Design of Blockchain-Enabled Secure Real Life Applications

A thesis submitted to Indian Statistical Institute
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

by

Debendranath Das
Senior Research Fellow

Under the joint supervision of

Dr. Sushmita Ruj
University of New South Wales, Sydney, Australia
&
Prof. Subhamoy Maitra
Indian Statistical Institute, Kolkata, India

Cryptology and Security Research Unit
Indian Statistical Institute
Kolkata, India

December, 2024

Dedicated to the Almighty

Declaration of Authorship

I, **Debendranath Das**, declare that this thesis titled, "Design of Blockchain-Enabled Secure Real Life Applications" and the work presented in it is the original work done by me. I confirm that:

– I have completed this work while pursuing my Ph.D. program in Computer Science at the Indian Statistical Institute, Kolkata. This thesis has not been submitted anywhere else for the award of any degree.

– Whenever I have referred to the work of others, I have always provided clear attribution and reference.

– I have acknowledged all the main sources of help.

– In the case of joint works with other co-authors, I have obtained a no-objection certificate from each co-author to include the work in my thesis. Each co-author has acknowledged my adequate contribution to merit the inclusion of the joint work in my thesis.

Date: 26.12.2024

Signature: Debendranath Das

DEBENDRANATH DAS
Cryptology and Security Research Unit
Indian Statistical Institute
203 Barrackpore Trunk Road
Kolkata - 700108
West Bengal, India.

List of Publications

Following is the list of publications/manuscripts that are included in this thesis. Chapter 3 is based on the paper **C1**. Chapter 4 is based on the paper **J1**. Chapter 5 is based on manuscript **J2**; Chapter 6 is based on paper **C2**, and Chapter 7 is based on paper **C3**.

Journals:

J1. **Debendranath Das**
“*BISECTION: BlockchaIn-enabled SECure healTh Insurance prOcessiNg.*”
International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Vol. 46, No. 1, PP. 44-63, (2024)
DOI: https://doi.org/10.1504/IJAHUC.2024.138744

J2. **Debendranath Das**, Sushmita Ruj, and Subhamoy Maitra
“*SecureVAX: A Blockchain-Enabled Secure Vaccine Passport System.*”
Communicated to a Journal
Preprint: https://doi.org/10.48550/arXiv.2407.13852

Conferences:

C1. **Debendranath Das**, Amudhan Muthaiah, and Sushmita Ruj
“*Blockchain-Enabled Secure and Smart Healthcare System.*”
International Conference on Design Science Research in Information Systems and Technology - DESRIST (2022)
Lecture Notes in Computer Science, Vol. 13229. PP. 97-109, Springer
DOI: https://doi.org/10.1007/978-3-031-06516-3_8

C2. **Debendranath Das**, and Atosi Das
“*Blockchain-Enabled Distributed Payment Card Tokenization System.*”
IEEE India Council International Subsections Conference - INDISCON (2024)
IEEE Xplore
DOI: https://doi.org/10.1109/INDISCON62179.2024.10744200

C3. **Debendranath Das**
“*Blockchain-enabled Multicurrency Supported Distributed e-Banking System.*”
International Conference on Advanced Communication and Intelligent Systems - ICACIS (2024) [Accepted. To Appear in *Springer CCIS Series*]

Acknowledgements

Words cannot fully capture my gratitude for the incredible support I have received throughout this Ph.D. journey. This experience has been more than just earning a degree; it has been an emotional journey with a great adventure.

I am deeply grateful to my supervisors, Prof. Subhamoy Maitra and Dr. Sushmita Ruj, for their unwavering support, insightful advice, and constant encouragement. Their guidance has been invaluable in conducting my research and completing this thesis. It has been an honor to have such supervisors who have supported me both academically and personally.

I would like to express my appreciation to my course teachers, Prof. Rana Barua, Prof. Mandar Mitra, Dr. Pinakpani Pal, Prof. Sandip Das, Prof. Palash Sarkar, Prof. Mridul Nandi, Dr. Gautam Pal, Dr. Debrup Chakraborty, Dr. Sabyasachi Karati, Dr. Arijit Bishnu, Dr. Sarbani Palit, Prof. Krishnendu Mukhopadhyaya, and Prof. Ansuman Banerjee. I am fortunate to have learned from such esteemed mentors during my Ph.D. coursework.

I am also deeply thankful to my co-authors, Mr. Amudhan Muthaiah and Mrs. Atosi Das, for their valuable contributions to my research work. I am blessed with wonderful friends who have been my pillars of support. Special thanks to Subhra, Abhinav, Bibhuti, Anupam, Shankar, Drimit, Rathindra, Animesh, Jyotirmoy, Debasmita, Sreyosi, Akanksha, Sreejit, Surochita, Manish, and Mohammad for the endless discussions and persistent encouragement. My sincere gratitude to my friends and seniors at ISI Kolkata, including Anirban da, Soumya da, Laltu da, Prabal da, Avijit da, Avik da, Binanda da, Nilanjan da, Anirudhha da, Avishek da, Samir da, Nayana di, Suprita di, Pritam da, Mostaf da, Diptendu da, Partha da, Suprita di, Suman, Gourab, Subha, Rakesh, Chandranan, Panchalika, and many more who have made my time at ISI Kolkata inspiring and enjoyable.

I extend my gratitude to the non-academic staff of ASU, CSRU, the accounts section, and the Dean's office for their assistance with logistics and official work. I also wish to thank my teachers and friends from my alma maters, from my primary school to postgraduate university: South Suburban Junior School, Mitra Institution (Bhowanipur Branch), Kalidhan Institution, Maulana Abul Kalam Azad University of Technology (formerly WBUT - in-house campus), and Jadavpur University.

Last but not least, I express my deepest gratitude to my family - my parents, elder sister, brother-in-law, niece - for their unwavering love, support, and sacrifices. Their faith gave me the strength to pursue academics. Words cannot suffice to repay their contributions.

Finally, I am grateful to all who have directly or indirectly contributed to this work. While I have endeavored to acknowledge everyone, any omissions are entirely unintentional. Your contributions, whether named or unnamed, are deeply appreciated and have been instrumental in shaping this research.

Debendranath Das

Abstract

Blockchain is a distributed, immutable, verifiable append-only ledger maintained across a peer-to-peer network. Bitcoin, proposed by Satoshi Nakamoto in 2008, is the first and most popular application based on blockchain technology, enabling digital transactions using a virtual currency called Bitcoin. In 2015, Ethereum, the next famous permission-less blockchain platform, supported writing smart contracts using a Turing-complete language. This allows developers to build any secure decentralized application on the platform.
In this thesis, we propose and implement several secure protocols applicable to various real-life applications, specifically focusing on the healthcare and banking sectors. These works leverage the power of the Ethereum blockchain, smart contracts, and relevant cryptographic tools. All our prototypes are implemented using the Ethereum platform: the first two were deployed on the Ropsten test network, and the remaining on the Sepolia test network.
Our *first* contribution proposed a blockchain-enabled secure and smart healthcare system, where blockchain integrated with smart contracts builds trust by providing transparency. The system guarantees fairness between the patient and the hospital, as well as the privacy and security of the patient's electronic healthcare records (EHRs). The *second* proposal extends the scope of the first by introducing a novel blockchain-enabled health insurance processing system that streamlines critical operations and ensures fairness for all stakeholders. Various insurance procedures are encoded through smart contracts, increasing transparency and trust in the claim settlement process. We present a secure and privacy-preserving access control policy for sharing EHR with insurance companies during verification to address privacy concerns.
A Vaccine Passport provides proof of vaccination, enabling mobility during pandemics like COVID-19. Challenges include preventing certificate forgery and protecting personal data. As our *third* contribution, we propose a blockchain-enabled vaccine passport system powered by smart contracts, ensuring secure and authentic vaccination certificates, with encrypted passports stored in the distributed InterPlanetary File System for added security.
The current payment card tokenization landscape is dominated by centralized entities like Visa and Mastercard, serving as Token Service Providers (TSPs). Our *fourth* contribution introduces a decentralized blockchain-powered tokenization system, where smart contracts generate tokens based on predefined criteria, ensuring robustness and transparent audit trails. Finally, our *fifth* contribution is an innovative e-banking system leveraging blockchain's decentralized features to support both fiat currencies and cryptocurrencies. Using Ethereum smart contracts, the system enables a single account to hold multiple currencies, with a functional prototype demonstrating its feasibility.

Contents

List of Figures

List of Tables

1 Introduction

"To succeed in your mission, you must have single-minded devotion to your goal."

— **A. P. J. Abdul Kalam**

In an era where digital interactions underpin nearly every aspect of our lives, ensuring their security and integrity is inevitable. Blockchain technology, known for its decentralized and immutable nature, offers a promising solution for creating secure digital systems. This dissertation, titled "Design of Blockchain-Enabled Secure Real Life Applications," embarks on a journey to explore and innovate at the intersection of blockchain technology and the security of practical applications.

Blockchain technology has demonstrated groundbreaking potential across various industries, revolutionizing traditional systems and processes by introducing decentralization, transparency, and security. Key industries benefiting from blockchain include:

– **Healthcare**: It ensures the secure storage and sharing of electronic health records, protects patient privacy, enables transparent tracking of pharmaceuticals, streamlines health insurance claims processing, and enhances interoperability between healthcare providers.

– **Finance**: Blockchain facilitates secure cross-border payments, enhances fraud detection in transactions, automates compliance processes, reduces transaction fees, and improves transparency in corporate governance.

– **Supply Chain Management**: Blockchain provides end-to-end traceability of goods, prevents counterfeit products, ensures compliance with ethical sourcing standards, and streamlines logistics operations.

– **Real Estate**: It simplifies property transactions, maintains tamper-proof property records, prevents ownership disputes, and reduces fraud in real estate dealings.

- **Voting Systems**: Blockchain ensures secure and tamper-proof elections, enables transparent remote voting, and verifies the integrity of election outcomes.
- **Digital Identity Verification**: It creates decentralized and secure digital identities, prevents identity theft, verifies credentials for education and employment, and empowers users with control over personal information.
- **Energy Trading**: Blockchain facilitates decentralized energy trading, enhances transparency in carbon credit systems, improves grid management, and promotes sustainable energy practices.
- **Decentralized Data Marketplaces**: It enables secure data sharing without intermediaries, allows individuals to monetize their data, prevents unauthorized access, and ensures fair compensation through verifiable data ownership.
- **Others**: Blockchain enables transparent tracking of pharmaceuticals in the pharmaceutical industry, supports self-sovereign identity by empowering individuals to manage their digital identities, simplifies intellectual property management, revolutionizes gaming with tokenized assets, enhances transparency in public sector governance, tracks disaster relief donations, streamlines educational certifications, and improves traceability in agricultural supply chains.

These are just a few examples, as the potential use cases of blockchain are vast and continually evolving, with applications emerging across diverse industries and domains. Its key features—decentralization, immutability, and transparency—provide unique opportunities to solve long-standing security issues. However, the path to effectively harnessing blockchain for real-world applications is laden with both technical and conceptual hurdles. In a broader context, while blockchain has a wide range of applications, this thesis focuses on addressing challenges related to fairness, security, and privacy in healthcare, as well as enhancing transparency in banking. In healthcare, issues such as safeguarding electronic health records (EHRs), ensuring patient privacy, and automating processes like insurance claims are critical pain points. In banking, the need for secure multicurrency transaction systems, transparent operations, and secure credit/debit card processes highlight the limitations of traditional centralized systems. This research narrows its scope to these domain-specific problems, proposing blockchain-enabled solutions that bridge the gap between its theoretical advantages and practical applications. By leveraging blockchain's inherent capabilities, this research aims to demonstrate practical solutions tailored to these domains. The proposed approaches seek to bridge the gap between theoretical potential and real-world applicability, addressing key pain points while laying the groundwork for future advancements in blockchain-enabled systems.

Healthcare and online banking are two pillars of modern society and have a profound impact on individuals and the society at large. The healthcare sector safeguards our well-being, while the banking industry underpins our economic stability. The security of digital systems in these fields is critical because breaches can lead to severe consequences, such as identity theft, financial loss, and compromised health records. Our research focuses on designing secure, user-friendly protocols for blockchain-based applications in these two areas by bridging theoretical foundations with practical implementations. Blockchain's inherent properties—such as immutability, decentralization, and transparency—make it uniquely suited for applications requiring trust and secure data management, such as healthcare and banking.

1.1 Healthcare Industry and Digitization

The healthcare industry is experiencing a significant transformation due to the rise of digital technologies, leading to an increase in the average life expectancy of human beings. Digitization has completely changed the way healthcare services are provided and managed. This has brought innovations such as electronic health records (EHRs), telemedicine, wearable health devices, and mobile health applications, all of which have improved the efficiency, accessibility, and quality of healthcare. These advancements have resulted in improved patient outcomes and more streamlined administrative processes. Nonetheless, the industry continues to grapple with significant issues like centralization, unfair practices, and financial exploitation. The COVID-19 pandemic has further highlighted these challenges, exposing inefficiencies in data sharing and coordination and raising concerns about privacy and fairness, especially with the introduction of vaccine passports.

Challenges Faced by the Healthcare Sector

1. **Centralization:** Patient data is often stored in siloed databases controlled by individual hospitals or healthcare providers, leading to inefficiencies and data breaches.

2. **Unfair Practices:** Patients may have difficulty accessing their health records, and disparities in the quality of care provided to different populations can exist.

3. **Financial Exploitation:** High healthcare costs and a lack of transparency in pricing policy can lead to financial burdens for patients.

4. **Data Privacy and Security:** Ensuring the privacy and security of sensitive health information is a major concern, particularly with the increasing digitization of records.

5. **Interoperability Issues:** Different healthcare systems often lack the ability to share and integrate data, leading to fragmented care effectively.

6. **Inefficiencies in Data Sharing:** Slow or inadequate sharing of data between providers can hinder patient care and public health responses.

7. **Equity and Access:** Not all patients have equal access to digital health technologies, creating disparities in healthcare delivery.

8. **Regulatory and Compliance Challenges:** Navigating complex regulatory environments can be difficult for healthcare providers and innovators.

9. **Adoption of New Technologies:** Resistance to change and the high cost of implementing new technologies can slow down the adoption of beneficial innovations.

10. **Emergency Response Preparedness:** The COVID-19 pandemic highlighted weaknesses in the healthcare system's ability to respond to sudden, large-scale health crises.

The question now is whether we could design a framework that addresses all the above challenges, improving transparency, securing data privacy, and ensuring equal access to healthcare services. Before we move on to discuss potential solutions to our problem, we discuss the online banking scenario and certain challenges in this area.

1.2 Online Banking and Payment Systems

Online banking has revolutionized the financial industry by providing customers with convenient access to banking services through digital platforms. It allows users to perform a variety of transactions, such as transferring funds, paying bills, and managing accounts, without needing to visit a physical bank branch. Advancements in Internet technology, mobile applications, and secure online payment systems have fueled the growth of online banking. Despite the convenience and efficiency of online banking, the centralized nature of traditional banking systems presents several significant drawbacks.

Several high-profile incidents have exposed the weaknesses of centralized data storage and IT systems. For instance, the 2017 Equifax data breach [167] compromised the personal information of 147 million people, and the 2018 TSB IT meltdown [15] left 1.9 million customers without access to their accounts. These events, along with the 2008 global financial crisis, emphasize the risks associated with institutions labelled "too big to fail", which required significant taxpayer-funded bailouts. The 2016 Bangladesh Bank heist [169] illustrated the security vulnerabilities in centralized financial systems, as hackers exploited weaknesses in the SWIFT network to steal $101 million. Additionally, the 2022 data breach at Flagstar Bank compromised the personal information of over 1.5 million customers [32]. These examples collectively underline the potential for widespread disruption, data breaches, and systemic financial risks inherent in centralized banking structures.

Next, we will discuss the drawbacks of the current banking system.

Drawbacks of Traditional Banking System

1. **Limited Fault Tolerance:** Centralized banking systems have poor fault tolerance. If a central server or database is compromised, it can result in widespread service disruptions and expose sensitive data to breaches.

2. **Security Risks:** Centralized systems are prime targets for cyberattacks. Hackers often focus on these systems due to the high volume of sensitive data stored in one place.

3. **Lack of Transparency:** Centralized control can lead to a lack of transparency in how transactions are processed and how fees are applied. This can result in mistrust among customers.

4. **Limited Accessibility:** Centralized banking systems may not be accessible to everyone, especially in regions with limited banking infrastructure or where people do not have easy access to bank branches or reliable Internet connections.

5. **Problems in Cross-Currency Trading:** Cross-currency trading, or foreign exchange (forex) trading, involves exchanging one currency for another in the global market. Several challenges, including high transaction costs, slow transaction time, exchange rate volatility, and complex regulatory and compliance requirements, complicate this process. These issues can significantly impact international trade, business operations, and individual remittances, making cross-currency trading a cumbersome and costly endeavor.

1.3 A Decentralized World

The shift towards a decentralized world is reshaping the foundations of various industries, addressing long-standing inefficiencies and injustices. Blockchain technology is at the heart of this transformation. It offers a strong solution to many issues found in centralized systems. Blockchain helps make things fairer, more secure, and more open. This makes it a powerful tool for improving various real-life problems, including healthcare and online banking, which have struggled with these issues for a long time.

Blockchain is a decentralized, immutable digital ledger that records transactions in a tamper-proof way across a distributed peer-to-peer network. Each block is cryptographically linked to the previous one, forming a chain of blocks that ensures data integrity. Blockchain validates new transactions through distributed consensus mechanisms, such as Proof of Work (PoW) or Proof of Stake (PoS), creating a transparent, tamper-resistant record of all transactions.

Blockchain technology gained significant recognition following the publication of a groundbreaking research paper by Satoshi Nakamoto in 2008 titled "Bitcoin: A Peer-to-Peer Electronic Cash System"[110]. Although initially associated with cryptocurrencies such as Bitcoin, the introduction of Ethereum in 2015 [25], which supports complete smart contracts, has highlighted the far-reaching potential of blockchain technology in diverse industries beyond just digital currencies.

The Distinct Benefits of Blockchain

Blockchain technology offers numerous advantages over traditional technologies, making it a superior choice for many applications:

– **Decentralization**: Unlike traditional centralized systems, blockchain operates on a distributed peer-to-peer network of nodes. This decentralization eliminates single points of failure, reduces the risk of data breaches, and ensures system robustness.

– **Immutability**: Once data is recorded on the blockchain, it cannot be altered or deleted in practice (although theoretically possible). This immutability ensures the integrity of records and prevents unauthorized tampering.

– **Transparency**: Blockchain's ledger is open and transparent to all participants in the network. Each node has a copy of the entire blockchain, making transaction histories fully visible and auditable. This transparency builds trust among stakeholders by providing a verifiable trail of all transactions.

– **Security**: Blockchain uses advanced cryptographic techniques, such as SHA-256 hashing, to secure data. Each block is linked to the previous one through a cryptographic hash, making it extremely difficult for malicious actors to alter the blockchain.

– **Verifiability**: Information stored on the blockchain is decentralized, allowing everyone to verify the correctness of data.

