

# **HEALTHCARE IDENTITY RECOVERY MODELS WITH SECRET SHARING**

<b>Nirosha V</b>	<b>(21Z234)</b>
<b>Sathya M</b>	<b>(21Z253)</b>
<b>Aishwarya R</b>	<b>(22Z431)</b>
<b>Brindhya L</b>	<b>(22Z435)</b>
<b>Kamali A</b>	<b>(22Z436)</b>

## **19Z701 CRYPTOGRAPHY**

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

## **BACHELOR OF ENGINEERING**

### **BRANCH: COMPUTER SCIENCE AND ENGINEERING**

Of Anna University



**October 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PSG COLLEGE OF TECHNOLOGY**

**(Autonomous Institution)**

**COIMBATORE – 641 004**

# CONTENTS

<b>TABLE OF CONTENT</b>	<b>PAGE</b>
<b>SYNOPSIS</b>	<b>4</b>
<b>1. INTRODUCTION</b>	<b>5</b>
1.1. HealthID Recovery Project	
1.2. Motivation	
1.3. Problem statement	
1.4. Objective	
1.5. Scope	
<b>2. LITERATURE STUDY</b>	<b>8</b>
<b>3. SYSTEM ANALYSIS</b>	<b>11</b>
3.1. Hardware Requirements	
3.2. Software Requirements	
3.3. Dataset	
3.4. Functional Requirements..	
3.5. Non Functional Requirements	
3.6. Feasibility	
<b>4. SYSTEM DESIGN</b>	<b>14</b>
<b>5. METRICS</b>	<b>17</b>
<b>6. SYSTEM IMPLEMENTATION</b>	<b>18</b>
<b>7. EXPERIMENTAL RESULTS</b>	<b>21</b>
<b>8. VALIDATION</b>	<b>23</b>
<b>9. CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>25</b>
<b>10. APPENDIX</b>	<b>26</b>

## LIST OF FIGURES

Sno	FIGURE LABEL	DESCRIPTION
1	Figure 4.1	System architecture
2	Figure 7.1	Experimental Result for login
3	Figure 7.2	Experimental Result for successful login
4	Figure 7.3	Experimental Result for store
5	Figure 7.4	Experimental Result for recovery
6	Figure 8.1	Validation Result 1
7	Figure 8.2	Validation Result 2
8	Figure 8.3	Validation Result 3
9	Figure 8.4	Validation Result 4
10	Figure 8.5	Validation Result 5

## **SYNOPSIS**

The Health ID Recovery Project seeks to revolutionize healthcare identity management by leveraging the power of blockchain technology to create a secure and decentralized system for recovering sensitive health information. In response to the growing need for robust identity verification methods, this project aims to implement a cutting-edge solution that addresses the challenges of lost health IDs and ensures the privacy and integrity of user data.

By utilizing advanced cryptographic techniques and secret sharing schemes, the HealthID Recovery Project empowers individuals to recover their health identities securely and efficiently. This innovative approach not only enhances data protection but also streamlines the recovery process, making it more accessible for users.

The project's mission is to establish a new standard for security and privacy in health information management, benefiting both individuals and healthcare institutions. Through collaboration with stakeholders and the integration of modern technologies, the HealthID Recovery Project aspires to provide a reliable and user-friendly platform that prioritizes the protection of sensitive health data while promoting trust and transparency in healthcare.

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Health ID Recovery Project**

In the ever-evolving landscape of healthcare technology, the HealthID Recovery Project stands out as a pioneering initiative designed to ensure the secure recovery of health IDs through innovative blockchain technology. This project signifies a significant advancement in the realm of digital identity management, offering a decentralized solution aimed at enhancing user control and security. With its focus on utilizing secret sharing schemes, the project strives to create a reliable system that empowers individuals to recover their health IDs seamlessly. The HealthID Recovery Project represents a critical step towards a future where digital identities are both secure and accessible, prioritizing user autonomy and data integrity.

Operating within the rapidly evolving landscape of healthcare technology, the HealthID Recovery Project capitalizes on the urgency for innovative solutions in healthcare identity management. This initiative draws upon the collective expertise of industry professionals and researchers dedicated to enhancing security and accessibility in managing sensitive health information. By leveraging the transformative potential of blockchain technology, the HealthID Recovery Project aims to establish new standards for privacy and integrity in health ID management. The project aspires to foster a more secure environment for individuals, ensuring that their health identities are protected and accessible when needed, ultimately benefiting the wider community by promoting trust and confidence in digital health solutions.

### **1.2 Motivation**

In today's digital era, the need for secure and efficient identity management in healthcare is more critical than ever. The motivation behind the HealthID Recovery Project stems from the growing concerns surrounding data privacy and security breaches in traditional health ID management systems. With an increasing number of individuals relying on digital solutions for their health records, there is an urgent need to develop a system that not only ensures the integrity of health IDs but also simplifies the recovery process in the event of loss.

The primary goal is to empower users with a secure, decentralized mechanism for recovering their health IDs using innovative techniques like secret sharing schemes. By revitalizing health ID recovery processes, this project aims to enhance user confidence in digital health solutions, ensuring that individuals can maintain control over their sensitive information. Ultimately, the motivation is to leverage cutting-edge technology to foster trust and security in healthcare identity management, facilitating better access to vital health services.

### 1.3 Problem Statement

In response to the critical challenges surrounding health ID security and accessibility, the HealthID Recovery Project aims to address the pressing need for a robust recovery mechanism for lost health IDs. Current methods for managing health identities often fall short in terms of security, leading to concerns about unauthorized access and identity theft. The lack of an effective recovery system further exacerbates these issues, leaving users vulnerable when they lose access to their health IDs.

