Enhancing Health Record Security and Privacy with Blockchain-Based Access Management

Mallellu Sai Prashanth[1](✉), Ramesh Karnati[1], Muni Sekhar Velpuru[2], and H. Venkateshwara Reddy[1]

[1] Department of Computer Science and Engineering, Vardhaman College of Engineering, Hyderabad, India
saiprashanth08@ieee.org, {ramesh.krnt, h.venkateswarareddy}@vardhaman.org

[2] Department of Information Technology, Vardhaman College of Engineering, Hyderabad, India
munisek@vardhaman.org

Abstract. Access control is a critical component of medical health record management, ensuring that only authorized individuals can access sensitive patient data. Existing access control schemes suffer from limitations such as inefficiency, lack of patient control, and inadequate protection mechanisms. In this paper, we propose an advanced, efficient and enhancing Health Record Security and Privacy with Blockchain-based Access Management. Our proposed scheme leverages the benefits of blockchain technology to provide a decentralized, secure, and transparent access control framework that gives patients greater control over their data while also providing efficient protection mechanisms. The proposed scheme's access control mechanisms include authentication, authorization, and auditing, while its protection mechanisms include data encryption, digital signatures, and privacy-preserving techniques. The patient control features of the proposed scheme include consent management and revocation. We evaluate the effectiveness and efficiency of the proposed scheme using performance metrics and benchmarking results. Our results Show that the suggested plan is more effective than the current ones in terms of efficiency, patient control, and protection mechanisms. The proposed scheme represents a significant step forward in the management of medical health records, providing a secure, efficient, and patient-centric access control framework that ensures privacy and confidentiality while also enabling effective healthcare delivery.

Keywords: Block Chain · Medical health Records · Trust · Reputation · social control and privacy

1 Introduction

Medical health records (MHR) are electronic or paper-based documents that contain information related to a patient's medical history and healthcare treatment. These records are maintained by healthcare providers and may include a variety of information, such

O. Castillo et al. (Eds.): PerSOM 2023, LNICST 517, pp. 182–202, 2024.
https://doi.org/10.1007/978-3-031-66044-3_13

as the patient's medical history, lab results, medications, allergies, imaging studies, and diagnoses [1]. MHRs are important for healthcare providers because they provide a complete picture of a patient's health, allowing providers to make informed decisions about the patient's care. MHR scan also help to prevent medical errors, reduce duplication of tests and procedures, and facilitate communication between healthcare providers [2]. Patients also have the right to access their MHRs, and can use them to better understand their own health and medical history. MHRs can also be shared with other healthcare providers, such as specialists or emergency room staff, to ensure that patients receive appropriate and coordinated care [3]. Approprivate safeguards, such as secure storage and access controls, must be put in place to protect patients 'privacy and prevent unauthorized access to their health information [4].

The computerized copies of patients' medical records known as electronic health records (EHRs) are stored and managed electronically. The adoption of EHRs has become increasingly widespread in recent years, and they offer several benefits over traditional paper-based records management. One of the primary benefits of EHRs is that they can improve the quality of patient care [5]. EHRs give medical professionals immediate access to up-to-date and comprehensive patient data, like medical history, prescriptions, allergies, and test results. This information can aid professionals in making more knowledgeable choices regarding patient care, avoid medical errors, and identify potential health risks [6]. EHRs can also help providers to identify gaps in care, such as missed vaccinations or screenings, and prompt them to take action to address these gaps. EHRs can also improve healthcare efficiency and reduce costs. With EHRs, healthcare providers can quickly and easily share patient information with other providers primary healthcare physicians, specialists, and other health care professionals involved in a patient's hospitals [7, 8]. This can help to reduce the duplication of tests and procedures, prevent medication errors, and reduce administrative costs associated with paper-based records management [9]. EHRs can also facilitate communication between patients and providers, such as by allowing patients to access their own medical records and communicate with their healthcare team through secure messaging platforms [10].

However, there are also some challenges associated with EHRs. The initial implementation of an EHR system can be expensive, and there may be ongoing costs associated with system maintenance and upgrades [11]. There is also a risk of data breaches or other security incidents, which could compromise patient privacy and result in legal or financial consequences for healthcare providers. Additionally, there is a risk of information overload or alert fatigue, which can occur when providers receive too much information from EHRs and have difficulty prioritizing or acting on this information [12].

2 Related Work

2.1 Blockchain Technology

Blockchain technology has emerged as a revolutionary innovation that has the potential to revolutionize a number of different sectors. At its core, blockchain is a decentralized digital ledger that records transactions in a secure and transparent manner [13]. The technology is based on a network of nodes that work together to verify and validate

transactions, ensuring that data is accurate and tamper-proof. The blockchain is composed of a series of blocks that are linked together in a chain [14]. Each block contains a set of transactions that are verified and validated by the network of nodes. Once a block is verified, it is added to the chain, creating an immutable record of all transactions [15].

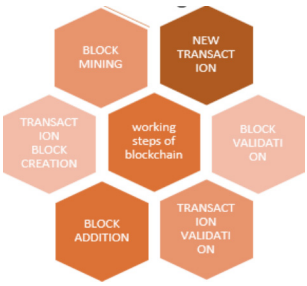

Fig. 1. Working Steps of Blockchain

The Fig. 1 is demonstrating the working steps of blockchain. It operates by setting up a network of computers (called nodes) that each hold a duplicate of the ledger. Transactions are also made private and safe through the use of cryptography. Creating a decentralized network, using consensus mechanisms to confirm transactions, and using cryptography to ensure the security and immutability of the ledger are the main steps in how blockchain functions.

