Blockchain Scalability Analysis With Encryption

Abhinav D (21Z203)
Gali Likith Sai (21Z216)
Hareesh S (21Z218)
Navaneetha Krishnan KS (21Z233)
S Karun Vikhash (21Z247)
Roohith M R (22Z432)

19Z701 - CRYPTOGRAPHY

report submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING
Branch: COMPUTER SCIENCE AND ENGINEERING
Of Anna University

September 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

Table of Contents

INTRODUCTION

In this report, we conduct a detailed scalability analysis of a blockchain-based communication system designed for secure data exchange between two hospitals. Each hospital maintains its own blockchain, which stores information related to doctors, patients, and appointments. The system ensures that appointment details are added to the blockchain only after receiving consent from both doctor and patient, with RSA encryption providing secure communication between hospitals. Data is encrypted on the sender side and decrypted on the receiver side before being stored in the blockchain. This setup guarantees privacy and security, but as the number of users and transactions increases, the scalability of the system becomes a crucial consideration.

The objective of this analysis is to assess how the system's performance is impacted as it scales, taking into account factors such as the growing size of the blockchain, increased transaction volume, and the computational overhead introduced by RSA encryption. We explore how the system handles increasing data load, focusing on areas like transaction throughput, network latency, and storage requirements. Additionally, we identify potential bottlenecks in the current setup and propose optimizations that could improve the scalability and efficiency of the system, ensuring it remains capable of supporting larger datasets and higher transaction volumes in a healthcare environment.

PROBLEM STATEMENT

The healthcare industry is increasingly adopting blockchain technology to manage sensitive data in a secure and transparent way. However, consortium blockchains, which are designed specifically for healthcare, encounter significant challenges when it comes to scaling up as the number of participants and transaction volumes grow. This project aims to explore the scalability of Ethereum-based consortium blockchains within the healthcare sector. It will assess how varying participant numbers affect key performance metrics such as transaction throughput (TPS), latency, resource utilization, and network bandwidth.

In addition to scalability, the project examines how integrating Rivest-Shamir-Adleman (RSA) encryption impacts blockchain performance. RSA is used to ensure that sensitive healthcare data is securely transmitted, but the goal is to understand if this added security measure affects the overall efficiency of the blockchain. By running simulations and conducting performance evaluations, the project seeks to uncover the optimal configurations that can enable healthcare blockchains to scale effectively, while maintaining high levels of security and performance. This research could help establish a more practical approach to building secure and efficient blockchain networks for the healthcare industries.

SCOPE OF THE PROJECT

The scope of this project includes:

- Analyzing scalability metrics such as transaction throughput (TPS), latency, resource utilization, and network bandwidth.
- Evaluating the impact of increasing participant numbers on the overall performance of the blockchain.
- Integrating and testing Rivest-Shamir-Adleman (RSA) encryption to ensure secure transmission of healthcare data.
- Applying these findings to real-world healthcare scenarios, with a focus on secure and efficient management of patient data within blockchain networks.

SOFTWARE REQUIREMENTS

Functional Requirements

1. Patient Record Management: The system must support the creation, update, and secure storage of patient records using blockchain technology, ensuring that sensitive information is protected.
2. Secure Data Sharing Between Hospitals: Implement encrypted data sharing between hospital nodes to enable secure and efficient exchange of patient information.
3. Transaction Processing: Handle transactions of varying sizes (1-2 KB, 5-10 KB, and 20-50 KB) at different transaction rates (10 TPS, 50 TPS, 100 TPS) to accommodate diverse healthcare needs.
4. RSA Encryption: All patient data must be encrypted using RSA before being stored or transmitted on the blockchain, ensuring privacy and security.
5. Decryption for Authorized Nodes: Provide authorized nodes with the ability to decrypt patient data when necessary, ensuring access control and compliance with data regulations.
6. Smart Contracts: Develop smart contracts in Solidity to manage healthcare transactions, enforce business rules, and control encryption processes.
7. Blockchain Monitoring: Continuously monitor performance metrics such as transaction throughput, latency, resource utilization, and network bandwidth under various load conditions.

Non-Functional Requirements

1. Scalability: The system must be able to grow and handle an increasing number of participants and transactions without significant performance degradation.
2. Performance: Target high throughput, with at least 50 TPS under medium load and up to 100 TPS under high load, while maintaining low latency for transaction confirmations (1-2 seconds).
3. Security: Ensure all patient data is encrypted using RSA before it is stored or transmitted on the blockchain, prioritizing data privacy.

4. Resource Efficiency: Optimize the system to make efficient use of CPU, memory, and storage resources across blockchain nodes.
5. Network Efficiency: Ensure that network bandwidth is managed efficiently, avoiding bottlenecks that could limit transaction speed or increase latency.

Technical Requirements

1. Ethereum and Ganache: Use the Ethereum blockchain for the project, with Ganache for local development and testing.
2. Truffle Framework: Use the Truffle framework to develop, test, and deploy smart contracts on the Ethereum network.
3. Node.js and Web3.js: Use JavaScript, with Web3.js, to interact with the Ethereum blockchain and simulate healthcare-related transactions.
4. Encryption Libraries: Implement RSA encryption with OpenSSL and ensure data integrity using Python's rsa library.
5. System Monitoring: Monitor system performance and network bandwidth using tools such as Prometheus, Grafana, Wireshark, or Netdata to ensure the system operates within expected parameters.

