

# **HEALTHCARE DATA PROVENANCE MODELS WITH MERKLE TREES**

Bonafide record of work done by

<b>MADDU HEMALI SAI PRAVALLIKA</b>	<b>(21Z225)</b>
<b>MADHUMITHA D R</b>	<b>(21Z228)</b>
<b>ROSHINI P</b>	<b>(21Z245)</b>
<b>SANTHOSHI R</b>	<b>(21Z251)</b>
<b>SHRUTI S</b>	<b>(21Z256)</b>

## **19Z701 - CRYPTOGRAPHY**

Dissertation submitted in the partial fulfillment of the requirements  
for the degree of

### **BACHELOR OF ENGINEERING COMPUTER SCIENCE AND ENGINEERING**



OCTOBER 2024

**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**

**PSG COLLEGE OF TECHNOLOGY**

**(Autonomous Institution)**

**COIMBATORE – 641 004**

<b>TABLE OF CONTENTS</b>	<b>PAGE</b>
<b>SYNOPSIS</b>	<b>3</b>
<b>1. INTRODUCTION</b>	<b>4</b>
1.1. PROBLEM STATEMENT	5
<b>2. LITERATURE SURVEY</b>	<b>6</b>
<b>3. SYSTEM REQUIREMENTS</b>	<b>9</b>
3.1 HARDWARE REQUIREMENTS	9
3.2 SOFTWARE REQUIREMENTS	9
3.3 FUNCTIONAL REQUIREMENTS	10
3.4 NON- FUNCTIONAL REQUIREMENTS	11
<b>4. IMPLEMENTATION AND SYSTEM DESIGN</b>	<b>12</b>
4.1. SYSTEM DESIGN - ARCHITECTURE	12
4.2. SYSTEM IMPLEMENTATION - WORKFLOW DIAGRAM	13
4.3. SYSTEM SEGMENTS	14
4.3.1 BACKEND(SMART CONTRACT)	
4.3.2 FRONTEND(REACT + WEB3.JS)	
4.3.3 LOCAL BLOCKCHAIN NETWORK(GANACHE)	
4.4. FUTURE ENHANCEMENTS	15
<b>5. RESULTS AND OBSERVATION</b>	<b>16</b>
<b>6. CONCLUSION</b>	<b>20</b>
6.1 SUMMARY	20
6.2 REFERENCES	21
<b>APPENDIX</b>	<b>22</b>

## **SYNOPSIS:**

This project addresses the need for secure tracking of healthcare data provenance by utilizing blockchain technology in conjunction with Merkle trees. Provenance in healthcare refers to the ability to track the origin and history of medical data, ensuring its integrity throughout its lifecycle. The blockchain component serves as a decentralized ledger that securely stores medical data in a tamper-proof manner, preventing unauthorized modifications or deletions. The integration of Merkle trees allows for the efficient generation of cryptographic proofs, ensuring that the integrity of the data can be verified quickly and reliably without the need to check the entire dataset.

The system provides multiple functionalities that are essential for ensuring the security and reliability of healthcare data. First, it tracks every instance of data access and modification, creating an immutable log of who accessed the data, when it was accessed, and what modifications were made. This capability is vital for maintaining transparency and ensuring accountability in handling sensitive medical information. Second, the system verifies the authenticity and integrity of the data at any point in time using Merkle tree proofs, which are cryptographic structures that enable fast and efficient verification of the data's consistency with the blockchain. This ensures that the data remains untampered and trustworthy throughout its lifecycle.

To implement this system, several technologies and tools were used. The project is built on the Ethereum blockchain, utilizing Ganache for local blockchain development. Solidity is employed to write the smart contracts that define the rules and logic for data access and modification. The MerkleTreeJS library is used to handle the generation and verification of Merkle proofs, while Web3.js is employed to facilitate interaction with the Ethereum blockchain. The frontend interface is built using React, providing users with an easy-to-use platform to interact with the system. This combination of tools and technologies ensures that the project is both scalable and efficient in addressing the core problem of healthcare data provenance.

# **CHAPTER 1**

## **INTRODUCTION**

The healthcare industry is highly reliant on data, and ensuring the integrity, authenticity, and traceability of this data is critical for patient safety and regulatory compliance. In traditional healthcare systems, data provenance—understanding the origin, history, and flow of data—is often difficult to track due to centralized storage systems that are vulnerable to tampering, unauthorized access, or inadvertent modifications. These issues can lead to compromised patient care, data breaches, and regulatory violations.

Blockchain technology, with its inherent properties of immutability, transparency, and decentralization, offers a robust solution to these challenges. By combining blockchain with cryptographic structures like Merkle trees, we can create a secure and efficient mechanism for tracking the provenance of healthcare data. Blockchain ensures that once data is recorded, it cannot be altered without the consensus of the network, making the system tamper-proof. Merkle trees, on the other hand, allow for efficient verification of data modifications, making it easier to validate the integrity of the data without needing to review the entire dataset.

This project explores the integration of blockchain and Merkle tree technology to provide a secure system for healthcare data provenance. The system ensures that any interaction with the data, whether it is accessed, modified, or transferred, is recorded on a decentralized ledger, providing a transparent audit trail. By leveraging Solidity for smart contracts, MerkleTreeJS for cryptographic proof generation, Ganache for local Ethereum blockchain deployment, Web3.js for interacting with the blockchain, and React for the front-end interface, this project delivers a comprehensive solution to the issue of data integrity in healthcare.