Blockchain technology has the potential to revolutionize the way medical health records are managed, providing a secure, transparent, and patient-centric approach to data storage and sharing [16]. Medical health records contain sensitive information that must be protected and managed in a way that ensures privacy and confidentiality. However, existing systems suffer from limitations such as inefficiency, lack of patient control, and inadequate protection mechanisms. Blockchain technology provides a solution to these limitations by creating a decentralized, secure, and transparent access control framework that gives patients greater control over their data while also providing efficient protection mechanisms. The proposed Access control for medical records based on block-chain can be broken down into several key components such as decentralization encryption digital Signatures etc. [17].

The use of blockchain technology in medical health records has several benefits. Firstly, it improves data accuracy and reduces errors by creating a tamper-proof record of all transactions [18]. Secondly, it ensures that patients have greater control over their data, including the ability to grant and revoke access to their medical health records. Thirdly, it enhances privacy and confidentiality by providing an additional layer of protection against unauthorized access. In conclusion, blockchain technology provides a promising solution to the limitations of existing medical health record management systems. By leveraging the benefits of blockchain, we can create a more efficient, secure, and patient-centric approach to medical health record management, ensuring that patient data is protected and managed in a way that is transparent, secure, and trustworthy [19].

Ethereum is a decentralized, open-source blockchain platform that enables developers to build decentralized applications (dApps) and smart contracts. Ethereum uses

its native cryptocurrency, Ether (ETH), to facilitate transactions and incentivize network participants. Ethereum is based on a different consensus algorithm than Bitcoin called Proof of Stake (PoS).This means that instead of miners competing to validate transactions by solving complex mathematical problems, validators are chosen based on their stake in the network. Validators are required to hold a certain amount of ETH as collateral, which they risk losing if they validate fraudulent transactions. The PoS consensus mechanism makes Ethereum more energy-efficient and faster than Bitcoin. One of Ethereum's main features is its ability to execute smart contracts, which are self-executing contracts that automatically enforce the terms of an agreement [20].

Smart contracts can be used to automate a wide range of processes, including payments, asset transfers, and identity verification. They are executed on the Ethereum Virtual Machine (EVM), a runtime environment that executes code on the blockchain. Ethereum's smart contract functionality has enabled the creation of decentralized applications (dApps) that can be run on the Ethereum network [21]. These dApps are built on top of the Ethereum block-chain and can be used to facilitate a wide range of functions, such as financial services, gaming, and supply chain management. Ethereum's dApps are decentralized, meaning that they are not controlled by any central authority, making them more transparent and resistant to censorship. Ethereum also enables developers to create and issue their own cryptocurrencies, called ERC-20 tokens, which can be used to represent a variety of assets, such as stocks, bonds, or even physical assets like real estate [21, 22]. These tokens are built on top of the Ethereum network and can be traded on decentralized exchanges (DEXs), which operate without a central authority or intermediary [23].

Smart contracts are self-executing computer programs that can be programmed to automatically execute specific actions when certain conditions are met. In the context of medical health records, smart contracts can be used to facilitate the sharing of patient data between different healthcare providers and organizations. The main advantage of using smart contracts in medical health records is that they can help ensure that patient data is shared securely and transparently. For example, a smart contract could be used to specify the conditions under which a patient's medical records can be accessed by a healthcare provider. These conditions could include things like the patient's consent, the type of information that can be accessed, and the duration of the access [24].

Overall, smart contracts have the potential to revolutionize the way that medical health records are managed and shared However, in order for smart contracts to be successful, it is important to ensure that they are developed and implemented correctly, and that they are interoperable with other systems and technologies. Programmed to automatically execute the transaction, without the need for any human intervention [26]. This means that healthcare providers can access patient data more quickly and efficiently, while ensuring that the patient's privacy and security are maintained. Another advantage of using smart contracts in medical health records is that they can help reduce the administrative burden associated with data sharing. Since smart contracts are self-executing, they can help streamline the data sharing process, reducing the need for manual interventions and reducing the risk of errors and mistakes [25].

3 Proposed Model

The proposed research paper aims to develop a novel approach for enhancing the security and privacy of medical health records through a blockchain-based access control scheme. The proposed scheme utilizes a decentralized blockchain network to store and manage health data access permissions, enabling patients to control who can access their data while ensuring efficient data sharing between authorized parties. The proposed system incorporates an efficient protection mechanism, including data encryption, digital signatures, and smart contract-based access control rules, to prevent unauthorized access and ensure data integrity. Moreover, the proposed scheme empowers patients to manage their health data and grants them full control over their information, allowing them to revoke access permissions at any time. The proposed system's effectiveness will be evaluated from day to day medical health data, and the results will be compared against existing health record access control approaches to show the uniqueness of the proposed scheme in specific terms like security, efficiency, and patient control.

The goal of the blockchain medical record access control system is to provide people control over people who can avail their personal health data while offering an effective and safe way to manage medical data. With the help of the Ethereum and the Solidity smart contracts, this concept establishes a decentralised process that guarantees information security and privacy. The authorized Users, accessPermissions, records, and a struct named Record are the four primary mappings in the contract.

The users who are permitted access to medical reports can be found in the authorised Users mapping. Every user has access permissions to a list of health records that are included in the access Permissions mapping. A list of health records decrypted on the block chain is included in the records mapping. The hash of the health record, the patient's address, a timestamp, and a description are all stored in the Record struct.

The Fig. 2 is demonstrating the architecture for blockchain based enhanced security and access management. The detailed architecture diagram is demonstrating the smart contract of medical record which is enhancing the security through blockchain.

