

# Smart IoT-Blockchain Security By Using Shuffled Random Starvation Link Encryption

A report submitted in partial fulfillment of the requirements for  
*the award of*

**Summer Internship**

to

**Ashish Kumar Yadav**  
(23UELEC007)

by



Department of Computer Science Engineering  
National Institute of Patna,  
Patna, 800005

June 2025

---

# CERTIFICATE

Department of Computer Science and Engineering  
National Institute of Technology Patna



This is to certify that the “**Internship Project Report**” submitted by **Ashish Kumar Yadav**, Registration No.: **23UELEC007**, Exam Roll No.: **23UELEC007**, is a registered candidate for the **B.Tech** program under the supervision of **Dr. Kakali Chatterjee**, Assistant Professor in the Department of Computer Science and Engineering at the National Institute of Technology Patna.

---

Supervisor  
Dr. Kakali Chatterjee  
Assistant Professor  
CSE Department  
National Institute of Technology Patna  
23rd may to 20th june (2025)

Date: \_\_\_\_\_

---

# CERTIFICATE OF INTERNSHIP

Department of Computer Science and Engineering  
National Institute of Technology Patna



This is to certify that **Ashish Kumar Yadav**, Registration No.: 23UELEC007, Exam Roll No. **23UELEC007**, student of **Central University of Karnataka**, successfully completed the summer Internship programme from 23rd may, 2025 to 20th June, 2025 at National Institute of Technology Patna.

---

Signature of Authorized

---

# CERTIFICATE OF APPROVAL TO THE PROJECT GUIDE

Department of Computer Science and Engineering  
National Institute of Technology Patna



This is to certify that the project entitled "**Smart IoT-Blockchain Security By Using Shuffled Random Starvation Link Encryption**" submitted by **Ashish Kumar Yadav**, Registration No.: 23UELEC007, Exam Roll No. **23UELEC007**, has been successfully completed under the Supervision of **Dr. Kakali Chatterjee**, Assistant Professor in the Department of Computer Science and Engineering at the National Institute of Technology Patna. This certificate is awarded in recognition of the partial fulfillment for the degree of Bachelor of Technology in Electronics and Communication (B.tech) at Central University of Karnataka.

---

Dr. Kakali Chatterjee  
Assistant Professor  
CSE Department  
NITP

---

EXTERNAL EXAMINER

---

HEAD OF DEPARTMENT

# Candidate's Declaration

I hereby certify that the work, which is being presented in the project report, entitled **Smart IoT-Blockchain Security By Using Shuffled Random Starvation Link Encryption**, in partial fulfillment of the requirement for the award of summer internship and submitted to the institution is an authentic record of my own work carried out during the period **start to end** under the supervision of Dr. Kakali Chatterjee.

I also cited the reference about the text(s)/figure(s)/table(s)/equation(s) from where they have been taken.

The matter presented in this report has not been submitted elsewhere for the award of any other degree or diploma from any institutions.

**Date:** \_\_\_\_\_ **Signature of the Candidate:** \_\_\_\_\_

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. The Viva-Voce examination of Ashish Kumar Yadav, intern student has been held on \_\_\_\_\_

\_\_\_\_\_  
Signature of Research Supervisor(s)

\_\_\_\_\_  
Signature of Head of the Department

# Acknowledgements

I would like to express my deepest gratitude to everyone who contributed to the successful completion of this internship project on developing a **Smart IoT-Blockchain Security By Using Shuffled Random Starvation Link Encryption**.

First and foremost, I extend my heartfelt thanks to my project supervisor, **Dr. Kakali Chatterjee**, for their invaluable guidance, insightful feedback, and constant encouragement throughout the project. Their expertise and support were instrumental in shaping the direction and success of this research.

I am also thankful to **National Institute of Patna** for providing me with the opportunity and resources to undertake this project. Special thanks to the IT(CSE) and Research departments for their technical support and access to essential tools and infrastructure.

A sincere thank you to all the healthcare professionals and domain experts who provided critical insights and real-world perspectives, ensuring the practical relevance and applicability of the system.

Last but not least, I am thankful to my family and friends for their unwavering support, patience, and encouragement throughout this journey. Their belief in my capabilities has been a constant source of motivation.

This project would not have been possible without the collective efforts and support of all these individuals. Thank you all for making this endeavor a rewarding and enriching experience.

# Abstract

As rapid digitization of healthcare sector safeguarding confidential personal medical information has emerged as an important issue of concern. Centralized architectures typical of traditional Electronic Health Record (EHR) systems expose them to data breaches, unauthorized disclosure, and privacy breaches. To counter these threats this project introduces a blockchain-based platform for secure Personal Health Records (PHR) management with a novel two-level encryption strategy inspired by the **Shuffled Random Starvation Link Encryption (SRSLE)** model. The system leverages Ethereum smart contracts developed in Solidity and deployed via Remix IDE providing a decentralized, transparent and tamper-resistant platform for the storing of an encrypted health data. A user-friendly **HTML/JavaScript** frontend, integrated with **MetaMask** and **Web3.js** enables patients to upload encrypted PHR data apply on-chain hashing for additional security and manage their records independently. Ownership verification is strictly enforced through wallet-based authentication and all interactions with the blockchain are immutably logged through event emissions. Our solution enables single and batch uploads of PHRs via `.txt` files and offers real-time monitoring of **gas usage**, **block confirmations** and execution **timestamps** to guarantee transparency and auditability. Testing and auditing prove that the system maintains patient privacy has strict access control and offers complete transparency of data use. Additions like transaction detail visibility and Excel export enhance usability and trust. While the existing implementation puts emphasis on foundational functionality it also provides the foundation for further improvements in the future, such as AI-driven sensitivity classification, off-chain storage through IPFS, and role-based access control.

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>8</b>
1.1	Motivation . . . . .	9
1.2	Objective . . . . .	10
1.3	Issues and Challenges . . . . .	10
<b>2</b>	<b>Related Work</b>	<b>13</b>
<b>3</b>	<b>Problem Statement</b>	<b>16</b>
<b>4</b>	<b>Proposed Work</b>	<b>17</b>
<b>5</b>	<b>Implementation</b>	<b>20</b>
5.1	Environment Setup and System Specifications . . . . .	20
5.1.1	System Specifications . . . . .	20
5.1.2	Software Tools . . . . .	21
5.2	Deployment via Remix IDE . . . . .	21
5.3	Frontend Integration . . . . .	22
5.4	Functional Workflow . . . . .	22
5.5	Deployment Prerequisites . . . . .	23
<b>6</b>	<b>Result and Anlysis</b>	<b>24</b>
6.1	Smart Contract Functionality . . . . .	24
6.2	User Interface Validation . . . . .	24
6.3	Security and Privacy Analysis . . . . .	25
6.4	Events and Transparency . . . . .	25
6.5	Performance and Transaction Insights . . . . .	25
<b>7</b>	<b>Conclusion and Future Work</b>	<b>29</b>
7.0.1	Future work . . . . .	29
	References . . . . .	31



## Chapter 1

# INTRODUCTION

Use of electronic technology in medical care has expanded exponentially over the past few years. Computer systems are now widely used by many hospitals, clinics, and even individuals to store and handle medical information. Such information is referred to as Electronic Health Records (EHRs) or Personal Health Records (PHRs). These records contain very personal information like a patient's name, age, diseases, diagnoses, laboratory results, and medication. Since such information is personal and confidential, it is extremely important to protect it from unauthorized access or misuse.

In conventional healthcare systems most patient medical records are consolidated in centralized repositories or cloud servers. These systems are quick and convenient they pose various risks. Here are some of them: If an unauthorized person hacks into the server or there is a technical fault a lot of private information can get leaked, destroyed or stolen. In certain instances these records may be accessed by even hospital or company employees without authorization. Such data breaches can cause serious privacy issues, identity theft and loss of confidence in healthcare.

The other significant problem is that patients generally don't have complete control over their own information. In most cases, hospitals, businesses or third-party providers retain control over the data, determine access to it and decide how it will be shared. This contradicts the notion of data privacy particularly in healthcare where data is extremely intimate.

To address these issues, researchers now look at applying blockchain technology in the healthcare sector. Blockchain is a unique form of database that is decentralized, clear and tamper-proof. This implies that data in a blockchain cannot be altered or removed once it is incorporated and all transactions are recorded clearly for everyone to view. Every entry on the blockchain is linked to the previous one forming an impenetrable chain of information.