## **1.1. PROBLEM STATEMENT:**

In healthcare, the integrity, authenticity, and traceability of medical data are of paramount importance. However, traditional healthcare data management systems often fall short of providing transparent, tamper-proof provenance tracking. Centralized databases are vulnerable to unauthorized access, data manipulation, and loss of historical data integrity, which can result in compromised patient care and non-compliance with regulatory standards. There is a pressing need for a decentralized solution that ensures secure, verifiable, and immutable data provenance throughout the lifecycle of medical records.

This project addresses the challenge of tracking and verifying healthcare data provenance by leveraging blockchain technology in combination with cryptographic structures like Merkle trees. The goal is to develop a decentralized system that ensures all interactions with medical data—whether accessing, modifying, or transferring—are securely recorded and easily verifiable. The solution must provide tamper-proof tracking of data access and modifications, while also offering an efficient mechanism for verifying data integrity without compromising performance or scalability.

## CHAPTER 2

### LITERATURE SURVEY

This chapter provides a literature survey from various papers on healthcare data provenance, blockchain technology, and Merkle trees.

S.No	Title of the Paper	Description	Keywords
1	Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis	This paper discusses the Merkle Signature Scheme, highlighting its security based on secure hash functions and one-time digital signatures, making it a potential alternative to conventional schemes despite its larger signature size and storage requirements.	Merkle Signature Scheme, Digital signatures, Cryptographic security
2	Optimal parameter selection for efficient memory integrity verification using Merkle hash trees	This paper analyzes the optimal parameter selection, including block size and tree depth, for Merkle hash trees to efficiently verify memory integrity, deriving a method to minimize the cost of tree updates based on memory size and update intervals.	Computer networks, Peer to peer computing, Data structures, Hardware, Cryptography
3	Fine-grained, secure and efficient data provenance on blockchain systems	This paper introduces LineageChain, a secure and efficient blockchain provenance system that captures and stores fine-grained data lineage during contract execution, supporting efficient querying via smart contracts. It is implemented on Hyperledger and optimized storage to benefit blockchain applications.	Blockchain, Data provenance, LineageChain
4	Merkle-Tree Based Approach for Ensuring Integrity of Electronic Medical Records	This paper proposes a Merkle tree-based approach to ensure the integrity of electronic medical records, designed for private network deployment. The method simplifies blockchain technology by eliminating mining.	Blockchain, Merkle-tree, electronic medical records, MIMIC-III, database, document integrity

5	Healthcare Data Management by Using Blockchain Technology	This chapter explores how blockchain technology can enhance healthcare data sharing by providing decentralized, immutable solutions for automated provenance tracking and credential verification, highlighting its application in international organ transplant processes.	Healthcare data sharing, Blockchain technology, Provenance tracking
6	Securing Healthcare Data by Using Blockchain	This chapter explores how blockchain and smart contracts can secure healthcare data during transfer, addressing the limitations of current cloud-based solutions and IoT applications in healthcare. It discusses blockchain's role in securely managing digital records, including lab results, clinical trials, and reimbursements.	Blockchain, Healthcare data security, Smart contracts
7	An improved convolution Merkle tree-based blockchain electronic medical record secure storage scheme	This paper proposes an improved blockchain electronic medical record (EMR) storage scheme using a convolution Merkle tree structure, enhancing storage efficiency and security by reducing the number of nodes and computational layers compared to traditional binary trees.	Electronic medical record, Blockchain Convolution operation, Improved convolution, Merkle tree
8	A blockchain and smart contract-based data provenance collection and storing in cloud environment	This paper proposes a blockchain and smart contract-based data provenance architecture for secure data storage in cloud environments, utilizing biometrics, cryptographic techniques, and IPFS to ensure data integrity, privacy, and efficient performance.	Data provenance, Blockchain, Cloud security
9	SciLedger: A Blockchain-based Scientific Workflow Provenance and Data Sharing Platform	This paper introduces SciLedger, a blockchain-based platform designed to securely store scientific workflow provenance, ensuring data integrity and promoting open-access sharing.	Scientific Workflow, Provenance, Blockchain

10	Blockchain for healthcare systems: Architecture, security challenges, trends and future directions	This paper provides a comprehensive overview of blockchain technology's applications, challenges, and future research opportunities in healthcare, discussing its architecture, security aspects, and interoperability requirements. It also reviews current blockchain-based healthcare solutions, outlines various security threats, and presents techniques to enhance network privacy and security.	Blockchain, Consensus algorithms, Cryptocurrency, Distributed ledger, Healthcare data, Security & privacy
----	--	---	---



## CHAPTER 3

### SYSTEM REQUIREMENTS

#### 3.1 HARDWARE REQUIREMENTS

For the successful implementation and deployment of the project, the following hardware requirements are necessary:

- **Processor:** A multi-core processor (e.g., Intel i5 or AMD Ryzen 5) to handle blockchain simulations and cryptographic operations efficiently.
- **RAM:** Minimum 8 GB of RAM is required to ensure smooth execution of blockchain nodes, local Ethereum development environments, and handling cryptographic processes.
- **Storage:** At least 100 GB of free storage space to manage blockchain data, development tools, and other project dependencies.
- **Network:** A stable internet connection (minimum 10 Mbps) to interact with the blockchain network, perform transactions, and test the application.
- **Graphics Card:** A dedicated graphics card is not mandatory, but it could be beneficial for front-end development, especially for rendering complex user interfaces.