– **Open Access**: Blockchain is accessible to anyone, allowing any individual to participate in the network without requiring permission, considering a permission-less model. This open access encourages widespread participation and innovation.

– **Availability**: Transaction records are stored across multiple nodes in the decentralized network. This redundancy ensures that data is not lost and remains available even if some nodes fail.

– **Censorship Resistance**: Blockchain is free from censorship because it does not rely on control by any single party. Trustworthy nodes validate transactions through consensus protocols, ensuring fairness and preventing unilateral control.

– **Smart Contracts**: Blockchain supports programmable smart contracts, which are self-executing contracts with the terms directly written into code. These contracts automate processes, reduce the need for intermediaries, and ensure fairness and efficiency.

– **Efficiency**: Blockchain removes third-party intermediaries, reducing errors and speeding up transactions. This efficiency makes settlement processes smoother and faster.

– **Cost Reduction**: By eliminating the need for intermediaries, blockchain reduces transaction costs for businesses and builds trust between partners through improved transparency.

– **Resilience**: The decentralized nature of blockchain makes it highly resilient to attacks. Even if some nodes in the network fail or are compromised, the rest of the network can continue to operate seamlessly.

1.3.1 Impact of Blockchain on Healthcare Sector

In Section 1.1, we discussed the numerous challenges confronting today's healthcare system. Blockchain technology offers a life-changing solution by addressing many of these issues to a significant extent, although regulatory and compliance challenges persist. By ensuring the integrity and immutability of medical records, blockchain enhances data security and privacy, safeguarding sensitive patient information from unauthorized access and tampering. Its decentralized nature fosters transparency throughout healthcare processes, thereby cultivating trust among patients, providers, and insurers. Moreover, blockchain facilitates efficient data sharing and interoperability among healthcare entities, enhancing care coordination and breaking down data silos. Through the automation capabilities of smart contracts, healthcare operations are streamlined, leading to reduced administrative costs and improved efficiency, particularly in insurance claim processing. The technology's transparency and immutability also strengthen fraud prevention measures, making it difficult to manipulate medical records, insurance claims, or vaccination certificates. Furthermore, blockchain empowers patients by granting them greater control over their medical data, enabling secure management and authorization of access. In public health, blockchain aids in tracking vaccination statuses and optimizing pandemic response management, highlighting its potential to revolutionize healthcare delivery and management.

1.3.2 Impact of Blockchain on Banking Sector

While Bitcoin has addressed several major problems associated with the centralized banking system, additional issues persist. For instance, traditional banks do not support the simultaneous holding of multiple currencies, limiting users' flexibility in managing their assets. Although cryptocurrencies have been invented to provide an alternative, many people remain unfamiliar with their usage and prefer fiat currencies. This creates a gap between the worlds of crypto and fiat, which has yet to be effectively bridged. Blockchain technology, however, presents a solution. By enabling the integration of multiple currencies into a single banking platform, blockchain can enhance the convenience of managing diverse assets. Its cryptographic features provide a higher level of protection for transactions against unauthorized access and cyberattacks, surpassing the security of traditional systems. The decentralized nature of blockchain ensures greater availability with zero system downtime, as the network is not reliant on a single point of failure. Smart contracts offer the potential to automate and streamline various banking functionalities, increasing efficiency and reducing human error. Furthermore, blockchain facilitates direct peer-to-peer transactions, reducing the need for intermediaries and consequently lowering transaction fees. Additionally, in the realm of payment card usage—such as credit and debit cards—blockchain can enhance security through tokenization. By removing centralized token service providers and employing decentralized mechanisms, blockchain ensures a higher level of security for card transactions. These advancements underscore the metamorphic potential of blockchain in creating a more integrated, secure, and efficient banking system.

1.4 Contributions and Organization of the Thesis

This research has primarily focused on designing blockchain-enabled secure real-life applications, concentrating on the healthcare and banking sectors. The rest of the chapters of the thesis are organized as follows: In **Chapter 2**, we have discussed the notations, cryptographic primitives, and background needed to understand the thesis. Related work is discussed within the relevant chapters. Our first three contributory chapters focus on blockchain's impact in the healthcare domain, while the latter two chapters concentrate on its impact on the banking sector. The contributions of this thesis are summarized in the remaining chapters, as outlined below.

***Contribution 1*:** **Blockchain-Enabled Secure Healthcare System**
In **Chapter 3**, we focus on developing a blockchain-enabled patient-centric hospital management system that addresses key challenges in healthcare. This system ensures fairness by protecting honest parties from financial loss, even in the presence of malicious actors. It enhances privacy by requiring patient consent for data access and protects data security by preventing tampering. The proposed secure and smart healthcare system coordinates interactions between patients and hospitals, ensuring that patient data is only accessible with permission and cannot be altered by unauthorized parties. It prevents overcharging and mandates timely treatment initiation by hospitals. Patients can choose to store their data either personally or in a semi-trusted medical database, with mechanisms in place to verify data authenticity.

***Contribution 2*:** **Blockchain-Enabled Secure Health Insurance Processing System**

A health insurance policy is crucial for providing essential medical coverage during emergencies. However, the health insurance sector faces challenges such as delayed claim processing, high administrative costs, and trust issues. In **Chapter 4**, we propose a transformative solution using blockchain technology to address these existing health insurance issues effectively. Our solution introduces a novel blockchain-enabled health insurance processing system that streamlines critical operations and ensures fairness for all stakeholders. Various insurance procedures are encoded through smart contracts, improving transparency and trust in the claim settlement process. We also present a secure and privacy-preserving access control policy for sharing electronic healthcare records (EHR) with insurance companies during verification to address privacy concerns.

***Contribution 3*: Blockchain-Enabled Secure Vaccine Passport System**
A vaccine passport serves as documentary proof, providing passport holders with a greater degree of freedom while roaming around during pandemics. It confirms vaccination against certain infectious diseases like COVID-19, Ebola, and flu. The key challenges faced by the digital vaccine passport system include passport forgery, unauthorized data access, and inaccurate information input by vaccination centers. Privacy concerns must also be addressed to ensure that the user's personal identification information (PII) is not compromised. Additionally, it is necessary to track vaccine vials or doses to verify their authenticity, prevent misuse and illegal sales, as well as to restrict the illicit distribution of vaccines. To address these challenges, we propose a blockchain-enabled secure vaccine passport system, in **Chapter 5**, leveraging the power of smart contracts. Our solution integrates off-chain and on-chain cryptographic computations, facilitating secure communication among various entities. We have utilized the InterPlanetary File System (IPFS) to store encrypted vaccine passports of citizens securely. Our prototype is built on the Ethereum platform, with smart contracts deployed on the Sepolia Test network, allowing for performance evaluation and validation of the system's effectiveness. By combining IPFS as a distributed data storage platform and Ethereum as a blockchain platform, our solution paves the way for secure, efficient, and globally interoperable vaccine passport management, supporting comprehensive vaccination initiatives worldwide.

***Contribution 4*: Blockchain-Enabled Secure Payment Card Tokenization System**
Payment card tokenization is a process that replaces sensitive card (Credit/Debit) information with a unique identifier

or token. This token is used for transaction processing, reducing the risk of unauthorized access and fraud. However, the current landscape is dominated by a few centralized entities, such as Visa, Mastercard, functioning as Token Service Providers (TSP). To address this, we propose a blockchain-powered decentralized tokenization system, in **Chapter** 6, where a smart contract generates tokens based on predefined criteria. Decentralization ensures the system's robustness against attacks and offers transparent audit trails. Our prototype is built on the Ethereum platform, with smart contracts deployed on the Sepolia test network, allowing for performance evaluation and validation of the system's efficacy.

***Contribution 5*: Blockchain-Enabled Multi-currency Supported Distributed e-banking System**
Blockchain technology has catalyzed a paradigm shift in the banking and financial sectors. In **Chapter** 7, we introduce an innovative e-Banking system leveraging blockchain's decentralized capabilities to create a hybrid platform accommodating both traditional fiat currencies and cryptocurrencies. The traditional banking system is susceptible to fraud as it is controlled by a central or consortium of power entities. Motivated by the limitations of traditional banking systems, we present a novel approach. Our system utilizes Ethereum's smart contracts to enable a bank account to hold multiple fiat currencies and cryptocurrencies simultaneously. The model boasts unique attributes: compatibility with various fiat currencies, decentralized banking operations, real-time currency exchange, and a user-friendly web application interface. We have developed a fully functional prototype of our proposed e-Banking system, demonstrating the system's capabilities, and experimental results establish its practicality. This research advances decentralized finance by bridging the gap between conventional financial systems and the emerging cryptocurrency world.

Finally, in **Chapter** 8, we conclude the thesis with a brief overview of our work, highlighting key contributions and discussing unresolved issues requiring further exploration in blockchain technology.

Through case studies and experiments, this dissertation demonstrates how blockchain enhances security, mitigates risks, and fosters trust in digital systems. In an era of growing digital threats, robust security is crucial. This work highlights blockchain's transformative potential in securing real-life applications, paving the way for a safer digital future.

2 Preliminaries and Background

“When it comes to privacy and accountability, people always demand the former for themselves and the latter for everyone else.”

— **David Brin**

This chapter defines important notations and terminology, provides formal definitions of various cryptographic concepts, and presents relevant background information for this thesis. Additional notations and tools specific to each chapter are detailed within their respective chapters.

2.1 Notations used in the thesis

The common notations or acronyms used in this thesis are listed in Table 2.1.

Table 2.1: General Notations/Acronyms used in this Thesis

Notations/Acronyms	**Description**
$ASKE$	Asymmetric Key Encryption
SKE	Symmetric Key Encryption
K	Shared Common Key in SKE setup
$< PK_e, SK_e >$	Public Key and Secret Key of an entity e in ASKE setup
$H(x)$ or $MD(x)$	Hash or Message Digest of x
$commit_x$ or $commit(x)$	Commitment of x
$MR(y)$	Merkle Root Computation over a set y
PRE	Proxy Re-encryption Scheme
$RK_{a \to b}$	Re-encryption Key, Delegator a and Delegatee b
BC	Blockchain
SC	Smart Contract
BTC	Bitcoin
ETH	Ether
$IPFS$	Interplanetary File System
T_{Event}	Timestamp, when the Event occurs

2.2 Cryptographic Preliminaries

The majority of the cryptographic-related concepts discussed in this chapter are borrowed from the textbooks authored by 1. Boneh & Shoup [20], 2. Katz & Lindell [82], 3. Stinson & Paterson [150], and 4. Stallings [149].

2.2.1 Encryption Scheme

Encryption is the process of converting an original plaintext message into a meaningless, unreadable ciphertext using an algorithm and a key. Decryption is the reverse process of encryption that converts the ciphertext back into plain text. Both encryption and decryption must co-exist for a working system [29]. Encryption takes place at the sender's end before transmitting confidential information over an insecure network, while decryption occurs at the receiver's end. The purpose of encryption is to ensure that information is only accessible to authorized parties.

– **Types of Encryption Scheme**:

1. **Symmetric Key Encryption (SKE)**: A symmetric key encryption scheme enables two communicating parties to encrypt and decrypt messages securely using the same key, which was shared apriori. Symmetric key encryption is also known as private key encryption. This system comprises three essential algorithms: 1. Key generation algorithm, 2. Encryption algorithm, and 3. Decryption algorithm. This encryption scheme is associated with three sets: the keyspace (K), the message space (M), and the ciphertext space (C).

   Here is how the algorithms work:

   – **KeyGen**(1^n) $\to k$: It takes a security parameter n as input and generates a secret key k belonging to K.

   – **Enc**(k, m) $\to c$: It takes a secret key k from K and a message m from M as inputs, producing a ciphertext c in C.

   – **Dec**(k, c) $\to m$: It uses a secret key k from K and a ciphertext c from C to output a message m in M.

   An encryption scheme is considered correct if, for all messages m in M and keys k in K, decrypting a ciphertext with the same key used for encryption yields the original message: $\mathrm{Dec}(k, \mathrm{Enc}(k, m)) = m$. However, it is important to note that even if an encryption scheme is correct, it may not necessarily be secure.

   There are two types of symmetric key encryption algorithms (or ciphers): **stream cipher** and **block cipher**. A block cipher divides the plaintext data into blocks (often 64-bit blocks, but newer algorithms sometimes use 128-bit blocks) and encrypts the data one block at a time. On the other hand, stream ciphers encrypt the data as a continuous stream of bits, encrypting one bit at a time.

- **Modes of Operation**: Encryption algorithms often use different modes of operation to enhance security and provide additional functionalities:
  - **ECB (Electronic Codebook)**: Each block of plaintext is encrypted independently. Simple but vulnerable to pattern attacks.
  - **CBC (Cipher Block Chaining)**: Each block of plaintext is XORed with the previous ciphertext block before being encrypted. Provides better security than the ECB.
  - **CFB (Cipher Feedback)**: Converts a block cipher into a self-synchronizing stream cipher. Suitable for encrypting data in real time.
  - **OFB (Output Feedback)**: Converts a block cipher into a synchronous stream cipher, avoiding the propagation of errors.
  - **CTR (Counter)**: Turns a block cipher into a stream cipher. Each block is combined with a counter value, which is incremented for each subsequent block.
- **Strengths**: Generally, symmetric key encryption is faster than asymmetric key encryption and suitable for encrypting large amounts of data.
- **Weaknesses**: Securely sharing the secret key between parties can be difficult, especially over long distances or insecure channels. If the key is intercepted during transmission, the encryption is compromised. Also, key management is challenging. As the number of users increases, the number of keys required grows exponentially. For n users, $\frac{n(n-1)}{2}$ unique keys are needed, which can become unmanageable.
- **A few examples of symmetric key encryption algorithms**: **Stream Ciphers**:
  - **RC4**:
    - RC4 is one of the most well-known stream ciphers [64].
    - RC4 generates a keystream by initializing a state array using the key-scheduling algorithm (KSA), and then produces a pseudorandom stream of bits using the pseudorandom generation algorithm (PRGA).
    - **Usage**: Widely used in various protocols such as WEP and SSL/TLS.
  - **Salsa20**:
    - Salsa20 is another stream cipher known for its performance and security [18].
    - The core function maps a 256-bit key, a 64-bit nonce, and a 64-bit counter to a 512-bit block of the key stream (a Salsa version with a 128-bit key also exists).
    - **Usage**: Salsa20 is used in various modern applications where efficient and secure encryption is needed.

Block Ciphers:

- **AES (Advanced Encryption Standard)**:
  - **Key Sizes**: 128, 192, or 256 bits.
  - **Block Size**: 128 bits.
  - **Usage**: Widely used in various applications, including securing sensitive data and in SSL/TLS for web security [37].
- **DES (Data Encryption Standard)**:
  - **Key Size**: 56 bits.
  - **Block Size**: 64 bits.
  - **Usage**: Historically used in various applications, now largely considered obsolete due to its vulnerability to brute force attacks [121].
- **3DES (Triple DES)**:
  - **Key Sizes**: 112 or 168 bits.
  - **Block Size**: 64 bits.
  - **Usage**: An enhancement of DES used in financial services and other industries for data encryption [36].
- **Blowfish**:
  - **Key Sizes**: 32 to 448 bits.
  - **Block Size**: 64 bits.
  - **Usage**: Used in various applications, including encryption for data storage and file encryption [113].

2. **Asymmetric Key Encryption (ASKE)**: In an asymmetric key encryption scheme, each involved party possesses a pair of keys: a public key and a secret key. The public key is publicly available and known to all, while the secret key is kept private and known only to the key owner. When sending a message, the sender encrypts it using the recipient's public key, ensuring that only the recipient, who has the corresponding secret key, can decrypt the message. This preserves the confidentiality of the communication. The public and secret keys are mathematically linked, making it computationally hard to deduce the secret key from the public key alone.

   This scheme consists of three algorithms: one for generating the key pair, one for encrypting a message using the public key, and one for decrypting a ciphertext using the secret key. Three sets—the keyspace K, the message space M, and the ciphertext space C—are associated with an asymmetric encryption scheme.

   The three algorithms have the following input/output behavior:

   – **KeyGen**$(1^n) \rightarrow (pk, sk)$: Takes a security parameter n as input and outputs a key pair (pk, sk), where $pk, sk \in K$.

   – **Enc**$(pk,\ m) \rightarrow c$: Takes a public key $pk \in K$ and a message $m \in M$ as inputs, and produces a ciphertext $c \in C$.

- **Dec**$(sk, c) \rightarrow m$: Takes a secret key $sk \in K$ and a ciphertext $c \in C$ as inputs, and outputs a message $m \in M$.

An asymmetric key encryption scheme is considered correct if, for all messages $m \in M$ and any key pair $(pk, sk) \in K$, decrypting an encrypted message with the corresponding secret key returns the original message: $\text{Dec}(sk, \text{Enc}(pk, m)) = m$.

- **Strengths**: Key sharing is simpler and more secure since the public key can be openly shared. Additionally, key management is easy. For n persons to communicate using asymmetric key encryption, the number of keys required is $2n$.
- **Weaknesses**: It is slower than symmetric encryption and is usually not used for encrypting large amounts of data directly.
- **A few examples of asymmetric key encryption algorithms**:
  * **RSA (Rivest-Shamir-Adleman)**:
    · **Key Sizes**: Typically, 2048 or 4096 bits.
    · **Strength**: Based on the difficulty of factoring large integers.
    · **Usage**: Often used for secure data transmission, digital signatures, and key exchange [141].
  * **ECC (Elliptic Curve Cryptography)**:
    · **Key Sizes**: Typically, 256, 384, or 521 bits.
    · **Strength**: Based on the algebraic structure of elliptic curves over finite fields.
    · **Usage**: Increasingly used in mobile devices and systems with limited computational resources due to its efficiency [21].
  * **ElGamal**:
    · **Key Sizes**: Variable, typically large primes.
    · **Strength**: Based on the difficulty of computing discrete logarithms.
    · **Usage**: Used for encryption and digital signatures, particularly in PGP (Pretty Good Privacy) [160].

- **Applications of Encryption**:
  - **Data at Rest**: Protecting stored data on devices and storage systems.
  - **Data in Transit**: Securing data transmitted over networks.
  - **Secure Communication**: Enabling confidential communication between parties.

2.2.2 Hash Function

Hash functions are fundamental cryptographic primitives used in various applications to ensure data integrity, provide unique identifiers called digest, and enable efficient data retrieval [38]. A hash function $H : \{0,1\}^* \to \{0,1\}^n$ is a one-way function, where $\{0,1\}^*$ represents inputs of arbitrary length and $\{0,1\}^n$ represents fixed-length outputs. This function takes an input x of any size and produces a fixed-size output y such that $H(x) = y$. Hash functions are one-way functions, meaning that it is computationally hard to reverse the process and retrieve the original input x from its hash value y.