One of the most significant advantages of applying blockchain technology to PHRs is that patients can own and control their information. Rather than relying on a central authority, patients can load their health records from their digital wallets (like MetaMask) to view, upload or share access. While all this is going on smart contracts which are self running code on the blockchain can be deployed to automate processes such as storing, encrypting and authenticating data.

This gives rise to a requirement for robust encryption techniques to provide that only the rightful owner can access the contents of the records. In this project a two-layered security technique is adopted. First, the data is encrypted prior to uploading onto the blockchain. Second, after it's placed on the blockchain, it's also secured through a keccak256 hash function. This process is based on the Shuffled Random Starvation Link Encryption (SRSLE) model, adding extra levels of security to sensitive data fields such as disease and diagnosis.

The system created in this project is implemented through Solidity smart contracts on the Ethereum blockchain with a basic HTML and JavaScript frontend. The web interface provides the capability to upload .txt files with encrypted PHR data and access the blockchain through Web3.js and MetaMask. It is also possible for users to encrypt or decrypt records and see transaction information like gas consumed, block number and timestamps. They are also able to export such information into Excel for better analysis.

## 1.1 Motivation

With the digital age of today the healthcare industry is facing a revolutionary change driven by improvement in data digitization, telemedicine and intelligent medical devices. Personal Health Records (PHRs) electronic records holding a patient’s medical history, diagnostic information, prescriptions, and lab results—are at the core of this change. Although PHRs bring with them enormous advantages in terms of access and continuity of care they also bring some grave issues of data safety, privacy, ownership, and interoperability.

Conventional healthcare models typically maintain patient records on centralized servers operated by hospitals or third-party vendors. Centralized architecture is naturally vulnerable to data breaches, unauthorized access and single-point failures. Instances have been reported where many different incidents pointed out the way in which sensitive health information gets leaked or abused—compromising confidentiality as well as trust of patients.

### Case in Point

In 2021, a well-known hospital chain in Asia suffered a **massive data breach** where hackers infiltrated their database and leaked personal health records of over **1.2 million patients** on the dark web. The leaked files included patients names, diseases (including HIV, cancer and mental health conditions), lab results and contact information. Among the victims was a young woman who had been undergoing treatment for a rare autoimmune disease. A week after the breach, she reported being harassed by an insurance agent who had somehow obtained her confidential diagnosis, offering “customized health plans” based on her sensitive condition. The incident not only caused emotional distress but also exposed the real-world dangers of unsecured healthcare data systems.

This story highlights the urgent need for privacy-first systems where patients control who sees their data and when. Beyond just technical failures, data breaches like this can have **psychological, social and financial** consequences especially for individuals with stigmatized or sensitive health conditions.

In order to tackle these challenges blockchain technology seems to be a hopeful solution. Its basic characteristics—**immutability, decentralization, transparency, and cryptographic security**—cooperatively fit the personal and sensitive nature of healthcare data. Blockchain provides tamper-proof data storage, record tracing of attempted accesses, and direct control of data by the data owners themselves. When coupled with lightweight encryption schemes and smart access control, it introduces next-generation healthcare systems.

The motivation behind this project arises from the a need to cover or minimize the gap between data availability and data privacy in digital healthcare. By leveraging the **Ethereum blockchain and smart contract** functionality we aim to create a transparent, secure and user controlled ecosystem for managing encrypted PHRs.

## 1.2 Objective

The main goal of this project is to create and deploy a secure blockchain-based PHR management system using a Solidity-written smart contract and deployed using the Remix IDE. The system is created to give users complete autonomy over their encrypted health information, provide privacy through hashing and access control and provide transparency through event logging.

**The following are the specific project objectives:**

1. **Secure Storage of Encrypted Health Data:** Make it possible for users to store encrypted health-related fields like disease history, diagnosis and lab results on a public blockchain without revealing sensitive information. This makes sure that although the blockchain is open the health data is kept private.
2. **Ownership-Based Access Control:** Implement a system where only the record owner (wallet address) may access or modify their PHR. Any unauthorized attempts are rejected, and those actions are stored for purposes of auditing.
3. **Sensitive Data Categorization:** Automatically label some fields (such as disease, diagnosis, lab test results) as sensitive using a rule-based categorization method, replicating the behavior of a simple AI classifier such as QBSAI. This tagging establishes which fields need to be encrypted and access-restricted.
4. **Data Integrity and Verification through Hashing:** Enable record owners to hash sensitive fields with **keccak256** to ensure the encrypted data has not been tampered or modified. This facilitates data integrity and a method of verification upon decryption.
5. **Auditability and Event Logging:** Create on-chain logs via event triggers for every primary operation including data storage, encryption, decryption, and attempted access. These logs produce an immutable audit trail that facilitates transparency and trustworthiness.
6. **User-Friendly Front-End Interface:** Create a lightweight HTML and JavaScript interface that communicates with the blockchain through Web3.js allowing non-technical users to upload PHR files, initiate encryption/decryption and examine transaction information.

## 1.3 Issues and Challenges

During the implementation of SecurePHR employing Solidity, Ethereum blockchain and the SRSLE encryption for personal health records (PHR) there were various problems and challenges that were encountered. These were based on both technical restrictions of blockchain platforms and real-world integration issues during the development of a secure healthcare application. The main problems encountered and how they were tackled are discussed below.

### Issue 1: Implementing the SRSLE Algorithm in Solidity

One of the main challenges was implementing the Shuffled Random Starvation Link Encryption (SRSLE) algorithm in a Solidity smart contract. As opposed to general-purpose programming languages Solidity is designed to be a limited programming language, optimized for security and deterministic execution on all Ethereum nodes. It does not have any built-in libraries for advanced randomisation and cryptography that the SRSLE algorithm relies on.

**Challenge:** In order to address this we took a hybrid strategy — executing some of the SRSLE logic on the frontend (in JavaScript) while securely storing and checking encrypted data and hashes on-chain using Solidity. This ensured data integrity on the blockchain while adhering to smart contract computational constraints.

### Issue 2: Creating Consistent Hashes for Sensitive Data

Another important concern was the generation of uniform hashes for sensitive fields of data (e.g. disease, diagnosis and lab results). There were mismatches in the beginning between hashes created on the frontend via JavaScript and those calculated within the Solidity smart contract. This was mainly caused by minor differences between encoding formats and string operations on different platforms.

**Challenge:** We resolved this by standardising the data flow and encoding method ensuring that all sensitive fields were properly serialised and encoded before being passed to the smart contract. Through repeated testing and validation we established consistency between off-chain and on-chain hash outputs.

### Issue 3: Handling Timestamps for Blockchain Transactions

A further challenge came in the form of proper timestamping. Ethereum blocks do contain timestamps but they can only provide rough time indications (as miners get to decide the absolute timestamp within some acceptable range). In the case of medical records where proper time tracking is paramount this did not meet the requirements.

**Challenge:** To alleviate this we added extra frontend-based timestamps upon user action (e.g., sending a PHR submission), which are recorded together with the transaction metadata. This established a more consistent point of reference for the event time from the user end in conjunction with the on-chain block timestamp.

### Issue 4: Gas Usage Metrics Calculation and Export

Monitoring gas usage was important to realize the cost of operation of different contract operations (e.g., storing a PHR, encryption, decryption, and access checking). It was hard though to pull these figures in an easy-to-use format — particularly for being included in Excel reports. Remix does not export transaction metadata natively, and MetaMask only supports minimalistic transaction information.

**Challenge:** We managed to do this by employing Web3.js in the frontend interface to retrieve transaction hashes, used gas, gas price, block number and timestamps at run-time. We also employed integration with SheetJS (xlsx) to support exporting all of this data to Excel for easier visualisation and analysis of system performance and costs.

### **Issue 5: MetaMask Connection and Transaction Error Handling**

Finally, users were commonly experiencing connection errors with MetaMask, especially when changing networks or with low internet connectivity. Some transactions also failed when there were incorrect gas estimates or nonce complications.

**Challenge** We improved the frontend robustness by adding better error handling and user feedback. The software now is able to effectively alert users in case Meta Mask is not in use. If the wrong network has been selected or if a transaction has been blocked. This significantly improved the overall user experience and reliability of the system.