#### 3.2 SOFTWARE REQUIREMENTS

The software stack comprises tools for blockchain development, cryptography, and front-end interaction. The following software is required:

- **Operating System:** Linux (Ubuntu/Debian), macOS, or Windows 10/11 (64-bit).
- **Blockchain Development Environment:**
  - **Ganache:** A local blockchain simulator for development and testing of Ethereum smart contracts.
  - **Truffle:** A development framework for Ethereum to compile, deploy, and test smart contracts.

- **Programming Languages:**
  - **Solidity:** For writing smart contracts.
  - **JavaScript (ES6+):** For developing the front-end and blockchain interaction using Web3.js.
  - **React:** For creating the user interface.
  - **Node.js:** For running the development server and package management.
- **Libraries and Tools:**
  - **MerkleTreeJS:** For generating and verifying Merkle proofs.
  - **Web3.js:** For interacting with the Ethereum blockchain from the front-end.
- **Version Control:** Git for version tracking and collaboration.
- **IDE/Code Editor:** Visual Studio Code or IntelliJ IDEA with appropriate plugins for Solidity and JavaScript development.

### 3.3 FUNCTIONAL REQUIREMENTS

These requirements describe the essential operations and features that the system must fulfill:

- **Smart Contract Implementation:** The system must include Solidity-based smart contracts that allow users to record and verify interactions with healthcare data, including access and modification history.
- **Merkle Tree Proof Generation:** The system must generate Merkle proofs that validate data integrity in an efficient and scalable manner.
- **Blockchain Interaction:** Users must be able to interact with the blockchain via a user-friendly interface to submit, modify, and verify healthcare data.
- **Provenance Tracking:** The system should log every access and modification to healthcare data, storing it on the blockchain to create a transparent, immutable record.
- **Data Access Control:** The system must ensure only authorized users can access or modify healthcare data, with permissions defined in the smart contracts.
- **Proof Validation:** The system must provide functionality to validate the integrity of medical data through Merkle proofs to ensure its consistency with the blockchain.

### 3.4 NON - FUNCTIONAL REQUIREMENTS

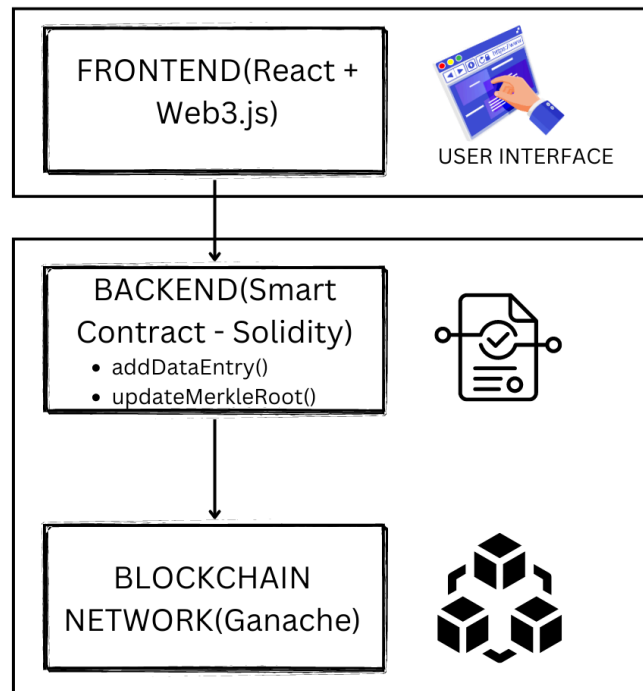
These requirements define the performance standards and qualities that the system should meet:

- **Security:** The system must ensure high levels of data security, with tamper-proof mechanisms enabled by blockchain technology and cryptographic proof verification.
- **Scalability:** The system should be able to scale efficiently, handling an increasing amount of healthcare data without a significant performance drop. This includes both blockchain transactions and Merkle proof validations.
- **Usability:** The front-end interface, built with React, must be intuitive and easy to navigate for healthcare administrators and other stakeholders.
- **Performance:** The system should provide fast interaction with the blockchain network and ensure the efficient generation and verification of Merkle proofs, minimizing latency.
- **Reliability:** The system must ensure consistent and reliable access to healthcare data provenance records, even as the network scales.
- **Maintainability:** The codebase, particularly the smart contracts and JavaScript libraries, must be modular and easy to maintain for future enhancements.
- **Compliance:** The system must be designed with potential future regulatory requirements in mind, especially those concerning healthcare data security and privacy.

# CHAPTER 4

## IMPLEMENTATION AND SYSTEM DESIGN

### 4.1 SYSTEM DESIGN - ARCHITECTURE



The system architecture comprises three main components: the blockchain network, the smart contract (backend), and the frontend interface for user interaction.

#### **Frontend (React + Web3.js):**

- Provides an interface for users to add healthcare data records (hashes) and view the computed Merkle root.
- Communicates with the smart contract deployed on the blockchain using web3.js to send transactions (e.g., adding data, updating the Merkle root).