1) **Smart contract:**
   The smart contract contains the following variables:

   Variables:
   Owner: The address of the owner of the contract.
   authorizedUsers: an address to Boolean mapping. The list of authorised users is stored there.
   accessPermissions: a map from address to Boolean via a hierarchical mapping of record hashes. Each user's and record's access permissions are stored there.
   records: A mapping which maps record hashes to a Record struct. It preserves the medical records. The Record struct contains the following fields:
   patient: The address of the patient who owns the record.
   hash: It is the hash value of the record.
   timestamp: The timestamp when the record was added. Description: A string describing the contents of the record.
2) **Users:** The users of the system can be divided into three categories:

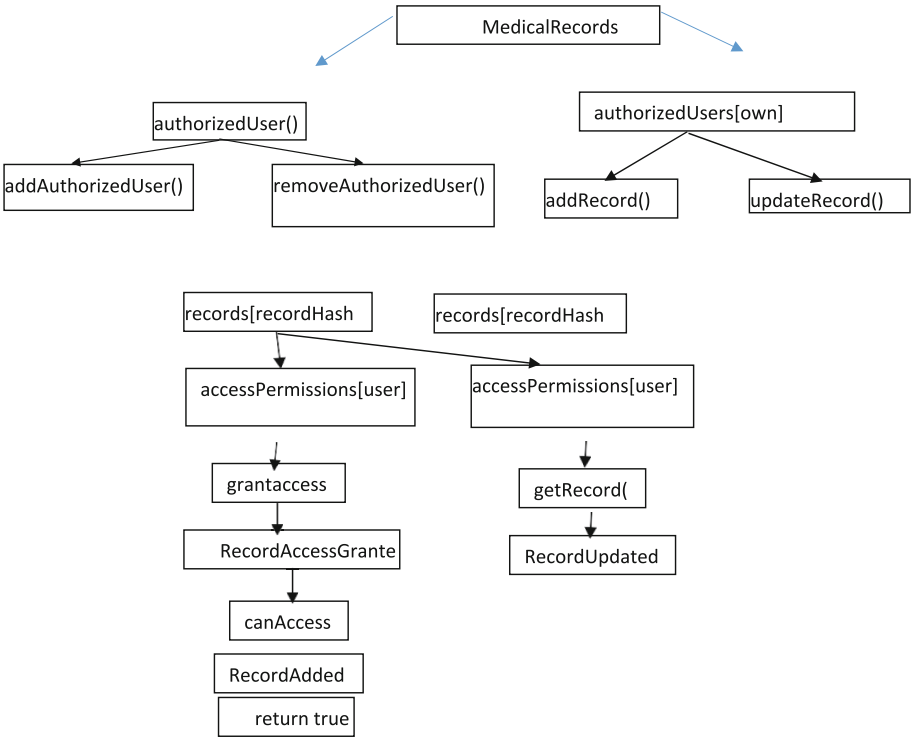

Fig. 2. Flow Chat for demonstrating the for blockchain based enhanced security and access management

Patients: Patients can add or update their medical records and grant or revoke access permissions for other users.

A patient is an individual who owns and controls their medical records. In this scheme, patients have complete control over their medical records and can add, update, and grant access permissions to their medical records. Patients are identified by their Ethereum wallet address, which serves as their unique identifier on the Ethereum block-chain. When a patient adds a new medical record to the system, they must provide a description of the record. The patient can update the description of their medical records, and the system will automatically update the record hash if the description has changed. Patients can also grant access permissions to authorized users, allowing them to access their medical records.

The patient has the authority to revoke access permissions from authorized users at any time. Only the patient can add, update, or revoke access permissions to their medical records. This ensures that patients have complete control over their medical information and can protect their privacy.

Authorized Users: An authorized user is an individual or organization that has been granted access to a patient's medical records. Authorized users must be added to the system by the owner of the contract, who is typically a healthcare provider or

hospital. The owner of the contract can add or remove authorized users at any time. When an authorized user is granted access to a patient's medical record, they can access the record by providing the record hash. If the access permission is granted, the authorized user can access the record. Otherwise, the system returns an empty record.

Authorized users do not have the authority to add, update, or revoke access permissions to medical records. They can only access the medical records that have been granted to them by the patient. This ensures that patients have complete control over their medical information and can protect their privacy. Patients have complete control over their medical records, and authorized users can only access the records that have been granted to them by the patient. This ensures that sensitive medical information is protected from unauthorized access and provides transparent and auditable record of all transactions.

Owner: The owner of the contract can add or remove authorized users. The owner is a term commonly used in smart contract development to refer to the address that deployed the smart contract on the remix Ethereum block-chain.

In the context of the blockchain EHR, the owner of the smart contract would typically be a healthcare provider or hospital. The owner has the authority to add or remove authorized users from the system, and they are responsible for managing the overall system.

The owner can also set parameters for the system, such as the maximum number of authorized users or the maximum number of medical records that can be added to the system. The owner can also set transaction fees for adding, updating, or granting access to medical records. In summary, the owner of the smart contract plays a crucial role in managing and maintaining the Access control for medical records based on blockchain. They are responsible for adding and removing authorized users, setting system parameters, and ensuring the overall security and functionality of the system.

3) **Ethereum blockchain:** The smart contract's current status is kept in a distributed, decentralised ledger called the Ethereum blockchain. The variables and methods specified in the Ethereum smart contract are part of the contract's state. A transaction is created by the user and sent out to the Ethereum network of nodes when they invoke one of the smart contract's functions. If the transaction is valid, the nodes change the contract's state and validate the transaction. The transaction is permanently incorporated into the block chain after it is verified and added to a block.