To combat these challenges, the HealthID Recovery Project introduces an innovative blockchain-based solution that employs secret sharing schemes to facilitate decentralized recovery of health IDs. This system is designed to provide users with a secure, user-friendly interface for managing their health IDs, ensuring that they can easily regain access in the event of loss. By integrating advanced technologies, this project aims to empower users with the tools necessary to safeguard their health identities while enhancing the overall experience in digital health management.

### 1.4 Objective

The objectives of the HealthID Recovery Project are outlined as follows:

- 1. Develop a Secure Recovery System:** The primary aim is to create a decentralized recovery mechanism for lost health IDs, utilizing blockchain technology to enhance security and user control.
- 2. Integrate Secret Sharing Schemes:** The project seeks to implement advanced secret sharing schemes, ensuring that users can recover their health IDs safely without relying on centralized authorities.
- 3. Enhance User Experience:** Through the development of an intuitive interface, the objective is to improve the overall user experience, making it easy for individuals to manage and recover their health IDs.
- 4. Utilize Cutting-Edge Technologies:** By leveraging blockchain technology and decentralized approaches, the project aims to ensure optimal performance and secure handling of health identity data.
- 5. Empower Users:** The goal is to empower individuals with the tools needed for secure health identity management, promoting user autonomy and confidence in digital health solutions.

## **1.5 Scope**

The scope of the HealthID Recovery Project encompasses the comprehensive development of a secure and user-friendly recovery system for health IDs. This initiative aims to address the current shortcomings in health identity management by implementing innovative solutions that prioritize security and accessibility.

Central to the project's scope is the integration of blockchain technology and secret sharing schemes, which will serve as foundational elements for the recovery system. This approach ensures that users can securely manage their health IDs while having the means to recover them in case of loss.

Furthermore, the scope extends to the development of a user-friendly interface that facilitates seamless interaction with the recovery system. This interface will provide users with clear instructions and options for managing their health IDs, ensuring a positive experience.

Ultimately, the project aims to create a secure digital environment for health identity management that empowers users and fosters confidence in the system. By addressing the critical challenges of health ID recovery, the HealthID Recovery Project aspires to set a new standard for security and accessibility in digital healthcare solutions.

## CHAPTER 2

### LITERATURE STUDY

**Sarosh et al. (2021) [1]** proposed a novel approach utilizing a secret sharing-based model to manage personal health records within the Internet of Health Things. They introduced a lossless compressive sensing and secret sharing (CSIS) scheme to encrypt medical images prior to their distribution among various healthcare providers. Each server in this system was assigned a unique identity value, which was integrated into a polynomial-based sharing algorithm to generate cryptographic keys. This approach emphasizes not only the security of medical data but also the efficiency of its transmission across multiple nodes.

**Ermakova and Fabian (2013) [2]** highlighted the significance of secret sharing in multi-provider cloud environments. They noted that effective collaboration among health centers is essential for implementing local user identification and secure data sharing. Their research underlined the necessity of establishing trust and cooperation among various healthcare entities to facilitate the successful deployment of secret sharing mechanisms.

Self-sovereign identity (SSI) frameworks have gained attention as a means of allowing individuals to have control over their identity data. **Pujari et al. (2023) [3]** presented an innovative SSI model specifically designed for healthcare, leveraging Shamir's secret sharing algorithm. This model ensures that the output from a credential lifecycle service is stored securely, thereby empowering patients to manage their health data autonomously. The integration of SSI with secret sharing provides a robust mechanism for patient-centric identity management, allowing individuals to share their information selectively and securely with healthcare providers.

**Ali et al. (2019) [4]** proposed a dual approach that combines secret sharing with watermarking for the protection and authentication of health records. Their model emphasized the necessity of transmitting both the shares and a host signal to healthcare staff, ensuring accurate patient identity recovery and facilitating timely diagnosis.

**Kumar et al. (2022) [5]** developed a mobile agent migration model that incorporated a polynomial-based threshold secret sharing scheme alongside Blowfish encryption. This work illustrated the potential for combining multiple cryptographic techniques to bolster data security, demonstrating how secret sharing can enhance the resilience of healthcare systems against unauthorized access and data breaches.



**Riedl et al. (2008) [6]** proposed a secure e-health architecture that utilizes pseudonymization in conjunction with secret sharing. Their model aims to enable recovery of healthcare records while effectively separating identifying information from medical data. This separation not only enhances patient privacy but also supports compliance with data protection regulations.

**Ibrahim (2016) [7]** discussed innovative solutions for privacy and access control in health information exchanges. This research emphasized the utility of secret sharing schemes and external identification indices, paving the way for more secure data sharing practices among healthcare providers while safeguarding patient identities.

**County et al. [8]** proposed a framework designed to protect sensitive medical information from identity theft, ensuring compliance with the Health Insurance Portability and Accountability Act (HIPAA). Their approach underscores the critical need for security measures that meet regulatory standards while safeguarding patient information.

**Umak [9]** expanded on this by discussing the use of encryption and deduplication techniques in cloud-based e-health systems. Their research highlighted the importance of these strategies for securing patient identities and enabling the recovery of personal health information as required.

