cost required for such queries. These improvements ensure that the sharing of EMR is both secure and efficient, offering a novel solution for the field of medical data sharing.

6.2.2. Time Overhead

The symbols and explanations for the time overhead of the main operations in this scheme are as shown in Table 4, where t_{ctd}, t_{qtd}, t_{cud}, t_{qud}, t_{cai}, t_{qai}, and t_{vp} represent the time overhead when executing certain chaincode operations.

Time overhead is a key indicator reflecting system performance, with a larger time overhead leading to a noticeable decrease in system efficiency. To evaluate the impact of each stage in the proposed scheme on system time overhead, as well as the influence of different data sizes on the time overhead of each stage, we conducted relevant tests. Selecting data of different sizes (64 KB/256 KB/1 MB) for time overhead tests during the data generation, uploading, and sharing phases yielded the results shown in Table 5. From the analysis, it can be concluded that the size of the data has a noticeable impact on the time overhead for each phase, with larger data resulting in greater time overhead.

Table 4. The symbols and explanations for the time overhead of the main operations in this scheme.

Symbol	Explanation
t_{pk}^{en}	Encrypting medical record data with public keys
t_{sk}^{de}	Decrypting medical record data with private key
$t_{md_{emr}}$	Calculating medical record data message digest
$t_{N_{emr_{hash}}}$	Calculating the hash value of the medical record number
$t_{N_{emr}}^{en}$	Encrypting medical record number
$t_{N_{emr}}^{de}$	Decrypting medical record number
t_{sign}	Doctor signs the medical record data
t_{ctd}	Creating *Treatement* to upload to data channel
t_{qtd}	Querying *Treatement* on the data channel
t_{AES}^{en}	Encrypting medical record data with AES
t_{AES}^{de}	Decrypting medical record data with AES
t_{grk}	Generate re-encryption key
t_{pre}	Proxy re-encryption
t_{cud}	Creating *DataUsageRecord* to upload to data channel
t_{qud}	Querying *DataUsageRecord* on the data channel
t_{uIPFS}	Uploading encrypted data to the IPFS to generate CID
t_{dIPFS}	Downloading encrypted data from the IPFS
t_{cai}	Creating $AttributionInfo$ to upload to attribution channel
t_{qai}	Querying $AttributionInfo$ on the attribution channel
t_{vi}	Verifying data integrity
t_{vp}	Verifying the validity of pf

Table 5. Time overhead for major phases in this scheme (ms).

Phase	Time Overhead	Result (64 KB/256 KB/1 MB)
Data generation	$t_{pk}^{en} + t_{md_{emr}} + t_{sign}$	87/295/1061
Data uploading	$t_{uIPFS} + t_{ctd} + t_{cai}$	127/136/148
Data sharing	$t_{vp} + t_{qtd} + t_{dIPFS} + 2t_{vi} + t_{sk}^{de} + t_{aes}^{en} + t_{grk} + t_{pre} + t_{aes}^{de} + t_{cud} + t_{md_{emr}} + t_{sign}$	616/2094/7756

To further explore the impact of data size on the performance of the blockchain network in our proposed scheme, we tested the time overheads of the main functions of the chaincode, and the results are shown in Table 6. The results indicate that regardless of variations in the size of the medical record data, the impact on the time overhead required for executing the chaincode in this scheme is minimal, suggesting a relatively small effect on the computational performance of the blockchain. Additionally, the average time required for verifying a zero-knowledge proof on the blockchain is 37 ms, indicating a similarly limited impact on the overall system operation.

Table 6. Time overheads of the main functions of the chaincodes in this scheme (ms).

Data Size	t_{ctd}	t_{qtd}	t_{cud}	t_{qud}	t_{cai}	t_{qai}	t_{vp}
64 KB	44.47	1.27	43.56	1.28	43.46	1.21	36.04
256 KB	47.37	1.23	46.81	1.23	42.71	1.28	37.21
1 MB	44.90	1.30	44.61	1.26	43.37	1.22	37.87

In combining the results from Tables 5 and 6, it can be concluded that the impact of data size on time overhead is primarily evident in the encryption and decryption processes, with some impact also seen during the IPFS upload and download processes. When the data size exceeds 1 MB, the time overhead for zero-knowledge verification is extremely low, significantly less than 1%. These results demonstrate that the blockchain-based scheme proposed in this paper maintains high computational efficiency and low resource utilization even when handling large-scale data.