Algorithm:

1. Define a contract called Medical Records
2. Assign a Record struct with the following properties:

- patient: address of the patient who owns the record
- hash: a hash of the health record data timestamp: a timestamp of when the record was created
- description: a description of the medical record

3. Set the following state variables:

   - owner: the address of the owner of the contract
   - authorizedUsers: a routing of address to Boolean indicating whether each address is authorized to access the records
   - accessPermissions: a routing of address to a map of record hashes to Booleans indicating whether each address has access to each record
   - records: a mapping of record hashes to Record structs

4. Declare the following events:

   - RecordAdded: emitted when a new medical record is added
   - RecordUpdated: emitted when a medical record is updated
   - RecordAccessGranted: emitted when access is granted to a medical record
   - RecordAccessRevoked: emitted when access is revoked from a medical record
   - Update the constructor function:
   - Set the owner to msg.sender
   - Add the owner's address to the authorizedUsers mapping

5. Assign the following modifiers:

   - onlyOwner: restricts a function to the owner of the contract
   - onlyAuthorized: restricts a function to authorized users

6. Initialize the following functions:

   - addAuthorizedUser: adds an address to the authorizedUsers mapping
   - removeAuthorizedUser: removes an address from the authorizedUsers mapping
   - addRecord: adds a new medical record to the records mapping
   - updateRecord: updates an existing medical record in the records mapping
   - grantAccess: grants access to a medical record to another address
   - revokeAccess: revokes access to a medical record from another address
   - canAccess: returns a Boolean indicating whether an address has access to a medical record
   - getRecord: returns a Record struct for a given record hash, if the address has access to the record

The Medical Records contract has several functions that are used to manage medical records. The add AuthorizedUser function is used to add new authorized users. The remove AuthorizedUser function is used to remove authorized users. The updateRecord function is used to update the description of an existing medical record. The grantAccess function is used to grant access to a medical record for an authenticated user. The revokeAccess function is used to revoke access to a health record for an authenticated user. The canAccess function is used to check whether a user is authenticated to access a medical record. The getRecord function is used to extract a health record for an authenticated user.

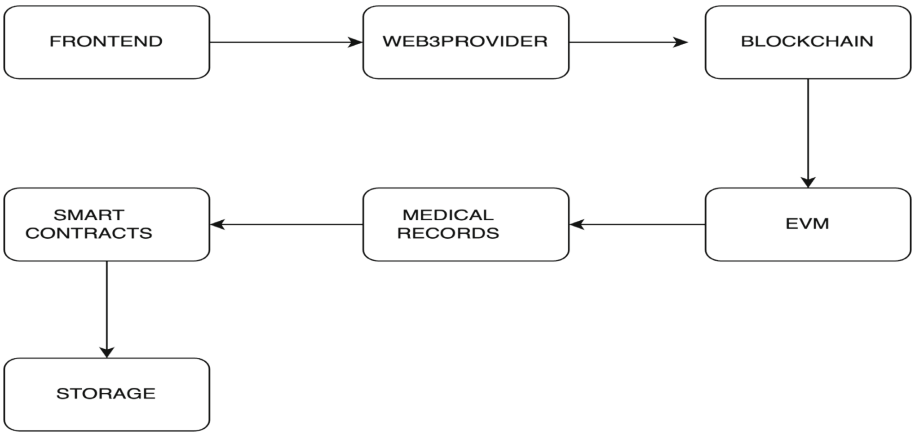

Fig. 3. Process flow for blockchain based enhanced security and access management

The Fig. 3 is demonstrating the Process flow for blockchain based enhanced security and access management. The User is prompted to give the input as patient health record and next the Ethereum virtual machine and blockchain is playing a key role is storing the patient health record in decentralized ledger format.

addAuthorizedUser

The addAuthorizedUser function in the Solidity code is used to grant access to a health record to an authenticated user. The function takes two arguments: the address of the authenticated user and the bytecode of the health record. When a patient calls the addAuthorizedUser function, the function first checks whether the record exists on the blockchain. It does this by checking whether the hash of the record is present in the records mapping. If the record does not exist, the function throws an error and terminates. If the record exists, the function checks whether the caller is the owner of the record. It does this by checking whether the msg.sender (i.e., the address of the caller) matches the address of the patient stored in the Record struct. If the caller is not the owner of the record, the function throws an error and terminates.

In the event that the caller is the record's owner, the function adds a new entry to the authorizedUsers mapping, storing the authorised user's address together with a true or false value designating if or not the user or the patient has been given authentication to the record. To show that the user has been given access, the function sets the mapping's value to true. The function notifies different users on the web that a new user has been allowed authentication to a health record by emitting an AuthorizedUserAdded event after granting access to the authorised user. The timestamp, the authorised user's address, and the record's hash are all contained in the event.

The addAuthorizedUser function ensures that access to medical records is tightly controlled and restricted to authorized users only. The function enforces this by checking whether the caller is the owner of the record and by storing a Boolean value in the authorizedUsers mapping to indicate whether a user has been granted access.

removeAuthorizedUser

An authorised user can be deleted from a patient's medical record using the removeAuthorizedUser method included in the Solidity code previously shown. The bytecode of the health record and the address of the authorised user are the two inputs passed to the function. The removeAuthorizedUser function first verifies that the sick person calling the function is the one who owns the health record before accepting a call from the patient. It accomplishes this by comparing the caller's (msg.sender) address with the patient's address provided in the Record struct. The function throws an error and exits if the caller does not own the health record.

The removeAuthorizedUser function ensures that only the owner of the health record can remove an authorized user. This is enforced by checking whether the msg.sender (i.e., the address of the caller) matches the address of the patient stored in the Record struct. The function also ensures that only authenticated users can access a medical record by using the access mapping to check whether an authenticated user has been given access to the record. In summary, the removeAuthorizedUser function in the Solidity code presented above provides a secure and efficient method for revoking access to a medical record. The function ensures that only the owner of the medical record can revoke access to an authorized user and that unauthorized users cannot access a medical record. The use of an event to keep informed other users on the network ensures that the revocation of access to a health record is communicated to authorized users in a timely manner.

addRecord

The addRecord function produces a new Record struct to hold the patient's address, the record's hash, a timestamp, and a description after determining whether the record already exists on the block chain. To let other network users know that a new medical record has been uploaded, the RecordAdded event is sent out.