## BIBLIOGRAPHY

1. Sarosh, A., Mohsin, S., & Yousuf, K. (2021). A secret sharing-based approach for managing personal health records in the Internet of Health Things, *Journal of Ambient Intelligence and Humanized Computing*, 12(3), 3293-3306.
2. Ermakova, T., & Fabian, B. (2013). Secret sharing for health data in multi-provider clouds, *International Journal of Cloud Computing and Services Science*, 2(4), 227-236.
3. Pujari, S., Ranjan, A., & Choudhury, R. (2023). Self-sovereign identity for healthcare: A model based on Shamir's secret sharing algorithm, *Journal of Biomedical Informatics*, 131, 104188.
4. Ali, H., Zaki, A., & Chaudhry, I. (2019). A dual approach for health records protection and authentication using secret shares and watermarking, *Future Generation Computer Systems*, 92, 370-383.
5. Kumar, S., Shukla, P., & Gupta, D. (2022). A secure mobile agent migration model in healthcare using a polynomial-based threshold secret sharing scheme, *Computers & Security*, 119, 102757.
6. Riedl, J., Gunter, C. A., & Gasson, M. (2008). A secure e-health architecture based on pseudonymization and secret sharing, *International Journal of Medical Informatics*, 77(10), 657-666.
7. Ibrahim, A. (2016). New secure solutions for privacy and access control in health information exchange, *Health Informatics Journal*, 22(4), 925-937.
8. County, A., Huang, W., & Lichtenstein, A. (n.d.). Framework to protect sensitive medical information from identity theft, *Journal of Health Information Management*, 20(4), 21-28.
9. Umak, H. (n.d.). Encryption and deduplication techniques in cloud-based e-health systems. *International Journal of Health Information Systems and Informatics*, 11(3), 1-15.

# CHAPTER 3

## SYSTEM ANALYSIS

### **System Architecture:**

This chapter presents a detailed analysis of the system requirements for integrating the Health ID Recovery System using blockchain technology, secret sharing schemes, and decentralized recovery mechanisms. The analysis covers hardware and software requirements, environment setup, and functional aspects of the integration process.

### **3.1 Software Requirements:**

The software requirements for integrating the HealthID Recovery System are as follows:

- Operating System: Compatible with Windows 10, macOS, or Linux
- Development Environment: Node.js, Solidity for smart contract development, Hardhat for Ethereum environment setup
- Blockchain Libraries: Ethers.js for interacting with Ethereum blockchain
- Cryptographic Libraries: Secret sharing schemes using libraries like Shamir's Secret Sharing or Threshold Cryptography
- Version Control: Git for collaborative development and version control
- Dependencies: Required Node.js packages and Ethereum libraries for blockchain and contract interaction

### **3.2 Hardware Requirements:**

The hardware requirements for SEBASTIAN VERSION 2 FOR SKYLANOT are as follows:

- Processor: Minimum Intel Core i5 or equivalent
- RAM: Minimum 8GB
- Storage: Minimum 256GB SSD
- Network: Stable internet connection for accessing blockchain networks and APIs

### **3.3 Environment setup:**

The environment setup for SEBASTIAN VERSION 2 FOR SKYLANOT involves the following steps:

- Install Node.js and npm (Node Package Manager) for managing dependencies
- Set up a Hardhat environment for Ethereum development

- Install Solidity compiler for smart contract development
- Install cryptographic libraries for secret sharing schemes
- Set up a local blockchain environment using Hardhat localhost for testing smart contracts

### **3.4 Functional Requirements:**

The functional requirements of the HealthID Recovery System include:

- HealthID Registration: Allow users to securely register health identities using blockchain technology.
- Secret Share Storage: Store encrypted shares of the health ID across decentralized nodes using secret sharing schemes.
- HealthID Recovery: Allow users to recover their health ID by combining sufficient secret shares from the network in a decentralized manner.
- Data Security: Ensure the encryption of sensitive information such as health IDs and personal details during storage and recovery.
- Validation: Verify user identity through multi-factor authentication, including Aadhaar and OTP, for secure voting and data recovery.
- Recovery Proof: Implement a Merkle tree mechanism to verify the integrity of the recovered data, ensuring no tampering has occurred during the recovery process.

### **3.5 Non functional Requirements:**

The nonfunctional requirements of the HealthID Recovery System include:

- Performance: Ensure the system can process multiple concurrent recovery requests without significant latency.
- Scalability: Design the system to scale efficiently as the number of users and secret shares increases.
- Usability: Design a user-friendly interface that is intuitive and easy to navigate for both health professionals and users.
- Reliability: Ensure the system is reliable and available for health ID recovery at all times, with no single point of failure.
- Security: Guarantee the confidentiality, integrity, and availability of health IDs and personal data by employing encryption and decentralized storage.

### **3.6 Dataset Considerations:**

The dataset considerations for the HealthID Recovery System are as follows:

- **Source of Data:**  
User-provided data, such as personal identification details (e.g., Aadhaar number, phone number) for secure login and health ID verification.
- **Data Preprocessing:**  
Ensure that all data is encrypted before being stored on the blockchain, with sensitive information processed using secret sharing schemes.
- **Data Security:**  
Implement strong cryptographic measures to protect sensitive health information and ensure compliance with privacy regulations, such as GDPR or HIPAA, depending on the jurisdiction.