- **Properties of a Cryptographic Hash Function**: A secure cryptographic hash function should have the following properties:
  - **Deterministic**: The same input x will always produce the same output y.
  - **Fast Computation**: The hash function should be able to process input quickly and efficiently.
  - **Preimage Resistance**: Given y, it should be computationally infeasible to find x such that $H(x) = y$.
  - **Second Preimage Resistance**: Given x and $H(x)$, it should be computationally infeasible to find $x' \neq x$ such that $H(x') = H(x)$.
  - **Collision Resistance**: It should be computationally infeasible to find any two distinct inputs x and x' such that $H(x) = H(x')$.
  - **Avalanche Effect**: A small change in the input x should produce a significantly different hash output y, making the hash function highly sensitive to input variations.
- **A few examples hash functions:**
  - **MD5 (Message Digest Algorithm 5)**:
    * **Output Size**: 128 bits.
    * **Usage**: Historically used for checksums and data integrity verification. Due to vulnerabilities, it is not recommended for cryptographic security.
  - **SHA-1 (Secure Hash Algorithm 1)**:
    * **Output Size**: 160 bits.
    * **Usage**: Previously widely used in digital signatures and certificates. Now considered insecure due to collision vulnerabilities.
  - **SHA-256 (Secure Hash Algorithm 256)**:
    * **Output Size**: 256 bits.
    * **Usage**: Part of the SHA-2 family, widely used for secure hashing in various security applications, including SSL/TLS, digital signatures, and blockchain.
  - **SHA-3 (Secure Hash Algorithm 3)**:
    * **Output Size**: Variable (commonly 224, 256, 384, or 512 bits).

    * **Usage**: The latest member of the Secure Hash Algorithm family, designed as an alternative to SHA-2 with enhanced security properties.

  - **RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest)**:
    * **Output Size**: 160 bits.
    * **Usage**: Used in various security applications, particularly in PGP and cryptocurrencies.
  - **Keccak**:
    * **Output Size**: Variable (commonly 224, 256, 384, or 512 bits).
    * **Usage**: The basis for the SHA-3 standard, Keccak is used in a wide range of cryptographic applications due to its robust security properties.

- **Applications of Hash Functions**

  1. **Data Integrity**: Hash functions are used to ensure data integrity by generating a fixed-size hash (digest) of data. Any changes to the data result in a different hash value, alerting to potential tampering or corruption.
  2. **Digital Signatures**: In digital signatures, hash functions are used to create a digest of a message. The digest is then signed with a private key, allowing verification of both the integrity and authenticity of the message using the corresponding public key.
  3. **Password Storage**: Hash functions securely store passwords by hashing them before storage. During authentication, the hash of the entered password is compared to the stored hash, ensuring that even if the hash is compromised, the original password remains protected.
  4. **Blockchain**: Hash functions play a crucial role in blockchain technology by linking blocks of transactions. Each block includes a hash of the previous block, forming a chain. This ensures the immutability and integrity of the entire blockchain.
  5. **Message Authentication Codes (MACs)**: Hash functions create MACs, which are tags added to messages to ensure their authenticity and integrity. The MAC is generated using a shared secret key, verifying that the message has not been altered.
  6. **File and Data Identification**: Hash functions are used to generate unique identifiers (hashes) for files or data sets. These identifiers are used to quickly compare and identify files, detect duplicates, or verify data integrity during transmission.
  7. **Cryptographic Salting**: In password hashing, salts (random values) are combined with passwords before hashing to prevent identical passwords from producing the same hash. This enhances security against dictionary attacks and rainbow table attacks.

8. **Data Deduplication**: Hash functions are used in data deduplication systems to identify and eliminate duplicate data by comparing hashes of data chunks. This optimizes storage space and improves data management efficiency.
9. **Fingerprinting and Digital Forensics**: Hash functions are used in digital forensics to create digital fingerprints of files and storage devices. These fingerprints help in identifying and verifying digital evidence without altering the original data.
10. **Content Addressing**: Hash functions uniquely identify data chunks in content-addressable storage systems (like BitTorrent). This enables efficient distribution, retrieval, and verification of data across distributed networks.

These applications demonstrate the versatility and critical role of hash functions in ensuring security, data integrity, and efficient data management across various domains of computing and information technology.

2.2.3 Digital Signature

Just like signing a document with a pen to prove authorship and prevent tampering, digital signatures serve a similar purpose in the digital world. They are cryptographic mechanisms used to validate the authenticity and integrity of a message, software, or digital document. They provide assurances that the message was created and sent by the claimed sender (authentication) and that it was not altered in transit (integrity) [81].

A digital signature scheme typically consists of three algorithms: *KeyGen*, *Sign*, and *Verify*.

– $KeyGen(1^n) \rightarrow (SK,\ PK)$: The *KeyGen* algorithm takes a security parameter n as input and outputs a pair of keys: the private key (SK) and the public key (PK).

– $Sign_{SK}(m) \rightarrow \sigma$: The *Sign* algorithm uses the private key (SK) and a message m to produce a digital signature σ.

– $Verify_{PK}(m, \sigma) \rightarrow \{\text{true, false}\}$: The *Verify* algorithm takes the public key (PK), the message m, and the digital signature σ as inputs and outputs a boolean value indicating whether the signature is valid (true) or not (false).

– **How Digital Signatures Work**:

  – **Signature Creation**:

    ∗ The sender creates a hash of the message. Signing large data is computationally expensive and time-consuming. By creating a hash, a small fixed-size digest of the data, signing becomes more efficient than signing the entire message.

    * The hash is then encrypted with the sender's private key to create the digital signature.
    * The original message, along with the digital signature, is sent to the recipient.
  - **Signature Verification**:
    * The recipient decrypts the digital signature using the sender's public key to obtain the hash.
    * The recipient also generates a hash of the received message.
    * If the two hashes match, the signature is verified as authentic, and the message integrity is confirmed.

- **Examples of Digital Signature Algorithms**:
  - **RSA (Rivest-Shamir-Adleman)**:
    * Relies on the difficulty of factoring large integers.
    * Widely used but less efficient compared to ECC.
  - **DSA (Digital Signature Algorithm)**:
    * A U.S. federal standard for digital signatures.
    * Based on the mathematical properties of discrete logarithms.
  - **ECDSA (Elliptic Curve Digital Signature Algorithm)**:
    * Uses elliptic curve cryptography.
    * Provides the same level of security with smaller key sizes compared to RSA and DSA, making it more efficient.

- **Properties of Digital Signatures**:
  - **Authenticity**: The signature confirms the identity of the signer.
  - **Integrity**: Any modification to the signed data will invalidate the signature.
  - **Non-repudiation**: The signer cannot deny having signed the message, as signing requires a secret key known only to the signer.

- **Applications of Digital Signatures**:
  - **Email Security**: Ensures that the email is from the claimed sender and has not been altered.
  - **Software Distribution**: Verifies that software or updates are from the legitimate developer and have not been tampered with.
  - **Financial Transactions**: Provides security for online transactions by ensuring the authenticity and integrity of transaction details.
  - **Legal Documents**: Digital signatures can be used to sign contracts and legal documents, providing a secure and verifiable method of consent.

2.2.4 Commitment Schemes

Commitment schemes are cryptographic protocols that enable one party (the committer) to commit to a chosen value while keeping it hidden from others, with the ability to reveal the committed value later [146]. These schemes play a crucial role in various cryptographic protocols and applications, such as secure multi-party computation, zero-knowledge proofs, and blockchain technology.

- **Properties of Commitment Schemes**: A secure commitment scheme has two main properties:
  - **Hiding**: Ensures that the committed value remains secret from any party other than the committer until it is revealed. This property guarantees that the commitment does not leak any information about the committed value.
  - **Binding**: Ensures that the committer cannot change the committed value after the commitment is made. This property prevents the committer from committing to a value and then changing it to another value at the time of revelation.
- **Phases of a Commitment Scheme**: A typical commitment scheme consists of two phases:
  1. **Commit Phase**:
     - The committer chooses a value v to commit to and generates a commitment C using a commitment function Commit(v, r), where r is a random value (often called a nonce) to ensure the hiding property.
     - The commitment C is then sent to the receiver.
  2. **Reveal Phase**:
     - The committer reveals the value v and the randomness r to the receiver.
     - The receiver verifies the commitment by checking if $C = Commit(v, r)$.
- **Types of Commitment Schemes**:
  - **Bit Commitment**:
    * Commits to a single bit (0 or 1).
    * Fundamental building block for more complex protocols.
  - **String Commitment**:
    * Commits to a string of bits or a more complex value.
    * Used in broader applications requiring commitment to larger datasets.
- **Construction of Commitment Schemes**:
  - **Hash-Based Commitment**:

    * **Commit**: To commit to a value v, choose a random value r and compute the commitment $C = H(v\|r)$, where H is a cryptographic hash function and $\|$ denotes concatenation.
    * **Reveal**: Reveal the values v and r. The receiver checks if the commitment C matches $H(v\|r)$.
  - **Pedersen Commitment**:
    * **Commit**: To commit to a value v, choose a random value r and compute the commitment $C = g^v h^r \mod p$, where g and h are generators of a cyclic group of prime order p, and h is chosen such that the discrete logarithm of h with respect to g is unknown.
    * **Reveal**: Reveal the values v and r. The receiver checks if the commitment C matches $g^v h^r \mod p$.

– **Applications of Commitment Schemes**:

  - **Zero-Knowledge Proofs**:
    * Used to prove the knowledge of a value without revealing the value itself.
    * Commitment schemes ensure that the prover cannot change their mind after committing to a value.
  - **Secure Multi-Party Computation**:
    * Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
    * Commitment schemes ensure that parties commit to their inputs before computation begins.
  - **Blockchain and Cryptocurrencies**:
    * Used in various protocols to ensure the integrity and confidentiality of transactions.
    * Commitment schemes are fundamental in constructing secure and verifiable ledgers.
  - **Electronic Voting**:
    * Ensures that votes are committed securely and cannot be altered.
    * Provides a mechanism to reveal votes at a later stage while maintaining voter privacy until then.

2.2.5 Proxy Reencryption

Proxy re-encryption (PRE) is a cryptographic technique that allows a proxy to convert ciphertexts encrypted under one public key to another public key without learning the plaintext. This method provides a way for users to securely delegate decryption rights without exposing their secret keys. Let us consider a scenario where Alice wishes to enable Bob to decrypt messages encrypted with her public key without giving him her private key. Alice also prefers not to give her private key to a proxy, as it demands too much trust. Instead, Alice seeks a solution where a proxy can transform messages encrypted with her public key into messages encrypted with Bob's public key without decrypting them. Proxy re-encryption serves this purpose. By providing the proxy with certain information, Alice allows it to carry out this transformation. In this setup, Alice is the delegator, and Bob is the delegate. For instance, Alice may need to forward her encrypted emails to Bob temporarily. She sends her encrypted emails to a proxy, which re-encrypts them under Bob's public key, enabling Bob to decrypt and read them. [108]

Workflow of Proxy Re-encryption Scheme

1. **Key Generation**:
   - Let $\mathcal{K} = (G, E, D)$ be a public-key encryption scheme, where G is the key generation algorithm, E is the encryption algorithm, and D is the decryption algorithm.
   - Alice has a key pair (PK_A, SK_A) generated by G, where PK_A is her public key and SK_A is her secret key.
   - Similarly, Bob has a key pair (PK_B, SK_B) generated by G, where PK_B is his public key and SK_B is his secret key.
2. **Encryption**:
   - Alice encrypts the message m using her public key PK_A:
   $$c_A = E(PK_A, m)$$
3. **Re-encryption Key Generation**:
   - Alice generates the re-encryption key $RKk_{A\to B}$ using her secret key SK_A and Bob's public key PK_B:
   $$RK_{A\to B} = \text{GenReencKey}(SK_A, PK_B)$$
4. **Re-encryption**:
   - The proxy uses the re-encryption key $RK_{A\to B}$ to transform $c_A = E(PK_A, m)$ into $c_B = E(PK_B, m)$ without learning the plaintext m:
   $$c_B = \text{ReEncrypt}(RK_{A\to B}, c_A)$$
5. **Decryption**:
   - Bob decrypts c_B using his secret key sk_B and obtains the original plaintext m:
   $$m = D(SK_B, c_B)$$

- **Security Properties:**

1. **Confidentiality**: The proxy cannot see the plaintext unless it colludes with Bob.
2. **Key Privacy**: The proxy cannot derive Alice's secret key, even if it colludes with Bob.
3. **Directionality**: The scheme can be:
   - **Bi-directional**: When Alice delegates to Bob, Bob can automatically delegate to Alice, requiring mutual trust.
   - **Uni-directional**: Alice can delegate to Bob without requiring Bob to delegate to her, so trust does not need to be mutual.
4. **Transitivity**: The scheme can be:
   - **Transitive**: Alice can delegate to Bob, and Bob can delegate to Tim, and so on.
   - **Non-transitive**: Bob cannot delegate to Tim after receiving a delegation from Alice.

- **Applications of Proxy Re-encryption Scheme:**

  PRE has several applications:

  - **Secure Data Sharing**: Allows users to securely share encrypted data without exposing their secret keys.
  - **Cloud Storage**: Enables secure and efficient sharing of data stored in the cloud by allowing the cloud provider to re-encrypt data for different users without accessing the plaintext.
  - **Access Control**: Enhances access control mechanisms by enabling fine-grained delegation of decryption rights without exposing the original decryption keys.
  - **Distributed Systems**: Supports secure communication and data sharing in distributed systems where trust levels vary among participants.

Proxy re-encryption provides a powerful mechanism for secure data sharing and delegation in various applications. By leveraging re-encryption keys, it maintains data confidentiality while enabling controlled access, making it a valuable tool in the realm of cryptographic security. In some proxy re-encryption schemes, the re-encryption process can be handled by the delegator (Alice) or the delegatee (Bob) instead of a proxy. This allows for flexibility in how the re-encryption is performed and can increase security by reducing the reliance on a third-party proxy.

2.3 Merkle Tree Data Structure

Merkle Trees, introduced by Ralph Merkle in 1979, are binary trees used for efficient and secure verification of data integrity and membership proofs. They derive their name from their inventor and have since become a foundational concept in cryptography, distributed systems, and blockchain technology [80], [153].

- Each leaf node contains the hash of a data block or element in the set.
- Non-leaf nodes store hashes computed from the concatenation or pairwise combination of their child nodes' hashes.
- The root node of the tree, known as the Merkle Root (MR), represents a condensed cryptographic summary of all data in the tree.

Properties:

1. **Data Integrity:** Merkle Trees enable efficient verification of data integrity. By comparing the Merkle Root against a trusted value, one can quickly determine if any part of the data has been altered.
2. **Efficient Updates:** When data changes, only a subset of the tree needs to be updated, typically logarithmic with respect to the number of elements, ensuring efficiency in dynamic environments.
3. **Compact Proof Generation:** Proof of membership or non-membership in a set can be efficiently generated and verified using a logarithmic number of hash comparisons proportional to the height of the tree.

Operations:

- **Construction:** Initially, all leaf nodes are populated with hashes of individual data elements. Non-leaf nodes are then constructed by hashing pairs of child nodes until a single root hash (Merkle Root) remains.
- **Verification:** To verify the integrity of a particular data element or set, a proof path from the leaf containing the element's hash to the Merkle Root is generated. This path consists of sibling hashes at each level, allowing quick verification.
- **Dynamic Updates:** Updating a Merkle Tree involves recalculating hashes along the path from the updated leaf node to the root. This ensures that the Merkle Root accurately reflects the current state of the data.

Applications of Merkle Tree:

- **Blockchain Technology:** In blockchain systems like Bitcoin and Ethereum, Merkle Trees are used to verify the inclusion of transactions in blocks efficiently and to detect tampering with block data.
- **Distributed File Systems:** Systems like IPFS (InterPlanetary File System) utilize Merkle Trees to verify the integrity of distributed data across multiple nodes.
- **Data Synchronization:** They are employed in data synchronization protocols to ensure that only changed data is transmitted between synchronized systems, optimizing bandwidth usage.

– **Cryptographic Commitments:** Merkle Trees serve as a form of cryptographic commitment, where the Merkle Root can be publicly shared to prove possession of specific data without revealing the data itself.

– **Secure Communication:** Used in protocols like TLS (Transport Layer Security) to verify the integrity of transmitted data and ensure resistance against tampering attacks.

Merkle Trees, which are fundamental in cryptography and distributed systems, come in several variations tailored to different applications. The basic Binary Merkle Tree serves as the foundation, using concatenated hashes of child nodes to ensure data integrity and enable efficient verification. Variants like the Quad Merkle Tree build upon this by accommodating up to four child nodes per parent, reducing tree height for large datasets. Other adaptations, such as the Patricia Tree (Radix Tree), focus on optimizing storage and retrieval by storing keys at leaf nodes and hashes at internal nodes. Each variation, from Sparse Merkle Trees for sparse datasets to Layered Merkle Trees for hierarchical aggregation in blockchains, addresses specific needs such as scalability, security, and efficiency in various technological contexts.

By leveraging Merkle Trees, developers can ensure data integrity, optimize performance, and enhance security across a wide array of technological implementations.

In the proposed healthcare system, digital signatures verify the authenticity of EHRs, while encryption protects sensitive patient data. Similarly, in the banking system, hash functions ensure the integrity of financial transactions, preventing unauthorized modifications. These cryptographic primitives are fundamental to blockchain operations: hash functions (e.g., SHA-256 in Bitcoin and Keccak-256 in Ethereum) ensure data integrity by organizing transactions into a Merkle tree structure, digital signatures (e.g., ECDSA in Bitcoin and Ethereum) provide authentication, and encryption secures communication between nodes.

2.4 Blockchain

As the name suggests, a blockchain is a chain of blocks, i.e., an ordered sequence of blocks, each containing a set of transactions [111]. The first block in the blockchain is known as the **genesis block**, serving as the root of the entire blockchain. The **height** of the blockchain is defined by the distance from the genesis block. Every block includes the hash of its preceding block within its header. This sequence of hashes creates a linked chain, with each block connecting to its parent. A block can have only one parent but may have multiple children, leading to a **fork** in the network. Consensus mechanisms are employed to resolve forks, determining which child block will continue the blockchain [110].

Blockchain records are immutable due to the cryptographic hashing mechanism. Altering the data in one block changes its hash, which in turn alters the hashes of all subsequent blocks. This cascading effect makes tampering computationally impractical, especially as the blockchain grows longer.

Consensus in Blockchain

Nakamoto's consensus algorithm enables all nodes in the blockchain network to agree on the validity of a block without needing a central authority. The process includes:

- Verifying transactions: Nodes validate the correctness of transaction inputs, outputs, and signatures and check for double-spending.
- Block creation: Miners collect verified transactions into a block and solve a cryptographic puzzle using the proof-of-work algorithm.
- Block propagation: The first miner to solve the cryptographic puzzle broadcasts the block to the network. Other nodes independently verify the block and, if valid, add it to their copy of the blockchain.

Mining and Proof-of-Work

Mining is the process that secures the Bitcoin network and enables decentralized consensus without a third party [110]. Any network node can become a miner tasked with validating transactions and recording them in the blockchain. Once included in a block, a transaction is confirmed, allowing the transaction outputs to be spent. Miners receive rewards for block mining and transaction fees included in the block. The proof-of-work mechanism requires miners to solve a cryptographic puzzle with a specific difficulty, ensuring that creating new blocks requires significant computational effort. This mechanism prevents the creation of numerous fake nodes, known as Sybil attacks. The winning miner, who solves the puzzle first, earns a reward for their computational effort.