#### **Smart Contract (Solidity):**

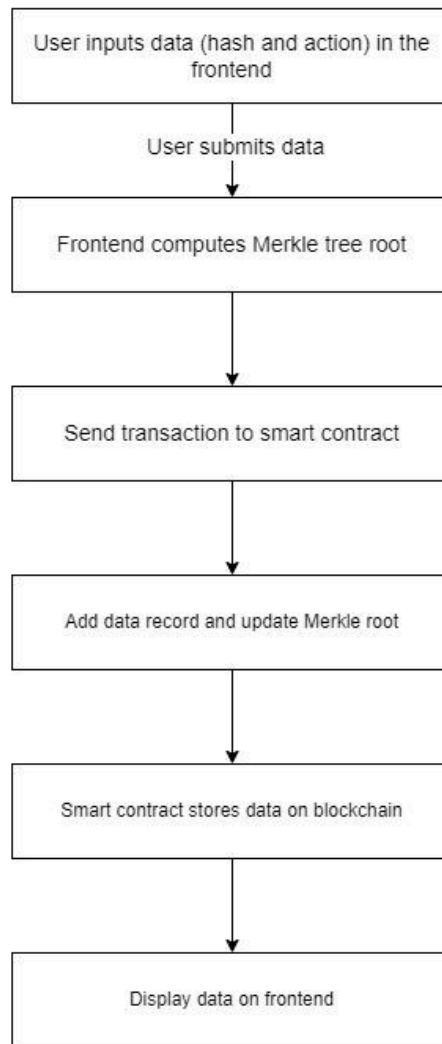
- Acts as the immutable backend for storing healthcare data records.

- Contains functions for adding data records, computing the Merkle root, and retrieving the total number of records.
- Is deployed on a local blockchain instance provided by Ganache.

#### **Blockchain Network (Ganache):**

- Provides a local Ethereum blockchain for testing and development.
- Stores the smart contract and all transaction records (e.g., data additions, Merkle root updates) securely and immutably.

## **4.2 SYSTEM IMPLEMENTATION - WORKFLOW DIAGRAM**



## 4.3 SYSTEM SEGMENTS

The individual components of the project and the functionality of each are explained in this section.

### 4.3.1 Backend (Smart Contract)

- **Technology:** Solidity
- **Purpose:** The smart contract acts as the core backend logic, responsible for storing healthcare data records and computing/verifying data integrity using the Merkle root.
- **Key Functions:**
  - `addDataRecord(string _dataHash, string _action)`: Adds a new data record (including the owner's address, data hash, timestamp, and action) to the contract's state.
  - `updateMerkleRoot(bytes32 _merkleRoot)`: Updates the stored Merkle root in the contract.
  - `getTotalRecords()`: Returns the total number of stored data records.
  - **Data Persistence:** The records and Merkle root are stored on the blockchain, ensuring they are immutable and tamper-proof.

### 4.3.2 Frontend (React + Web3.js)

- **Technology:** React, Web3.js, merkle-treejs library
- **Purpose:** Provides the user interface for interacting with the smart contract and displaying data. This is where the user adds new healthcare records and views the computed Merkle root.
- **Components:**
  - **User Input Forms:** Collects the data hash and action from the user to add a new record.
  - **Merkle Tree Computation:** Utilizes the merkle-treejs library and keccak256 hashing to compute a Merkle Tree using the healthcare data records.
  - **Web3 Integration:** Uses web3.js to interact with the Ethereum blockchain, sending transactions to the smart contract and fetching contract data (e.g., total records, Merkle root).

### 4.3.3 Local Blockchain Network (Ganache)

- **Technology:** Ganache (local blockchain)
- **Purpose:** Provides a simulated Ethereum blockchain environment for deploying and testing the smart contract. It allows for fast, cost-free development.
- **Functionality:**
  - Simulates the Ethereum blockchain, providing a set of accounts and private keys for testing transactions.
  - Stores the smart contract and tracks all interactions with it, including data additions and state changes (e.g., Merkle root updates).

## 4.4 FUTURE ENHANCEMENTS

To further strengthen the system, several future enhancements can be implemented. Firstly, data encryption and privacy can be ensured by integrating encryption algorithms like AES or RSA in the frontend, allowing sensitive healthcare data to be encrypted before it is added to the blockchain. This approach prevents the storage of plaintext data on the blockchain, enhancing security and confidentiality.

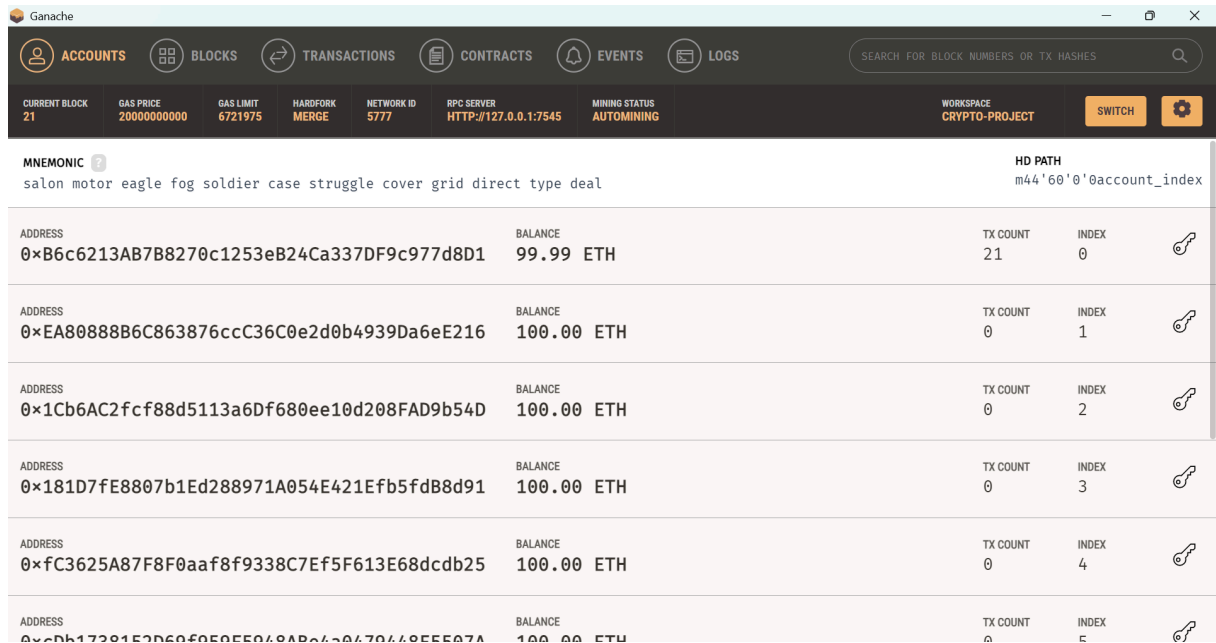
Secondly, implementing a role-based access control mechanism in the smart contract can ensure that only authorized accounts, such as healthcare providers, can add or modify data records, thus securing interactions with the contract. Additionally, for efficient storage, off-chain solutions like IPFS or Filecoin can be used to store large healthcare data while keeping only their hashes on the blockchain, ensuring data integrity without overwhelming the blockchain's capacity. Another important enhancement involves integrating the system with a public blockchain, such as the Ropsten test network, after thorough testing on the local Ganache network. This step would validate the system's scalability and real-world functionality in a decentralized environment.