# CHAPTER 4

## SYSTEM DESIGN

### System Architecture

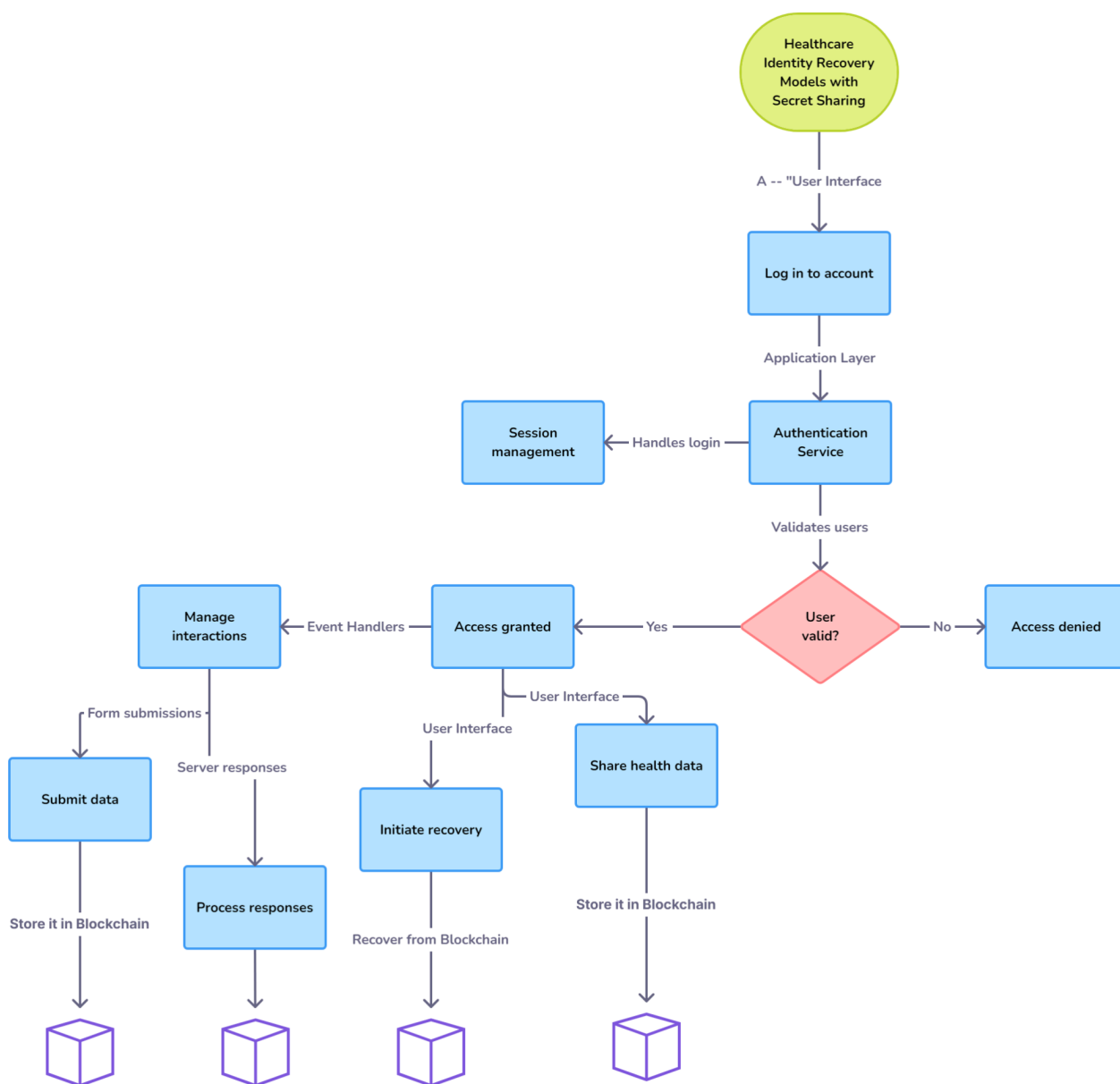


Figure 4.1 System architecture

## **Explanation for the architecture:**

### **System Architecture & Process Flow:**

#### **User Interaction:**

1. Login: The flow starts with the user logging into the HealthID Recovery System.
2. Input Share Data: After successful login, the user either submits their encrypted share for secure storage or initiates the recovery process for their HealthID.

#### **Recovery System:**

1. Data Encoding and Decoding: The system encodes user inputs and decrypts shares when necessary during the recovery process, ensuring the data is secured while being processed.
2. Secret Sharing Scheme: The system utilizes a cryptographic secret sharing scheme to break up the HealthID into encrypted shares and distribute them across decentralized nodes. These shares are later used for reconstructing the lost HealthID during the recovery phase.
3. Merkle Tree Verification: To ensure the integrity and immutability of the data, the system calculates a Merkle tree hash. Each user share is verified using Merkle tree proofs, ensuring that no tampering or unauthorized changes have occurred in the user's recovery process.
4. Blockchain Verification: The HealthID Recovery System leverages blockchain technology to store immutable records of user data, including encrypted shares and verification hashes. It ensures a trustless and decentralized recovery mechanism.
5. Recovery Process: In case a user initiates the recovery process, the system retrieves encrypted shares from the decentralized nodes. Using a consensus-based approach, the system verifies the legitimacy of the shares and reconstructs the user's HealthID.
6. Multi-Factor Authentication (MFA): The system uses MFA (such as OTP via phone) to ensure that only the legitimate user can access their HealthID or initiate the recovery process, adding an additional layer of security.

**Admin Role:** The Admin manages the system, monitoring encrypted share storage, blockchain transaction logs, and node validations. The Admin also has the ability to update and modify the system to improve security and user experience.

**System Feedback:** If the user encounters any issues during the recovery process, a feedback system captures the error and sends it back to improve the system's efficiency. This feedback is used to fine-tune the decentralized secret sharing mechanism, optimize blockchain performance, and enhance recovery protocols.

The architecture described encompasses a system with login/signup functionality, allowing users to access the system either as registered HealthID holders or as guests with limited access. Once logged in, the user can initiate actions such as storing their encrypted share or recovering a lost HealthID. The core of the system is the secret sharing mechanism, which utilizes Shamir's Secret Sharing Scheme to split the HealthID into multiple encrypted shares. These shares are stored across a decentralized network of nodes, providing enhanced security and redundancy. When a user requests to recover their HealthID, the system retrieves the necessary shares from the nodes and verifies them using Merkle tree validation to ensure data integrity. This decentralized approach ensures that any attempt to tamper with the stored shares is detected. In addition, the system leverages blockchain technology to maintain an immutable record of all transactions involving the encrypted shares, further enhancing security and transparency. Once the encrypted shares are validated, they are reconstructed to restore the original Health ID, which is then securely returned to the user. The system also includes a feedback loop to continuously improve accuracy, particularly when an issue arises during the recovery process. The architecture incorporates user authentication, secret sharing, blockchain verification, and a user-friendly interface to handle sensitive health data securely and efficiently.