Due to the complexity of verification and the proof-of-work process, blockchain transactions are slower compared to traditional payment systems.

2.5 Bitcoin

Bitcoin, introduced by Satoshi Nakamoto in 2008 [110], represents a groundbreaking decentralized digital currency system. It ensures secure and efficient transactions while addressing the double-spending problem and enabling participants to join and leave the network seamlessly.

2.5.1 Key Properties

Bitcoin achieves several key properties:

- **Permissionless**: Users can participate in the network without requiring identities. They can join or leave the network anytime.

- **Decentralized**: It operates without a central authority or trusted third party.
- **Secure**: Transactions are secured by cryptographic mechanisms and a resilient consensus protocol [35].
- **Immutable Ledger**: All transactions are permanently recorded in a verifiable append-only ledger, known as the blockchain.

2.5.2 Bitcoin Addresses

Users generate private/public key pairs to create Bitcoin addresses, which serve as pseudonymous identities. Ownership of funds is tied to these addresses, enabling secure transactions.

2.5.3 UTXO Model

Bitcoin uses the Unspent Transaction Output (UTXO) model [8]. Each transaction output consists of coins (BTC) and conditions (script) that must be met to spend these coins. This model differs from the account-based model used in some other cryptocurrencies, as it tracks individual "coins" rather than account balances.

2.5.4 Transaction Structure

A Bitcoin transaction $\mathrm{tx} = (\mathrm{id}, \mathrm{input}, \mathrm{output})$ includes:

- **Transaction ID**: Hash of the transaction.
- **Transaction Input**: References to existing UTXOs that the spender has the right to use.
- **Transaction Output**: New UTXOs created, specifying coins and spending conditions.

2.5.5 Bitcoin Script

Transactions are valid if each input is accompanied by a valid signature or other required conditions and if the inputs have not been spent before. Bitcoin Script plays a crucial role in this validation process by allowing users to define these conditions directly within transaction outputs. This stack-based, turing in-complete scripting language enables complex spending conditions, such as multi-signature requirements or time-locked transactions, which are verified autonomously. Valid transactions meeting these specified conditions are then added to the blockchain through the consensus mechanism, ensuring the integrity and security of the Bitcoin network.

2.5.6 Consensus Mechanism

Bitcoin's consensus mechanism ensures agreement among network nodes on the validity of transactions without relying on a central authority. It involves:

– **Verification**: Nodes validate transactions and create blocks.

– **Proof-of-Work (PoW)**: Miners compete to solve cryptographic puzzles to append new blocks to the blockchain, requiring significant computational effort.

– **Blockchain Security**: The longer the blockchain, the more secure it becomes against tampering due to the computational difficulty of altering past transactions.

2.5.7 Mining and Monetary Policy

New bitcoins are created through mining rewards. Miners receive newly minted bitcoins as a reward for successfully adding a block to the blockchain. The reward amount undergoes a "halving" event approximately every four years, reducing by 50%. This deflationary mechanism is crucial to Bitcoin's monetary policy and scarcity. However, the mining process is not without potential exploits. Selfish mining is a strategy where miners withhold newly discovered blocks to create a private chain, potentially claiming more rewards than their fair share. Additionally, in mining pools, malicious actors may engage in block withholding attacks [10], where they submit partial proofs of work but withhold full solutions, undermining the pool's efficiency. These strategies, while detrimental to the network's intended fairness, highlight the complex game theory aspects of Bitcoin's consensus mechanism.

2.5.8 Network Security

The Proof-of-Work system secures the network against various attacks:

– **51% Attacks**: An attacker would need to control more than half of the network's computational power to alter the blockchain potentially.

– **Double-Spending**: The consensus mechanism and blockchain structure make it extremely difficult to spend the same bitcoin twice.

2.5.9 Full Nodes and Light Nodes

Bitcoin network participants can run different types of nodes:

Full Nodes

Full nodes download and verify the entire blockchain, including all transactions and blocks. They play a crucial role in maintaining and validating the network by:

– Independently verifying all transactions and blocks

- Enforcing the consensus rules of the Bitcoin protocol
- Providing a complete and authoritative record of all transactions

Running a full node requires significant storage and bandwidth resources but offers the highest level of security and network support.

Light Nodes and SPV
Light nodes, also known as SPV (Simplified Payment Verification) clients, provide a more resource-efficient way to interact with the Bitcoin network:

- **Lightweight**: They do not store the entire blockchain, only downloading block headers and specific transactions relevant to the user's wallet.
- **SPV Mechanism**: Light nodes use SPV to verify transactions without downloading full blocks. They rely on the longest proof-of-work chain to confirm transaction inclusion.
- **Merkle Proofs**: Light nodes request Merkle proofs from full nodes to verify specific transactions, allowing them to confirm payments without storing the entire blockchain.
- **Trade-offs**: While SPV nodes are more efficient in terms of resources, they offer reduced security compared to full nodes and must trust full nodes to some extent for transaction verification.

SPV, introduced in the original Bitcoin whitepaper [110], enables Bitcoin wallets on resource-constrained devices like smartphones. However, users should be aware of the security trade-offs when using light nodes instead of full nodes.

2.5.10 Privacy and Pseudonymity

While Bitcoin transactions are pseudonymous, they are not entirely anonymous. All transactions are publicly recorded on the blockchain, which can potentially be analyzed to link addresses to real-world identities [132]. Users face challenges in maintaining complete anonymity, leading to the development of privacy-enhancing techniques and alternative cryptocurrencies focused on privacy.

2.5.11 Scalability Challenges

Bitcoin faces scalability challenges, particularly in terms of transaction throughput. The current block size and block time limit the number of transactions that can be processed per second. Solutions like the Lightning Network, a second-layer protocol, aim to address these limitations by enabling faster and cheaper off-chain transactions.

2.5.12 Recent Developments

Notable recent developments in the Bitcoin ecosystem include:

- **Segregated Witness (SegWit)**: An upgrade that increased block capacity and fixed transaction malleability.
- **Taproot**: An upgrade that improved privacy, scalability, and smart contract functionality.

2.5.13 Transaction Finality

The time taken to finalize a transaction, ensuring it is irreversible and added to the blockchain, is bounded by Δ, dependent on network confirmation processes. Generally, six confirmations (approximately one hour) are considered sufficient for most transactions.

Bitcoin's design ensures robustness against arbitrary deviations (Byzantine faults) and provides a foundation for decentralized financial applications and trustless peer-to-peer transactions. Its ongoing development and the growing ecosystem around it continue to shape the future of digital currencies and blockchain technology.

2.6 Ethereum

Ethereum, proposed by Vitalik Buterin in 2013 and launched in 2015 [24], [170], is a decentralized, open-source blockchain platform that enables the creation and execution of smart contracts and decentralized applications (DApps).

2.6.1 From Bitcoin to Ethereum: Addressing Limitations and Introducing New Paradigms

Limitations of the Bitcoin Ecosystem

While Bitcoin revolutionized digital currencies, it had several limitations:

- **Limited Programmability**: Bitcoin's script language is intentionally non-Turing complete, restricting complex smart contract functionality.
- **Specialized Use Case**: Bitcoin primarily focuses on peer-to-peer value transfer, limiting its use to more complex applications.
- **Slow Transaction Finality**: Bitcoin's block time and confirmation requirements lead to slower transaction finality compared to traditional payment systems.
- **Scalability Issues**: The fixed block size limits transaction throughput, leading to higher fees during network congestion.

The Need for Ethereum

Ethereum was conceived to address these limitations and expand blockchain capabilities:

- **General-Purpose Platform**: Ethereum aimed to create a blockchain that could support a wide range of decentralized applications beyond simple value transfer.
- **Turing-Complete Smart Contracts**: By implementing a Turing-complete programming language, Ethereum enabled complex, self-executing contracts and applications.
- **Faster Block Time**: Ethereum's shorter block time (originally 15 seconds, compared to Bitcoin's 10 minutes) allows for quicker transaction confirmations.
- **Flexible State Storage**: Ethereum's account-based model provides more flexibility for complex state management compared to Bitcoin's UTXO model.

UTXO vs. Account-Based Model

Bitcoin and Ethereum use fundamentally different models for tracking user balances and facilitating transactions:

Bitcoin's UTXO (Unspent Transaction Output) Model

- **Transaction-Centric**: Each transaction consumes one or more UTXOs as inputs and creates new UTXOs as outputs.
- **Stateless**: There is no concept of an account balance; a user's balance is the sum of their unspent outputs.
- **Privacy Advantage**: Each transaction can use a new address, enhancing privacy.
- **Parallel Processing**: UTXOs can be processed in parallel, potentially improving scalability.
- **Complexity in Smart Contracts**: Implementing complex smart contracts is more challenging in a UTXO model.

Ethereum's Account-Based Model

- **Account-Centric**: Each account has a balance, and transactions directly update these balances.
- **Stateful**: The system maintains the state (balance and other data) for each account.
- **Simplified Logic**: Easier to understand and implement complex smart contract logic.
- **Nonce System**: Uses a nonce (transaction count) to prevent double-spending and ensure transaction order.

– **Potential Privacy Concerns**: Reuse of addresses can make transaction tracking easier.

Key Differences

– **State Management**: UTXO tracks unspent outputs, while the account model tracks account balances.

– **Transaction Verification**: UTXO requires verifying the entire transaction history, while the account model only needs to check the current state.

– **Smart Contract Suitability**: The account model is generally more suitable for complex smart contracts and state-dependent operations.

– **Scalability Trade-offs**: UTXO offers potential parallelization benefits, while the account model simplifies state management.

Ethereum's choice of the account-based model aligns with its goal of being a general-purpose blockchain platform. This model facilitates the easier implementation of complex smart contracts and decentralized applications. This fundamental difference in design philosophy and implementation underscores the distinct roles that Bitcoin and Ethereum play in the blockchain ecosystem.

2.6.2 Key Properties

Ethereum extends the blockchain concept beyond simple value transfer, offering:

– **Turing-complete Programming**: Allows for complex, arbitrary computation on the blockchain.

– **Smart Contracts**: Self-executing contracts with the terms directly written into code.

– **Decentralized Applications (DApps)**: Applications that run on a peer-to-peer network rather than a single computer.

– **Tokenization**: Ability to create and manage custom tokens (e.g., ERC-20, ERC-721).

2.6.3 Ethereum Virtual Machine (EVM)

The EVM is a Turing-complete virtual machine that executes smart contract bytecode:

– **State Machine**: Maintains the global state of all accounts and smart contracts.

– **Gas Mechanism**: Uses "gas" to measure computational cost and prevent infinite loops.

– **Opcodes**: Provides a set of instructions for smart contract execution.

2.6.4 Accounts

Ethereum has two types of accounts:

- **Externally Owned Accounts (EOAs)**: Controlled by private keys, can initiate transactions.
- **Contract Accounts**: Controlled by their code, cannot initiate transactions on their own and activated by transactions from EOAs or messages from other contracts.

2.6.5 Transactions and Messages

- **Transactions**: In the Ethereum network, transactions and messages are two distinct concepts that facilitate operations and communication. Transactions are signed data packages initiated by Externally Owned Accounts (EOAs). They can transfer Ether, deploy smart contracts, or trigger contract code execution. Transactions are recorded on the blockchain and modify the global state. They require gas for execution and include fields such as recipient, value, data, gas price, and gas limit. Every transaction is recorded on the Ethereum blockchain and represents a state transition in the network.
- **Messages**: Messages, on the other hand, are virtual objects that exist only within the Ethereum execution environment. They represent internal calls between contracts and are created when one contract calls another contract's function. Unlike transactions, messages are not directly serialized or recorded on the blockchain. While messages themselves are not visible on the blockchain, their impact is reflected in the overall state changes recorded as part of the transaction that triggered them. This mechanism allows for complex interactions between contracts while maintaining the integrity and traceability of the blockchain state.

2.6.6 Consensus Mechanism

Ethereum has undergone significant changes in its consensus mechanism:

- **Proof-of-Work (PoW)**: Initially used, similar to Bitcoin's mechanism.
- **Proof-of-Stake (PoS)**: Transitioned to PoS in 2022 with "The Merge", significantly reducing energy consumption.
- **Validators**: In PoS, validators replace miners, staking ETH to propose and attest to blocks.

2.6.7 Smart Contracts

Smart contracts are self-executing programs stored on the blockchain:

- **Solidity**: Primary high-level language for writing smart contracts.

- **Immutability**: Once deployed, the contract code cannot be changed.
- **Deterministic Execution**: Given the same input, contracts always produce the same output.

2.6.8 Tokenization Standards

Ethereum introduced several token standards:

- **ERC-20**: Standard for fungible tokens.
- **ERC-721**: Standard for non-fungible tokens (NFTs).
- **ERC-1155**: Multi-token standard supporting both fungible and non-fungible tokens.

2.6.9 Gas and Ether

- **Ether (ETH)**: Native cryptocurrency of Ethereum.
- **Gas**: Unit to measure computational effort, priced in Ether.
- **Gas Limit**: Maximum amount of gas a transaction or block can consume.
- **Gas Price**: Amount of Ether the sender is willing to pay per unit of gas.

2.6.10 Transaction Cost and Latency

- **Transaction Cost**:
  When conducting transactions on Ethereum, a gas fee is incurred. Every transaction executed on the Ethereum platform consumes a certain amount of gas, known as the "gas cost", which depends on the complexity and computational resource requirements of the transaction. Gas refers to the monetary cost associated with completing a transaction or the execution of a contract on the Ethereum platform. The term "gas cost" refers to the amount of gas a transaction consumes to execute successfully. On the other hand, the "gas price" is the unit price of gas measured in GWei (1 Gwei = 10^{-9} ETH) [7]. Each blockchain transaction has a gas limit to avoid running out of gas if the code contains bugs, providing a safety mechanism.

  Transaction cost is derived by multiplying gas cost by gas price. While gas cost is a fixed parameter, gas price is variable. The transactor initiator has the provision to select the gas price. Choosing the appropriate gas price can be a challenge task. To calculate the correct gas price, one must monitor network congestion and current gas price trends. Opting for higher gas prices provides a lucrative incentive for validators/miners to include transactions more quickly into a block of a blockchain. The transaction initiator needs to pay a transaction fee to the validator or miner for including the transaction into a block of a blockchain.

– **Transaction Latency:**
  Latency refers to the duration of time that a user must wait after initiating a transaction by broadcasting it to the network before it is processed and then included in a block. In other words, it's the duration between the moment a user triggers a transaction and the moment it becomes a part of the blockchain. In public blockchains like Ethereum, transactions must go through a distributed consensus mechanism before they can be added to the blockchain. This means that multiple nodes in the network must agree on the transaction's validity. As a result, there will be some delay. The delay or latency may vary based on network congestion. High transaction latency can impact the user experience by causing delays in transaction confirmation, which can be critical in time-sensitive applications.

2.6.11 Scalability Solutions

Ethereum faces scalability challenges, addressed through various solutions:

– **Layer 2 Solutions**: Off-chain scaling solutions like Optimistic Rollups and zk-Rollups.

– **Sharding**: Planned future upgrade to split the network into multiple shards, increasing throughput.

2.6.12 Ethereum 2.0 (Eth2) Upgrade

A series of upgrades to improve Ethereum's scalability, security, and sustainability:

– **Beacon Chain**: Introduced the Proof-of-Stake mechanism.

– **The Merge**: Transitioned the main network from PoW to PoS.

– **Future Upgrades**: Include sharding and other improvements.

2.6.13 Ethereum Node Types

1. **Full Nodes:** Download and verify the entire blockchain, including all transactions and state changes.

2. **Light Nodes:** Download block headers and use Merkle proofs to verify specific data without storing the entire blockchain.

3. **Archive Nodes:** Store the entire history of the blockchain, including all historical states, requiring significant storage.

Ethereum's programmable blockchain and smart contract capabilities have made it a foundational platform for numerous blockchain applications, from decentralized finance to non-fungible tokens and beyond. Its ongoing development and the growing ecosystem continue to drive innovation in the blockchain space.

2.7 Smart Contract

Invented by Nick Szabo in 1994, years before the advent of blockchain technology, smart contracts are self-executing agreements between parties written in programming languages like Solidity, Serpent, and Vyper, and deployed on blockchain networks. They automate the enforcement of contract terms without requiring intermediaries, thereby enhancing security and transparency. By executing automatically based on predefined conditions, smart contracts reduce the need for trust and eliminate opportunities for fraud or manipulation. Integrating smart contracts with blockchain technology has catalyzed the development of decentralized applications (DApps) that securely manage transactions and processes across various industries [83], [107], [135].

Initially used in the Bitcoin network for transferring digital value, early smart contracts were limited by a Turing-incomplete language, restricting their functionality. Ethereum later revolutionized smart contracts by introducing a Turing-complete language, empowering developers to create more sophisticated and powerful custom contracts.

- **Key Features:**
    - *Autonomy:* Operate without intermediaries, ensuring direct and transparent interactions.
    - *Immutability:* Once deployed on the blockchain, contracts cannot be altered, ensuring reliability and trustworthiness.
    - *Transparency:* Contract terms and execution are visible to all participants on the blockchain, promoting openness.
    - *Automation:* Automatically execute predefined actions based on conditions, reducing human intervention.
    - *Customization:* Modifiable before deployment to suit specific contract requirements.
- **Advantages:**
    - Enhance recordkeeping accuracy and transparency.
    - Reduce fraud and manipulation through automated enforcement.
    - Improve operational efficiency and cost-effectiveness by eliminating intermediaries.
    - Foster trust among parties by ensuring programmable contract execution.
    - Enable secure and reliable execution of complex contractual agreements.
- **Challenges:**
    - Security Risks: Vulnerabilities in smart contract code can lead to breaches and financial losses.

  - Regulatory Uncertainty: Legal frameworks for smart contracts are still evolving, posing compliance challenges.
  - Implementation Complexity: Developing robust code and ensuring security measures are crucial but challenging.
  - Immutability Concerns: Once deployed, contracts cannot be easily updated, necessitating careful planning.
  - Alignment of Intentions: Ensuring all parties agree on contract terms and their implications is critical.
  - Scalability Issues: Increasing transaction volume and complexity can strain blockchain networks.

This innovation promises to redefine traditional business models by enabling trustless, efficient, and autonomous digital interactions across industries.

The first two works in this thesis were deployed on the Ropsten test network, utilizing Ethereum's Proof-of-Work consensus algorithm, while later works were deployed on the Sepolia test network under Ethereum's Proof-of-Stake mechanism. Solidity-based smart contracts were used for executing autonomous operations, integral to the proposed solutions in healthcare and banking.

2.8 Distributed File Storage

Distributed file storage is a system that allows files to be stored across multiple locations or nodes rather than on a single server. This approach enhances data redundancy, availability, and accessibility. Key benefits include improved fault tolerance, as data is replicated across multiple nodes, ensuring that the system can recover from individual node failures. Additionally, distributed storage systems often provide better scalability, allowing for seamless expansion as storage needs grow. One prominent example of a distributed file storage system is the InterPlanetary File System (IPFS).