## Chapter 2

# Related Work

Security and privacy of health have grown increasingly more important with advancements in digital technologies and the vast majority of personal information accumulated through networked systems. Applications of blockchain technology in health seek to resolve imperative issues concerning the centralized control of patient data absence of transparency, and susceptibility to unauthorized access and tampering. In the last few years, numerous approaches have been suggested to enhance Personal Health Record (PHR) systems security, efficiency and scalability. This chapter discusses and examines current work in this area and points out their strengths, contributions, limitations and potential for future improvement.

The idea of decentralizing health data management picked pace with blockchain-based PHR systems. An early contribution in this direction is by K. Riad et al. (2019), who proposed the Sensitive and Energetic Access Control (SEAC) mechanism. They proposed a method to manage PHR on cloud servers and included a dynamic permission control mechanism for securing patient data. While SEAC guarantees confidentiality and flexibility of access controls, it has inherent security problems related to centralized data hosting which leaves it vulnerable to breaches. The authors proposed improving the framework of SEAC to strengthen security layers and protect patient privacy from threats.

Y. Sun et al. (2022) presented the PPMRSS model, which is centered on securely sharing health information among IoT environments. The technique effectively solves interoperability challenges and maintains confidentiality representing a groundbreaking approach to managing PHRs in distributed environments. The model however, shows shortcomings in processing data that are maintained in decentralized formats and has limited robustness in file residency control. The authors stressed the importance of enhancing scalability and strengthening security measures in order to facilitate secure and effective sharing of healthcare records at the large-scale level.

Another key contribution was from P.P. Ray et al. (2021) with their research on BloTHR PHR management. Their framework ventured into a blockchain-integrated framework that facilitated privacy-preserving data exchange over swarm-based platforms. This approach introduced a secure data-sharing strategy that dramatically improved privacy and transparency. However, the transition from IoT edge devices to cloud-based backends created new threats. Data migration operations raised issues of authentication, confidentiality, and vulnerability to external attacks. The authors recommended reinforcing EHR control processes for ensuring uniform data security at all levels of engagement.

Mayer et al. (2021) investigated blockchain employment in conjunction with fog computing for PHR data exchange. The study proposed a decentralized data storage system that could guarantee reliable access and improved security. By utilizing fog nodes, the model enhanced latency and accommodated efficient data access. In spite of these advantages, the method required high implementation costs and advanced infrastructure. The authors recognized that the effectiveness of the system had to be confirmed by more extensive simulations and test cases with several network configurations and health scenarios.

A.N. Gohar et al. (2022) resolved communication issues with their Patient-Centric Healthcare Framework (PCHF), highlighting cloud-based blockchain support. Although the model supported secure data exchange, the authors conceded that integration introduced complexity to operations. Particularly, decentralized access needed new policies and tools to manage dynamic data flows. They saw the necessity of developing more efficient and scalable data handling strategies, particularly for large datasets in longitudinal medical studies.

A. Haddad et al. (2023) carried out a comprehensive review of electronic storage of healthcare data and blockchain-based solutions. The review emphasized the improvement of privacy measures in PHR systems. Although the research contributed to categorizing diverse strategies and available methods it did not have an original proposed system. The review was a valuable starting point for learning about important concepts of safe data handling but emphasized the need for follow-up work aimed at interoperability and integration with larger healthcare infrastructure.

Z. Pang et al. (2022) proposed implementing a Practical Byzantine Fault Tolerance (PBFT) mechanism in blockchain-based PHR communication. The technique was intended to maximize cross-hospital data sharing by maintaining the authenticity and integrity of health data. The authors proposed incorporating adaptive consensus policies to facilitate efficient real-time data sharing, ensuring efficiency and minimizing synchronization delays.

Throughout these studies, some of the challenges that are common include the threat of centralized control of data, the challenge to preserve interoperability, the expensiveness in deploying infrastructure and the absence of uniform access policies for various systems. In response to these recent studies—such as M. K. Kala and M. Priya (2024)—have suggested an innovative framework that combines blockchain, IoT and a novel dual-encryption technique referred to as Shuffled Random Starvation Link Encryption (SRSLE). Their framework proposes Quasi-Sensitive Attribute Identification (QBSAI) for high-fidelity data classification and Key Authentication Policy (KAP) to authenticate peer access using behavioral logs and time-stamped transactions.

Ref and Year	Proposed/approach	Contribution	Pros	Cons	Future Study
K. Riad et al. (2019)	Sensitive and Energetic access Control (SEAC)	To handle PHR that was stored on cloud servers	Ensures confidentiality of patient data and provides dynamic adaptation of permissions	Serious issues with EHR security and Risk to consumers' privacy	To enhance the security and privacy of EHR data
Y. Sun et al. (2022)	PPMRSS approach	For safe sharing of healthcare information across IoT	Ensures confidentiality addresses interoperability issues	Flaws in file residency based on hash value and decentralized data	Scalability and security features to be enhanced
P.P Ray et al. (2021)	BloTHr PHR management	Revealed a secure data-sharing plan that protects privacy	Incorporating swarm exchange Paradigms and also enhances Privacy and security levels of EHR transmission	Moving data from these devices to backend servers raises security and privacy issues	EHR management should be enhanced
R. Akkaoui et al. (2020)	EdgeMedichain	To safeguard sensitive data during medical PHR exchange in a Centralized Platform	Increased computational capabilities, ensure data integrity and efficient data sharing	Complexity architecture and Implementation of the framework and increases potential security risks	Further research in ensuring and addressing issues associated with sharing data in a decentralized form
Y. Zhang et al. (2021)	Efficient Identity based distributed decryption Scheme	Investigates a fruitful identity-based algorithm for PHR sharing system	Improved security, Key management, and sharing data without compromising privacy	Crucial problems and difficulty for user's security and privacy during accessing health records	Failed to conduct real-world trials and deployments of the proposed scheme
Mayer et al. (2021)	Blockchain with Fog Computing	To incorporate blockchain-based PHR data exchange	Provides immutable platform with enhanced security	Challenges like technological, Infrastructural cost and Implementation costs of Fog computing will be high	Further explored additional parameters and scenarios to validate the effectiveness of Fog Chain
Zaabar et al. (2021)	Health Block	Efficient and secure remote patient monitoring and EHR management	Provides enhanced security with improved privacy	Better performance, latency	PHR data processing and scalability issues are not solved
A.N. Gohar et al. (2022)	(PCHF) Patient-centric healthcare Framework	For safe data exchange of data in the Cloud	Primarily focused on blockchain-supported decentralized communication	New framework integration is a complex and time-consuming diagnosis in some operations	Enhancement needed to handle large volumes of healthcare data
A. Haddad et al. (2022)	A brief review was carried out	Detailed review carried out with an emphasis on digitally stored healthcare management	Explains all existing schemes related to healthcare data storage management	Provides a robust solution for security and private PHR sharing	Scalability, interoperability, and integration with AI should be enhanced
Ghavat et al. (2022)	Socially Linked data (SoLiD) approach	Used for IoT-based exchange PHR	Secure authentication through decentralized solutions	Includes security flaws, inconsistent data and insufficient security	Focus on the scalability and real-world implementation of the proposed method



## Chapter 3

# Problem Statement

The digital transformation of healthcare is being rapidly accelerated by the integration of IoT enabled medical devices and advanced data analytics. In this landscape an enormous volume of Sensitive Personal Medical Data (SPMD) is continuously generated, transmitted and stored by wearable devices, healthcare systems and cloud platforms. While this data holds the potential to drive breakthroughs in precision medicine and improve patient outcomes it also raises serious concerns regarding data privacy, security and control.

The majority of current healthcare data management systems are based on centralized models in which patient information is kept on third-party cloud servers. This places patient information at risk of a wide range of threats such as unauthorized access, data breaches and misuse since control is taken away from the patient. In the context of IoT-based healthcare, these vulnerabilities are compounded by the restricted computational powers of IoT devices that are incapable of executing intricate encryption protocols. Also, the absence of adequate audit trails and access transparency in legacy systems results in patients being unable to monitor easily how their data is accessed or shared.