Performance Constraints

1. Transaction Throughput (TPS): The blockchain must handle at least 50 TPS under medium load and scale to 100 TPS under high load.
2. Latency: Transactions should be confirmed in 1-2 seconds on average.
3. Resource Utilization: CPU usage should remain under 70% and memory usage under 75%, while ensuring efficient storage management.
4. Network Bandwidth: Manage bandwidth to ensure it does not become a limiting factor for transaction throughput or latency.

IMPLEMENTATION

Pre-Requisites

- Node.js (v14.x or later)
- Git

Steps to Set Up and Install Truffle and Ganache

1. *Install Truffle:* Truffle is a development framework for Ethereum. Install it globally using npm:

```
npm install -g truffle
```

2. *Install Ganache:* Ganache is a personal blockchain for Ethereum development.

```
https://archive.trufflesuite.com/ganache/
```

3. *Create a Project Directory:* Create a new directory for your project and navigate into it

```
mkdir temp
cd temp
```

4. *Initialize Truffle Project:* Initialize a new Truffle project

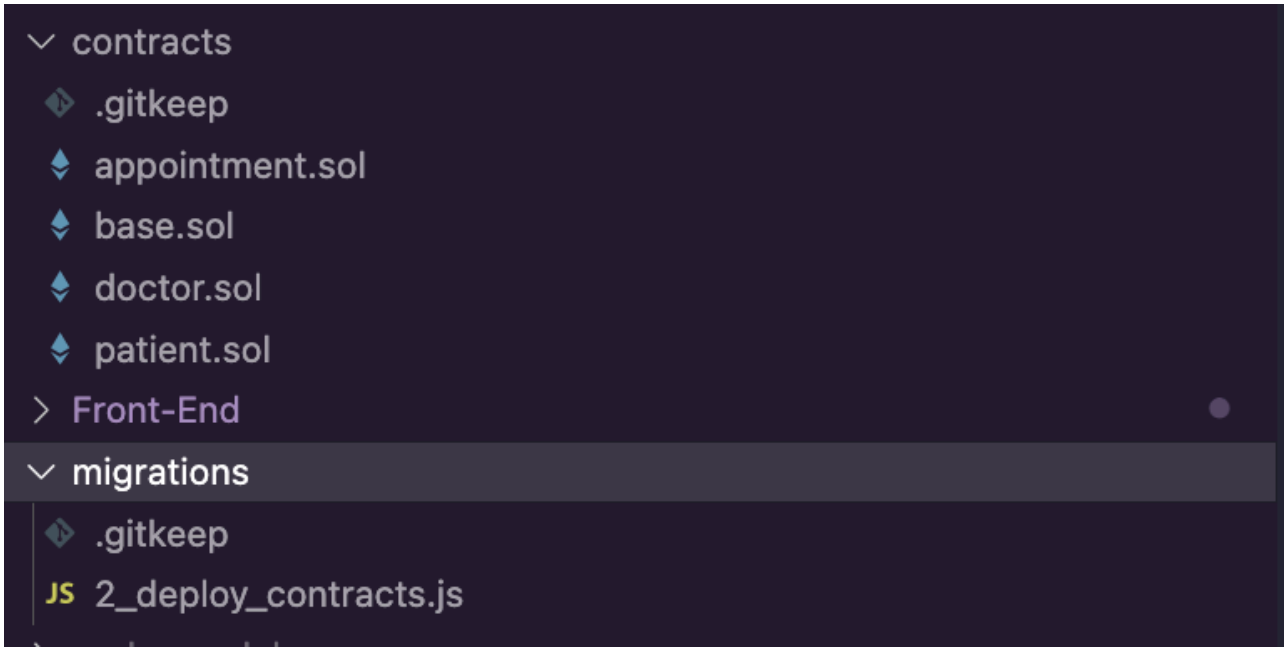
```
npm install @truffle/hdwallet-provider // if not installed
truffle init
```

This command sets up the basic project structure.

```
hareeshsenthil@Hareeshs-MacBook-Air temp % truffle init

Starting init...
================

> Copying project files to /Users/hareeshsenthil/Desktop/temp

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs
```

5. The project directory will look as mentioned below:

6. *Create Contracts and Migrations*
    a. Contracts: Place your Solidity contracts in the contracts/ directory.
    b. Migrations: Create deployment scripts in the migrations/ directory.