InterPlanetary File System

InterPlanetary File System (IPFS) is a decentralized protocol for storing and sharing files on a distributed network. Instead of relying on a centralized server, IPFS uses a peer-to-peer network of nodes to store and retrieve files, making it more resilient to censorship and failure. In IPFS, files are addressed by their content rather than their location. When a file is added to the IPFS network, it is split into multiple smaller pieces, each assigned a unique cryptographic hash. These hashes create a unique identifier for the entire file, called a Content Identifier (CID) [16].

When a user requests a file from the IPFS network, they use the CID to locate the file. The IPFS network employs a distributed hash table (DHT) to store information about which nodes have a copy of the file. The DHT is a decentralized system that allows nodes to communicate and find the file's location. The file is then retrieved from the node that has a copy of it.

IPFS offers significant advantages, including enhanced data resilience, improved access speeds through decentralized retrieval, and resistance to single points of failure or censorship. It is particularly useful for applications requiring robust, decentralized data management, such as blockchain technologies and content distribution networks.

This innovation in file storage ensures that data is more secure, accessible, and efficiently managed, offering a reliable solution for modern data storage needs.

2.9 Incentive Mechanism

An incentive mechanism is created to reward honest participants and penalize malicious or dishonest parties in a decentralized system. This mechanism is essential for ensuring security, reliability, and proper functioning, as well as for maintaining trust and encouraging participation in decentralized networks. Typically, rewards come in the form of cryptocurrency tokens or other digital assets and are given to participants who adhere to protocol rules, validate transactions, and contribute to network security [66], [100]. For instance, in blockchain networks, miners or validators receive rewards for successfully adding a new block to the chain. These rewards act as a motivator for honest behavior and consistent participation.

On the other hand, parties that engage in malicious activities or try to compromise the system will face penalties. These penalties may involve losing staked tokens, paying fines, or facing temporary or permanent exclusion from participating in the network. These penalties are in place to discourage malicious behavior by making it expensive and unprofitable.

In a smart contract-enabled system, participants typically lock a certain amount of stake in the smart contract. If all parties adhere to the protocol, the stake is released back to them at the end. However, if a participant behaves maliciously, their stake is deducted and transferred to the honest party as a reward for their compliance. This system of staking and penalizing ensures that participants are financially motivated to act honestly and follow the protocol, thereby enhancing the security and integrity of the decentralized application.

By balancing rewards and penalties, incentive mechanisms play a critical role in ensuring the integrity and robustness of decentralized networks, promoting honest participation, and discouraging harmful activities. It is an important mechanism towards designing a secure decentralized application on a blockchain platform.

3 Blockchain-Enabled Secure and Smart Healthcare System

"For he who has health has hope, and he who has hope has everything."

— **Owen Arthur**

IoT devices in the healthcare sector generate massive amounts of patients' medical data. This digital data is a part of EHRs (Electronic Health Records), typically stored in databases. Owing to the personal nature of EHRs, they must not be made available publicly. Failure to do so can have grave implications on the patient's life – such as discrimination by an employer based on medical history or failure to get insurance. Further, tampering of medical data has the potential to jeopardize a person's life. A third party might also use such sensitive data to inflict harm or sell it to other parties.

Each entity involved in providing medical service to the patient must be made accountable for their action. Unethical behavior on the part of medical professionals includes hospitals overcharging patients or providing inadequate medical service. According to a survey conducted by *Jan Arogya Abhiyan* and *Corona Ekal Mahila Punarvasan Samiti* in September 2021, 75% of the patients admitted for COVID-19 were overcharged. This happened despite the government capping the treatment expenses of COVID patients.

Patients may make false accusations of doctors and hospitals of mistreating them. Relying on a third party to resolve a dispute is not a good solution. Further, rules and regulations imposed by the country on medical treatment can be easily bypassed. Keeping all these problems in mind, we inferred that blockchain-based solutions would perfectly fit in this case.

The proposed patient-centric healthcare system ensures privacy, security of patients' medical data and fairness of the entities involved. If a patient has given consent to a party, the latter can access the medical data. An access control matrix stored in blockchain disallows any sort of malicious intervention. The digital footprint of the medical data is stored in the blockchain to provide integrity and immutability, while digital signature ensures accountability. The hospital authority cannot extort any arbitrary amount from a patient by providing unfair treatment or denying treatment. In that case, our system will penalize the hospital authority. On the other hand, if a patient claims that s/he has received an invalid medical report or denies receiving treatment, the logic encoded in the smart contract prevents such behavior by penalizing the patient.

3.1 Objectives

We intend to propose a *patient-centric hospital management system* which realizes the following objectives:

– *Fairness*: An honest party will never lose money even if the rest of the parties are malicious and try to cheat and claim money without providing the desired service or data.

– *Privacy*: Any party cannot view a patient's data until and unless it gets consent from the patient.

– *Data Security*: The data of a patient stands protected and cannot be tampered.

3.1.1 Contributions

We briefly discuss the salient features of our proposed patient-centric healthcare system that bring novelty to the design:

– We propose a *Secure and Smart Healthcare System* which coordinates the interaction between patient and hospital while the patient is getting treated;

– The proposed system ensures that no one has access to the patient's data stored in the medical database until the patient grants permission. Any external agent without access cannot tamper with the data. Any malicious behavior can be detected using the digital fingerprint of the data recorded in the blockchain.

– A patient cannot be overcharged for seeking treatment from a hospital. Simultaneously, a hospital has to start the treatment within a specified period. If they fail to do so, the patient can withdraw any deposit made.

– A patient can either keep the data with himself or store it in a medical database, which is assumed to be semi-trusted. The database owner checks the validity of the data provided by the patient before storing it in the database to prevent the storage of any spurious data.

– We have implemented the prototype in the Ethereum platform and Ropsten test network and have evaluated the performance. Code for the protocol is publicly available on GitHub[*].

3.1.2 Organization

The rest of the chapter is structured as follows - we have discussed the state-of-the-art in Section 3.2. In Section 3.3, the system model and high-level view of our construction are presented. In Section 3.5, we have addressed our security claims. Section 3.6 shows the results of our proposed system and also discusses the outcome. Finally, we have concluded the chapter in Section 3.7.

3.2 Related Work

We discuss the state-of-the-art blockchain solution for the healthcare system. Although state-of-the-art tries to enhance the security of the healthcare system using blockchain framework, these have certain drawbacks. Xiao et al. [173] have proposed a blockchain architecture model to store and enable different parties to view EHRs. However, this blockchain model is prone to a single point of failure. The current healthcare system struggles with fragmented, non-interoperable, and non-interpretable EHRs, complicating data sharing and analysis. To address this, Talukder et al. [155] propose an Ethereum-based "Proof of Disease" (PoD) consensus protocol. This protocol aims to ensure a consistent, secure, and easily interpretable version of medical information, potentially transforming health data management and utilization. However, the system faces significant challenges in adoption, regulatory compliance, and technical complexity. Xia et al. [172] had proposed a cloud-based blockchain platform for sharing files with untrustworthy parties seeking access to medical files. However, the solution is not scalable and suffers from key management problems. Jiang et al. [79] designed a medical data exchange system using blockchain by developing off-chain and on-chain verification for the security of the system's storage. Their work has addressed the problem of scalability. However, the solution does not guarantee the fairness of the entities involved.

The data preservation system in work proposed by Li et al. [91] basically contains two programs - the data access program and the blockchain interaction program. Zhang et al. [179] proposed PSN-based healthcare by designing two protocols for the authentication and sharing of healthcare data. The drawback of these two systems is that it lacks a data access control policy. Additionally, [179] does not provide a protocol for sharing EHR.

A *healthcare data gateway* was proposed by Yup et al. [177]. It is a blockchain approach to healthcare intelligence to address users' privacy by proposing a data access control for privacy. Liang et al. [97] proposed a mobile-based healthcare record-sharing system using blockchain. They designed a secure user-centric approach to provide access control and privacy using a channel formation scheme. Zhang and Poslad [180] proposed an access control policy for electronic medical records with finer granular access. Yang and Li [176] proposed an architecture for securing EHR based on distributed ledger technology. However, these works lack any formal algorithm or proper implementation, and the authors have not evaluated system performance.

Table 3.1: Drawbacks of State-of-the-Art Healthcare Systems

Related Works	Drawbacks
[173]	Single point of failure
[97], [176], [177], [180]	Lack implementation and do not evaluate system's performance
[97], [172], [176], [179], [180]	Lack proper framework
[179]	Do not provide data access control policy
[97], [176], [177], [180]	No algorithm is given for the protocol
[59], [79], [152], [176], [177], [179], [180]	Lack EHR sharing protocol
[172], [173]	Scalability issue
[63], [97]	Interoperability problem
[172]	Key management issue
[79], [152]	Performance and fairness issues
[50], [59], [152]	Require high Storage, power and/or computation cost

Fan et al. [50] proposed an improved consensus mechanism to enhance the security and privacy of medical data. Sun et al. [152] designed a distributed attribute-based signature scheme for medical systems based on blockchain and proposed a blockchain-based record-sharing protocol. Gorenflo et al. [59] proposed a performance optimization for the Hyperledger blockchain framework. However, this work requires high storage, high power and computation cost.

Table 3.1 has summarized the drawbacks of various healthcare improvement proposals. Our model has addressed most of the existing problems by proposing a decentralized, distributed healthcare system using a permissionless blockchain framework. It also ensures the privacy of patient's medical data (using the access control policy), data security, and fairness of various entities involved in the system.

3.3 High-Level View of The System

Remark 1. *The problem is to design a patient-centric healthcare system that ensures privacy, as well as security of patient's medical data and fairness of the entities involved. Specifically, the system must enforce access control using a blockchain-based access control matrix that is resistant to malicious interventions. Digital footprints of medical data are stored on the blockchain to ensure integrity and immutability, while digital signatures provide accountability. To uphold fairness, the system penalizes hospital authorities for extortion or denial of treatment and discourages patients from making false claims or denying received services through smart contract logic.*

3.3.1 System Model

The major actors or parties involved in our proposed system, as shown in Figure 3.1, are as follows -

1. **Patient (P)/User**

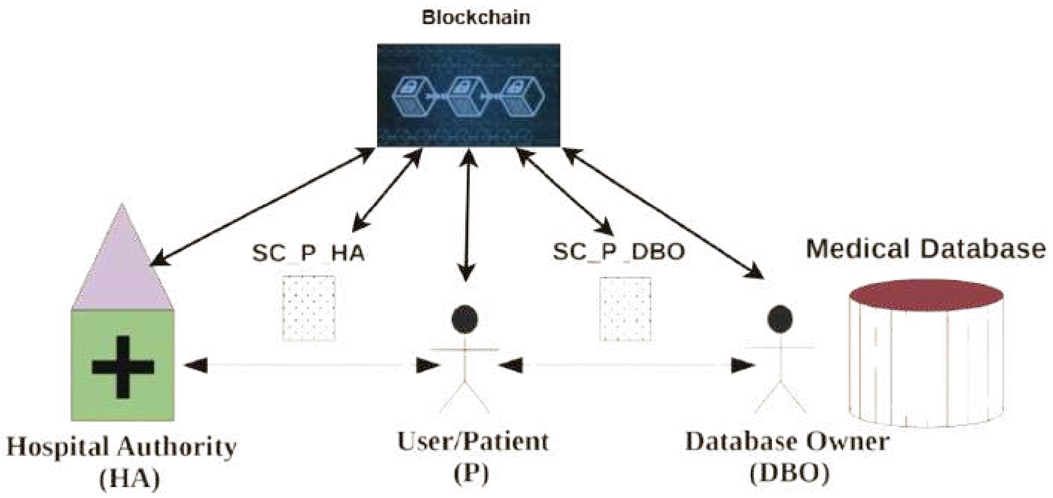


FIGURE 3.1: *Architecture of the proposed healthcare system illustrating the connections between key components*

2. **Hospital Authority (HA)**
3. **Database Owner (DBO)**

In our system, we have the following smart contracts. The protocol suite is written in the form of functions inside the smart contracts.

– Smart Contract between Patient and Hospital Authority: **SC_P_HA**

– Smart Contract between Patient and Database Owner: **SC_P_DBO**

We introduce an additional smart contract, **SC_Registration**, where system entities register themselves before participating in the protocol. Thus, the system is built using three smart contracts. Blockchain's immutability ensures tamper-proof patient data, while smart contracts enforce secure and automated access controls. For example, every access request and update to EHRs is recorded on the blockchain, providing a transparent audit trail.

In the following subsections, we discuss how these actors interact within the system. Before that, we outline the assumptions taken.

3.3.2 Assumptions

a) In our system, every single party has a unique ID. A smart contract generates these IDs at the time of registration in the system. One unique ID corresponds to a particular *PublicKey*. Every party must register first to be a part of the system.

b) Medical Data Repository Owner or simply the Database Owner (DBO) is considered semi-trusted in our system. DBO can have open access to the stored data as the data is stored in plain-text form. However, this assumption can be removed by adding the scope of handling encrypted data in the database, and the model can be changed accordingly. The DBO's activity log is maintained in the blockchain. In case DBO misbehaves, then it can be questioned and penalized accordingly. Patients can also keep their records in their private storage locally or appeal to the DBO to remove their records in case of privacy concerns.

c) Database Owner (DBO) must satisfy certain prerequisite conditions to be a part of the system and appeal to the government expressing their interest. The conditions or criteria may vary for different Governments of various countries. If all the necessary criteria are satisfied, the government introduces DBO into the system.

3.3.3 Communication Protocol between Patient and Hospital

When a patient visits a hospital, the hospital initially analyzes the patient's problems. After preliminary scanning, the hospital generates an estimated cost of the treatment. Hospitals and patients need to lock this amount in the smart contract (SC_P_HA).

Access Control: For accessing the patient's medical records from the medical repository, the hospital authority asks the patient to grant proper access permission. The hospital can read the patient's medical history if permission is granted. The patient can revoke access permission, if needed, at any instant. Information related to this access control is stored in the blockchain.

Locking of Hospital Treatment Cost: The patient locks the estimated cost in the smart contract. Then, the hospital must start the treatment within a fixed time window and register the timestamp of treatment in the blockchain. If the hospital fails to do so, the patient can unlock their money.

Storing Patient's Record after Treatment: The hospital generates the patient's medical files - reports, prescriptions, etc. However, these files are not transferred to the patient immediately because of security reasons, which are addressed in Section 3.5. With the help of some cryptographic computations and fair exchange protocol, as shown in Figure 3.2, the hospital sends the medical files to the patient. The hospital stores the following crucial attributes as the metadata in the blockchain - MR of the file chunks M_1, MR of the encrypted file M_2, the signature of the hospital on MR of the file, and the signature of the hospital on H(Patient ID || Date of Report || MR of Encrypted File). MRs M_1 and M_2 are used for verification by the protocol on behalf of the patient and other associated entities like DBO. The signature on the MR makes the hospital accountable for its encrypted file. The signature on the hash of the patient's attributes and the hash of the encrypted file add much more accountability, giving the patient the chance to complain if the hospital misbehaves.

Upon receiving these file attributes offline, the patient verifies and gives consent. If the attributes match, the patient invokes a function to give consent and sign on the file. If the patient finds a mismatch in the file attributes, s/he can withdraw the locked amount.

Hospital Bill Settlement: Meanwhile, the hospital provides the final medical bill to the patient. We assume that the final bill amount is not greater than the

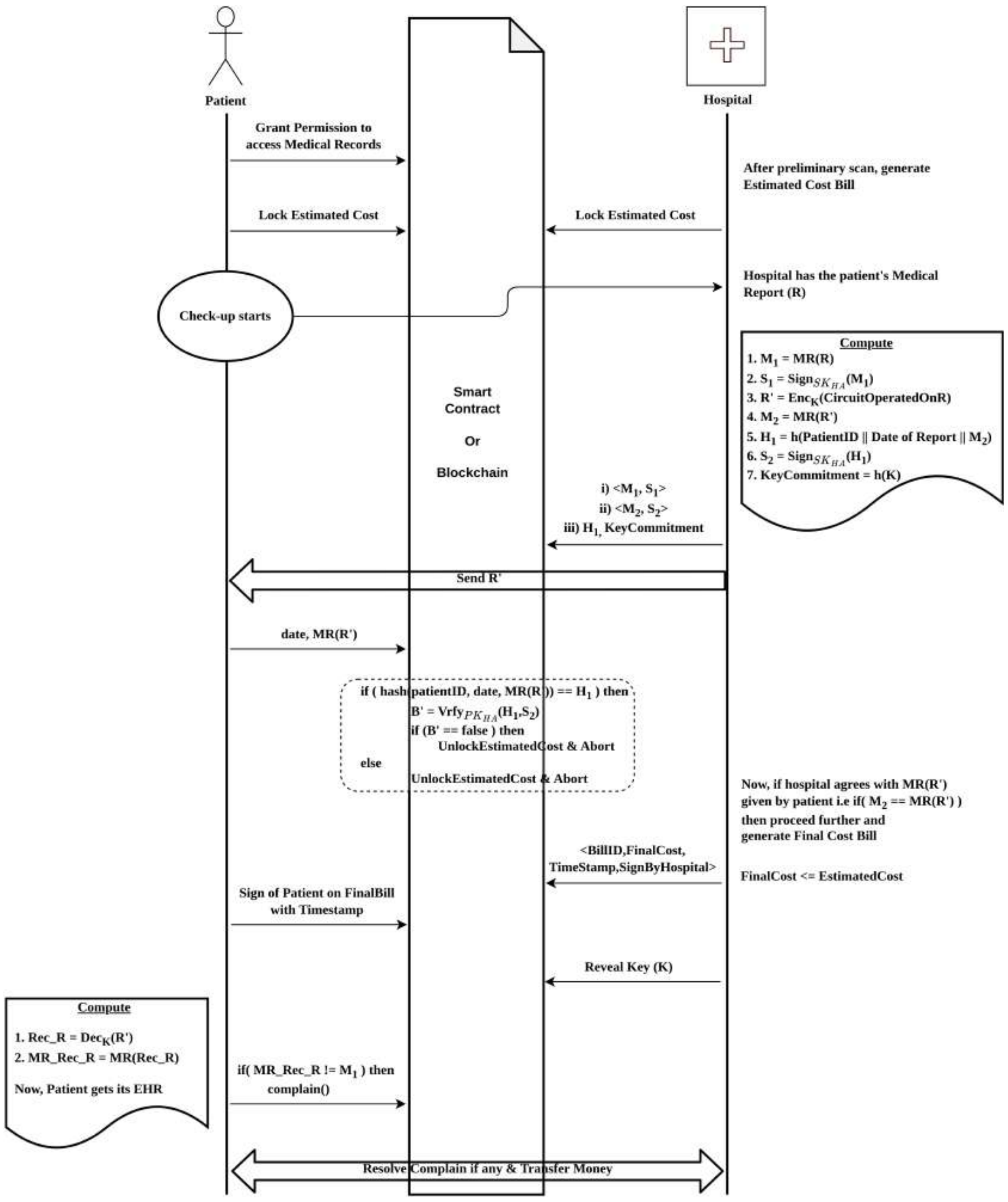

Patient
Smart Contract Or Blockchain
Hospital
Grant Permission to access Medical Records
After preliminary scan, generate Estimated Cost Bill
Lock Estimated Cost
Lock Estimated Cost
Hospital has the patient's Medical Report (R)
Check-up starts
Compute
1. M_1 = MR(R)
2. S_1 = $\mathrm{Sign}_{SK_{HA}}(M_1)$
3. R' = Enc_K(CircuitOperatedOnR)
4. M_2 = MR(R')
5. H_1 = h(PatientID \|\| Date of Report \|\| M_2)
6. S_2 = $\mathrm{Sign}_{SK_{HA}}(H_1)$
7. KeyCommitment = h(K)
i) <M_1, S_1>
ii) <M_2, S_2>
iii) H_1, KeyCommitment
Send R'
date, MR(R')
if (hash(patientID, date, MR(R')) == H_1) then
B' = $\mathrm{Vrfy}_{PK_{HA}}(H_1, S_2)$
if (B' == false) then
UnlockEstimatedCost & Abort
else
UnlockEstimatedCost & Abort
Now, if hospital agrees with MR(R') given by patient i.e if(M_2 == MR(R')) then proceed further and generate Final Cost Bill
<BillID,FinalCost, TimeStamp,SignByHospital>
FinalCost <= EstimatedCost
Sign of Patient on FinalBill with Timestamp
Reveal Key (K)
Compute
1. Rec_R = Dec_K(R')
2. MR_Rec_R = MR(Rec_R)
Now, Patient gets its EHR
if(MR_Rec_R != M_1) then complain()
Resolve Complain if any & Transfer Money

FIGURE 3.2: *Interaction between Patient and Hospital*

estimated cost. The patient has two options: give consent or raise a dispute for being overcharged. The hospital and patient agree on the price through offline communication. Furthermore, this pathway involves two additional transactions by the parties before agreeing to the revised final bill.