Finally, implementing Merkle proof verification in the frontend would allow users to verify the inclusion of specific data within the Merkle tree, providing an extra layer of transparency and integrity verification within the system. Together, these enhancements will significantly improve the system's security, efficiency, and robustness.

# CHAPTER 5

## RESULTS AND OBSERVATION

### (i) Ganache Setup:



The screenshot shows the Ganache application window. At the top, there's a navigation bar with icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this is a status bar displaying various network parameters like CURRENT BLOCK (21), GAS PRICE (2000000000), GAS LIMIT (6721975), HARDFORK (MERGE), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). The main area shows the MNEMONIC (salon motor eagle fog soldier case struggle cover grid direct type deal) and the HD PATH (m44'60'0'0account\_index). Below this is a table of accounts.

ADDRESS	BALANCE	TX COUNT	INDEX
0xB6c6213AB7B8270c1253eB24Ca337DF9c977d8D1	99.99 ETH	21	0
0xEA80888B6C863876ccC36C0e2d0b4939Da6eE216	100.00 ETH	0	1
0x1Cb6AC2fcf88d5113a6Df680ee10d208FAD9b54D	100.00 ETH	0	2
0x181D7fE8807b1Ed288971A054E421Efb5fdB8d91	100.00 ETH	0	3
0xfC3625A87F8F0aaf8f9338C7Ef5F613E68dcdb25	100.00 ETH	0	4
0x6Dh17281E2D60f05055049ABc43047044855507A	100.00 ETH	0	5

### (ii) Smart Contract Deployment

```
D:\healthcare-provenance>truffle migrate --reset

Compiling your contracts...
=====
> Compiling .\contracts\HealthcareData.sol
> Artifacts written to D:\healthcare-provenance\build\contracts
> Compiled successfully using:
  - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang

Starting migrations...
=====
> Network name: 'development'
> Network id: 5777
> Block gas limit: 6721975 (0x6691b7)

2_deploy_healthcare_data.js
=====

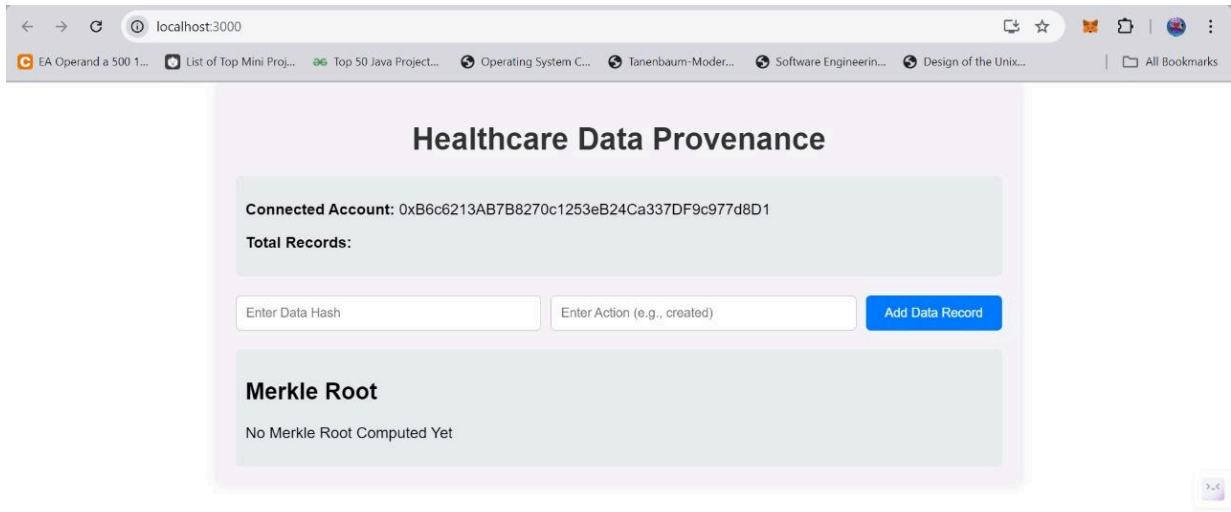
Deploying 'HealthcareData'
=====
> transaction hash: 0x91ade024b6ec1b7c23b44170a2cd47ef4d30e3be82efe5b20e68923ccf792c0f
> Blocks: 0 Seconds: 0
> contract address: 0x640F3fd981B10825b86F3A7338E8e16bA4Ac9EdC
> block number: 3
> block timestamp: 1727894214
> account: 0xB6c6213AB7B8270c1253eB24Ca337DF9c977d8D1
> balance: 99.995032726557821987
> gas used: 712477 (0xadf1d)
> gas price: 3.192852551 gwei
> value sent: 0 ETH
> total cost: 0.002274834006978827 ETH

> Saving artifacts
=====
> Total cost: 0.002274834006978827 ETH

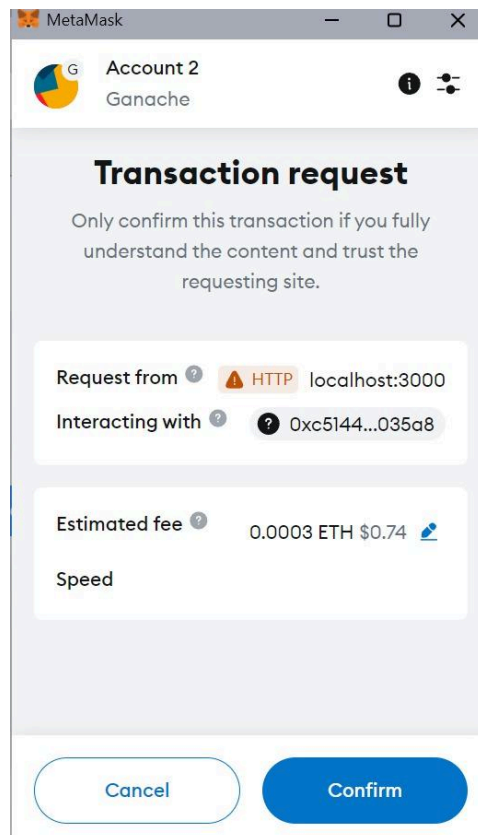
Summary
=====
> Total deployments: 1
> Final cost: 0.002274834006978827 ETH
```