Conditional querying is an important part in implementing EMR sharing, and the efficiency of conditional querying affects the efficiency of the whole sharing process. We propose a method for creating composite keys to improve the efficiency of system conditional querying. To demonstrate the feasibility of the method and evaluate the advantages of the optimization scheme, we conducted a separate time overhead test specifically for this functional module.

We used the scenario of conditional querying described in Section 5.4 as an example for the subsequent tests. In this scenario, we only need two attributes, hospital name and diagnosis result, to construct composite keys. Then, we use the hash value of the EMR's number as the value corresponding to this composite key. For this purpose, we simulated 1479 records and divided the records into five groups based on the ratio P as shown in Equation (13):

$$P = \frac{m}{n}. \tag{13}$$

where n in Equation (13) represents the total number of EMRs for a certain diagnosis result DR_i, and m represents the number of EMRs containing DR_i in hospital H. The grouping results are shown in Table 7.

Table 7. Grouping results.

Group	1	2	3	4	5
Ratio $P(\%)$	100–20	20–15	15–10	10–5	5–0
Number of records	606	138	188	207	340

We compared the query time overhead of our proposed method with the general method that does not use composite key. We repeated the query multiple times for each group of data in Table 7 using both methods and took the average of the query time for each group as the experimental result.

As shown in Table 8, with decreasing P, the query time overhead of the general method gradually increases. This is because as the P decreases, the total number of EMRs corresponding to the given diagnosis result in the conditional query increases, leading to an increase in the number of records that need to be filtered, thus resulting in a decreased query performance. In contrast, the query time spent by the proposed method based on the composite key is significantly less than that of the general method. This is because using the method of creating composite keys can accurately locate the data that satisfy all query conditions, requiring only the retrieval of the data that meet the conditions, thereby highlighting the superiority of the composite key-based method in terms of the query time performance.

Table 8. Query time overhead results (ms).

Group	General Method	Our Method
1	182	72
2	293	68
3	344	59
4	518	56
5	847	51

Data transmission is a crucial process in electronic medical record sharing, with AES-PRE being the key technology ensuring secure data transmission. While ensuring data security, AES-PRE can improve transmission efficiency. In order to test the performance impact and benefits of using the AES-PRE method in the data sharing phase, we compared it with the method of directly re-encrypting the ciphertext of medical data (DReC) in terms of the time overhead. Figure 4 illustrates the processing time results of our scheme and DReC for different sizes of medical data. As the size of the shared data gradually increases, DReC's time overhead also increases. In contrast, the time overhead of the AES-PRE process varies very little with the size of the data. For the same data size, the incurred time overhead was also smaller than that of DReC, demonstrating that our scheme is more efficient in data sharing.

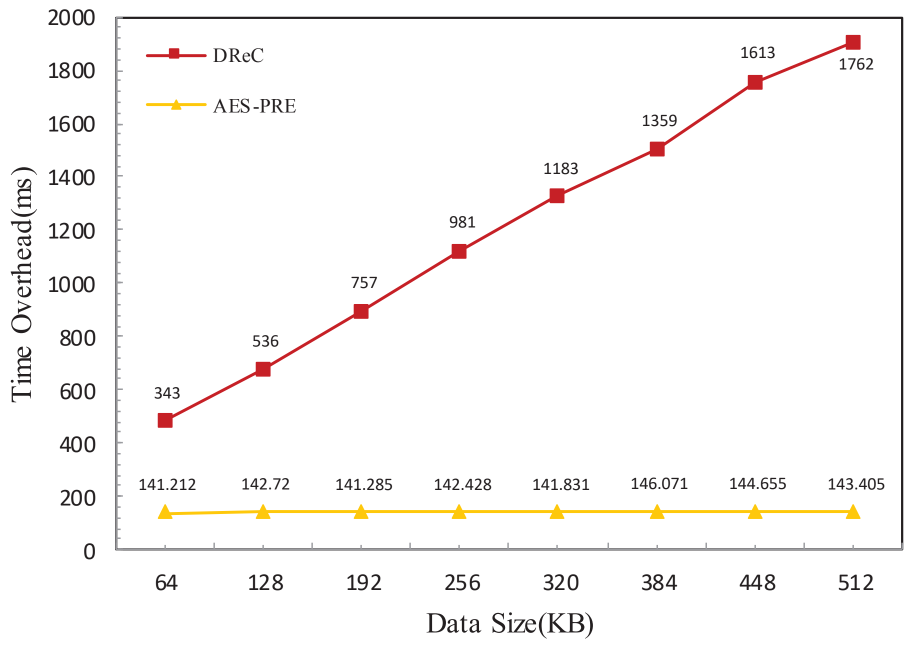

Figure 4. Time overhead comparison between AES-PRE and DReC.