Receiving the patient's consent on the final bill, the hospital sends the decryption key to the patient. Suppose the MR of the decrypted file does not match the one in the contract. In that case, the patient raises a complaint by providing a *Proof of Misbehavior* [45]. In that case, the patient provides the positions and the two witnesses corresponding to both file chunks and encrypted file chunks where a mismatch occurred. It also provides intermediate nodes in the Merkle tree as proof so that the root can be reconstructed. Once the complaint is verified and the counterparty is found to be malicious, s/he is penalized accordingly. Vice versa, if a false complaint has been raised, the party gets penalized. Suppose the patient does not respond within the time-out period. In that case, the hospital withdraws the locked money (patient's and hospital's) and aborts the protocol. There is a timeline check in all the functions to ensure that each process in the protocol runs within the allocated time window. Also, at each stage of the protocol, both parties are given functions to abort the protocol to avoid indefinite waiting if one of the parties stops responding. The above communication model between patient and hospital is depicted in Figure 3.2

Uploading Medical Data: When the treatment is completed successfully, the patient can store the medical files in their local storage devices or store them in some medical repository/cloud server owned by a semi-trusted third-party vendor (a.k.a. DBO). And DBO provides the storage space service for some charges. Any two-party fair exchange protocol can be used. In this protocol, we implement the **Fairswap Protocol** [45], as part of the smart contract **SC_P_DBO**, for ensuring a fair exchange of information between patient and DBO. The system leverages off-chain storage solutions for large data volumes while maintaining essential metadata on-chain, ensuring scalability without overloading the blockchain network.

Having described the architecture and components of the proposed healthcare system, we now transition to the implementation details. The next section focuses on the technologies, frameworks, and protocols used to translate the system design into a functional solution, validating its feasibility and robustness.

3.4 Implementation and Technical Details

Prior to describing the actual algorithms, let us discuss some important terminologies and notations that we have used in our implementation (Table 3.2). The readers can also refer to Table 3.3 for the various structure definitions used in the Algorithms. These algorithms are then converted to smart contract codes written in the Solidity programming language.

1. Entity Registration: The patient can sign up in the Healthcare System by calling the function **PatientRegistration** (Algorithm 1). The parameter $hash\_of\_Personal\_Info \leftarrow H(name||age||mob||address)$ refers to a hash value. It helps to preserve the patient's privacy. Once the registration is finished, the patient will receive a unique identifier. Similarly, other entities

Algorithm 1 Patient Registration Function

```
Function PatientRegistration($hash\_of\_Personal\_Info$)                ▷ Caller: $P$:
    if ( caller is already registered ) then
        Exit
    Generate: a unique $pID$
    Create: a mapping or association between $pAddr$ and corresponding $pID$
    Record: the $hash\_of\_Personal\_Info$ corresponding to $pID$ on BC
end
```

TABLE 3.2: Terminology & Notation used in our Scheme

Abbreviation	**Interpretation**
BC	Blockchain
SC	Smart Contract
P	Patient
HA	Hospital Authority
DBO	Database Owner
pAddr	Patient Address
hAddr	Hospital Address
dboAddr	Database Owner Address
pID	Patient ID
hID	Hospital ID
dboID	Database Owner ID
ebID	Estimated Cost Bill ID
fbID	Final Cost Bill ID
msID	Multi-Signature ID
CST	Current System Time
TTL	Threshold Time Limit

TABLE 3.3: Structure Definition

Structure Name	**Member Variables**
EstimatedCheckUpCost	ebID, pID, hID, estimatedCost, $T_{Estimate}$, $T_{LockingByHA}$, $T_{LockingByP}$, $T_{CheckUpStart}$, $T_{UnlockingByHA}$, $T_{UnlockingByP}$
FinalCheckUpCost	fbID, pID, hID, finalCost, $T_{FinalBilling}$, $T_{ComplaintByP}$, $T_{UnlockingByHA}$, $T_{UnlockingByP}$, $T_{FinalConsentByP}$
MultiSigOnMedicalData	msID, pID, hID, MR_MedData, MR_EncCircuitOperatedMedData, H_x, $Sign_{SK_{HA}}MR\_MedData$, $Sign_{SK_{HA}}x$, $T_{SigningByHA}$, $T_{VerificationByP}$, $T_{UnlockingByHA}$, $T_{UnlockingByP}$ *x=(pID||Date||MR_EncCircuitOperatedMedData)
FileProperties	fileSize, fileChunkSize, depth

can also register in the system -
HA: $hID \leftarrow$ HospitalRegistration($name$)
However, a **DBO** cannot register itself directly into the system. A trusted third party or TTP (here, we consider the Government as TTP) is responsible for introducing a DBO into the system.

2. Providing Medical Services to the Patient and Settling the Final Bill: Here, we will discuss the algorithms related to generating the new medical reports or EHRs of a patient, fairly communicating the same to the patient party, preserving the integrity of the files, generating the final cost bill, and finally, discharging the patient by settling the final bill and transferring EHR to the patient. We break it into two modules - **I. Pre-Treatment** and **II. Generation and Transfer of EHR to Patient and Settlement of Final Bill**, described in Algorithm 2 and Algorithm 3 respectively.

I. *The Algorithms for* ***Pre-Treatment*** *(Algorithm 2)*

–The patient, having ID - pID, reports to the hospital the medical problem s/he is facing. Hospital, having ID hID, estimates the cost for the medical service and locks the amount with the invocation of the ***GenerateEstimatedCostBill*** function (say, at time $T1$). The function generates a unique estimated cost bill ID - $ebID$ and also instantiates a structure of $EstimatedCheckUpCost$ with this $ebID$. The $ebID$ is communicated with the patient.

–Upon receiving the $ebID$, the patient needs to invoke the function ***LockEstimatedAmount*** (say, at time $T2$) within a limited time period (TTL) to lock the *estimatedCost* amount.

Algorithm 2 Pre-Treatment Functions

```
Function GenerateEstimatedCostBill($pID$, $estimatedCost$)                ▷ Caller: $HA$:
    Check: if (caller's ID == $hID$)
    Check: if (msg.value == $estimatedCost$)
    Generate: a unique $ebID$
    Instantiate: a structure $EstimatedCheckUpCost$ $ec$
    Lock: the $estimatedCost$ in the SC
    Record: the CST (say, $T1$) on BC
end
Function LockEstimatedAmount($hID$, $ebID$)                               ▷ Caller: $P$:
    Check: if (caller's ID == $pID$)
    Fetch: the structure from BC : $EstimatedCheckUpCost$ $ec$ (corresponding to $ebID$)
    if ( (CST - T1) > TTL OR ( msg.value $\neq$ ec.estimatedCost) ) then
        Exit
    Lock: the $estimatedCost$ amount in the SC
    Record: the CST (say $T2$) on BC
end
```

II. *The Algorithms for* ***Generating and Transferring EHR and Settlement of Final Bill*** *(Algorithm 3)*

–Upon patient locking the estimated amount, the hospital needs to invoke the function ***StartTreatment*** (say, at time $T3$) within a threshold time limit (TTL) to start the treatment process. The medical files are generated after the function invocation. Considering the variable nature of medical processes, there is no set time limit for the next immediate function - (***KeepSignedHashToBlockchain***)'s invocation.

–The generated medical report (R)is broken into chunks of predetermined size. A Merkle tree is constructed with the chunks, and the Merkle root as $M_1 \leftarrow MR(R)$ is obtained. The chunks are operated according to a circuit. In our case, the Boolean circuit maps to the construction of the Merkle tree. Hash is the primary operation of the gates used. In any gate of this circuit, the child gate's output vectors are concatenated and used as the inputs to the next predecessor gate (i.e., bottom-up approach). The concatenated vector is hashed, and the resultant value is the output of the gate. The gate outputs are encrypted with a key, and all the encrypted chunks are used to construct a different Merkle tree and obtain a Merkle root as $M_2 \leftarrow MR(Enc(CircuitOperatedOnR))$. This Merkle root is concatenated with the patient ID, date of report, etc. It is hashed as $H_1 \leftarrow H(pID||Date||Enc(CircuitOperatedOnR))$ and signed as $S_2 \leftarrow Sign_{SK_{HA}}(H_1)$. The hospital also signs on the Merkle root of the original file R as $S_1 \leftarrow Sign_{SK_{HA}}(MR(R))$. The two hashes - M_1 and M_2, the corresponding signatures - S_1 and S_2, and a few file-related variables are the parameters of the function ***KeepSignedHashToBlockchain***. Let us say the function is invoked at the time $T4$.

–The patient, by this time, might have received the encrypted file chunks from the hospital. The patient constructs the Merkle tree with the received $Enc(CircuitOperatedOnR)$ and checks whether the constructed Merkle root and the root kept on the blockchain are the same. The patient also verifies the matching of the concatenated hash (H_1) discussed in the last function. The patient also verifies the two signatures - S_1 and S_2. If these parameters are valid, then the patient invokes the function ***VerifyAnd-GiveConsent*** (say, at time $T5$) within a limited period (TTL).

–Next, the hospital invokes the function ***DischargeAndGenerateFinal-CostBill*** (say, at time $T6$) within a limited period (TTL) specifying the final cost - $finalCost$. It is important to note that the final cost is always less than or equal to the estimated cost because the estimated cost is an overestimation of the actual treatment cost, serving as an upper bound.

–The patient calls the function ***consentOnFinalBill*** (say, at time $T7$) if the final cost provided by the hospital is satisfactory.

–The hospital reveals the key by invoking the function ***KeyReveal*** (say, at time $T8$) within a limited time period (TTL). Then the key is hashed and checked with the key's commitment stored in the contract.

–When the hospital discloses the key in the blockchain, the patient decrypts the file chunks and performs the circuit operation of the gates one by one. If all the checks are valid, then the patient gives consent by invoking the function ***PatientFinalConsent*** (say, at time $T9$) within a limited period (TTL) and then the service charges are transferred to the hospital. Also, the locking amount gets unlocked and transferred to the individual parties. If the above check finds a mismatch, the patient lodges a complaint with the SC, providing proof of misbehavior, as defined in Fairswap protocol [45]. SC verifies the legitimacy of the complaint and identifies the faulty party for appropriate penalties.

After completing treatment, the patient can choose to store their EHRs on their local device or upload it to a repository or cloud storage. If the patient opts for repository storage, they must pay service charges to DBO. A secure fair exchange protocol facilitates data exchange between the patient and DBO. We have implemented the fairswap protocol [45] for this purpose. However, since this is not central to our thesis, details are omitted here. For further information, refer to the original paper. The implementation code is available on GitHub[*] for reference. The functional sequence diagram is depicted in Figure 3.3.

With the technical implementation of the proposed system detailed, we now proceed to analyze the security of our system. This section evaluates the robustness of the system against potential threats, demonstrating how the design ensures fairness, privacy, and data integrity while addressing key security challenges.

Algorithm 3 Generating and Transferring EHR and Settlement of Final Bill

```
Function StartTreatment($pID$)                                              ▷ Caller: $HA$:
    Check: if ($caller's\ ID == hID$)
    Generate: the medical file or EHR of the patient with $pID$
    Record: the CST (say $T3$) on BC
end
Function KeepSignedHashToBlockchain($pID, M1, S1, H1, M2, S2, keyCommitment, EHRParams$) ▷
Caller: $HA$:
    Check: if ($caller's\ ID == hID$)
    Generate: a unique $msID$
    Instantiate: a structure $MultisigOnMedicalData$
    Populate: $MultisigOnMedicalData$ with $M1$, $S1$, $H1$,$M2$, $S2$, $keyCommitment$
    Instantiate: a structure $FileProperties$
    Populate: $FileProperties$ with $EHRParams$
    Record: the CST (say, $T4$) on BC
end
Function VerifyAndGiveConsent($msID, H(pID||Date||offlineReceivedEncryptedData)$)    ▷ Caller: $P$:
    Check: if ($caller's\ ID == pID$)
    if ($CST - T4) > TTL$) then
        Exit
    Fetch: the structure from BC : $MultisigOnMedicalData$ corresponding to $msID$
    Verify: if the two hashes are matched and also their respective signatures were signed by the right $HA$
    Record: the CST (say, $T5$) on BC
end
Function DischargeAndGenerateFinalCostBill($ebID, pID, finalCost$)         ▷ Caller: $HA$:
    Check: if ($caller's\ ID == hID$)
    if ($CST - T5) > TTL$) then
        Exit
    Fetch: the structure from BC : $estimatedCost$ corresponding to $ebID$
    if ($finalCost > estimatedCost$) then
        Exit
    Generate: a unique $fbID$
    Instantiate: a structure $FinalCheckUpCost$ and update the $finalCost$
    Record: the CST (say, $T6$) on BC
end
Function ConsentOnFinalBill($fbID, hID$)                                    ▷ Caller: $P$:
    Check: if ($caller's\ ID == pID$)
    if ($CST - T6) > TTL$) then
        Exit
    Fetch: the structure from BC : $FinalCheckUpCost$ corresponding to $fbID$
    Validate: necessary members of struct $FinalCheckUpCost$
    Record: the CST (say, $T7$) on BC
end
Function KeyReveal($pID, msID, key$)                                        ▷ Caller: $HA$:
    Check: if ($caller's\ ID == hID$)
    if ($CST - T7) > TTL$) then
        Exit
    Fetch: the structure from BC : $MultisigOnMedicalData$ corresponding to $msID$
    Validate: if struct $MultisigOnMedicalData$ belongs to $pID$
    if ($H(key) \neq MultisigOnMedicalData.keyCommitment$) then
        Exit
    Store: the $key$ on the BC
    Record: the CST (say, $T8$) on BC
end
Function PatientFinalConsent($consent$)                                     ▷ Caller: $P$:
    Check: if ($caller's\ ID == pID$)
    if ($CST - T8) > TTL$) then
        Exit
    if ($consent == true$) then
        Transfer: final cost to the HA's account
        Unlock: the remaining locked amount and transfer the same to the respective parties
    Record the CST (say, $T9$) on BC;
end
```

3.5 Security Analysis

Blockchain technology uses some cryptographic primitives (e.g., hash function, digital signature). As long as the underlying cryptographic primitives are secured, the blockchain is secure, and so is our system. Assuming that the