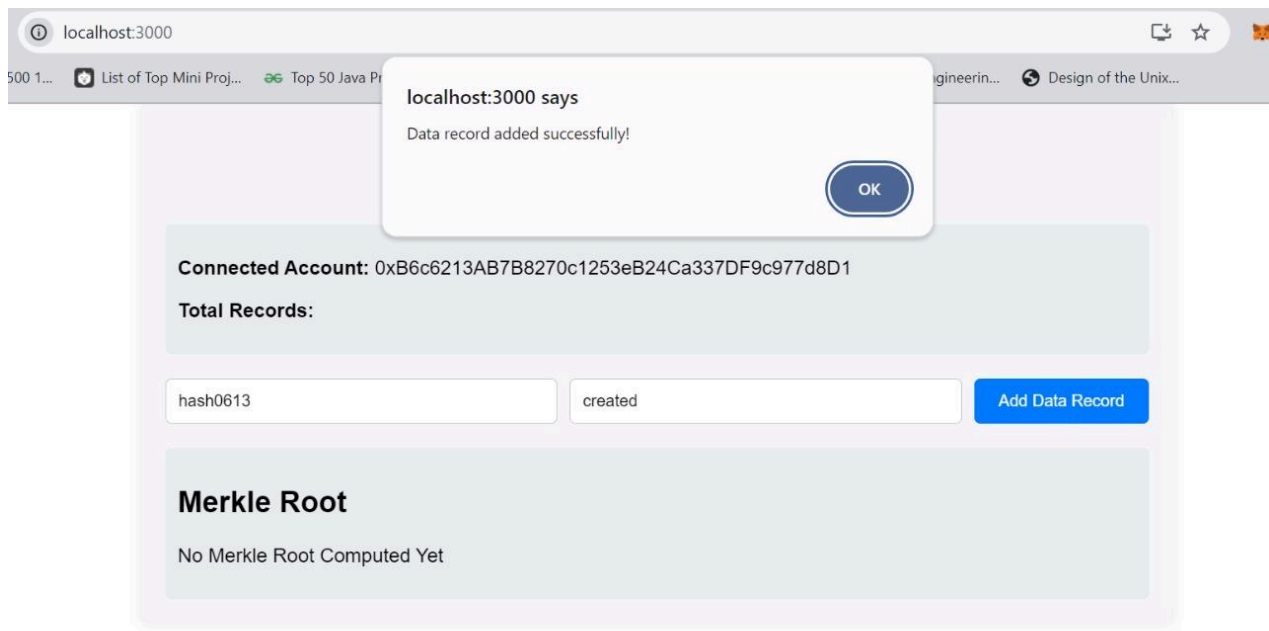
### (iii) User Interface(Frontend) - Initial State