Another similarly critical issue is that existing encryption techniques employed for secure protection of PHR during transmission frequently do not offer granular, individualized protection. Additionally, there is no universal implementation of a mechanism that confirms the trustworthiness of each entity accessing the sensitive information. With increasing regulatory obligations like GDPR and HIPAA, healthcare systems now have to ensure traceable, patient-controlled and tamper-resistant data management solutions.

It is especially difficult to bridge these issues in the context of IoT-based healthcare scenarios. Exchanging data between resource-limited IoT devices and cloud servers, frequently across public communication networks can cause privacy issues and security threats. Additionally without a Key Authentication Policy (KAP) and two-level encryption processes attackers can take advantage of vulnerabilities in peer-to-peer networks and centralized databases to access unauthorized information.

To overcome these challenges, this project introduces a new system called: **"Smart IoT-Blockchain Security to Secure Sensitive Personal Medical Data Using Shuffled Random Starvation Link Encryption (SRSLE)"** The process utilizes Ethereum blockchain smart contracts, **Shuffled Random Starvation Link Encryption (SRSLE)** for secure data encryption and **Quasi Sensitive Attribute Identification (QBSAI)** to dynamically identify and encrypt sensitive attributes in PHR data. By using blockchain in conjunction with imposing **Key Authentication Policy (KAP)** this architecture provides an immutable, tamper evident and open ledger for all transactions involving sensitive medical data.

## Chapter 4

# Proposed Work

The proposed project works around designing and developing a **decentralized, privacy-preserved, and secure PHR management system** based on blockchain and smart contracts. The concept arises out of the need to overcome the weaknesses of centralized Electronic Health Record (EHR) systems, where patients do not have control over their data, and where unauthorized access or breach of data is highly likely. With healthcare data becoming more sensitive and digital by the day, there is an urgent need for systems that are not just protecting patient privacy but also offering transparency, traceability, and fine-grained access control. Our system responds to these challenges through the creation of an end-to-end solution based on **Solidity smart contracts running on Ethereum (through Remix IDE)**, a basic yet easy-to-use **HTML/JavaScript** for use, and conceptual integration of **advanced encryption and categorization methods** based on state-of-the-art research.

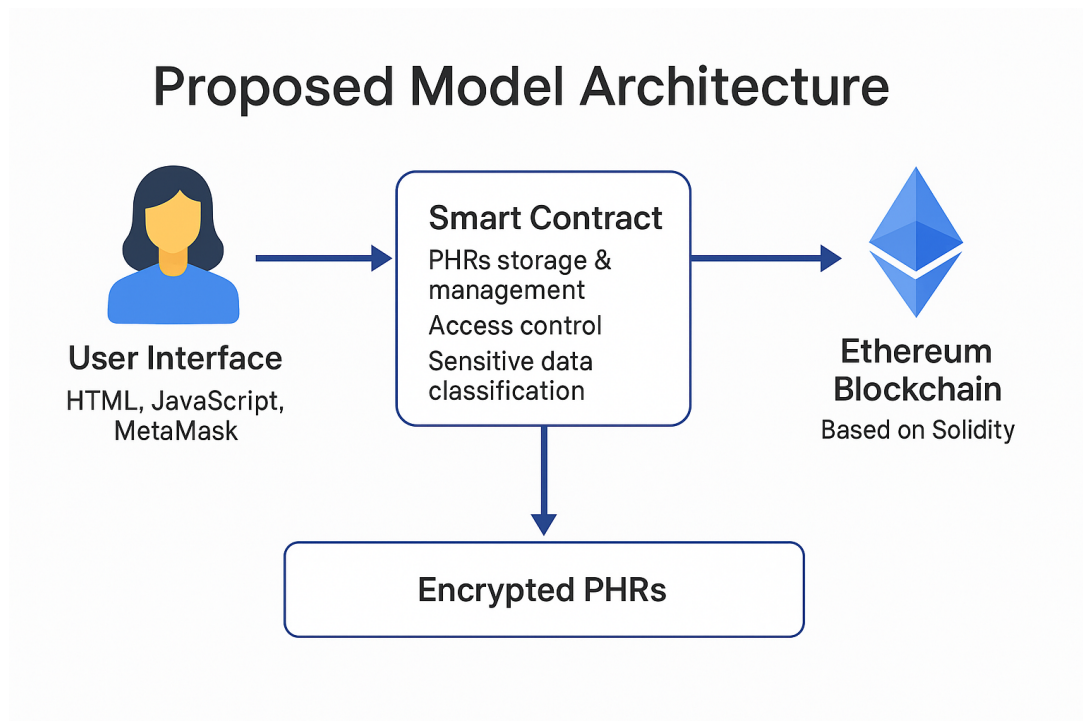


Figure 4.1: Proposed Model Architecture (PHR Flow)

## Module 1: Smart Contract.

This module is the system backbone. A Solidity smart contract SecurePHR is authored and deployed to the Ethereum blockchain using Remix IDE. Every single PHRs record is indexed uniquely and has fields like the **patient id, name, patient age, encrypted diagnosis, encrypted disease and laboratory test**. The contract structure also contains metadata such as entry's **timestamp, owner's wallet address and hashed versions of sensitive data** for integrity verification. This structure ensures that once a PHR is stored, its information cannot be modified or accessed by any individual other than the owner. The decentralized nature of Blockchain assures data immutability and tamper resistance which forms the foundation for secure management of healthcare data.

## Module 2: User Interface

A basic frontend using HTML and JavaScript is implemented to enable users (patients) to make it simple to interact with the blockchain. The interface has the capability to upload files with .txt extension holding PHR data, initiating encryption or decryption operations and viewing transaction information like gas consumed and block number. It is hooked to the smart contract with Web3.js.

## Module 3: Blockchain Interaction

This module handles all interactions with the blockchain via MetaMask and Web3.js. Wallet-based authentication using MetaMask is done for identity verification, so only valid users have the permission to act. Transactions are signed and sent via MetaMask a secure and user-managed environment.

## Module 4: Security and Encryption.

For safeguarding sensitive health information, the system employs a two-layer encryption process. Encrypted PHR information is uploaded by users and the smart contract then adds another layer of protection by utilizing the keccak256 hashing algorithm (adapted from the Shuffled Random Starvation Link Encryption process). This process protects even if blockchain information is exposed because it keeps the data unreadable for unauthorized users.

One of the highlights of this system is that it can identify sensitive and non-sensitive data at the attribute level. Similar to the **Quasi Sensitive Attribute Identification (QBSAI)** method described in applicable literature, our system emulates sensitivity classification by automatically labeling fields like disease, diagnosis, and lab results as sensitive when receiving data.

## Module 5: Event Logging and Export.

For transparency, all critical activities (record storage, encryption, attempts at access) are tracked through smart contract events. The frontend offers a feature for exporting transaction information into Excel so users can view and track their records securely.

## Module 6: Access Control or verification.

Access control is strictly enforced throughout the contract. Only the owner uploader as identified by their Ethereum wallet address, can read or update their PHR. Any access attempts by unauthorized users to view, encrypt, or decrypt the data are rejected and recorded through the AccessAttempt event. This mechanism produces a decentralized and verifiable access pattern record, inhibiting nefarious acts and promoting trust among users. The access control policy is also consistent with the Key Authentication Policy (KAP) model presented in the supporting research which insists on the need to authenticate user identity prior to admitting access to private records. While our current deployment utilizes address ownership, it sets a good foundation for more sophisticated key verification and peer authentication protocols.