6.2.3. Storage Overhead

To test the impact of the different sizes of data on the blockchain's storage overhead in our scheme, we conducted tests on the storage overhead in both the blockchain and IPFS. By uploading data of various sizes and performing multiple tests to obtain averages, we compared the storage overhead generated by the blockchain and IPFS. In this context, the returned CID from the IPFS is a constant 46 bytes (SHA-256, Base64 encoding). As shown in Figure 5, it can be observed that the data size has almost no effect on the storage overhead on the blockchain network.

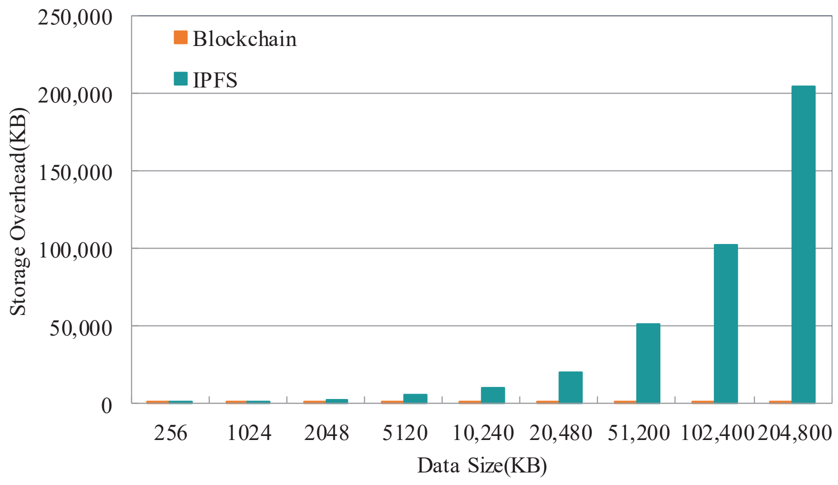

Figure 5. Storage overhead comparison between blockchain and IPFS.

6.2.4. Throughput

The efficiency of the blockchain network is crucial for the overall system performance. Throughput is an important indicator of the performance of a blockchain network. Therefore, this study conducted throughput tests on the blockchain network to evaluate the system's performance. The Caliper tool was utilized to conduct throughput tests for adding and querying relevant records in the blockchain. Figure 6 illustrates the impact of transaction request volume on the throughput of the blockchain network when adding *TreatmentData* and *DataUsageRecord* to the blockchain. It also depicts the effect of different data sizes on the throughput of the blockchain network when medical data are directly stored (DS) on the blockchain. The throughput represents the number of successfully submitted transactions per second (TPS). As the transaction request volume gradually increases, the throughput of our scheme also increases. When the transaction request volume reaches around 550, the throughput of our scheme tends to stabilize at around 50 TPS.

In contrast, the throughput of the DS is significantly lower than our scheme. Additionally, as the size of the stored data increases, the throughput of the DS obviously decreases. As shown in Figure 5, the size of the data stored by our scheme on the blockchain remains relatively constant at about 6 KB. Therefore, our scheme has a clear advantage in this aspect.