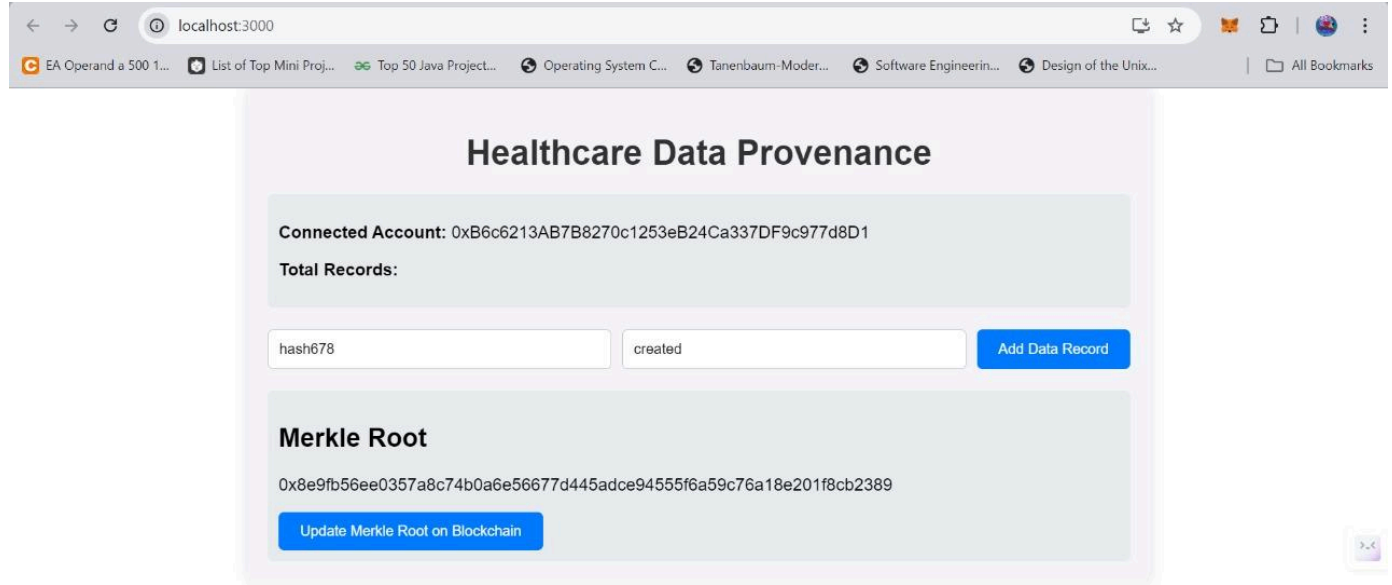
### (iv) MetaMask Integration



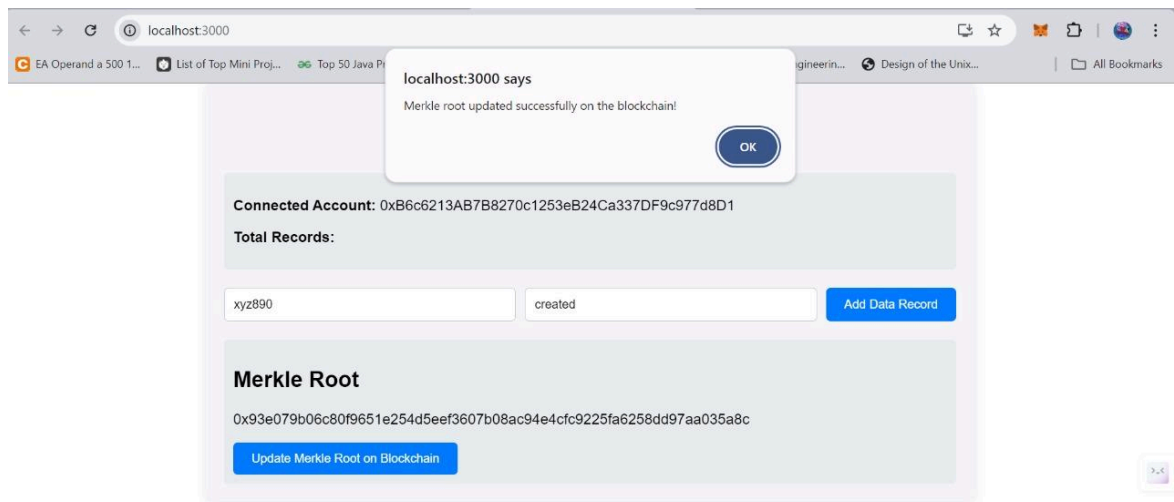
## (v) Adding a new Data Record



## (vi) Merkle Tree computation and Display



(vii) Updating the Merkle Root on Blockchain:



(viii) Smart Contract Interaction Logs in Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK

21

GAS PRICE

20000000000

GAS LIMIT

6721975

HARDFORK

MERGE

NETWORK ID

5777

RPC SERVER

HTTP://127.0.0.1:7545

MINING STATUS

AUTOMINING

WORKSPACE

CRYPTO-PROJECT

SWITCH

TX HASH

0xfa0f38f6cb2215b8c58c62b5357a46da5fca9b8fe1c74b1ecb5e9af41c0597fd

CONTRACT

CALL

FROM ADDRESS

0xB6c6213AB7B8270c1253eB24Ca337DF9c977d8D1

TO CONTRACT ADDRESS

HealthcareData

GAS USED

44140

VALUE

0

TX HASH

0x35711c98605b37948f752f183841539115623c9ef9bf72553300e67e54c1547c

CONTRACT

CALL

FROM ADDRESS

0xB6c6213AB7B8270c1253eB24Ca337DF9c977d8D1

TO CONTRACT ADDRESS

0xc51447F2477F31470bAA75E8Af084671299035a8

GAS USED

122058

VALUE

0

TX HASH

0x4250231007f8fdecdd2de97cc3fab89301bdc9197c846e2ed49ca8649363ece

CONTRACT

CREATION

FROM ADDRESS

0xB6c6213AB7B8270c1253eB24Ca337DF9c977d8D1

CREATED CONTRACT ADDRESS

0x66816F10C8B9a09013ec4C70B04d0CEFD0403b50

GAS USED

712477

VALUE

0

TX HASH

0xd9fdd9283f65a8dd0f52bf7bbc67d6a468c1ae76e0ae3e88e6cad05a149cea4

CONTRACT

CALL

FROM ADDRESS

0xB6c6213AB7B8270c1253eB24Ca337DF9c977d8D1

TO CONTRACT ADDRESS

0xc51447F2477F31470bAA75E8Af084671299035a8

GAS USED

122082

VALUE

0

## **CHAPTER 6**

### **CONCLUSION**

#### **6.1 SUMMARY**

This project successfully implements a blockchain-based system for healthcare data provenance using smart contracts and Merkle trees to ensure data integrity. The integration of a React frontend with a Solidity smart contract on a local blockchain (Ganache) enables secure data record-keeping and verification through cryptographic proofs. The system demonstrates a reliable way to manage sensitive healthcare data with a tamper-proof method. While the current implementation meets the project's goals, future enhancements such as data encryption and access control can further improve its security and scalability.