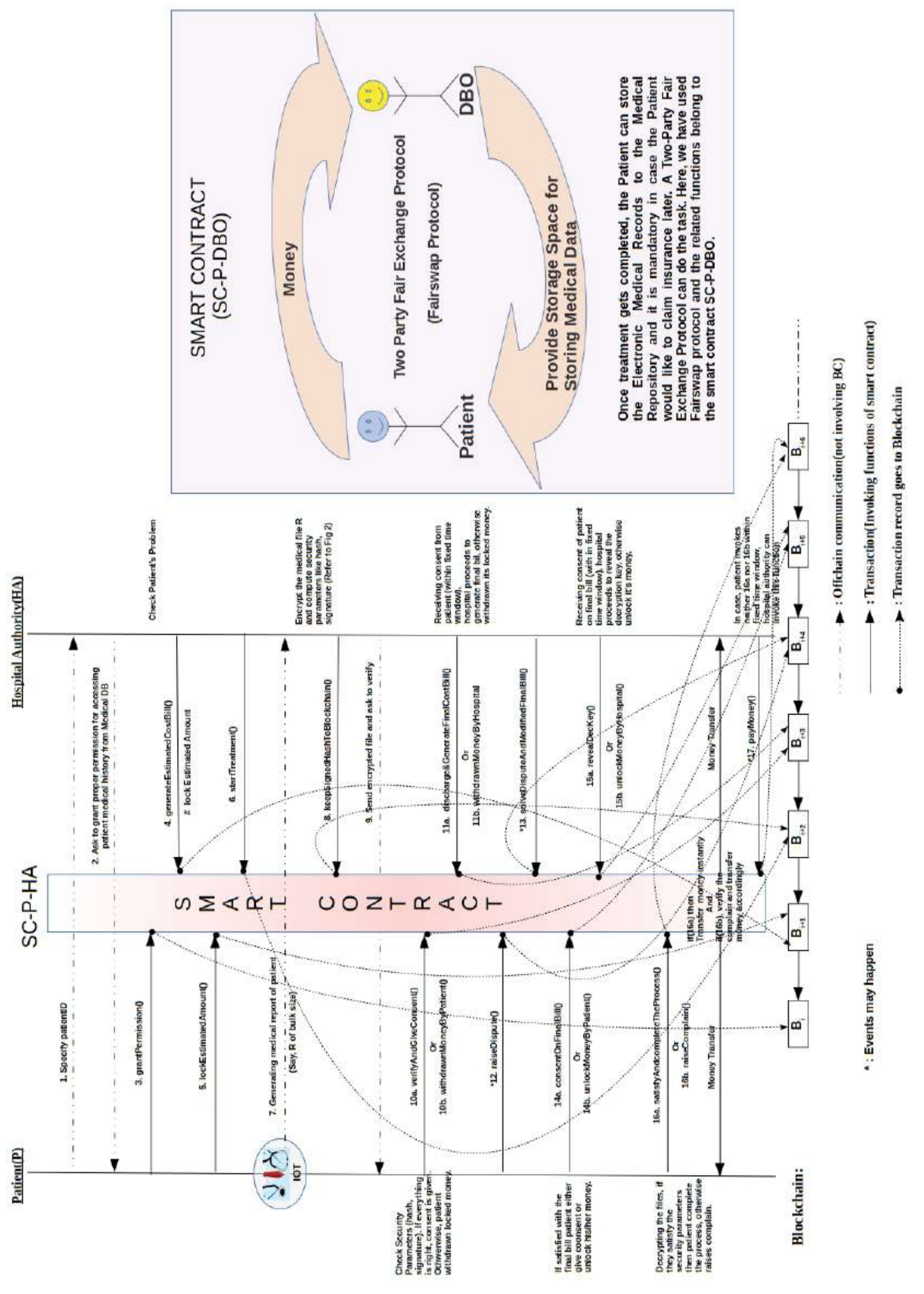

SMART CONTRACT (SC-P-DBO)
Money
Two Party Fair Exchange Protocol
(Fairswap Protocol)
Patient
DBO
Provide Storage Space for Storing Medical Data
Once treatment gets completed, the Patient can store the Electronic Medical Records to the Medical Repository and it is mandatory in case the Patient would like to claim insurance later. A Two-Party Fair Exchange Protocol can do the task. Here, we have used Fairswap protocol and the related functions belong to the smart contract SC-P-DBO.
Patient(P)
SC-P-HA
Hospital Authority(HA)
SMART CONTRACT
1. Specify patientID
2. Ask to grant proper permission for accessing patient medical history from Medical DB
3. grantPermission()
Check Patient's Problem
4. generateEstimatedCostBill()
5. lockEstimatedAmount()
# lock Estimated Amount
6. startTreatment()
7. Generating medical report of patient (Say, R of bulk size)
IOT
Encrypt the medical file R and compute security parameters like hash, signature (Refer to Fig 2)
8. keepSignedHashToBlockchain()
9. Send encrypted file and ask to verify
Check Security Parameters (hash, signature). If everything is right, consent is given. Otherwise, patient withdrawn locked money.
10a. verifyAndGiveConsent()
Or
10b. withdrawnMoneyByPatient()
Receiving consent from patient (within fixed time window), hospital proceeds to generate final bill, otherwise withdrawn its locked money.
11a. discharge&GenerateFinalCostBill()
Or
11b. withdrawnMoneyByHospital
*12. raiseDispute()
*13. solveDisputeAndModifiedFinalBill()
If satisfied with the final bill patient either give consent or unlock his/her money.
14a. consentOnFinalBill()
Or
14b. unlockMoneyByPatient()
Receiving consent of patient on final bill (with in fixed time window), hospital proceeds to reveal the decryption key, otherwise unlock it's money.
15a. revealDecKey()
Or
15b. unlockMoneyByHospital()
Decrypting the files, if they satisfy the security parameters then patient complete the process, otherwise raises complain.
16a. satisfyAndCompleteTheProcess()
Or
16b. raiseComplain()
If(16a) then Transfer money instantly And If(16b) verify the complain and transfer money accordingly
Money Transfer
In case, patient invokes neither 16a nor 16b within fixed time window, hospital authority can invoke this function
*17. payMoney()
Blockchain:
B_i
B_{i+1}
B_{i+2}
B_{i+3}
B_{i+4}
B_{i+5}
B_{i+6}
* : Events may happen
: Offchain communication(not involving BC)
: Transaction(Invoking functions of smart contract)
: Transaction record goes to Blockchain

FIGURE 3.3: *Functional Sequence Diagram of SC_P_HA and Fair Exchange Protocol between Patient and Database Owner*

blockchain is secure, the money locked in the blockchain is protected, and hence, the payment involved in the system is also safe. We claim that our system takes care of essential security aspects and provides fairness to the parties involved in the system.

3.5.1 Fairness

We discuss the fairness of each party, i.e., Patient and Hospital Authority. Even if one of the parties acts maliciously and tries to cheat, the malicious party is penalized, and the money is used to compensate the honest party.

Proposition 1. *(Patient's Fairness) The honest patient must not lose money or get mistreated, no matter if the other party (e.g., the hospital) behaving maliciously, under the assumption that the owner of the Medical Data Repository is semi-trusted and the underlying blockchain is secure.*

Proof. We will prove the proposition in the cases where an honest patient's interest might get compromised. Specifically, we will be analyzing the security on the grounds of price, data, and responsiveness.

When Hospital Authority (HA) is malicious:

HA tries to overcharge after treatment

Initially, at the time of admission, the Hospital Authority examines the medical history of the patient. It generates a bill with an estimated cost, and it gets recorded in the BC. If the patient agrees with it, then only s/he would proceed further to take the services from the hospital. The final cost of the check-up must not be higher than this estimated cost. Therefore, in our system, a patient knows the upper limit of the total check-up cost beforehand. It might be possible that the hospital has overcharged, or the patient is not fully satisfied with the treatment, then both the parties can negotiate on this bill and come to a joint agreement (however, it can never be higher than the estimated cost, that was given initially).

HA sends wrong medical files to patient

Medical Data or EHR (Electronic Health Record) is a crucial factor in determining the fairness of the protocol. The patients have to be ensured of receiving the correct file, and if not, then there is a provision for raising a complaint and aborting the protocol. Before actually sending the medical reports to the patient, the hospital first splits the file into multiple chunks of equal size. The Merkle root of the file chunks is obtained; the hospital keeps this Merkle root and the signature on it in the blockchain. Let this root be M_1. The hospital encrypts the entire Merkle tree's input, starting from the leaf level, which is the chunks of the file, to the intermediate level up to the root, using a key and then computes the Merkle root of the encrypted inputs. Let this root be M_2. This Merkle root is concatenated with the patient identifier and the date before being signed by the hospital. The encrypted file Merkle root, along with the digital signature, is put in the immutable blockchain (refer to $< M_2, S_2 >$ in Figure 3.2). Next, the hospital sends the encrypted inputs to the patient using some fair exchange protocol (in our case, we are using *fairswap* protocol [45]). Upon receiving the encrypted inputs, the patient re-computes the Merkle root, verifies the result with the stored root in the blockchain, and verifies the signatures to ensure that the file indeed came from the same hospital (proof of authenticity).

Date and patient identifier have been used along with M_2 for the signature to ensure that the hospital is held accountable if in case it sends encrypted inputs of a file belonging to a different person or sends a patient's previous medical file. If in case the hospital keeps the date and patient identifier correct with a mismatched file, the patient on deriving the file after the revelation of the key might come that the file is corrupted. The Merkle root of the file and the signature on it that was kept in the blockchain serve as a sufficient means to hold the hospital accountable for its maliciousness.

The key closely, coupled with the processed file, also needs to be rightly served to the patient to ensure a fair protocol. There are two ways in which the key can become an issue for the patient. In the first case, the hospital keeps the commitment of one key in the blockchain and reveals a different key to be used for decryption. In the second case, right after encrypting with a key, the hospital commits to a different key and reveals the same. In this case, the patient will decrypt the file with the second key and on finding a mismatch in the gate output computation, it will lodge a complaint, which will be resolved on-chain by penalizing the malicious party.

HA repudiates

In case the hospital authority provides a wrong file and computes the digital fingerprint as per the wrongly generated file. The contract cannot detect whether things are out of place. When the patient later detects it has received the wrong medical report, in such a case the transcript of the interaction between hospital and patient present in blockchain can be used for accusing the authorities and taking legal action if needed.

HA becomes unresponsive in the midst of the protocol

A timely response is necessary for a fair protocol. A party can stop responding, which can lead to an indefinite delay in the termination of the protocol. So, when the patient locks the estimated bill amount in the smart contract, the hospital needs to start the treatment within a fixed time window (a tunable parameter that can be set as per the system requirement). Hospital immediately invokes a function *StartTreatment* (defined in SC_P_HA) and updates the state. That essentially makes a transaction entry in the blockchain. Failure to start treatment within a limited period causes the patient to unlock their money. A patient can then seek treatment from some other hospitals. We can also penalize the hospital for this negligence (by deducting some amount of locked estimated cost of the hospital). That is the reason behind locking the estimated cost amount of the hospital. Suppose the hospital becomes unresponsive at any point in the protocol. In that case, the patient can call the respective exit function to abort the protocol, and we can also penalize a hospital for any such malpractice by deducting its locked money. □

Proposition 2. *(Hospital's Fairness) An honest hospital authority will receive money for all the services provided to the patient, despite the patient's misbehavior (say, the patient tries to take services from the hospital without paying the bill amount and then leaves), under the assumption that the underlying blockchain is secure.*

Proof. If a patient is malicious, they may try to cheat in the following ways:

Patient does not pay for the medical service

A patient can not deny paying the hospital after taking medical services. Because at the time of taking admission to the hospital, the hospital authority would guess the expected cost of the treatment. The patient is supposed to lock this amount in the smart contract beforehand, and only then the treatment will begin. Again, at the time of discharge, the hospital authority calculates the exact/final treatment cost. In our case, we have considered that the final cost should not be greater than the estimated cost. Once the patient gives consent over the final bill by putting a signature, then the patient receives all the medical files (i.e., Prescriptions, reports, etc.). Otherwise, the patient does not obtain these medical files (The patient can have the encrypted files, but the decryption key is still with the hospital. It is only revealed after the final bill has been locked in the contract by the patient).

Patient produces a wrong complaint

Receiving the correct file, the patient might try to raise a false complaint to exonerate money from the hospital without paying any money for the obtained services. The encrypted Merkle root ensures that the complaint should be part of the Merkle tree. This is used for checking the validity of the complaint, and the fact that the patient is lying gets detected.

Patient becomes unresponsive in the middle of protocol

Different exit points prevent an honest party from waiting indefinitely and keep its money locked in the smart contract. If this happens at any time in the middle of the protocol, the hospital can exit and unlock the money locked in the contract. □

3.5.2 Privacy

A patient's medical data is sensitive information. If a person's health record is available publicly, s/he may face embarrassment and might be subjected to discrimination in daily life. Hence, it must be ensured that access to patient data is provided only with the patient's consent.

Proposition 3. *(Patient's Privacy) In our proposed system, no entity can access a patient's data unless granted permission. At the same time, the patient's personally identifiable attributes remain hidden from public view.*

Proof. When a patient registers on the blockchain, they do not provide personal details such as name, age, address, and contact number. Instead, the hash of these attributes is stored in the blockchain at the time of registration. Our healthcare model prioritizes patient control by managing consent through smart contracts that record and enforce consent policies. Patients define an access control matrix, specifying which identities have access to their data, and can update this matrix to grant or revoke permissions as needed. This access control matrix is securely stored in the blockchain, ensuring that no one can access the patient's medical data without their explicit consent. □

3.5.3 Data Security

Tampering with a patient's medical records can have catastrophic consequences, such as administering incorrect medication dosages or altering diagnoses, which could endanger lives. In addition, users might attempt to falsify their medical records to submit fraudulent insurance claims, leading to financial exploitation and a loss of trust in the system. Ensuring the security of medical data while maintaining privacy is therefore critical.

Proposition 4. *(Data Security) In our proposed system, no individual, including the patient or the hospital, can tamper with medical data or misuse it for fraudulent purposes.*

Proof. Our protocol uses blockchain as its foundational framework to secure the system while addressing privacy concerns. Instead of storing medical data directly on the blockchain, we store its digital footprint—such as cryptographic hashes — to ensure integrity and immutability. The immutability of blockchain guarantees that any unauthorized modification to the data will be detectable. This approach not only prevents tampering and misuse but also safeguards patient privacy and reduces costs associated with on-chain storage. By combining off-chain storage of sensitive medical data with on-chain verification, our system effectively balances security, privacy, and cost-efficiency. □

3.6 Result and Discussion

Implementation Setup: We have implemented the Healthcare Management System on Ethereum test networks in a system having Intel(R) Core(TM) i7-6700HQ running Linux Mint 18.04 19.1 (Tessa), a 64-bit operating system using 16.00 GiB of RAM. We have used the Ropsten test network and an infura endpoint. Source code is available on GitHub repository[*].

The two main factors determining the feasibility of any blockchain model are the cost of implementation and the time required. The details of Ethereum transaction cost and latency are discussed in Chapter 2 - Section 2.6.10.

Table 3.4: Deployment Addresses of Smart Contracts

Smart Contract	**Address**
SC_Registration	0x5a818296705cC24Feec4CfEAF1DfdaE056fEf037
SC_P_HA_1	0x9528dA5753ae928Eb1e0284C7b1771e2FC17a766
SC_P_HA_2	0x7b88e153aC1b2BCA865CD58E1082f50Ed69f4c3c
SC_P_DBO	0xC062E1eF5EdB815bcF5B93C6BaD497ABCA407f31

Table 3.4 specifies the addresses of the deployed contracts. The contract deployment is a one-time occurrence. The transaction cost and the time taken for each contract deployment have been depicted in Figure 3.4 and Figure 3.5 respectively. The gas price was 18.9 Gwei, and the ether cost was 2300.54 dollars at the time of deployment. Depending upon the size of the contracts, the deployment cost varies (Table 3.5). These are one-time costs. So, once deployed,

[*] https://github.com/Debendranath-Das/Blockchain-Enabled-Secure-and-Smart-Healthcare-System

TABLE 3.5: Deployment Cost of Smart Contracts

Smart Contract	Deployment Cost(Ether)
SC_Registration	0.0353157
SC_P_HA_1	0.0994202
SC_P_HA_2	0.0583187
SC_P_DBO	0.04783231

we can get the benefits throughout the usage of this protocol.
The smart contracts for patients and hospitals (SC_P_HA_1 & SC_P_HA_2) have been split into two parts, citing the limited gas limit for blocks in Ethereum. The high gas for the collective patient and hospital contracts reflects the slightly higher steps involved in the protocol.

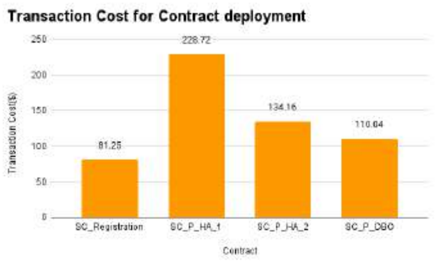


FIGURE 3.4: *Transaction Cost for Contract Deployment*

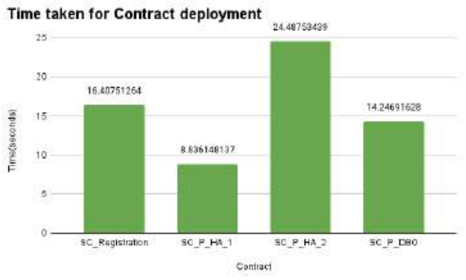


FIGURE 3.5: *Time taken for Contract Deployment*

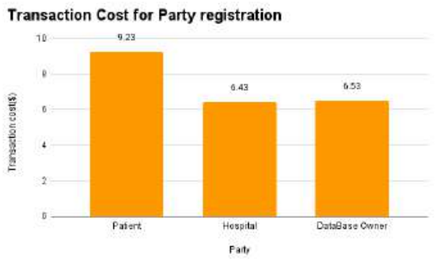


FIGURE 3.6: *Transaction Cost for Party Registration*

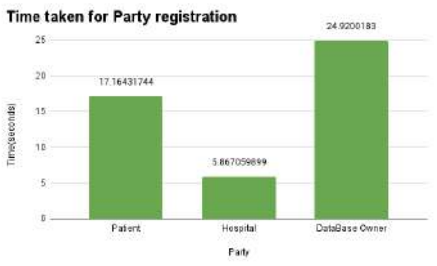


FIGURE 3.7: *Time taken for Party Registration*

The transaction costs and time taken associated with the entities' registration process, depicted in Figure 3.6 and Figure 3.7 respectively, are similar in scale for different parties except for the patients, which are slightly higher due to the few more variables involved in the registration for patients.

The transaction cost for certain functions involved in the protocol depends on the size of the files. The file size may vary depending upon the treatment and the corresponding result produced. The base file is constructed indifferently for subsequent usage in the protocol. The file is divided into numbers of chunks.

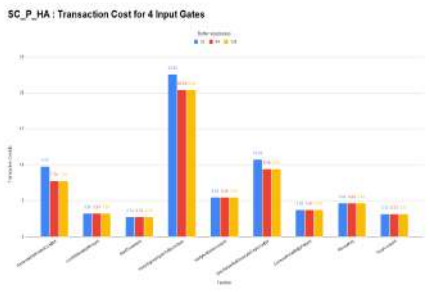


FIGURE 3.8: *SC_P_HA Transaction Cost for 4 Input Gates*

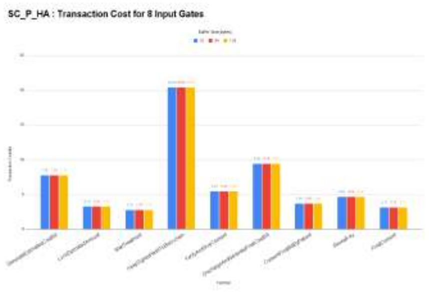


FIGURE 3.9: *SC_P_HA Transaction Cost for 8 Input Gates*

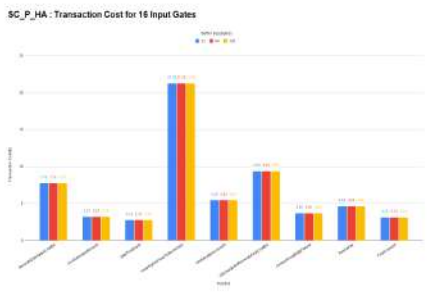


FIGURE 3.10: *SC_P_HA Transaction Cost for 16 Input Gates*

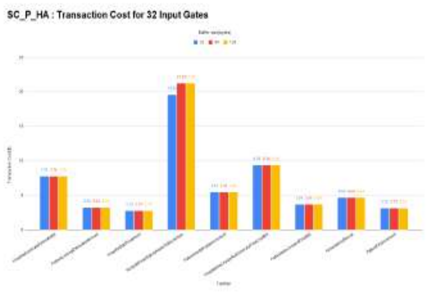


FIGURE 3.11: *SC_P_HA Transaction Cost for 32 Input Gates*

The number and size of the chunks are the varying parameters, referred to as the number of input gates and the buffer size of the gate, respectively.

We have shown the cost associated with various functions call for the contracts SC_P_HA (Figure 3.8, 3.9, 3.10, 3.11) with varying number of input gates and the buffer size. The number of input gates varies in the range of 4, 8, 16, 32 and the buffer sizes used are 32, 64, and 128. The file size can be derived by multiplying the number of input gates and the buffer size. So, a file having 4 input gates and a buffer size of 32 would have a file size of 128 bytes.

We find from the graphs of SC_P_HA (Figure 3.8, 3.9, 3.10, 3.11) that they follow a similar kind of trend with varying numbers of input gates. The transaction cost for different functions hardly varies while keeping the number of input gates constant and varying the buffer size. The graphs show that some functions require a higher cost due to their heavy functionalities. However, this is viable as the functions' utilities outweigh the cost.

3.7 Conclusion

We have discussed a novel *Secure and Smart Healthcare System* where every involved party's fairness is preserved without trusting each other. EHRs are tamper-proof and free from unauthorized access in our healthcare system enabled by blockchain technology. Our system also ensures that the patient's privacy does not get compromised. We proposed, prototyped, and deployed our healthcare system, which works fine in the private and Ropsten test networks. Experimental result shows the satisfactory outcome of various performance metrics. Our protocol demonstrates blockchain's capability and importance in the healthcare sector and proves that it could be the next revolutionary technology to replace current healthcare systems. Future work could focus on integrating the proposed system with existing healthcare standards, such as *HL7* [72], to enhance interoperability and ensure seamless adoption within current healthcare infrastructures.