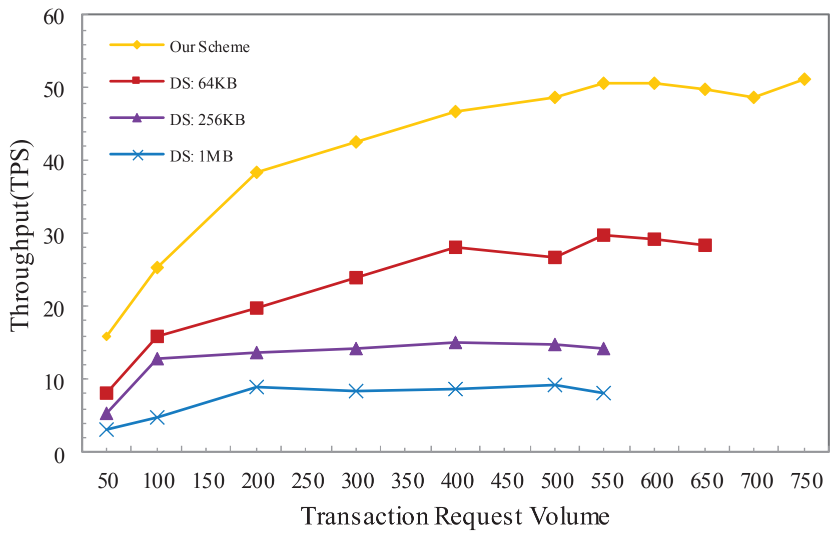

Figure 6. Throughput when adding *TreatmentData* and *DataUsageRecord* to the blockchain.

7. Conclusions

This study proposed an EMR storage and sharing solution based on Hyperledger Fabric and IPFS. Storing patient information and medical data in separate channels mitigates the privacy threat posed by data mining that could leak patient identities. Introducing an IPFS alleviates the deficiency in the blockchain storage performance. zk-SNARKs are used as an implementation of decentralized authentication, avoiding third-party trustworthiness issues. The efficient and secure transmission of medical data is ensured through AES-PRE. In addition, creating composite key improves the efficiency of queries in the Hyperledger Fabric network. Security analyses and experimental results show that these efforts provide significant advances in EMR sharing, offering solutions to the privacy, security, and efficiency challenges.

Our scheme can be further improved in several aspects. Firstly, we will explore how to strengthen patient key management to further enhance privacy protection in our future work. Secondly, creating composite keys results in a large amount of redundant storage, and the introduction of indexes can be considered in the future to reduce redundancy. These efforts can further improve system security and efficiency.

Author Contributions: Conceptualization, methodology, software, and writing, J.G.; validation, supervision, and revision, K.Z., Z.L. and K.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China under Grant No. 2020YFB1805400.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Han, Y.; Fang, X. Systematic review of adopting blockchain in supply chain management: Bibliometric analysis and theme discussion. *Int. J. Prod. Res.* **2024**, *62*, 991–1016. [CrossRef]
2. Mathur, S.; Kalla, A.; Gür, G.; Bohra, M.K.; Liyanage, M. A survey on role of blockchain for IoT: Applications and technical aspects. *Comput. Netw.* **2023**, *227*, 109726. [CrossRef]
3. Khashan, O.A.; Khafajah, N.M. Efficient hybrid centralized and blockchain-based authentication architecture for heterogeneous IoT systems. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 726–739. [CrossRef]
4. Merlo, V.; Pio, G.; Giusto, F.; Bilancia, M. On the exploitation of the blockchain technology in the healthcare sector: A systematic review. *Expert Syst. Appl.* **2023**, *213*, 118897. [CrossRef]
5. Khalid, M.I.; Ehsan, I.; Al-Ani, A.K.; Iqbal, J.; Hussain, S.; Ullah, S.S. A comprehensive survey on blockchain-based decentralized storage networks. *IEEE Access* **2023**, *11*, 10995–11015. [CrossRef]
6. Ren, Y.; Huang, D.; Wang, W.; Yu, X. BSMD: A blockchain-based secure storage mechanism for big spatio-temporal data. *Future Gener. Comput. Syst.* **2023**, *138*, 328–338. [CrossRef]
7. Cao, S.; Zhang, X.; Xu, R. Toward secure storage in cloud-based ehealth systems: A blockchain-assisted approach. *IEEE Netw.* **2020**, *34*, 64–70. [CrossRef]
8. Carter, G.; Shahriar, H.; Sneha, S. Blockchain-based interoperable electronic health record sharing framework. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019 ; Volume 2, pp. 452–457.
9. Huang, H.; Sun, X.; Xiao, F.; Zhu, P.; Wang, W. Blockchain-based eHealth system for auditable EHRs manipulation in cloud environments. *J. Parallel Distrib. Comput.* **2021**, *148*, 46–57. [CrossRef]
10. Xia, Q.; Sifah, E.B.; Asamoah, K.O.; Gao, J.; Du, X.; Guizani, M. MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access* **2017**, *5*, 14757–14767. [CrossRef]
11. Fu, J.; Wang, N.; Cai, Y. Privacy-preserving in healthcare blockchain systems based on lightweight message sharing. *Sensors* **2020**, *20*, 1898. [CrossRef]
12. Wang, Y.; Zhang, A.; Zhang, P.; Wang, H. Cloud-assisted EHR sharing with security and privacy preservation via consortium blockchain. *IEEE Access* **2019**, *7*, 136704–136719. [CrossRef]