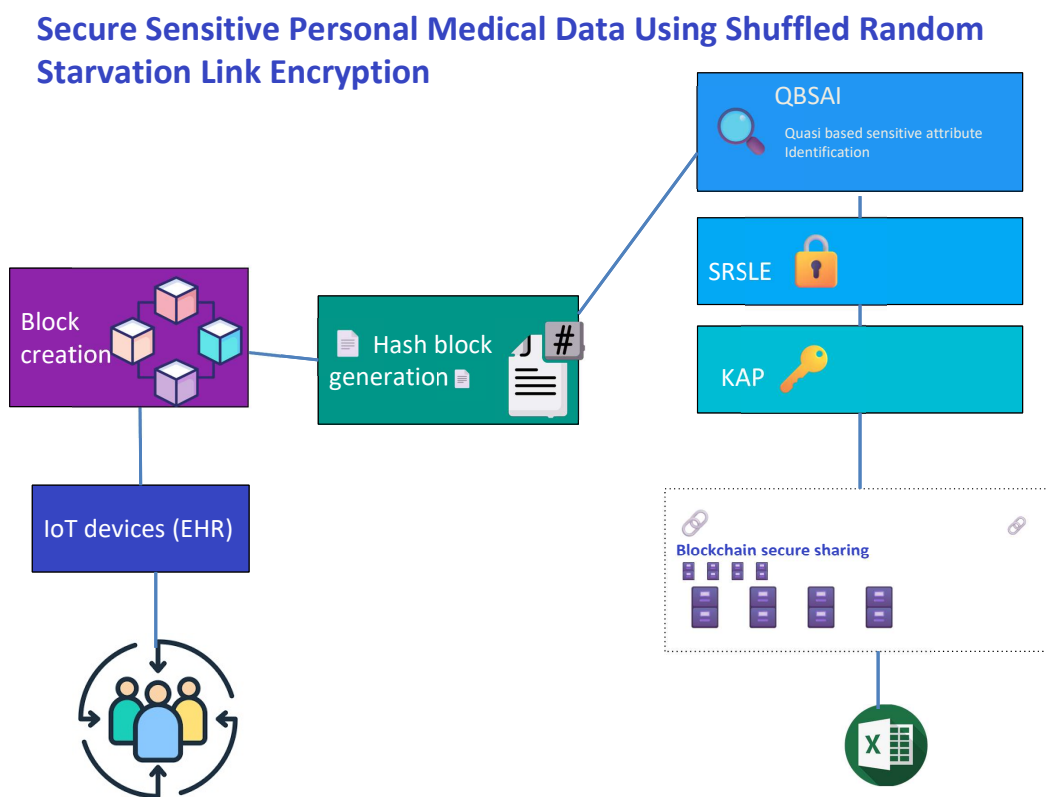


Figure 4.2: Smart IoT-Blockchain Security to Secure Sensitive Personal Medical Data Using Shuffled Random Starvation Link Encryption(PHR Flow)

## Chapter 5

# Implementation

The system is developed based on a Solidity smart contract named SecurePHR, which is deployed on the Ethereum blockchain through the Remix IDE. The contract is meant to secure Personal Health Records (PHRs) adding functionality such as encryption, ownership verification, and sensitivity categorization.

The central element is a PHR Struct that stores encrypted fields like disease, diagnosis, and lab results, along with the patient's address and a timestamp. There is a two-layered security model where hashing and encryption are applied for sensitive data. There are two mappings: (i) PhrRecords maps a unique PHR ID to the patient record. (ii) classified Sensitive flags very sensitive fields based on QBSAI-based logic.

Events are issued for each critical event, including record storage, attempted accesses and sensitivity levels to create a transparent and auditable history. Access is controlled through modifiers to guarantee that only approved users have access or update privileges to information.

The smart contract is deployed on the "Injected Web3" in MetaMask environment. Upon deployment the contract address is utilized to access a web-based frontend that allows users to safely and smoothly interact with the blockchain. This decentralized approach guarantees privacy, traceability and tamper-proof storage of health information.

## Implementation Steps

### 5.1 Environment Setup and System Specifications

The development and testing of the proposed system was performed using the following hardware and software environment:

#### 5.1.1 System Specifications

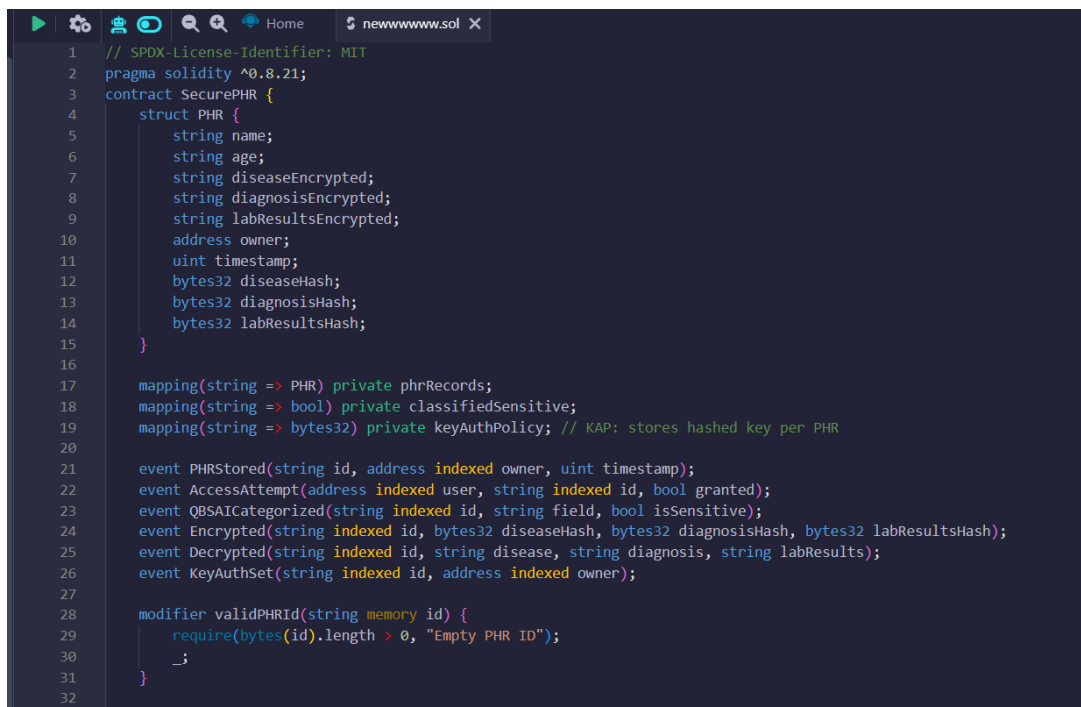
- **Processor:** Intel Core i5 or equivalent
- **RAM:** 8 GB or higher
- **Operating System:** Windows 10 / Windows 11 (64-bit)
- **Web Browser:** Google Chrome (latest version)

### 5.1.2 Software Tools

- **Remix IDE:** Online (Solidity version 0.8.21)
- **MetaMask Extension:** v11.x (latest version as of 2025)
- **Ethereum Network:** Sepolia Testnet (via MetaMask connection)
- **Solidity Compiler:** v0.8.21
- **Web3.js:** v1.8.2
- **Frontend Development:** HTML5, JavaScript.

## 5.2 Deployment via Remix IDE

1. Create a Smart Contract in Remix Ide in Solidity Programming Language.
2. Compile the solidity file in Remix IDE.
3. Deploy the Contract by using the “Injected Web3” environment with MetaMask.
4. After successful deployment, note down the contract address for frontend integration.
5. Go to **”Deploy and Run Transactions”** panel:
  - Set the environment to **”Injected web3”** (connect MetaMask).
  - Deploy **”SecurePHR”** contract.
  - Copy the deployed contract address.