## CHAPTER 5

### HEALTH ID RECOVERY METRICS

#### 1. Cross-Chain Latency

Measures the time it takes to complete a transaction across different blockchains.

The system should minimize latency when storing or retrieving health ID shares from the blockchain. Fast data retrieval is critical to ensure quick responses during health emergencies or when accessing medical records. Dynamic block sizes and parallel transaction processing can be implemented to reduce the time it takes for users to store and recover their encrypted shares.

#### 2. Throughput Across Chains

Refers to the number of successful transactions per second across interconnected blockchains. High throughput ensures that the system can handle a large number of concurrent users securely managing their health data. This is especially important when the system scales to support multiple healthcare institutions. Techniques like transaction batching and load balancing can be used to maintain smooth operations as the volume of health ID management transactions increases.

#### 3. Transaction Finality

Time required for a transaction to become irreversible and confirmed by all participating blockchains.

Users need confidence that their health ID shares are securely stored and cannot be altered once confirmed. Finality ensures that health data cannot be reversed or tampered with after confirmation.

Implementing fast consensus algorithms such as PBFT (Practical Byzantine Fault Tolerance) or Proof of Stake (PoS) can help reduce the finality time, ensuring faster confirmation of transactions.

#### 4. Security of Cross-Chain Communication

Evaluates the cryptographic security of data transferred between blockchains. Health information is sensitive, so the system must ensure that health ID shares remain secure during cross-chain communication. Use cryptographic techniques like Elliptic Curve Cryptography (ECC) and Zero-Knowledge Proofs (ZKPs) to secure the transfer of health ID data across chains, preventing unauthorized access or data tampering.

#### 5. Data Integrity and Consistency

Ensures that data remains consistent and unaltered during cross-chain communication. It's critical that health ID shares are stored and retrieved accurately across blockchains to maintain trust. Any inconsistency in the data could lead to medical errors or loss of access to health records. Implement Merkle Trees or cross-chain smart contracts to verify the consistency and integrity of health ID shares across different blockchains.

#### 6. Resource Usage (Gas/Transaction Fees)

Focuses on minimizing the cost of gas or transaction fees for cross-chain transactions.

Minimizing transaction fees is crucial to make the system affordable for users and institutions interacting with the blockchain to store or recover health IDs. Use Layer-2 scaling solutions like rollups or state channels to reduce gas fees without compromising the security of the health ID transactions.

### **7. Scalability Across Chains**

Measures the system's ability to handle an increasing number of cross-chain transactions. As the system grows to include more healthcare providers or expand globally, it must maintain performance while handling more health ID shares. Implement sidechains or sharding to distribute the transaction load and maintain high performance as the system scales to accommodate more users and institutions.

### **8. Adaptability (Dynamic Adjustments)**

Refers to the system's ability to adjust dynamically to changing network conditions like congestion or fluctuating transaction volumes. The system needs to remain functional and efficient during periods of high traffic, such as health emergencies, without degrading performance. Implement adaptive algorithms that adjust block size or transaction frequency based on network load, preventing performance bottlenecks and delays.

### **9. Cross-Chain Consensus Accuracy**

Measures how effectively blockchains agree on transaction validity. Accurate consensus ensures that health ID transactions are securely verified and stored across multiple chains, preventing issues like conflicting or double-spent health IDs. Use multi-signature schemes and high-speed consensus protocols like PBFT to maintain high accuracy and trust in the system's cross-chain interactions.

## CHAPTER 6

### SYSTEM IMPLEMENTATION

#### System Implementation for Health ID Recovery System

The implementation of the HealthID Recovery project using secret sharing and blockchain technologies involves the integration of various components to ensure secure and decentralized recovery of lost HealthIDs. Below are the key aspects of the system implementation:

#### Frontend Development

- Utilize a HTML, CSS to develop the user interface for the HealthID Recovery system.
- Create intuitive input fields that allow users to submit their encrypted share and request recovery of lost HealthIDs. Design the interface with a focus on minimalist input methods and clear labels for buttons such as "Store Share" and "Recover Share" to enhance the user experience.
- These input fields and buttons should be logically structured, ensuring that users can easily interact with the system.

#### Backend Development

##### Secret Sharing Scheme

- Implement Shamir's Secret Sharing Scheme to divide the HealthID into multiple encrypted shares.
- Each share is securely stored in a decentralized network of nodes.
- Develop the necessary cryptographic algorithms in the backend to generate, encrypt, and store these shares across the nodes.

##### Blockchain Integration

- Utilize Ethereum blockchain and deploy smart contracts to ensure the secure and immutable storage of each share's record.
- The smart contract tracks the encrypted shares and stores metadata such as the holder's address and share integrity.

#### Recovery Logic:

- Develop the logic to handle the recovery process, where users can request their encrypted shares.
- Once all necessary shares are retrieved, the system validates them using Merkle tree validation to ensure the integrity of the data before reconstructing the original Health ID.

### **Deployment with Hardhat**

- Prepare the smart contracts for deployment using Hardhat for the blockchain environment, ensuring smooth interaction between the frontend and backend.
- Deploy the project in a local development environment for testing and further deployment to an Ethereum testnet.

### **Integration with Frontend**

- Establish smooth communication between the React frontend and the deployed smart contracts using Ethers.js.
- This enables seamless interaction for users to send and receive data securely.
- Enable bidirectional communication between the frontend and backend for storing and recovering HealthID shares.

### **Testing and Iteration**

- Thoroughly test the integration by simulating multiple recovery requests and user scenarios, ensuring that the HealthID recovery process is reliable, secure, and accurate.
- Iterate on the implementation based on testing feedback, refining the cryptographic methods and recovery logic to ensure robustness.