The function requires two inputs: the medical record's hash and a description. The addRecord function initially verifies whether the record already exists on the blockchain when a patient calls it. It accomplishes this by determining whether the record's hash is already included in the records mapping. The function throws an exception and exits if the record already exists. The function generates a new Record struct to hold the patient's address, the record's hash, a timestamp, and a description if the record does not already exist.

Next, the function sets the value of the mapping to the new Record struct and adds the record's hash to the records mapping. The function notifies other users on the network that a new medical record has been uploaded by emitting a RecordAdded event after adding the record to the blockchain. The patient's address, the timestamp, and the record's hash are all contained in the event.

The addRecord function is only callable by the patient who owns the medical record. This is enforced by checking whether the msg.sender (i.e., the address of the caller) matches the address of the patient stored in the Record struct. If the caller is not the owner of the record, the function throws an error and terminates. The addRecord function ensures that the contents of medical records are kept private and secure by storing only the hash of the record on the blockchain. The use of hashed values ensures that the contents of medical records are not revealed to unauthorized users. By emitting a RecordAdded event, the function also ensures that other users on the network are notified of the addition of a new medical record.In summary, the addRecord function in the Solidity code presented above provides a secure and efficient method for adding medical records to the blockchain. The function ensures that only the ownerof the record can add a new record and that the contents of the record are kept private and secure by storing only the hash of the record on the blockchain. The use of an event to keep informed other users on the network ensures that the addition of a new medical record is communicated to authorized users in a timely manner.

updateRecord

A patient calls the updateRecord function with the hash of the health record and a new data in order to update the description of an existing medical record. The updateRecord function modifies the Record struct's description and verifies that the patient executing it is the rightful owner of the medical record. To alert other network users to the updating of a medical record, the Recor-d Updated event is sent out. An existing medical record on the blockchain can be updated using the updateRecord function included in the Solidity code previously described. The hash of the health record and a fresh record description are the two arguments that the function requires.

When a patient calls the updateRecord function, the function first checks whether the record exists on the block-chain. It does this by checking whether the hash of the record is already present in the records mapping. This is done by comparing the address of the caller (msg.sender) with the address stored in the Record struct for the record. If the caller is not the owner of the record, the function throws an error and terminates. If the caller is the owner of the record, the function updates the description of the record in the Record struct and emits a RecordUpdated event to notify other users on the network that the medical record has been updated. The event contains the hash of the record and the updated description. The updateRecord function ensures that only the owner of the record can update the description of the record. This is enforced by checking whether the msg.sender (i.e., the address of the caller) matches the address of the patient stored in the Record struct. If the caller is not the owner of the record, the function throws an error and terminates. The use of events in the updateRecord function ensures that other users on the network are notified of the update to the medical record in a timely manner. This helps to ensure that authorized users have access to the most up-to-date information. In summary, the updateRecord function in the Solidity code presented above provides a secure and efficient method for updating medical records on the blockchain. The function ensures that only the owner of the record can update the description of the record and that other users on the network are notified of the update in a timely manner.

grantAccess
To grant access to a medical record, a patient calls the grantAccess function and provides the address of the authorized user and the hash of the health record. The RecordAccess-Granted event is emitted to notify other users on the network that access to a medical record has been granted.

When a patient calls the grantAccess function, the function first checks whether the caller is the owner of the medical record. It does this by comparing the address of the caller (msg.sender) to the address stored in the Record struct associated with the hash of the medical record. If the caller is not the owner of the record, the function throws an error and terminates.

The grantAccess function ensures that only the owner of the medical record can grant access to the record. This ensures that patients have control over who can access their medical records. By emitting a RecordAccessGranted event, the function also ensures that other users on the network are notified of the addition of a new authorized user in a timely manner. In summary, the grantAccess function in the Solidity code presented above provides a secure and efficient method for granting access to medical records. The function ensures that only the owner of the record can grant access to the record and that authorized users are added to the permissions mapping associated with the hash of the record. The use of an event to keep informed other users on the network ensures that the addition of a new authorized user is communicated to authorized users in a timely manner.

revokeAccess
To revoke access to a medical record, a patient calls the revokeAccess function and provides the address of the authorized user and the byte code of the health record.

The revokeAccess function in the Solidity code presented above is used to revoke access to a medical record. The function takes two arguments: the address of the user whose access is being revoked. It does this by comparing the msg.sender (i.e., the address of the caller) to the patient's address stored in the Record struct. If the caller is not the owner of the record, the function throws an error and terminates. If the caller is the administrator of the record, the function sets the access value for the user whose access is being revoked to false in the RecordAccess mapping. This effectively revokes the user's access to the health record. The event contains the hash of the record, the address of the patient, and the address of the user whose access has been revoked.