7. *Compile Contracts:* Compile your smart contracts

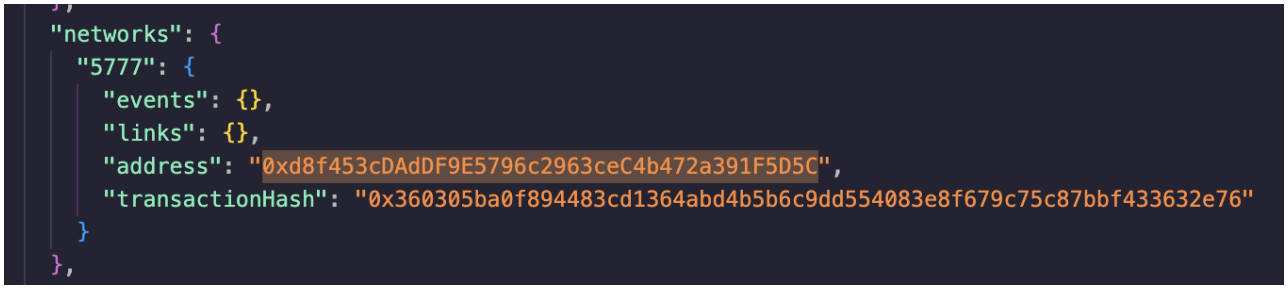
```
truffle compile
```

8. Follow the below *list of commands for additional setup*

```
// for migrating without hyperledeger 'comment 'fabric-evm'
in truffle-config.js':
truffle migrate --reset --network development
// for migrating with hyperledeger:
truffle migrate --network besu
```

9. *Update app.js with Contract Addresses:*

   After compiling, search for "address": in the compiled contract JSON files to get the hex addresses.



   Paste all the contract addresses into your app.js file to interact with them.

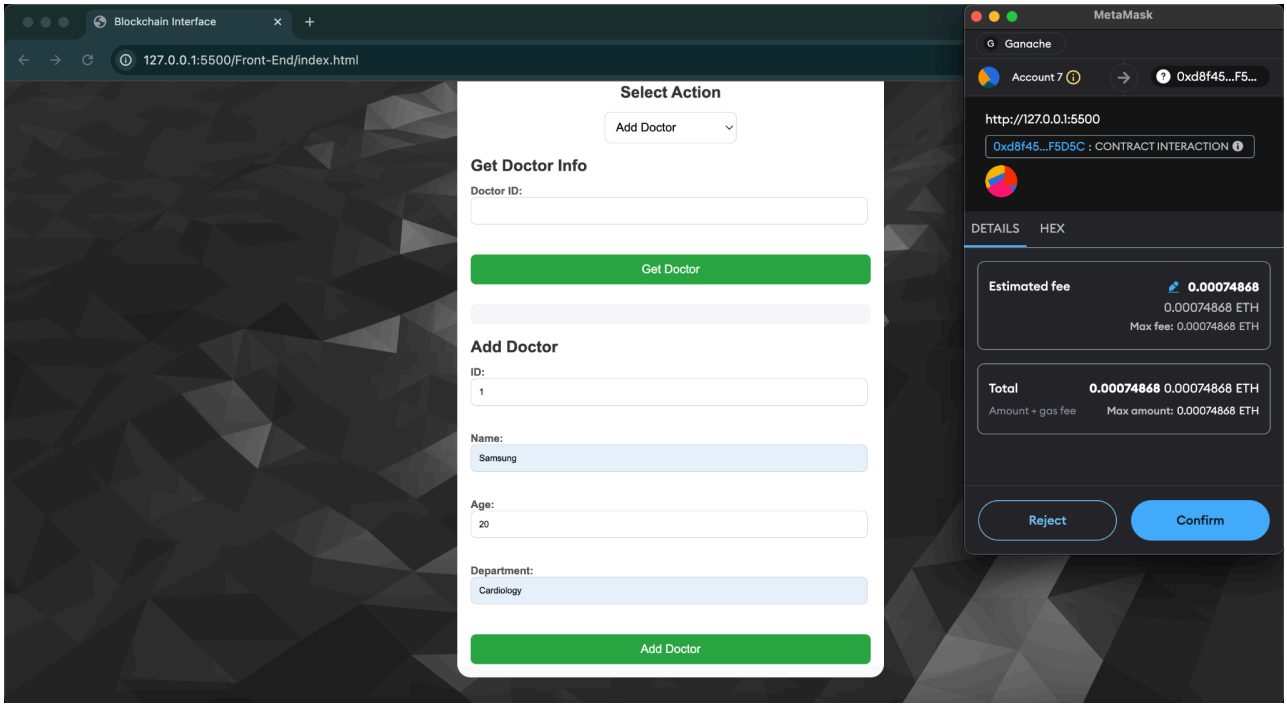
10. *Change the Mnemonic in truffle-config.js*

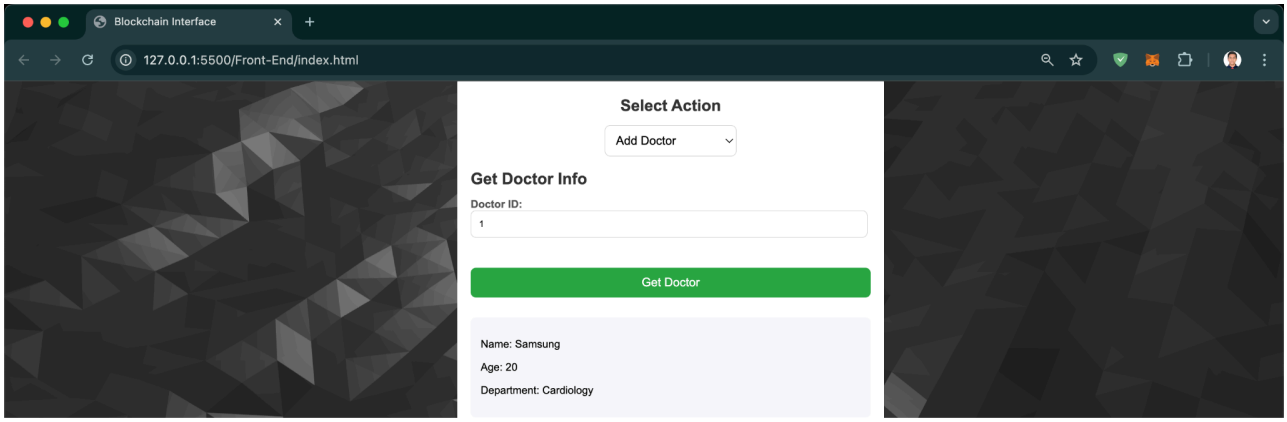
    Edit **truffle-config.js** and replace the mnemonic with your own for secure deployment.