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.21;
3 contract SecurePHR {
4     struct PHR {
5         string name;
6         string age;
7         string diseaseEncrypted;
8         string diagnosisEncrypted;
9         string labResultsEncrypted;
10        address owner;
11        uint timestamp;
12        bytes32 diseaseHash;
13        bytes32 diagnosisHash;
14        bytes32 labResultsHash;
15    }
16
17    mapping(string => PHR) private phrRecords;
18    mapping(string => bool) private classifiedSensitive;
19    mapping(string => bytes32) private keyAuthPolicy; // KAP: stores hashed key per PHR
20
21    event PHRStored(string id, address indexed owner, uint timestamp);
22    event AccessAttempt(address indexed user, string indexed id, bool granted);
23    event QBSAICategorized(string indexed id, string field, bool isSensitive);
24    event Encrypted(string indexed id, bytes32 diseaseHash, bytes32 diagnosisHash, bytes32 labResultsHash);
25    event Decrypted(string indexed id, string disease, string diagnosis, string labResults);
26    event KeyAuthSet(string indexed id, address indexed owner);
27
28    modifier validPHRId(string memory id) {
29        require(bytes(id).length > 0, "Empty PHR ID");
30        _;
31    }
32 }
```

Figure 5.1: Sample code of Smart contract in Remix IDE (Solidity)

## 5.3 Frontend Integration

A web-based frontend is created using HTML, CSS, and JavaScript, and connects to the deployed smart contract using Web3.js. and copy the ABI code and Contract Address from the Remix IDE and paste to the HTML/Javascript in VS code. The frontend provides a user-friendly interface for interacting with the smart contract.

- **Upload ‘.txt’ Files:** Reads file and parses into key-value format.
- **Copy the ABI code from the remix IDE**
- **Copy the Contract Address from the remix IDE**
- **Paste the ABI and Contract Address to Html/Javascript in Vs code**
- **Buttons:**
  - upload and submit: Call `storePHR()` on chain.
  - encrypt by ID: Calls `encryptSensitiveById()` to hash sensitive fields.
  - Decrypt by ID: Calls `decryptSensitiveById()` to log values.
- **Transaction Details Panel:** Displays `txHash`, block number, Gas used, timestamp

### Sample File Format

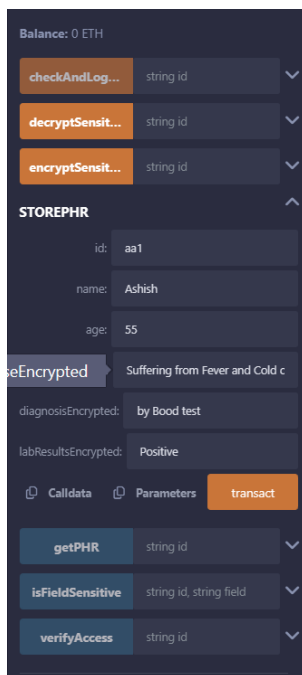
```
id: aa2
name: Ashish
age: 55
diseaseEncrypted: Suffering from Fever and cough
diagnosisEncrypted: By Blood test
labResultsEncrypted: Positive
```

## 5.4 Functional Workflow

1. **PHR Creation:** User uploads a ‘.txt’ file. The system parses the data and invokes the `storePHR()` function to store it on-chain.
2. **QBSAI-Based Field Sensitivity:** Fields such as disease, diagnosis, and lab results are automatically classified as sensitive using a field-mapping logic (e.g., `id_disease`).
3. **Encryption Simulation:** The `encryptSensitiveById()` function hashes each sensitive field using `keccak256`, simulating a second layer of encryption (SRSLE logic).
4. **Decryption Simulation:** The `decryptSensitiveById()` function emits the encrypted values via logs. Although no actual decryption is performed, this emulates secure field retrieval.
5. **Access Control:** Functions such as `getPHR()`, `verifyAccess()`, and `checkAndLogAccess()` enforce strict owner-only access to sensitive PHR data and log all access attempts.
6. **Field Sensitivity Verification:** The `isFieldSensitive()` function checks whether a specific field is flagged as sensitive based on prior classification.

## 5.5 Deployment Prerequisites

- MetaMask browser extension must be installed and connected.
- Remix IDE must be used for contract compilation and deployment.
- The deployed contract address and ABI must be added to the frontend JavaScript and HTML.
- A test network (Ganache CLI, Sepolia) must be connected for testing transactions.



(a) Storing PHRs Manually in Solidity

### Upload a PHR Record (.txt)

Choose File id aa3.txt Upload and Submit

Enter PHR ID: e.g., phr1234 Encrypt by ID Decrypt by ID

#### Last Transaction Details

**Transaction Hash:** 0x1715bdaaaae368d1dfe6bb39f6b43f812c71975fdd3c7311340f04bd13b15ec9  
**Block:** 8579633  
**Timestamp:** 19/6/2025, 7:18:36 am  
**Gas Used:** 78998  
**Gas Price:** 0.001013912 Gwei

Download This Tx as Excel

Export ALL Transactions to Excel

(b) Interface Integration for .txt file

Figure 5.2: Smart contract code in solidity and Interface Integration in browser



(a) before Encryption



(b) After Encryption

Figure 5.3: Encryption of data for the privacy



# Result and Anlysis

## 6.1 Smart Contract Functionality

The core of the SecurePHR system is Solidity smart contract deployed via the Remix IDE.the contract includes functionalities for storing,encrypting,decrypting and verifying access to health data records.Several vital features were validated through testing:

- **StorePHR()** successfully accepts encrypted PHR values along with metadata like name,age and timestamps.
- Access control was also tested with `verifyAccess()` and `checkAndLogAccess()` function to restrict view or editing of sensitive field only to the PHR's owner.
- The `encryptSensitiveById()` function generated keccak256 hashes(via keccak256) for disease,diagnosis and lab result fields.These hashes act as a verification layer and immutable fingerprint of encrypted data.