13. Dagher, G.G.; Mohler, J.; Milojkovic, M.; Marella, P.B. Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustain. Cities Soc.* **2018**, *39*, 283–297. [CrossRef]
14. Akkaoui, R.; Hei, X.; Cheng, W. EdgeMediChain: A hybrid edge blockchain-based framework for health data exchange. *IEEE Access* **2020**, *8*, 113467–113486. [CrossRef]
15. Chen, L.; Lee, W.K.; Chang, C.C.; Choo, K.K.R.; Zhang, N. Blockchain based searchable encryption for electronic health record sharing. *Future Gener. Comput. Syst.* **2019**, *95*, 420–429. [CrossRef]
16. Azaria, A.; Ekblaw, A.; Vieira, T.; Lippman, A. Medrec: Using blockchain for medical data access and permission management. In Proceedings of the 2016 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, 22–24 August 2016; pp. 25–30.
17. Cachin, C. Architecture of the hyperledger blockchain fabric. In Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, USA, 25 July 2016; Volume 310, pp. 1–4.
18. Zhong, B.; Wu, H.; Ding, L.; Luo, H.; Luo, Y.; Pan, X. Hyperledger fabric-based consortium blockchain for construction quality information management. *Front. Eng. Manag.* **2020**, *7*, 512–527. [CrossRef]
19. Tanwar, S.; Parekh, K.; Evans, R. Blockchain-based electronic healthcare record system for healthcare 4.0 applications. *J. Inf. Secur. Appl.* **2020**, *50*, 102407. [CrossRef]
20. Al-Sumaidaee, G.; Alkhudary, R.; Zilic, Z.; Swidan, A. Performance analysis of a private blockchain network built on Hyperledger Fabric for healthcare. *Inf. Process. Manag.* **2023**, *60*, 103160. [CrossRef]
21. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
22. Dwivedi, A.D.; Singh, R.; Ghosh, U.; Mukkamala, R.R.; Tolba, A.; Said, O. Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for Internet of Things. *J. Ambient. Intell. Humaniz. Comput.* **2022**, *13*, 4639–4649. [CrossRef]
23. Chen, T.; Lu, H.; Kunpittaya, T.; Luo, A. A review of zk-snarks. *arXiv* **2022**, arXiv:2202.06877.
24. Ben-Sasson, E.; Bentov, I.; Horesh, Y.; Riabzev, M. Scalable, transparent, and post-quantum secure computational integrity. *Cryptol. ePrint Arch.* **2018**. Available online: https://eprint.iacr.org/2018/046 (accessed on 16 April 2024).
25. Bünz, B.; Bootle, J.; Boneh, D.; Poelstra, A.; Wuille, P.; Maxwell, G. Bulletproofs: Short proofs for confidential transactions and more. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 315–334.
26. Eberhardt, J.; Tai, S. Zokrates-scalable privacy-preserving off-chain computations. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1084–1091.
27. Groth, J. On the size of pairing-based non-interactive arguments. In Proceedings of the Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, 8–12 May 2016; Proceedings, Part II 35; Springer: Berlin/Heidelberg, Germany, 2016; pp. 305–326.
28. Blaze, M.; Bleumer, G.; Strauss, M. Divertible protocols and atomic proxy cryptography. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 127–144.
29. Chen, Z.; Xu, W.; Wang, B.; Yu, H. A blockchain-based preserving and sharing system for medical data privacy. *Future Gener. Comput. Syst.* **2021**, *124*, 338–350. [CrossRef]
30. Jayabalan, J.; Jeyanthi, N. Scalable blockchain model using off-chain IPFS storage for healthcare data security and privacy. *J. Parallel Distrib. Comput.* **2022**, *164*, 152–167. [CrossRef]
31. Saidi, H.; Labraoui, N.; Ari, A.A.A.; Maglaras, L.A.; Emati, J.H.M. DSMAC: Privacy-aware Decentralized Self-Management of data Access Control based on blockchain for health data. *IEEE Access* **2022**, *10*, 101011–101028. [CrossRef]
32. Oksuz, O. A System For Storing Anonymous Patient Healthcare Data Using Blockchain And Its Applications. *Comput. J.* **2024**, *67*, 18–30. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.