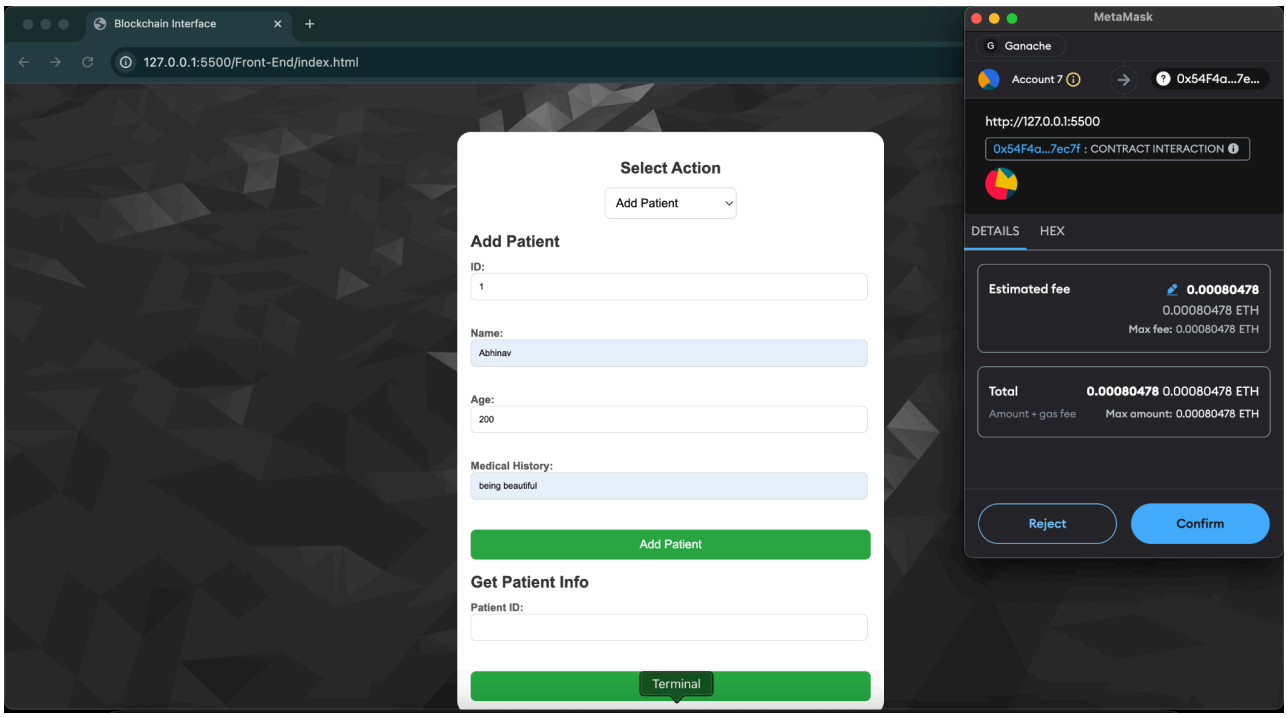
```
const HDWalletProvider =
require('@truffle/hdwallet-provider');
const mnemonic = 'your mnemonic phrase here'; // Change this
to your mnemonic

module.exports = {
  // ... rest of the configuration
};
```

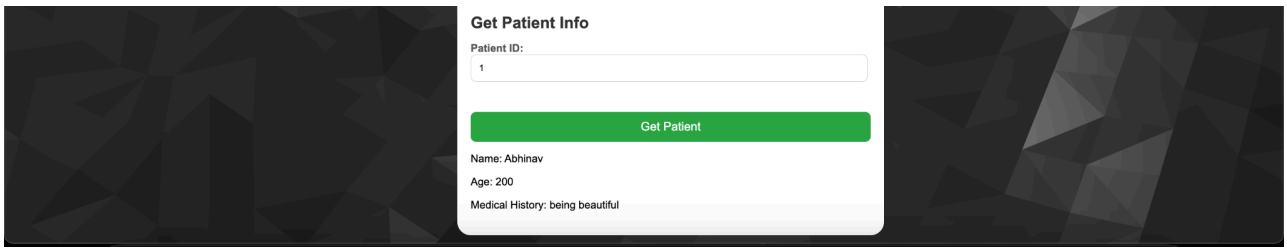
```
// require('dotenv').config();
const HDWalletProvider = require('@truffle/hdwallet-provider');

module.exports = {
    networks: {
        development: {
            host: "127.0.0.1",     // Localhost (default: none)
            port: 7545,            // Standard Ethereum port (default: none)
            network_id: "5777",    // Any network (default: none)
            type: "fabric-evm",
        },
        besu: {
            provider: () => new HDWalletProvider("
            r", "http://127.0.0.1:7545"),
            network_id: "5777",    // Match any network id
            gas: 4500000,          // Gas limit for Besu
            gasPrice: 1000000000,  // Gas price in wei
            timeoutBlocks: 50,     // Wait 50 blocks before considering transaction as failed
            skipDryRun: true       // Skip dry run before migrations
        }
    },
```