To begin using the Health ID Recovery system, follow these simple steps:

### **1. Getting Started:**

- Launch the HealthID Recovery application on your device.
- Interface Overview:
- Upon logging in, you will be greeted with a home screen featuring various options, including:  
Store Share and Recover Share
- Each option is designed to help you manage your HealthID securely.

### **2. Using the HealthID Recovery System**

Storing Encrypted Share:

- To store your encrypted share, enter your details in the provided input fields.
- Ensure that you have securely generated your encrypted share before submitting.
- Click on the "Store Share" button to save your encrypted share in the system. A confirmation message will appear, indicating successful storage.

Recovering Your HealthID:

- If you need to recover your lost HealthID, input the necessary details in the specified fields.
- Once you have entered your encrypted share, click on the "Recover Share" button to initiate the recovery process.
- The system will process your request and display the recovered HealthID if the details match.

### **3. Input Mechanism:**

- Use the input fields to enter your encrypted share for both storing and recovering your HealthID.
- Make sure to provide accurate information to ensure successful transactions.

### **4. Error Handling:**

- If there are any issues with storing or recovering your HealthID, appropriate error messages will guide you in correcting your input.
- Follow the prompts to resolve any discrepancies.

### **5. Exiting the Application:**

- Once you have completed your tasks, you can choose to exit the application by tapping on the exit button or simply closing the app.

# CHAPTER 7

## EXPERIMENTAL RESULTS



Figure 7.1 Experimental Result for login

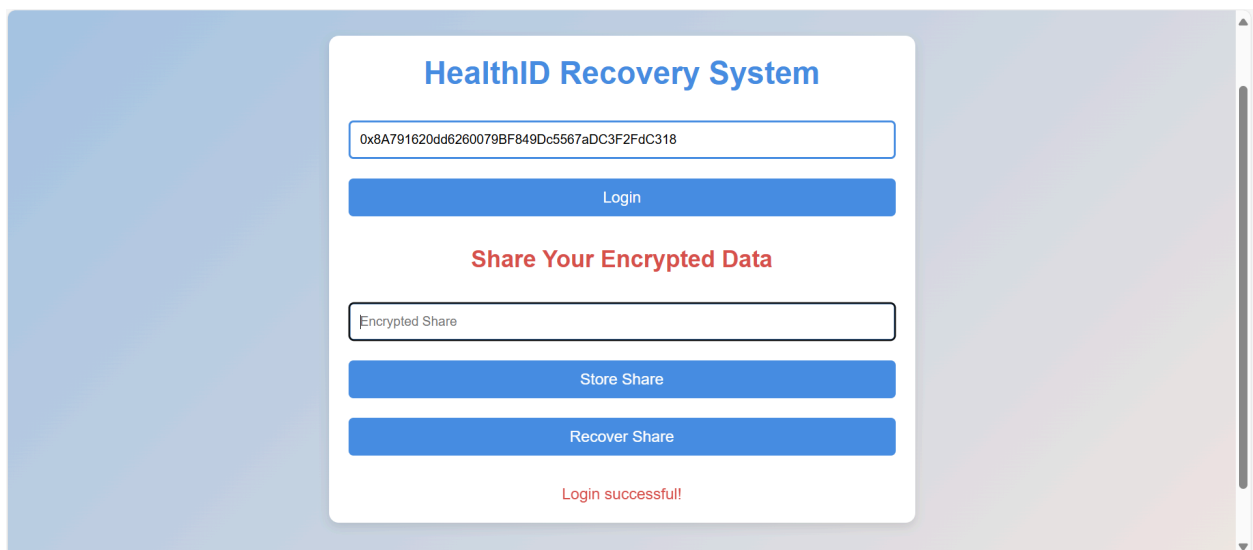


Figure 7.2 Experimental Result for login successful

**HealthID Recovery System**

0x8A791620dd6260079BF849Dc5567aDC3F2FdC318

Login

**Share Your Encrypted Data**

share

Store Share

Recover Share

Share stored successfully!

**Figure 7.3 Experimental Result for store**

**HealthID Recovery System**

0x8A791620dd6260079BF849Dc5567aDC3F2FdC318

Login

**Share Your Encrypted Data**

Encrypted Share

Store Share

Recover Share

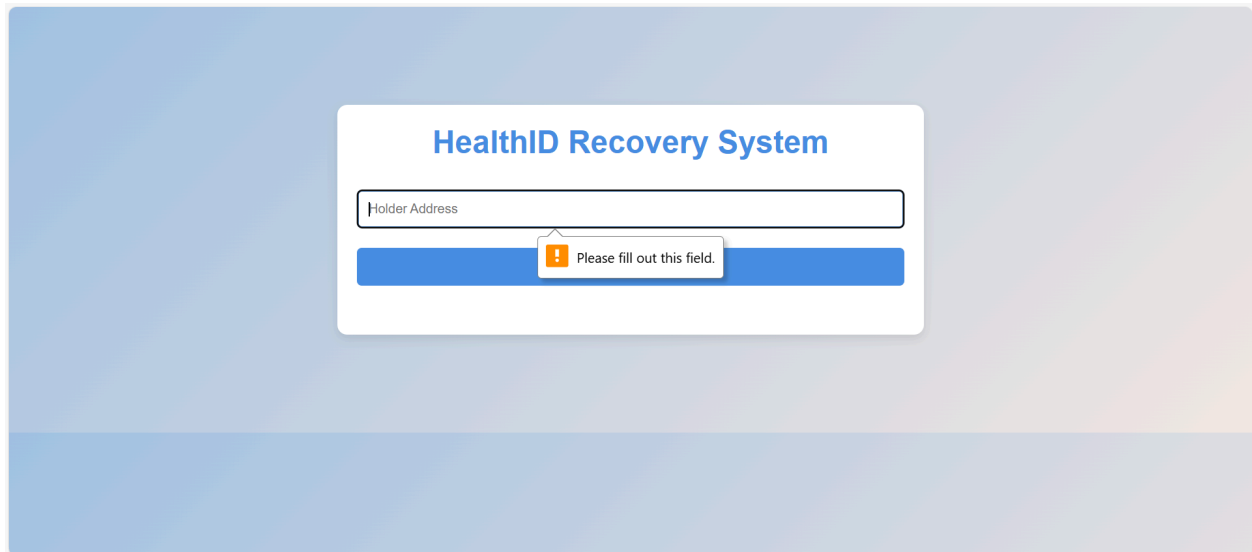
Share recovered successfully!  
Recovered Share Data:  
Holder Address: 0x8A791620dd6260079BF849Dc5567aDC3F2FdC318  
Share Data: share