- `decryptSensitiveById()` simulated the decryption by emitting the stored encrypted values,reinforcing privacy by not storing plain data on-chain.

## 6.2 User Interface Validation

- The front-end interface implemented in simple HTML and Web3.js offers easy access to blockchain functionality without profound technical expertise:
- File upload capability interprets `.txt` files holding PHR information and assigns each field to the proper smart contract arguments.
- "Encrypt by ID" and "Decrypt by ID" interaction buttons enables users to process sensitive information securely.
- A committed transaction details panel programmatically retrieves and shows transaction hashes,block numbers,gas consumption and timestamps providing users with transparency and confidence.

### 6.3 Security and Privacy Analysis

- **Sensitive information is never exposed in plaintext** on-chain maintaining patient confidentiality.
- Only the owner wallet address that stores a PHR is authorized to update or access the data enforced via `require()` checks.
- The **sensitivity classification** via the `classifiedSensitive` mapping mimics a lightweight AI based flagging mechanism (QBSAI) that automatically labels critical fields.
- Tamper detection is feasible through the consistency of `keccak256` hash outputs for encrypted values, ensuring any off-chain tampering or corruption of encrypted data can be quickly detected.

### 6.4 Events and Transparency

- `PHRStored` logs creation date and time as well as owner information.
- `Encrypted` and `Decrypted` events send data hashes or encrypted field values for transparency and auditability.
- `AccessAttempt` offers logs that may be analyzed later to detect unauthorized access attempts adding auditability and support for healthcare data policies.

### 6.5 Performance and Transaction Insights

- Gas usage for storing a PHR was **32,000-35,000 gas** depending upon inputs string length.
- Encryption and decryption processes were take **75,000-135000 gas**, as there were only internal memory operations involved (no external contract calls).
- The web3 interface displayed consistent interactions with MetaMask environments, so the solution was suitable to use for a larger blockchain test network such as Sepolia.

Function Name	Average Gas Consumed
Contract Deployoment	231781.9
Get PHRs	0
Verify Access	0
Decryption	75860.4
Encription	131467.4
By Is Field Sensitive	0
check login	43824.1
By Uploading Txt file	328083.4
Storing PHRs	32873.7

Table 6.1: Average Gas Consumed by Different Smart Contract Functions

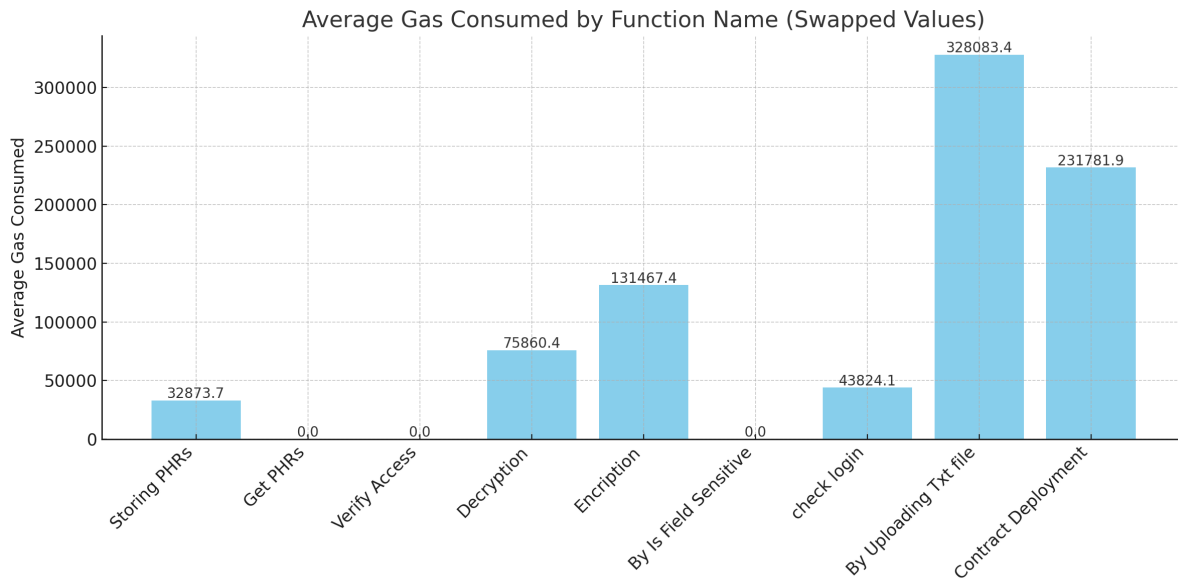


Figure 6.1: Gas Consumed By different function in Solidity

#### Gas Consumed by different function call

**In Fig 6.1,** The graph shows the amount of average gas used by various smart contract functions. The most consumed gas is by "By Uploading Txt file" ( 328,083 units), then by "Contract Deployment" ( 231,782 units). Encryption and Decryption also use a large amount of gas but check login and Storing PHRs use less. Functions such as Get PHRs, Verify Access and By Is Field Sensitive have zero gas consumption, probably because they are read only or not being used. This comparison assists in determining what functions are most resource-intensive on the blockchain.

#### Block size vs Gas Consumed

**In Fig 6.3,** The graph shows the correlation between block size and the gas consumed in blockchain transactions. As displayed, the gas consumed does not always grow with the size of the block. Sometimes even with a lower block size gas consumption is higher probably because there are complicated transactions in the block. Some time Larger blocks consume less gas when they hold simpler or more effective transactions. This graph helps us analyze how the nature of transactions not just block size influences gas consumption in the blockchain network. Such information can assist us in streamlining transaction processing and enhancing overall network efficiency.

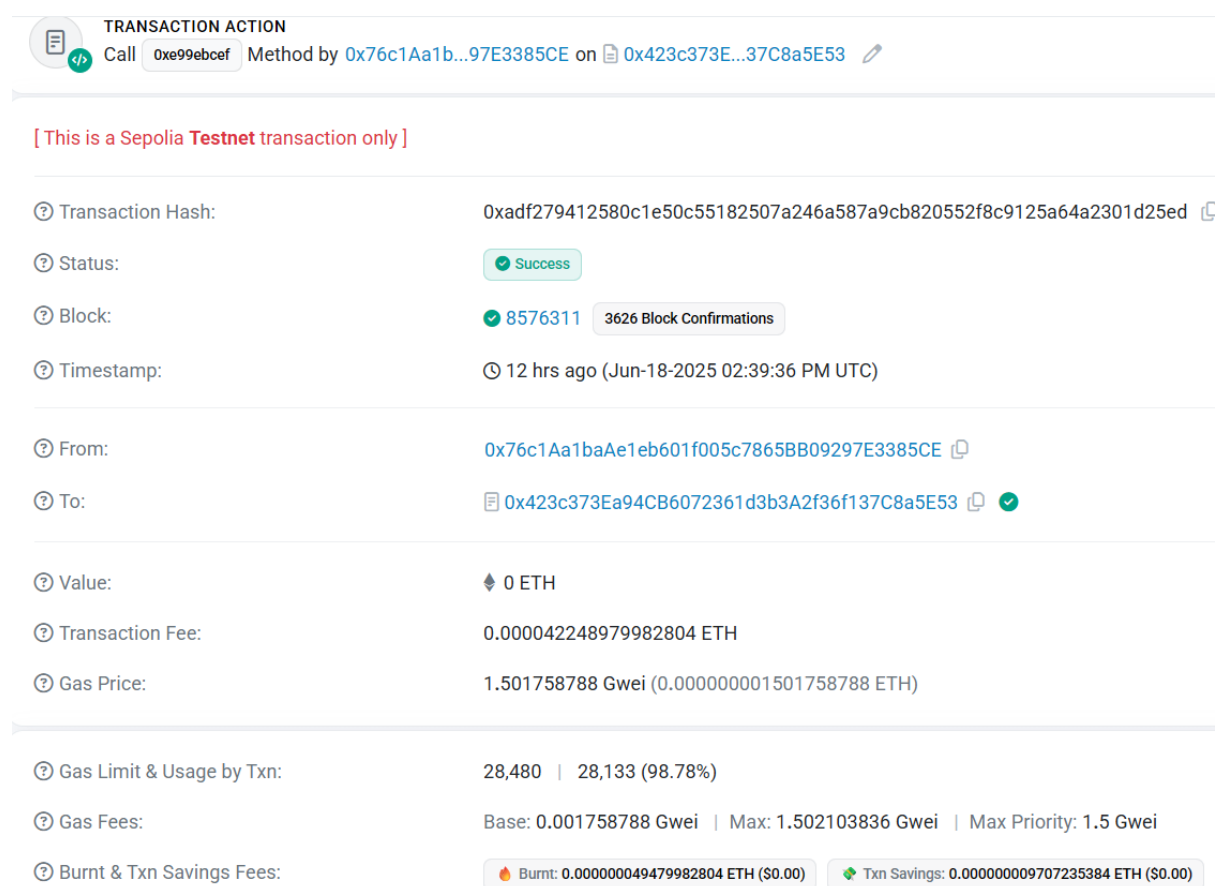


Figure 6.2: Transaction History of Metamask for Contract Deployment

### Block number vs Gas used

**In Fig 6.4**, The following graph shows that the block number against the gas consumed per transaction on the blockchain network. From the graph we can see that gas consumption is not uniform it keeps fluctuating with each block. In certain blocks, gas consumption is large reflecting complex transactions while in others it is small perhaps due to simpler operations. This trend informs us about how gas consumption acts over time within our system. To be more clear we enhanced the graph by placing markers on the data points and employing a more perceptible line style. This allows it to be easier to follow individual blocks and their respective gas consumption.

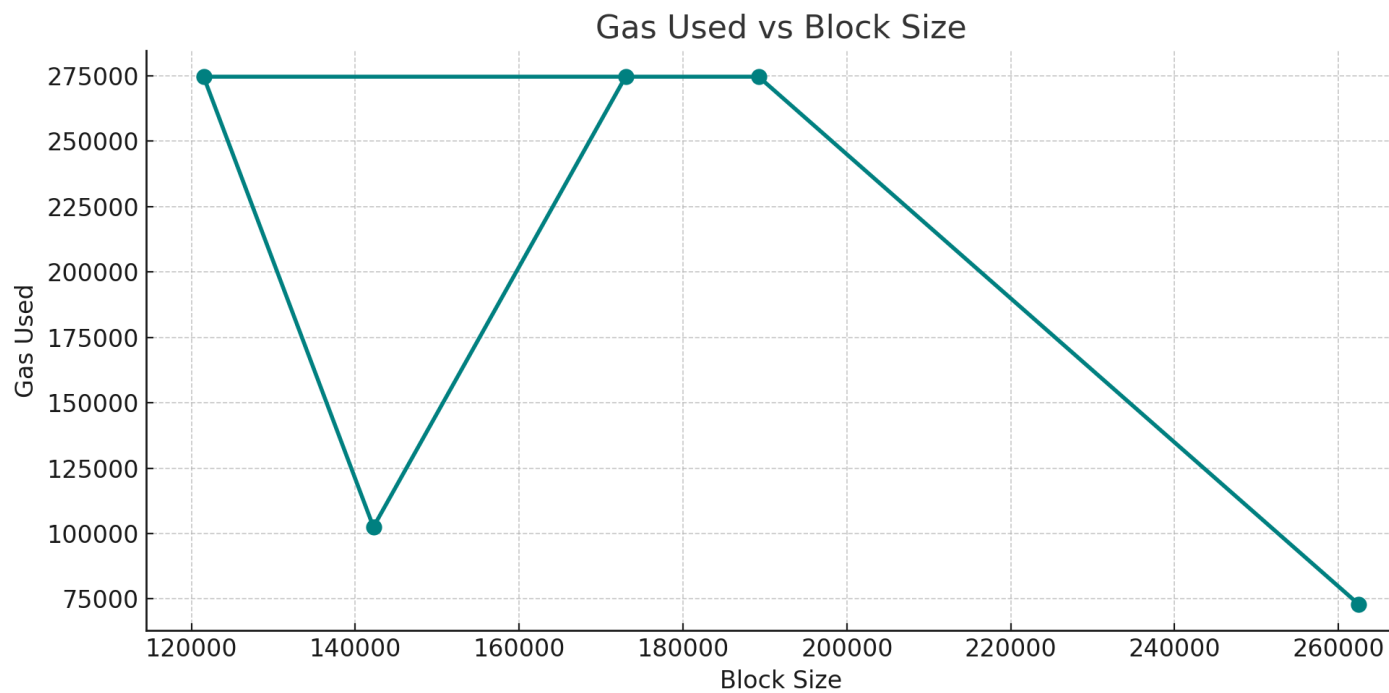


Figure 6.3: Graph of Gas Used vs Block Size

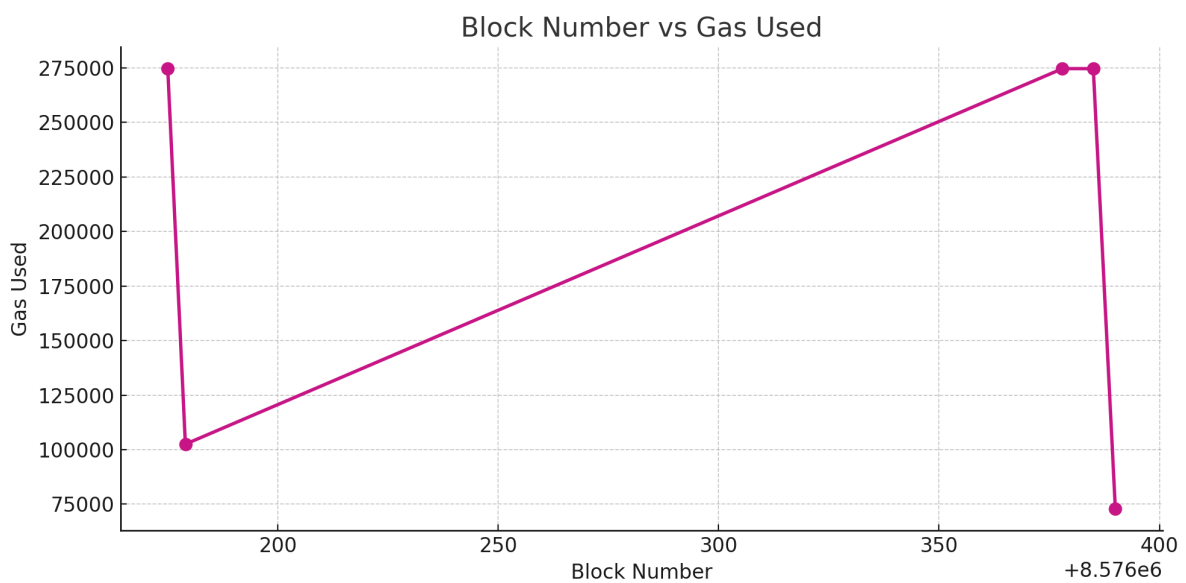


Figure 6.4: Transaction History of Metamask for Contract Deployment

## Chapter 7

# Conclusion and Future Work

The growing need for secure and privacy-preserving management of sensitive personal medical data (SPMD) motivated the design and development of this project: “Smart IoT-Blockchain Security to Secure Sensitive Personal Medical Data Using Shuffled Random Starvation Link Encryption (SRSLE)”.