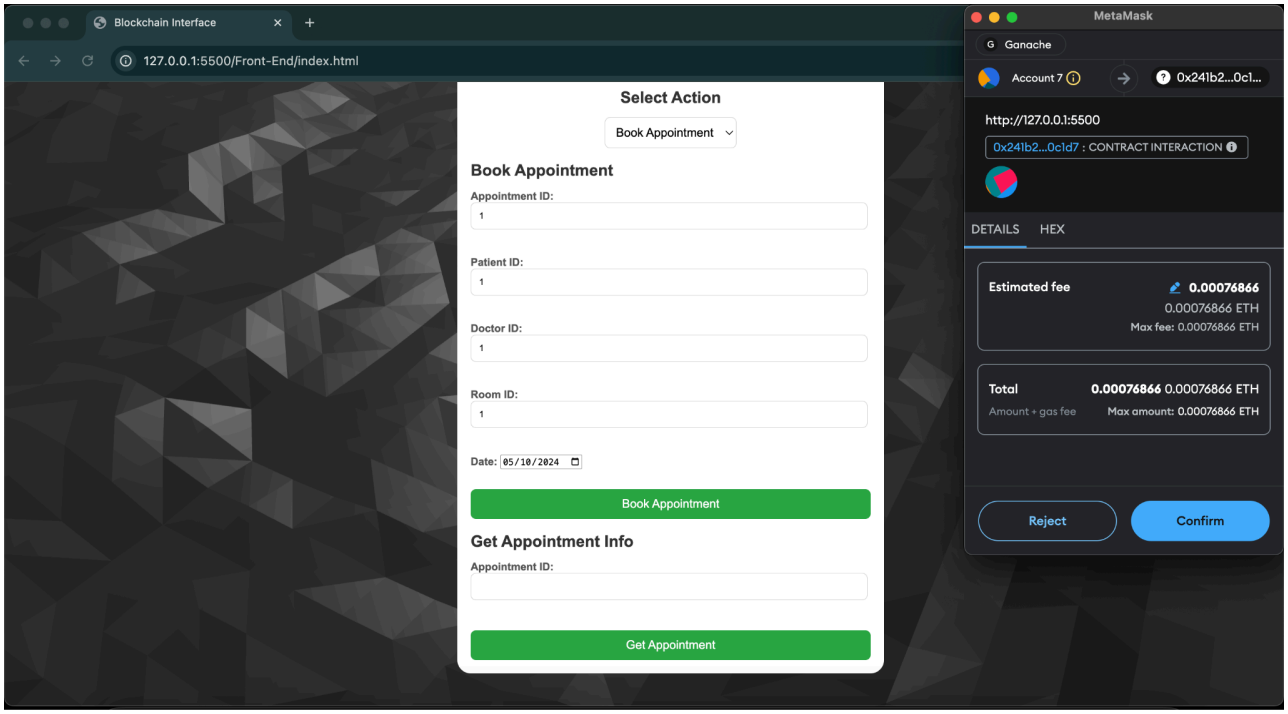
11. Corresponding HTML, CSS and JS are written and deployed.

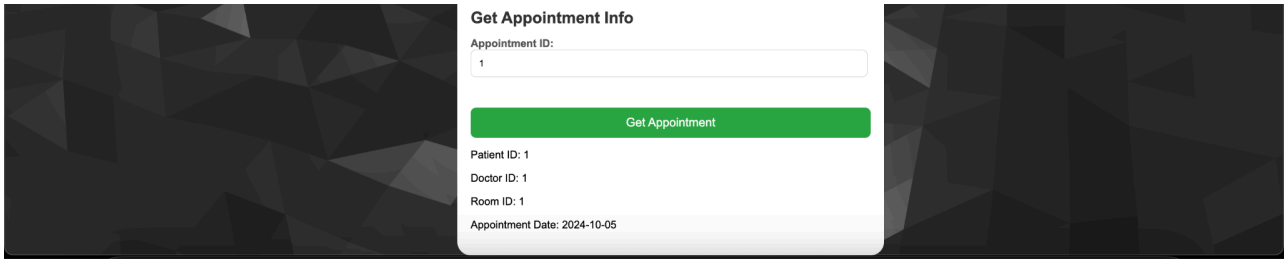



Blockchain Interface
127.0.0.1:5500/Front-End/index.html
Select Action
Add Doctor
Get Doctor Info
Doctor ID:
1
Get Doctor
Name: Samsung
Age: 20
Department: Cardiology