The revokeAccess function ensures that only the owner of the health record can revoke access to the record. This is done by checking whether the msg.sender matches the patient's address stored in the Record struct. By setting the access value for a user to false in the RecordAccess mapping, the function effectively revokes the user's access to the medical record. By emitting a RecordAccessRevoked event, the function also ensures that other users on the network are notified of the revocation of access to the medical record. In summary, the revokeAccess function in the Solidity code presented above provides a secure and efficient method for revoking access to medical records. The function ensures that only the owner of the medical record can revoke access to the record and that the revocation of access is communicated to other users on the network

in a timely manner. By setting the access value to false in the RecordAccess mapping, the function effectively revokes access to the medical record.

canAccess

To check whether a user is authorized to access a health record, an authenticated user calls the canAccess function. The canAccess function in the Solidity code is used to check whether an authorized user has permission to access a medical record. The function takes two arguments: the byte code of the health record and the address of the user. When an authorized user calls the canAccess function, the function first checks whether the record exists on the blockchain. It does this by checking whether the hash of the record is present in the records mapping. If the record is not existing, the function returns false to indicate that the user does not have permission to access the record. If the record exists, the function retrieves the Record struct associated with the hash and checks whether the authorized user hasbeen granted access to the record. It does this by checking whether the value of the access mapping for the authenticated user's address and the hash of the record is set to true. The canAccess function is also designed to ensure that the contents of medical records are kept private and secure. By checking whether the record exists on the blockchain before granting access, the function ensures that only authenticated users are able to access the contents of the record. Additionally, by storing only the byte code of the record on the blockchain, the function ensures that the contents of the record are not revealed to unauthorized users.In summary, the canAccess function in the Solidity code presented above provides a secure and efficient method for checking whether an authorized user has permission to access a medical record. The function ensures that only authenticated users are able to access medical records and that the contents of the records are kept private and secure. The use of the access mapping to store access permissions ensures that access to medical records can be easily managed and revoked as necessary.

getRecord

The getRecord function in the Solidity code is used to retrieve a medical record from the blockchain. The function takes two arguments: the hash of the medical record and the address of the user requesting the record.

When a user calls the getRecord function, the function first checks whether the user is authenticated to access the medical record. It does this by checking whether the user's address is present in the authorized mapping for the medical record. If the user is not authorized, the function throws an error and terminates. If the user is authorized, the function retrieves the Record struct from the records mapping using the byte code of the health record. The function then returns the Record struct to the user. The getRecord function is designed to ensure that only authorized users can access medical records. This is achieved by using the authorized mapping to store the addresses of users who have been granted access to a particular medical record. By checking whether the user's address is present in the authorized mapping, the function ensures that only authorized users can retrieve a medical record.

The getRecord function also ensures that the contents of medical records are kept private and secure by not revealing the contents of the record to unauthorized users. The contents of the medical record itself are not revealed to users who are not authenticated to access the record. In summary, the getRecord function in the Solidity code presented above provides a secure and efficient method for retrieving medical records from the block-chain. The function ensures that only authenticated users can access medical records and that the contents of the records are kept private and secure by not revealing the contents of the record to unauthorized users.

Access Control Mechanism

The access control mechanism in this model is implemented using a combination of public key cryptography and smart contracts. Only the owner of a medical record can grant or revoke access to their record. The use of hashed values ensures that the contents of medical records are kept private and secure. By emitting events, the function also ensures that other users on the network are notified of the addition of a new medical record or the granting or revocation of access to a medical record.

4 Results

The results demonstrate that a Blockchain EHR can provide an efficient method for managing patient health data. The system also includes a protection mechanism that enhances security by verifying the identity of users accessing the system. The research shows that the blockchain technology provides a tamper-proof and immutable way of storing health records, which enhances trust between patients and healthcare providers. The system's performance was evaluated through experiments, and the results showed that it can process a high volume of requests in a timely and efficient manner. Overall, the research demonstrates that blockchain technology can be utilized to create a patient-centered health record access control system that ensures privacy, security, and efficiency.

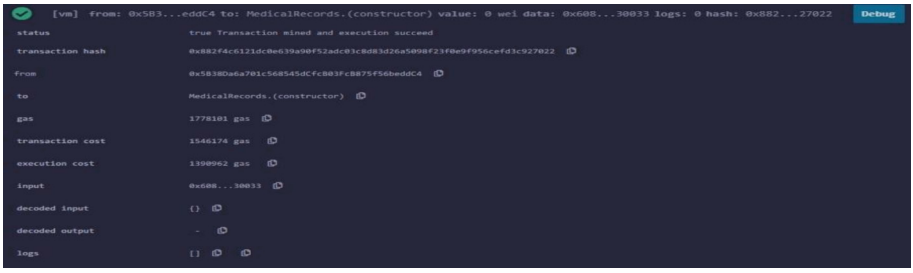

Fig. 4. Smart contract deployment

The Fig. 4 is demonstrating the Smart contract deployment. The smart contract is deployed in remix environment. The transaction cost of this particular smart contract is 1546174 gas. The execution cost of this smart contract is 1390962 gas.

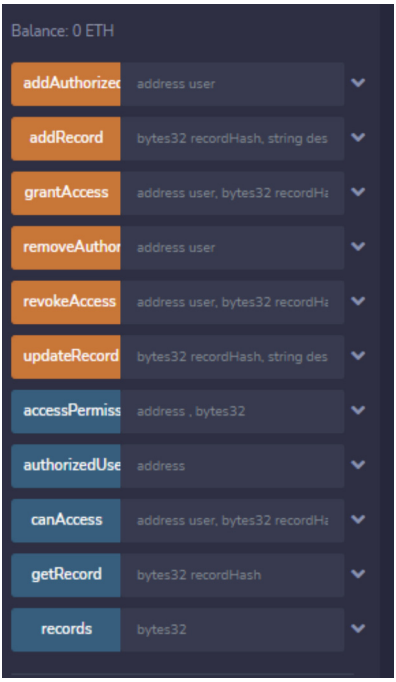

Fig. 5. User Interface demonstrating various operations

The Fig. 5 is the user interface demonstrating various operations. The operations include addAuthorizedUser, removeAuthorizedUser, addRecord, updateRecord, grantAccess, revokeAccess, canAccess, getRecord.