**The project implemented a two-level security architecture:**

- The first layer of encryption is user-provided (e.g., via AES or other methods prior to upload).
- The second layer applies keccak256 hashing of sensitive fields (Disease, Diagnosis, Lab Results), providing a tamper-evident and irreversible verification mechanism. This is inspired by the SRSLE architecture from current research.

**Key functionality was validated through testing:**

- PHR records were securely stored on-chain.
- Ownership verification was enforced—only the wallet address that created the record could read, update or encrypt it.
- Attempts at unauthorized access were prevented and immutably recorded.
- All transactions — storage, encryption, decryption — were visible to the user along with gas consumed and blockchain metadata.
- Event logging (PHRStored, Encrypted, Decrypted, AccessAttempt) was fully transparent.
- Excel export enabled users to simply analyze transaction history.

The findings demonstrate that blockchain-powered PHR systems can offer a secure, valid and user-centric substitute for conventional centralized EHR systems. Patients have more control over their personal medical information and are able to observe how their records are utilized — a significant leap toward privacy-first digital health.

### 7.0.1 Future work

**The several opportunities for further enhancement is:**

1. **Full SRSLE Algorithm:** The project at the moment employs keccak256 hashing as an easy implementation. Implementing the entire Shuffled Random Starvation Link Encryption process would add more robust security particularly for large-scale healthcare data.
2. **Off-chain Storage via IPFS:** All data now is currently stored directly on-chain within PHR. Integrating IPFS (InterPlanetary File System) for encrypted file storage, with only hash references stored on-chain, would drastically lower gas expenses and enhance scalability.
3. **AI-driven Dynamic Sensitivity Classification:** The system presently employs a static QBSAI-inspired classification for sensitive fields. Incorporating AI or NLP-driven sensitivity detection would enable the system to automatically detect sensitive information from various PHR formats — enhancing flexibility and usability.
4. **Role-Based Access Control (RBAC):** Using RBAC would enable data owners to give controlled access to medical providers, researchers, or relatives — with specific permissions and time expirations. This is a crucial feature for real-world usability in clinical settings.
5. **Enhanced User Interface** The frontend is working but minimal. Upgrading to a new UI framework like React.js or Bootstrap would enhance usability and make the system easier to use for healthcare providers and patients.
6. **Mainnet Deployment and Gas Optimization:** Subsequent releases can be deployed on the Ethereum mainnet or similar L2 implementations (e.g., Polygon) to test real-world scalability and further optimize gas usage.

## References

1. K. Riad et al., "Sensitive and Energetic Access Control (SEAC) to handle PHR stored on cloud servers," 2019.
2. sun2022 Y. Sun et al., "PMMRSS approach for safe sharing of healthcare information across IoT," 2022.
3. ray2021 P. P. Ray et al., "BloTThR PHR management: A secure data-sharing plan that protects privacy," 2021.
4. akkaoui2020 R. Akkaoui et al., "EdgeMedichain: Safeguarding sensitive data during medical PHR exchange on a centralized platform," 2020.
5. zhang2021 Y. Zhang et al., "Efficient identity-based distributed decryption scheme for secure PHR sharing," 2021.
6. mayer2021 Mayer et al., "Blockchain with Fog Computing for PHR data exchange and enhanced security," 2021.
7. zaabar2021 Zaabar et al., "Health Block: Secure remote patient monitoring and EHR management," 2021.
8. gohar2022 A. N. Gohar et al., "PCHHF: Patient-centric healthcare framework using blockchain for cloud-based data exchange," 2022.
9. haddad2022 A. Haddad et al., "A brief review of digitally stored healthcare management systems," 2022.
10. ghavat2022 Ghavat et al., "SoLiD: Socially Linked Data approach for IoT-based secure PHR exchange," 2022.
11. M. Kumari Kala, M. Priya (2024). DCAP: Smart IoT-Blockchain Security to secure Sensitive Personal Medical Data using Shuffled Random Starvation Link Encryption DOI: 110.1109/ACCESS.2024.3483437
12. Wang, Y., Zhang, A., Zhang, P., Qu, Y., Yu, S. (2021). Security-aware and privacy-preserving personal health record sharing using consortium blockchain. *IEEE Internet of Things Journal*, 9(14), 12014-12028. DOI: 10.1109/JIOT.2021.3132780.
13. Blockchain and its Applications By Prof. Sandip Chakraborty, Prof. Shamik Sural — IIT Kharagpur— NPTEL
14. Karan Parekh Sudeep Tanwar and Richard Evans. Blockchain-based electronic healthcare record system for healthcare 4.0 applications. *Journal of Information Security and Applications*, 50(2), 2020.
15. P. P. Ray, B. Chowhan, N. Kumar and A. Almogren, "BIOTHR: Electronic Health Record Servicing Scheme in IoT-Blockchain Ecosystem," in *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10857-10872, 1 July 1, 2021. DOI: 10.1109/JIOT.2021.3050703.