Blockchain Interface
127.0.0.1:5500/Front-End/index.html
Select Action
Add Patient
Add Patient
ID:
1
Name:
Abhinav
Age:
200
Medical History:
being beautiful
Add Patient
Get Patient Info
Patient ID:
Terminal
MetaMask
Ganache
Account 7
0x54F4a...7e...
http://127.0.0.1:5500
0x54F4a...7ec7f : CONTRACT INTERACTION
DETAILS
HEX
Estimated fee
0.00080478
0.00080478 ETH
Max fee: 0.00080478 ETH
Total
0.00080478 0.00080478 ETH
Amount + gas fee
Max amount: 0.00080478 ETH
Reject
Confirm


Get Patient Info
Patient ID:
1
Get Patient
Name: Abhinav
Age: 200
Medical History: being beautiful


Blockchain Interface
127.0.0.1:5500/Front-End/index.html
Select Action
Book Appointment
Book Appointment
Appointment ID:
1
Patient ID:
1
Doctor ID:
1
Room ID:
1
Date: 05/10/2024
Book Appointment
Get Appointment Info
Appointment ID:
Get Appointment
MetaMask
Ganache
Account 7
0x241b2...0c1...
http://127.0.0.1:5500
0x241b2...0c1d7 : CONTRACT INTERACTION
DETAILS
HEX
Estimated fee
0.00076866
0.00076866 ETH
Max fee: 0.00076866 ETH
Total
0.00076866 0.00076866 ETH
Amount + gas fee
Max amount: 0.00076866 ETH
Reject
Confirm


Get Appointment Info
Appointment ID:
1
Get Appointment
Patient ID: 1
Doctor ID: 1
Room ID: 1
Appointment Date: 2024-10-05

SCALABILITY CONSIDERATIONS

In the context of this project, scalability refers to the system's ability to handle an increasing number of transactions (appointments) and participants (doctors and patients) while maintaining performance, security, and reliability.

Blockchain Size and Growth on the Besu Network

As contracts are deployed and transactions increase, the size of the blockchain on the Besu network will expand. The growing number of appointments and interactions between doctors and patients will lead to more data being stored on the blockchain, which introduces the following challenges:

- **Transaction Volume:** Increased transaction volume can lead to longer block validation times and higher resource consumption. The Besu network's ability to handle a large number of transactions efficiently will be crucial, especially as multiple hospitals and users interact with the system simultaneously.
- **Storage Constraints:** Besu nodes will need to store the growing blockchain data. If the system scales to multiple hospitals and adds more transactions, storage demands may increase rapidly, requiring optimized storage solutions or archiving mechanisms to prevent bottlenecks.

RSA Encryption and Integration with Front-End JavaScript

The use of RSA encryption for secure communication between hospitals adds an extra layer of computational complexity, which affects the scalability of the system. In particular:

- **Encryption and Decryption Overhead:** RSA encryption on the sender side and decryption on the receiver side introduces latency, particularly as the number of transactions increases. This process may slow down interaction times between the front-end JavaScript applications and the back-end Besu contracts. As the system scales, it's critical to minimize the time spent on encryption to prevent bottlenecks in real-time appointment handling.
- **Front-End Integration with Fabric EVM:** The integration of the front-end with Fabric EVM needs to be seamless to prevent delays in processing. As the user base grows, optimizing how the front-end interacts with the back-end will be essential to avoid performance degradation when handling large numbers of requests.

Access Control on the Test Net

The test net is used to validate access control mechanisms, ensuring that only authorized users can execute specific transactions. However, as the system scales, the following factors may impact performance:

- Role-Based Access Control Overhead: As more doctors and patients are added to the system, the complexity of managing permissions and access rights will increase. The test net's performance in validating access control must scale efficiently to accommodate a larger number of participants without increasing transaction latency.

Transaction Throughput on the Besu Network

The transaction throughput, or the number of transactions that can be processed per second, is a key scalability factor on the Besu network. As the system scales, more appointments and interactions will need to be processed, and the Besu network must handle this load without compromising performance. Considerations include:

- **Block Creation and Validation:** As more contracts and transactions are deployed on the Besu network, the time required for block creation and validation may increase, potentially limiting transaction throughput.

Network Latency and Geographical Distribution

The system may need to support hospitals located in different geographical regions. Network latency between hospitals interacting with the blockchain could affect the speed of data transfer and contract execution:

- **Cross-Hospital Communication:** If hospitals are geographically distributed, network latency could affect how quickly transactions are propagated across the Besu network and verified by nodes. Ensuring low-latency communication between hospitals is crucial for the system to handle increased volumes without performance bottlenecks.
- **Scalability of Multi-Hospital Systems:** As more hospitals join the network, the time required for cross-hospital transactions to be processed and synchronized across the Besu nodes may increase, affecting overall system performance.

Proof of Work Consensus Overhead

Besu's use of Proof of Work introduces significant scalability challenges due to its computationally expensive nature:

- **Mining Difficulty:** As more transactions are processed and the blockchain grows, the difficulty of the PoW consensus algorithm increases. This means that more computational power is required to validate transactions, which can slow down the time it takes to confirm new blocks.
- **Energy Consumption:** PoW consumes significant energy, especially as more nodes compete to solve the cryptographic puzzle. This can limit the system's scalability because high energy costs and computational power may become prohibitive as the network expands.
- **Transaction Throughput:** The time it takes to mine a new block in a PoW system can become a bottleneck for transaction throughput. If many hospitals are interacting and generating large numbers of transactions, the network could become congested, reducing the number of transactions processed per second.