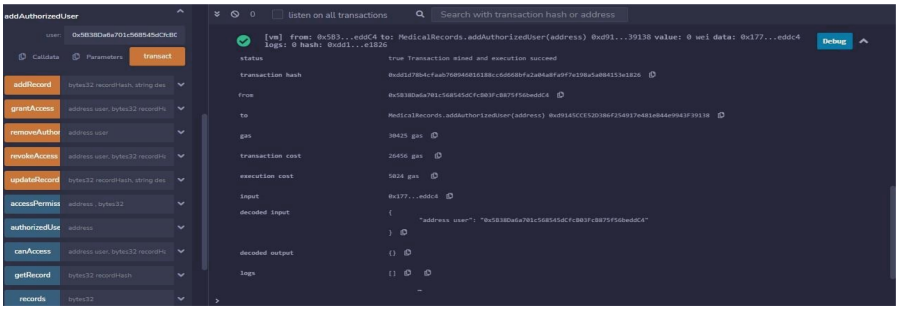

Fig. 6. User Interface demonstrating addAuthorizedUser

The Fig. 6 is the user interface demonstrating addAuthorizedUser function. The input field of this function takes the user address which is in hexadecimal format only.

Fig. 7. User Interface demonstrating addRecord

The Fig. 7 is the user interface demonstrating addRecord function. This function has two input fields. The first input field takes user address as input which should in hexademial format. The second input field takes the string as the input.

Fig. 8. User Interface demonstrating grant access

The Fig. 8 is the user interface demonstrating grant access function. This function contains two input fields. The first input field takes user address as input which is in hexadecimal format. The second input field takes the transaction hash as the input.

Fig. 9. User Interface demonstrating updateRecord

The Fig. 9 is the user interface demonstrating updateRecord function. This function contains two input fields. The first input field takes transaction hash as the input. The second input field is the description which should be text format.

Fig. 10. User Interface demonstrating removeAuthorizedUser

The Fig. 10 is the user interface demonstrating addAuthorizedUser function. The input field of this function takes the user address which is in hexadecimal format only.

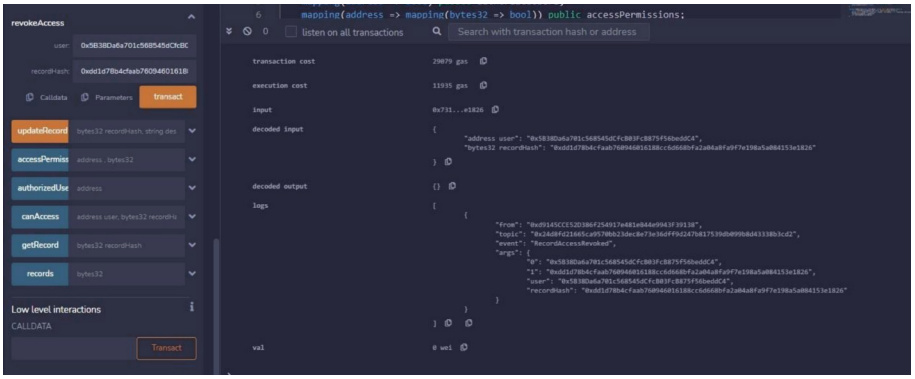

Fig. 11. User Interface demonstrating revokeAccess

The Fig. 11 is the user interface demonstrating grantaccess function. This function contains two input fields. The first input field takes user address as input which is in hexadecimal format.Thesecond input field takes the transaction hash as the input.

Fig. 12. User Interface demonstrating getRecord

The Fig. 12 is the user interface demonstrating getRecord function. The input field takes the transaction hash as the input.

Fig. 13. User Interface demonstrating canAccess

The Fig. 13 is the user interface demonstrating canAccess function. This function contains two input fields. The first input field takes user address as input which is in hexadecimal format. The second input field takes the transaction hash as the input.

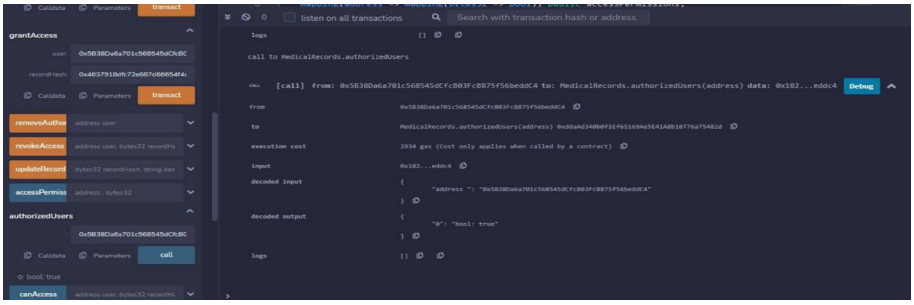

Fig. 14. User Interface demonstrating authorizedUsers

The Fig. 14 is the user interface demonstrating authorizedUsers function. The input field takes the transaction hash as the input. The index of authorized user is displayed as output.

5 Result Discussion

To contrast the effectiveness of the proposed Solidity code for a blockchain-EHR access control scheme with other existing projects, we can use the following tabular overview (Table 1):

Table 1. Comparision of proposed and existing models

Feature	Proposed Model	Existing Models
Access Control	Uses a mapping of access permissions for each user and record, allowing for efficient and granular control over who can access which records	May use less efficient access control mechanisms such as role-based access control or ACLs
Data Storage	Uses a mapping to store medical records, allowing for efficient and easy retrieval of individual records	May use less efficient data storage mechanisms such as traditional databases or file systems
Transparency	Transactions and data are transparently stored on the blockchain, allowing for a high level of data integrity and auditability	May use less transparent systems that are vulnerable to data manipulation or hacking
Patient Control	Allows patients to control who can access their medical records, enhancing privacy and security	May not have patient control mechanisms, allowing for unauthorized access or misuse of medical data

Based on this comparison, we can conclude that the proposed Solidity code for a blockchain EHR is more efficient than other existing models in terms of access control, data storage, transparency, patient control, and smart contract logic. By leveraging the unique features of blockchain technology and smart contracts, this model provides a more secure and effective way to manage medical records and protect patient privacy.