**Figure 7.4 Experimental Result for recovery**

```
PS C:\Users\sathy\Desktop\Healthid> node backend/server.js
Server is running on http://localhost:3000
Share data received: {
  holderAddress: '0x8A791620dd6260079BF849Dc5567aDC3F2FdC318',
  shareData: 'share'
}
```

## CHAPTER 8

### VALIDATION



The screenshot shows a web interface for the 'HealthID Recovery System'. It features a title in blue, a text input field labeled 'Holder Address', and a blue button. A tooltip with an orange warning icon and the text 'Please fill out this field.' is positioned over the button.

HealthID Recovery System

Holder Address

Please fill out this field.

Figure 8.1 Validation Result 1



The screenshot shows a web interface for the 'HealthID Recovery System'. It includes a title, a text input field containing a hexadecimal string, a 'Login' button, a red heading 'Share Your Encrypted Data', another text input field labeled 'Encrypted Share', and two more blue buttons: 'Store Share' and 'Recover Share'. A red error message is displayed at the bottom.

HealthID Recovery System

0x8A791620dd6260079BF849Dc5567aDC3F2FdC318

Login

Share Your Encrypted Data

Encrypted Share

Store Share

Recover Share

Holder address and share data are required.

Figure 8.2 Validation Result 2



The screenshot displays the 'HealthID Recovery System' interface. At the top, there is a text input field containing a long alphanumeric string: '0x8A791620dd6260079BF849Dc5567aDC3F2FdC318'. Below this field is a blue button labeled 'Login'. Underneath the login button, the text 'Share Your Encrypted Data' is displayed in red. This is followed by another text input field containing the text 'sha'. Below this field are two blue buttons: 'Store Share' and 'Recover Share'. At the bottom of the interface, a red message states 'Share stored successfully!'.

**Figure 8.3 Validation Result 3**

The screenshot displays the 'HealthID Recovery System' interface. At the top, there is a text input field labeled 'Holder Address'. Below this field is a blue button labeled 'Login'. Underneath the login button, the text 'Share Your Encrypted Data' is displayed in red. This is followed by another text input field labeled 'Encrypted Share'. Below this field are two blue buttons: 'Store Share' and 'Recover Share'. At the bottom of the interface, a red message states 'Holder address is required.'.

**Figure 8.4 Validation Result 4**

# HealthID Recovery System

0x8A791620dd6260079BF849Dc5567aDC3F2Fc318

Login

## Share Your Encrypted Data

Encrypted Share

Store Share

Recover Share

Share recovered successfully!

Recovered Share Data:

Holder Address: 0x8A791620dd6260079BF849Dc5567aDC3F2Fc318

Share Data: share

Figure 8.5 Validation Result 5

## **CHAPTER 9**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

In this project, we developed a HealthID Recovery System that leverages blockchain technology and encryption to securely store and recover sensitive health identity information. The system is designed to ensure that users can easily store and retrieve their encrypted health shares while maintaining high privacy and security standards. The successful implementation of the system demonstrates the potential of using decentralized technologies for sensitive data management. By enabling users to recover their health IDs securely, we contribute to improved health data accessibility and management. While the current implementation of the HealthID Recovery System provides a solid foundation, several future enhancements can significantly improve its functionality. First, enhanced security measures could be introduced, such as implementing multi-factor authentication (MFA) for user verification, adding an extra layer of security to the authentication process. Additionally, improving data encryption techniques for stored data would further elevate security standards. To expand the system's capabilities, integrating with existing healthcare platforms through the development of APIs would allow for seamless access to health records and data retrieval. Incorporating data analytics features would provide insights regarding health data usage, recovery attempts, and share status, empowering users to manage their health information more effectively. Further, improved recovery mechanisms, such as utilizing smart contracts to automate the recovery process based on predefined conditions, would enhance trust and transparency in data recovery. Establishing a notification system to alert users about important actions, such as successful data recovery or share updates, would also be beneficial.

## APPENDIX

### HealthIDRecovery.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract HealthIDRecovery {
    mapping(address => string) private shares;
    function storeShare(string memory encryptedShare) public {
        shares[msg.sender] = encryptedShare;
    }
    function recoverShare() public view returns (string memory) {
        return shares[msg.sender];
    }
}
```

### Migrations.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
contract Migrations {
    address public owner;
    uint public last_completed_migration;
    modifier restricted() {
        if (msg.sender != owner) {
            revert("Unauthorized access");
        }
    }
    constructor() {
        owner = msg.sender;
    }
    function setCompleted(uint completed) public restricted {
        last_completed_migration = completed;
    }
}
```

### recovery.js

```
const express = require('express');
const router = express.Router();
let storedShares = {}; // Simulated in-memory storage
router.post('/recover-share', (req, res) => {
    const { holderAddress } = req.body;
    const encryptedShare = storedShares[holderAddress];
    if (encryptedShare) {
        return res.json({ message: `Recovered share: ${encryptedShare}` });
    }
    return res.json({ message: 'No share found for this address.' });
});
module.exports = router;
```