CURRENT SCALABILITY FEATURES

Decentralized Data Management

- **Blockchain Implementation**: Each hospital has its own blockchain, ensuring that data related to doctors, patients, and appointments is securely managed in a decentralized manner. This decentralized structure inherently supports scalability by allowing each hospital to operate independently while still enabling communication through encrypted transactions.
- **Inter-Hospital Communication**: The use of separate blockchains for each hospital reduces the likelihood of network congestion as each hospital manages its data autonomously. This decentralized architecture enables the system to scale to multiple hospitals without centralized bottlenecks.

Secure Communication via RSA Encryption

- **End-to-End Encryption**: RSA encryption secures communication between hospitals, ensuring that appointment data remains private and tamper-proof. Despite the overhead introduced by RSA, the encryption guarantees the security of sensitive healthcare information as the system scales.
- **Scalability of Encrypted Data Transmission**: The use of RSA allows the system to handle an increasing number of secure communications between hospitals. As the number of transactions grows, the encryption process ensures secure data transmission without compromising the integrity of the information.

Test Net for Access Control

- **Scalable Role-Based Access Control**: By leveraging a test net for validating access control mechanisms, your implementation ensures that only authorized users can initiate or approve transactions. This system is scalable as more users are added. It allows flexible role management without performance degradation.
- **Dynamic Access Control**: As the number of users grows, the access control mechanism ensures that permissions can be dynamically updated and validated for new doctors, patients, and other stakeholders without slowing down the system.

Proof of Work Consensus on Besu Network

- **Scalability Through Decentralization:** PoW on the Besu network provides a secure and decentralized method for validating transactions. As more hospitals and participants are added, the PoW consensus mechanism ensures the network's integrity, allowing it to grow without a single point of failure.
- **Mining Flexibility:** Although PoW can be resource-intensive, it scales by increasing the number of participating nodes that contribute to mining new blocks. This allows the system to maintain its security and consensus even as the number of transactions increases.
- **Support for Multiple Nodes:** The Besu network can support a growing number of validator nodes, which ensures that as more transactions are added, the load is distributed across multiple miners. This prevents bottlenecks in block validation.

Smart Contract Deployment on Fabric EVM

- **Efficient Contract Execution:** The use of Fabric EVM for integrating smart contracts with the front-end JavaScript allows for scalable interaction between users and the blockchain. The smart contracts are optimized to handle large numbers of transactions without significant performance degradation.

SCALABILITY SOLUTIONS

Optimizing Proof of Work (PoW) Consensus

While PoW offers strong security, it is inherently resource-intensive and can slow down transaction processing. Several optimizations can be made to improve scalability in a PoW-based system:

- **Reduce Block Size or Transaction Data**: By reducing the size of each block or the amount of data per transaction, you can decrease the time it takes to mine new blocks. This involves optimizing the structure of the stored data and compressing unnecessary metadata.
- **Increase Block Frequency**: Adjusting the network to produce blocks more frequently can increase the number of transactions processed per second. This requires balancing security with performance, as frequent block production can lower the difficulty level of mining, making it easier to confirm transactions.

Optimizing RSA Encryption Overhead

RSA encryption, while secure, can introduce latency and computational overhead. Several strategies can be applied to optimize the encryption process for better scalability:

- **Hybrid Encryption Approach**: To reduce RSA's computational burden, implement a hybrid encryption model that combines RSA with a symmetric encryption algorithm. RSA can be used to securely exchange a symmetric key, which is then used to encrypt and decrypt data for the remainder of the session. Symmetric encryption is much faster and more efficient for large datasets.
- **Parallel Encryption**: Instead of processing encryption and decryption sequentially, use parallel encryption techniques to handle multiple transactions at once. This can improve the system's ability to handle large volumes of secure communication without introducing significant latency.

Smart Contract Optimization for Fabric EVM

Optimizing smart contracts deployed on the Besu network is crucial for ensuring that the system can handle high transaction loads efficiently:

- **Gas Optimization**: Reduce gas costs by optimizing the logic within smart contracts. This includes simplifying complex calculations, reducing the number of state changes, and minimizing on-chain storage. Efficient smart contract design can lower transaction fees and increase throughput.
- **Lazy Evaluation**: Implement lazy evaluation in smart contracts to defer certain computations or state updates until they are absolutely necessary. This can improve efficiency by reducing the number of on-chain operations required to process transactions.

Optimizing Network Latency and Communication

Reducing network latency is essential for ensuring fast communication between hospitals, especially when they are geographically distributed:

- **Geographic Node Distribution**: Ensure that nodes are distributed geographically in a way that minimizes the physical distance between key participants. Closer proximity reduces the time it takes for data to propagate between nodes.
- **Content Delivery Networks**: Use CDNs to store frequently accessed blockchain data closer to the nodes that need it. This can reduce the time it takes to retrieve and verify data, improving overall network performance.