## 6.2 REFERENCES

- [1] Becker, Georg. "Merkle signature schemes, merkle trees and their cryptanalysis." Ruhr-University Bochum, Tech. Rep 12 (2008): 19.
- [2] D. Williams and Emin Gun Sirer, "Optimal parameter selection for efficient memory integrity verification using Merkle hash trees," Third IEEE International Symposium on Network Computing and Applications, 2004. (NCA 2004). Proceedings., Cambridge, MA, USA, 2004, pp. 383-388, doi: 10.1109/NCA.2004.1347805.
- [3] Pingcheng Ruan, Gang Chen, Tien Tuan Anh Dinh, Qian Lin, Beng Chin Ooi, and Meihui Zhang. 2019. Fine-grained, secure and efficient data provenance on blockchain systems. Proc. VLDB Endow. 12, 9 (May 2019), 975–988. <https://doi.org/10.14778/3329772.3329775>.
- [4] B. Sharma, C. N. Sekharan and F. Zuo, "Merkle-Tree Based Approach for Ensuring Integrity of Electronic Medical Records," 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2018, pp. 983-987, doi: 10.1109/UEMCON.2018.8796607.
- [5] Bittins, S., Kober, G., Margheri, A., Masi, M., Miladi, A., Sassone, V. (2021). Healthcare Data Management by Using Blockchain Technology. In: Namasudra, S., Deka, G.C. (eds) Applications of Blockchain in Healthcare. Studies in Big Data, vol 83. Springer, Singapore. [https://doi.org/10.1007/978-981-15-9547-9\\_1](https://doi.org/10.1007/978-981-15-9547-9_1).
- [6] Gupta, M., Jain, R., Kumari, M., Narula, G. (2021). Securing Healthcare Data by Using Blockchain. In: Namasudra, S., Deka, G.C. (eds) Applications of Blockchain in Healthcare. Studies in Big Data, vol 83. Springer, Singapore. [https://doi.org/10.1007/978-981-15-9547-9\\_4](https://doi.org/10.1007/978-981-15-9547-9_4).
- [7] Hegui Zhu, Yujia Guo, Libo Zhang, An improved convolution Merkle tree-based blockchain electronic medical record secure storage scheme, Journal of Information Security and Applications, Volume 61, 2021, 102952, ISSN 2214-2126, <https://doi.org/10.1016/j.jisa.2021.102952>.
- [8] Jyoti, A., Chauhan, R.K. A blockchain and smart contract-based data provenance collection and storing in cloud environment. Wireless Netw 28, 1541–1562 (2022). <https://doi.org/10.1007/s11276-022-02924-y>.
- [9] Hoopes, Reagan, et al. "Sciledger: A blockchain-based scientific workflow provenance and data sharing platform." 2022 IEEE 8th International Conference on Collaboration and Internet Computing (CIC). IEEE, 2022.
- [10] Andrew J, Deva Priya Isravel, K. Martin Sagayam, Bharat Bhushan, Yuichi Sei, Jennifer Eunice, Blockchain for healthcare systems: Architecture, security challenges, trends and future directions, Journal of Network and Computer Applications, Volume 215, 2023, 103633, ISSN 1084-8045, <https://doi.org/10.1016/j.jnca.2023.103633>.

## APPENDIX

### Smart Contract - Solidity code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HealthcareData {
    // Struct to store healthcare data provenance
    struct DataRecord {
        address owner;
        string dataHash; // A hash representing the data for integrity
        uint256 timestamp;
        string action; // Description of action (e.g., "created", "modified", "accessed")
    }

    // Array to store data records
    DataRecord[] public dataRecords;

    // Merkle root storage
    bytes32 public merkleRoot;

    // Event to track provenance actions
    event DataProvenance(
        address indexed owner,
        string dataHash,
        uint256 timestamp,
        string action
    );

    // Function to add a new data record
    function addDataRecord(string memory _dataHash, string memory _action) public {
        DataRecord memory newRecord = DataRecord({
```

```

        owner: msg.sender,
        dataHash: _dataHash,
        timestamp: block.timestamp,
        action: _action
    });
    dataRecords.push(newRecord);
    emit DataProvenance(msg.sender, _dataHash, block.timestamp, _action);
}

// Function to update the Merkle root
function updateMerkleRoot(bytes32 _merkleRoot) public {
    merkleRoot = _merkleRoot;
}

// Function to get the total number of records
function getTotalRecords() public view returns (uint256) {
    return dataRecords.length;
}

// Function to retrieve a specific data record
function getDataRecord(uint256 _index) public view returns (address, string memory, uint256, string
memory) {
    require(_index < dataRecords.length, "Invalid index");
    DataRecord memory record = dataRecords[_index];
    return (record.owner, record.dataHash, record.timestamp, record.action);
}
}

```