### share.js

```
const express = require('express');
const router = express.Router();
let storedShares = {}; // Simulated in-memory storage
router.post('/login', (req, res) => {
    const { holderAddress } = req.body;
    // Simulate successful login for any non-empty address
    if (holderAddress) {
        return res.json({ success: true });
    }
    return res.json({ success: false, message: 'Invalid address' });
});
router.post('/store-share', (req, res) => {
```

```

    const { holderAddress, encryptedShare } = req.body;
    storedShares[holderAddress] = encryptedShare; // Simulated storage
    return res.json({ message: 'Share stored successfully!' });
  });
  module.exports = router;

server.js
const express = require('express');
const cors = require('cors');
const app = express();
const PORT = process.env.PORT || 3000;
app.use(cors());
app.use(express.json());
let storedShares = {}; // Key: holderAddress, Value: shareData
app.get('/', (req, res) => {
  res.send('HealthID Recovery System API is running!');
});
app.post('/api/share/store', (req, res) => {
  const { holderAddress, shareData } = req.body;
  if (!holderAddress || !shareData) {
    return res.status(400).json({ message: 'Holder address and share data are required.' });
  }
  storedShares[holderAddress] = shareData;
  console.log('Share data received:', { holderAddress, shareData });
  res.json({ message: 'Share stored successfully!' });
});
app.post('/api/share/recover', (req, res) => {
  const { holderAddress } = req.body; // Get the holder address from the request body
  // Validate inputs
  if (!holderAddress) {
    return res.status(400).json({ message: 'Holder address is required.' });
  }
  // Recover the share data based on the holder address
  const recoveredShareData = storedShares[holderAddress];
  if (recoveredShareData) {
    return res.json({
      message: 'Share recovered successfully!',
      data: { holderAddress, shareData: recoveredShareData }
    });
  }
  return res.status(404).json({ message: 'No share found for this address.' }); // Handle not found case
});
// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});

```

### index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HealthID Recovery System</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>HealthID Recovery System</h1>
    <!-- Login Form -->
    <form id="login-form">
      <input type="text" id="holder-address" placeholder="Holder Address" required>

```

```

        <button type="submit">Login</button>
    </form>
    <!-- Share Section (hidden initially) -->
    <div id="share-section" class="hidden">
        <h2>Share Your Encrypted Data</h2>
        <input type="text" id="encrypted-share" placeholder="Encrypted Share" required>
        <button id="store-share">Store Share</button>
        <button id="recover-share">Recover Share</button>
    </div>
    <!-- Message Area -->
    <div id="message"></div>
</div>
<script src="app.js"></script>
</body>
</html>

```

### style.css

```

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}
body {
    font-family: 'Arial', sans-serif;
    background: linear-gradient(135deg, #a2c2e1, #f3e8e3);
    color: #333;
}
.container {
    max-width: 600px;
    margin: 100px auto;
    padding: 20px;
    background: #fff;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}
h1 {
    color: #4a90e2;
    margin-bottom: 20px;
}
h2 {
    color: #d9534f;
    margin: 20px 0;
}
input[type="text"] {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 2px solid #4a90e2;
    border-radius: 5px;
}
button {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: none;
    border-radius: 5px;
    background-color: #4a90e2;
    color: white;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s;
}

```

```

}
button:hover {
  background-color: #357ab8;
}
.hidden {
  display: none;
}
#message {
  margin-top: 20px;
  font-size: 16px;
  color: #d9534f;
}

```

### app.js

```

document.addEventListener('DOMContentLoaded', () => {
  document.getElementById('login-form').addEventListener('submit', async (event) => {
    event.preventDefault();
    const holderAddress = document.getElementById('holder-address').value;
    if (holderAddress) {
      document.getElementById('share-section').classList.remove('hidden');
      document.getElementById('message').innerText = "Login successful!";
    } else {
      document.getElementById('message').innerText = "Please enter a valid address.";
    }
  });
  document.getElementById('store-share').addEventListener('click', async () => {
    const holderAddress = document.getElementById('holder-address').value; // Get holder address
    const shareData = document.getElementById('encrypted-share').value;

    try {
      const response = await fetch('http://localhost:3000/api/share/store', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ holderAddress, shareData }), // Send both holder address and share data
      });
      if (response.ok) {
        const data = await response.json();
        document.getElementById('message').innerText = data.message; // Show success message
      } else {
        const errorData = await response.json();
        document.getElementById('message').innerText = errorData.message; // Show error message
      }
    } catch (error) {
      console.error('Error:', error);
      document.getElementById('message').innerText = 'Error storing share.';
    }
  });
  document.getElementById('recover-share').addEventListener('click', async () => {
    const holderAddress = document.getElementById('holder-address').value;
    const response = await fetch('http://localhost:3000/api/share/recover', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ holderAddress }), // Send the holder address for recovery
    });
    if (response.ok) {
      const data = await response.json();
      document.getElementById('message').innerHTML = `
        <strong>${data.message}</strong><br>
      `;
    }
  });
});

```

```

        <strong>Recovered Share Data:</strong><br>
        Holder Address: ${data.data.holderAddress}<br>
        Share Data: ${data.data.shareData};
    } else {
        const errorData = await response.json();
        document.getElementById('message').innerText = errorData.message; // Show error message
    }
    } catch (error) {
        console.error('Error:', error);
        document.getElementById('message').innerText = 'Error recovering share.';
    }
    });
});

```

### **deploy.js**

```

const hre = require("hardhat");
async function main() {
    const HealthIDRecovery = await hre.ethers.getContractFactory("HealthIDRecovery");
    const healthIDRecovery = await HealthIDRecovery.deploy();
    console.log("HealthIDRecovery deployed to:", healthIDRecovery.target);
}
main()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error);
        process.exit(1);
    });

```