Improving Access Control Validation on the Test Net

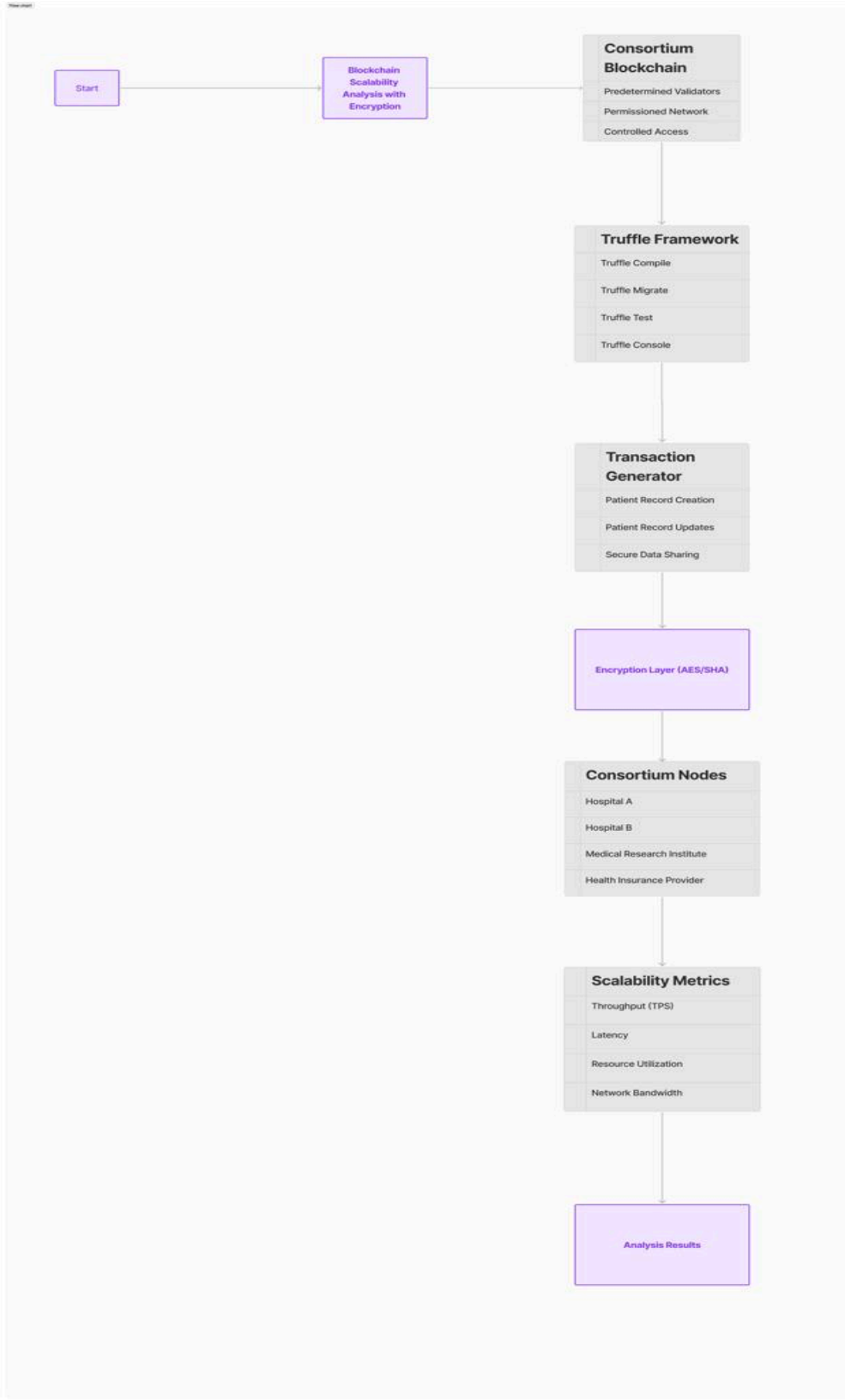
The test net is used to validate access control for users. Optimizing this process can improve scalability as the number of participants grows:

- **Role-Based Access Caching**: To avoid repeated access control validation for the same users, implement a caching mechanism that temporarily stores access permissions. This can reduce the need for real-time validation for every transaction, improving performance.
- **Permissioned Blockchain**: Use a permissioned blockchain model where specific users are pre-approved based on their roles. By limiting the number of access control checks to key actions, the system can scale more efficiently.

CONCLUSION

This project tackles the challenges of both scalability and security in consortium blockchains designed for healthcare. By analyzing the performance of an Ethereum-based blockchain under varying participant numbers and transaction loads, while incorporating RSA encryption for Secure data transmission, the study aims to offer valuable insights into how to configure blockchain networks that are both scalable and secure. The findings will help strike the right balance between performance and security, ensuring blockchain technology can effectively support healthcare data management. Ultimately, this project can assist healthcare providers in adopting blockchain solutions that are efficient, secure, and scalable for managing patient data, paving the way for wider use in the industry.

APPENDIX


Start
Blockchain Scalability Analysis with Encryption
Consortium Blockchain
Predetermined Validators
Permissioned Network
Controlled Access
Truffle Framework
Truffle Compile
Truffle Migrate
Truffle Test
Truffle Console
Transaction Generator
Patient Record Creation
Patient Record Updates
Secure Data Sharing
Encryption Layer (AES/SHA)
Consortium Nodes
Hospital A
Hospital B
Medical Research Institute
Health Insurance Provider
Scalability Metrics
Throughput (TPS)
Latency
Resource Utilization
Network Bandwidth
Analysis Results