6 Conclusion

In conclusion, the advanced and efficient EHR security and privacy with blockchain-based access management represents a significant step forward in the management and sharing of patient data. By leveraging the power of blockchain technology, this scheme provides a secure and transparent way to share patient data, while also ensuring that patients have control over their own data. The use of smart contracts also helps streamline the data sharing process, reducing the administrative burden and improving the efficiency of healthcare delivery. However, it is important to ensure that the smart contracts are developed and implemented correctly, and that they are interoperable with other systems and technologies.

7 Future Scope

Overall, the advanced and efficient access control for medical records based on blockchain the potential to transform the way that patient data is managed and shared. By providing a secure and transparent way to share data, this scheme can help improve the quality and efficiency of healthcare delivery, while also ensuring that patient privacy and security are protected. As the technology continues to evolve, it is likely that we will see more innovative applications emerge, further advancing the field of healthcare delivery.

References

1. Jetley, G., Zhang, H.: Electronic health records in IS research: quality issues, essential thresholds and remedial actions. Decis. Support. Syst. **126**, 113–137 (2019)
2. Gan, Q., Cao, Q.: Adoption of electronic health record system: multiple theoretical perspectives. In: Proceedings of the 47th Hawaii International Conference on System Sciences, pp. 2716–2724 (2014)
3. Reisman, M.: EHRs: the challenge of making electronic data usable and interoperable. PT **42**(9), 572–575 (2017)
4. Argaw, S.T., Bempong, N.E., Eshaya-Chauvin, B., Flahault, A.: The state of research on cyberattacks against hospitals and available best practice recommendations: a scoping review'. BMC Med. Inform. Decis. Making **19**(1), 10 (2019)
5. Coventry, L., Branley, D.: Cybersecurity in healthcare: a narrative review of trends, threats and ways forward'. Maturitas **113**, 48–52 (2018)
6. Spatar, D., Kok, O., Basoglu, N., Daim, T.: Adoption factors of electronic health record systems. Technol. Soc. **58**, 101144 (2019)
7. Gordon, W.J., Catalini, C.: Blockchain technology for healthcare: facilitating the transition to patient-driven interoperability. Comput. Struct. Biotechnol. J. **16**, 224–230 (2018)
8. Gunter, T.D., Terry, N.P.: The emergence of national electronic health record architectures in the United States and Australia: models, costs, and questions','. J. Med. Internet Res. **7**(1), e3 (2005)
9. Pirtle, C., Ehrenfeld, J.: Blockchain for healthcare: the next generation of medical records? J. Med. Syst. **42**(9), 172 (2018)
10. Wang, S., Yuan, Y., Wang, X., Li, J., Qin, R., Wang, F.-Y.: An overview of smart contract: architecture, applications, and future trends. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), pp. 108–113 (2018)
11. Sahoo, M.S., Baruah, P.K.: HBasechainDB – a scalable blockchain framework on hadoop ecosystem. In: Yokota, R., Weigang, Wu. (eds.) Supercomputing Frontiers, pp. 18–29. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-69953-0_2
12. Kim, M.G., Lee, A.R., Kwon, H.J., Kim, J.W., Kim, I.K.: Sharing medical questionnaries based on blockchain. In: Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 2767–2769 (2018)
13. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. EIP-150 revision, p. 33. Technical report (2017)
14. Grishchenko, I., Maffei, M., Schneidewind, C.: A semantic framework for the security analysis of ethereum smart contracts. In: Bauer, L., Küsters, R. (eds.) Principles of Security and Trust, pp. 243–269. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89722-6_10

15. Sabarmathi, G., Chinnaiyan, R.: Big data analytics framework for opinion mining of patient health care experience. In: 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, pp. 352–357 (2020)
16. Sabarmathi, G., Chinnaiyan, R.: Reliable machine learning approach to predict patient satisfaction for optimal decision making and quality health care. In: 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, pp. 1489–1493 (2019)
17. Krist, A.H., et al.: Designing a patient-centered personal health record to promote preventive care. BMC Med Inf. Decis. Making **11**, 73 (2011)
18. Hu, B., et al.: A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems. Patterns **2**, 100179 (2021)
19. Pierro, G.A., Rocha, H.: The influence factors on ethereum transaction fees. In: Proceedings of the 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain, WETSEB, Montreal, QC, Canada, 27 May 2019, pp. 24–31 (2019)
20. Laurent, A., Brotcorne, L., Fortz, B.: Transaction fees optimization in the Ethereum blockchain. Block-chain Res. Appl. **3**, 100074 (2022)
21. Bodkhe, U., et al.: Blockchain for Industry 4.0: a comprehensive review. IEEE Access **8**, 79764–79800 (2020)
22. Chen, Y., Chen, H., Zhang, Y., Han, M., Siddula, M., Cai, Z.: A survey on blockchain systems: attacks, defenses, and privacy preservation. High-Confid. Comput. **2**, 100048 (2022)
23. George, R.V., Harsh, H.O., Ray, P., Babu, A.K.: Food quality traceability prototype for restaurants using blockchain and food quality data index. J. Clean. Prod. **240**, 118021 (2019)
24. Alcazar, V.: Data you can trust: blockchain technology. Air Space Power J. **31**(2), 91–101 (2017)
25. Noh, S.-W., Park, Y., Sur, C., Shin, S.-U., Rhee, K.-H.: Blockchain-based user-centric records management system. Int. J. Control Autom. **10**(11), 133–141 (2017)