In the next chapter, we will demonstrate a generalization of the above system incorporating the involvement of the Medical Insurance Company. In this, we aim to build trust between a policy buyer and an insurance company by making the processes transparent through the blockchain framework.

4

Blockchain-Enabled Secure Health Insurance Processing

"A tremendous amount of needless pain and suffering can be eliminated by ensuring that health insurance is universally available."

— **Daniel Akaka**

The insurance industry consists of businesses that provide policyholders with risk management services. The insurer guarantees protection against unforeseen adverse events. In contrast, the policyholder pays a premium in exchange for risk protection [52]. Nowadays, health insurance has become necessary as medical expenses continue to rise. Health insurance covers medical treatments, ensuring quality care without financial worries. However, the process of filing health insurance claims is often manual, complex and time-consuming, making it challenging for the patients (or policyholders) to receive the payments.

Challenges of Existing Health Insurance Processing System: The traditional health insurance processing system (Figure 4.1) involves a series of steps that begin with a patient seeking medical treatment from a healthcare provider. After providing the service, the healthcare provider charges the patient for the services rendered. The patient then submits a claim to the insurance company seeking reimbursement for the medical expenses incurred. The insurance company reviews the claim, verifies the services rendered, and decides on the reimbursement amount. Unfortunately, despite its widespread adoption, the legacy

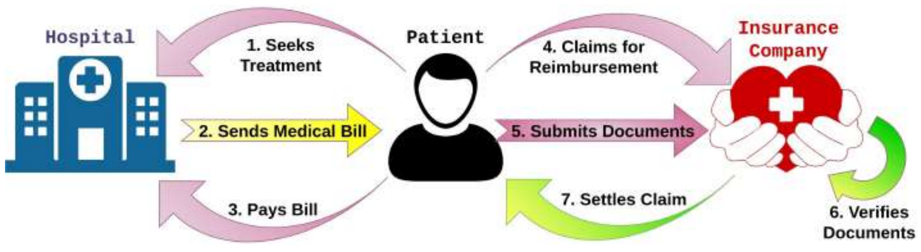


FIGURE 4.1: *Process of Legacy Health Insurance System*

health insurance processing system has several shortcomings (Figure 4.2) [52], [125].

– **Lack of Transparency:** The patients and healthcare providers are often unaware of the exact reimbursement process and how decisions are made. Sometimes, insurance providers cheat the policyholders during claim settlement by refusing to approve the financial reimbursement, citing various justifications. This leads to mistrust between policyholders and insurance providers, negatively impacting the insurance industry's reputation. On the other way, the policyholders may attempt to seek medical reimbursement from insurers by submitting false medical invoices. A recent survey report estimates that around 15% of the total claims in the health insurance industry are false [144]. However, today's insurance providers have limited ability to verify the authenticity of such claims, leading to potentially fraudulent activities [112]. These fraudulent activities can lead to significant losses for insurance companies, resulting in higher premiums for policyholders and decreased trust between the parties.

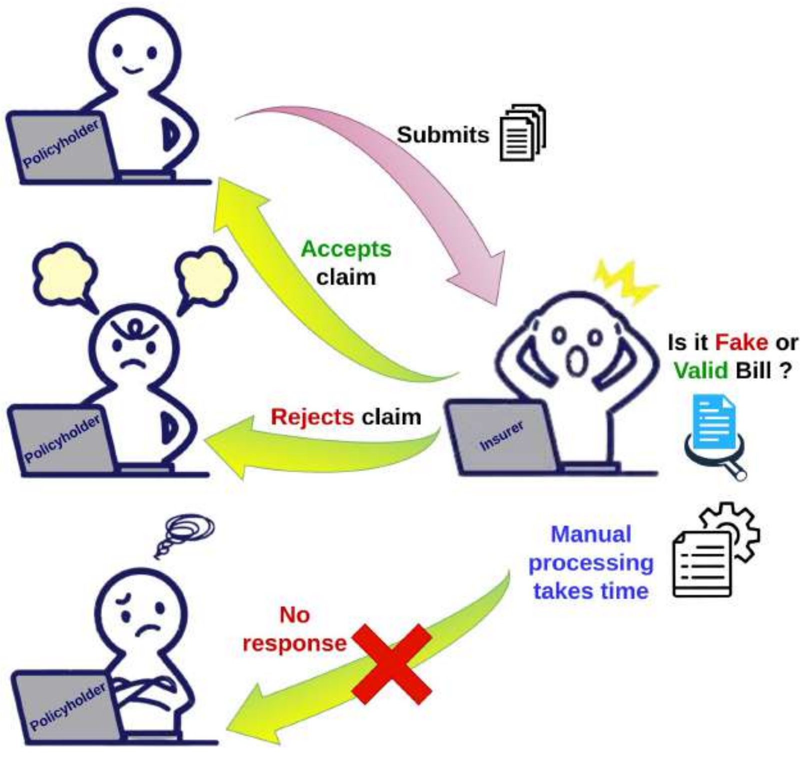


FIGURE 4.2: *Challenges of Legacy Health Insurance System*

– **Slow Processing:** The traditional system is slow and cumbersome, involving numerous intermediaries. While considerable automation processing has been accomplished in the current insurance systems, they continue to experience performance bottlenecks because of a single reliable source of transaction state data. In addition, the manual interactions required by the general insurance systems for various transaction procedures cause slow processing and long payment settlement times. This slow processing can be frustrating for patients who require quick reimbursement.

- **Higher Administrative Cost:** The legacy health insurance processing system has a higher administrative cost due to the need for multiple intermediaries and manual processing steps. These administrative costs also lead to higher insurance premiums, making healthcare insurance inaccessible for some individuals [4].

- **Poor Data Security and Privacy:** Another significant challenge of the traditional health insurance processing system is poor data security and privacy. Healthcare data breaches have become a significant concern in recent years, and the insurance industry has not been immune to such breaches. For example, in 2015, Anthem Inc. suffered a data breach that affected 78.8 million individuals, making it the largest healthcare data breach in history [168]. The same year, Premera Blue Cross [139] and Excellus BlueCross BlueShield [123] also experienced data breaches that affected 11 million and 10 million individuals, respectively. Another notable healthcare data breach in the insurance industry was the 2011 Health Net data breach, which affected 1.9 million individuals [30]. These incidents highlight the importance of implementing robust cybersecurity measures to protect sensitive patient data and prevent such breaches from occurring in the future [14], [87].

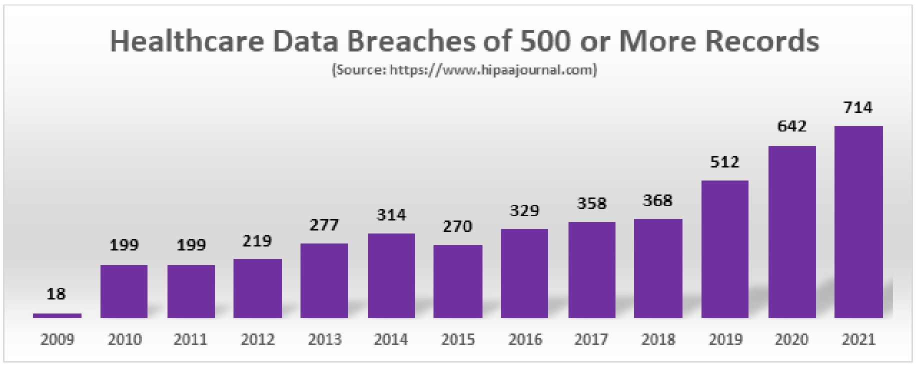


FIGURE 4.3: *Healthcare Data Breaches Statistics*

According to HIPAA, 912992 patient records were stolen or made public in March 2019. Compared to the average rate over the previous five years, there was an abrupt rise of 14% in data breaches [71]. As per the HIPAA Medical Journal, in 2018, an average healthcare data breach of 500 or more records was reported daily (Figure 4.3). According to these statistics, medical data is neither safe nor secure because it is sometimes utilized for personal or financial advantage or research [134]. Numerous types of insurance coverage are available (e.g., car, health, life insurance). The fundamental requirements for different insurance companies are essentially the same. However, health insurance requires significantly more focus. As it deals with patients' medical data, maintaining data privacy is one of the significant challenges.

Potential of Blockchain to Resolve the Existing Problems of Legacy Health Insurance Processing Systems: Introducing transparency to the

legacy health insurance processing systems through the use of blockchain technology can enable more trust among the involved parties. Blockchain technology can potentially revolutionize the health insurance industry by offering a secure, transparent, and efficient platform for processing claims. By leveraging the power of blockchain and smart contracts, healthcare providers can securely store and share medical data with insurance companies, streamlining the claims process and reducing the likelihood of errors and fraud [57], [124]. Smart contracts, which are self-executing computer programs, can automate insurance claims' verification and approval process, thereby reducing the need for intermediaries and eliminating the potential for human error. The smart contract-based distributed ledger will automatically stop any fraudulent transaction if the requests or actions do not adhere to the rules of the contracts. Also, blockchain technology can ensure that medical data is safe, can't be changed, and is easy to access. This cuts down on administrative costs and processing times.
In summary, a blockchain-based insurance industry should -

1. increase system transparency;
2. increase automation;
3. speed up key business processes, such as client registration, policy issuance, and claims processing;
4. eliminate the need for intermediaries or insurance agents;
5. facilitate fraud detection easier;
6. ensure the privacy of the client's data;
7. reduce administrative or operational costs, and
8. make it simpler for auditors to identify suspicious trading patterns [124].

4.1 Objectives

In light of the challenges posed by legacy-based health insurance systems, our objective is to explore the potential use of blockchain technology to design a fair, efficient health insurance framework with a shorter turnaround time while eliminating the need for trust assumptions. Given that blockchain offers many of the essential features required for this purpose, we aim to investigate its potential to realize the following properties:

- *Fairness*: An honest entity (whether policyholder or insurer) would never forfeit money, even if the other players are dishonest and make every effort to defraud.
- *Automation*: Introducing greater automation to the key stages of the insurance processing system to reduce manual interactions and operational costs.

– *Privacy*: A third party (e.g., an insurance company) cannot view a policyholder's data unless the user grants consent to access their data.

– *Data Security*: policyholder's medical data stands shielded and cannot be modified.

4.1.1 Contributions

The chapter aims to explore the potential of blockchain technology integrated with smart contracts to revolutionize the health insurance industry. The chapter will highlight the potential benefits of adopting blockchain technology through an in-depth analysis of the existing health insurance processing system. We briefly highlight the distinguishing features of the proposed system that make our model unique.

– We propose a *Smart and Transparent Health Insurance Processing System* which coordinates the interaction between policyholder and insurer during various phases of the insurance process, from policy offering to claim settlement. Our strategy can expedite key phases of the insurance process by using greater automation and online engagement.

– Fairness is assured for both the insured and the insurer. Nobody can manipulate another by engaging in malicious behavior.

– Our system ensures the privacy of users' data. Policyholders' data remains confidential and is only shared with those who have been granted authorised access. When needed, users can also revoke access permission.

– In our system, an insurance firm would rapidly determine whether a claim submitted by a policyholder is legitimate. Various critical security parameters are stored in the blockchain to prevent fraudulent activity, such as message digests, digital signatures, and so on.

– Because blockchain serves as the backbone of our system, some security features such as immutable data, transparency, and non-reliance on centralized authority are implicitly inherited.

– We have implemented and assessed the prototype on the Ethereum platform and the Ropsten test network. The protocol's source code is available on GitHub[*].

In a nutshell, we have focused on the health insurance industry and designed a blockchain-based solution to address its performance and security issues in this work.

[a]https://doi.org/10.5281/zenodo.8232704

4.1.2 Organization

The remaining chapter is structured as follows - We have discussed the state-of-the-art in Section 4.2. Section 4.3 presents the system model and high-level view of the construction. In Section 4.4, implementation details are discussed. Next, we have addressed our security assertions in Section 4.5. After that, Section 4.6 shows the results of our proposed system and discusses the outcome. Finally, we have concluded the chapter in Section 4.7.

4.2 Related Work

The finance industry was the first to incorporate blockchain technology, led by the pioneering cryptocurrency, Bitcoin [110]. Since then, numerous other cryptocurrencies have emerged and continue to grow in the financial market [31]. Ethereum [24], introduced by Buterin in 2014, expanded the possibilities of blockchain by integrating smart contracts, leading to its adoption across various sectors. Here, we will explore a state-of-the-art blockchain solution for insurance processing systems.

Using machine learning approaches to identify fraudulent auto insurance claims, [130] developed a system to lower risks and fraud in processing insurance claims. Their future goals include improving algorithms and processing efficiency to produce better results. [89] suggested employing blockchain's smart contracts and sensors to reduce fraud in auto insurance. Sensors continuously monitor the location and motion of the car, and updates are made in the blockchain's smart contract to reduce fraud. The B-FICA: Blockchain-based Framework for Auto-Insurance Claims and Adjudication [115] provides a two-way verification in which sensor and entity data are tracked to decrease the fictitious claims for automated vehicles.

Besides the car insurance system, a few works are also done on health insurance. [124] suggested a framework for processing health insurance claims using private blockchain Hyperledger Fabric. [148] elaborates on a framework for processing health insurance transactions using role-based access on parties using the Ethereum blockchain. [181] proposed MIStore, a blockchain-based solution for the medical insurance storage system. There are n servers in addition to patients, hospitals, and insurance providers. Few nodes are selected as servers, and the system's security heavily depends on these servers.

[57] use IBM Blockchain to secure insurance claim transactions and counter fraud attempts. The author of the papers [57] has not offered any solution for other essential functionalities except claim operation. Goyal et al. [62] also proposed a blockchain-based solution for the insurance management system employing a machine learning tool. They applied a random forest classifier for predicting the risk to compute the risk-rated premium rebate. Ismail et al. [76] suggest Block-HI, a blockchain-based healthcare insurance fraud detection architecture based on various fraud scenarios. Their study focused on detecting claim fraud; however, no precise picture of how insurance companies handle claims or offer policies was provided. [6] propose a blockchain-based solution

to enhance Ghana's National Health Insurance Scheme (NHIS), aiming to improve data management, transparency, and security while streamlining claims processing. The work is criticized for its limited scope, lack of comparative analysis, and insufficient evaluation, potentially oversimplifying the challenges of implementing blockchain in a healthcare setting.

[133] explore the potential of blockchain technology in transforming the insurance industry, emphasizing consensus models, distributed ledgers, and integrating technologies like smart contracts and machine learning. However, criticisms point to insufficient analysis of security and privacy issues, implementation constraints, and the need for more empirical evidence. [138] contribute a theoretical model for optimizing health insurance claims processes in Indonesia through blockchain. The proposed model has not been implemented yet. Therefore, there is no empirical testing of its performance with extensive data. Additionally, there is a lack of detailed information on data collection and processing methodologies.

[78] focus on health insurance fraud detection, proposing an insurance chain model with Hyperledger Fabric and leveraging Support Vector Machines and Random Forest for fraud claim detection. However, challenges such as high costs, complexity, data privacy, security, scalability, and limitations in empirical setup are acknowledged. Detailed security analysis and scalability challenges are potential drawbacks.

While the existing literature on blockchain-based insurance systems has explored several solutions, their implementations still have significant drawbacks. For example, many of these solutions focus solely on claim operations, neglecting other essential functionalities such as policy offering and handling [57], [158]. Few solutions rely heavily on a few servers, leaving the system vulnerable to security breaches [133], [181]. Furthermore, existing solutions need to adequately address data privacy concerns, especially in the context of healthcare insurance, where patient confidentiality is crucial [76]. Also, some papers convey ideas that are region-specific (e.g.,[6] on Ghana and [138] on Indonesia), so their proposed models may have limitations in terms of generalizability to diverse healthcare and insurance systems. We propose a decentralized, distributed health insurance processing system that utilizes a permission-less blockchain foundation to address these limitations. In addition, our solution incorporates an Access Control Policy to ensure patients' medical information remains private while ensuring all system participants are treated fairly. Unlike existing solutions that primarily focus on claim operations, our model addresses all critical functions of the insurance process, including policy offering and handling. By implementing our system, we can assure data security, protect patients' privacy, and promote fairness among all system participants. In Table 4.1, a comparative analysis of our model and existing solutions is presented, focusing on 7 key features.

4.3 High-Level View of The System

Remark 2. *The traditional health insurance processing systems face significant challenges, including inefficiencies in claims settlement, lack of transparency,*

TABLE 4.1: A Comparative Analysis of Current State-of-the-Art

Article	Policy offering	Timely management of claim	Access control mechanism	Privacy of EHR	Safety	Application Liveness	Implementation
[124]	✘	✔	✔	✔	✔	✘	✔
[148]	✘	✔	✘	✘	✔	✘	✔
[181]	✘	✘	✘	✔	✘	✘	✔
[57]	✘	✔	✘	✘	✔	✘	✘
[62]	✔	✔	✘	✘	✘	✘	✘
[76]	✘	✔	✘	✘	✘	✘	✔
[6]	✘	✘	✔	✔	✔	✘	✔
[133]	✘	✘	✔	✔	✔	✘	✘
[138]	✘	✔	✘	✔	✔	✘	✘
[78]	✔	✔	✘	✔	✔	✘	✔
Proposed Model	✔	✔	✔	✔	✔	✔	✔

and vulnerabilities to fraudulent activities. These issues not only delay the reimbursement process for policyholders but also impose financial and operational burdens on insurers. Furthermore, centralized architectures compromise data security and privacy, leaving sensitive medical information exposed to unauthorized access and tampering. In this context, it becomes imperative to design a system that ensures fairness, privacy, and data security, while eliminating reliance on trust assumptions. The problem is to create a framework where honest entities are protected from malicious behavior, sensitive data remains confidential, and the entire process is streamlined through automation. This chapter addresses these challenges by leveraging blockchain technology to propose a decentralized, secure, and transparent health insurance processing system that overcomes the limitations of legacy systems.

4.3.1 System Model

The significant actors or parties involved in our proposed system are as follows -

1. **Policyholder/User/Patient (P):** The entity that seeks treatment in healthcare (acted as *Patient*), purchases a health insurance policy and later claims for reimbursement of the medical bill (served as *Policyholder*). Sometimes simply called as *User*.

2. **Insurer/Insurance Company (IC):** The entity that offers and sells insurance. Insurance firms are financial organizations that offer direct insurance or reinsurance and provide financial protection against future risks. In this context, we are specifically interested in health *Insurance Company*, also referred to as *Insurer*.

3. **Database Owner (DBO):** The semi-trusted entity that the Government appoints in our blockchain ecosystem after checking their credibility. *Database Owner* provides storage space/database to store the patients' EHRs and, in return, obtains money. We can think of a DBO as today's cloud service providers such as iCloud, Google Drive, Microsoft One Drive, Dropbox, etc.

Figure 4.4, illustrates the various actors involved in the system and also sketches the smart contracts between multiple entities. It is worth mentioning that part of the figure (surrounded by a red dotted box) depicts our prior work [39], in which we provided a prototype for secure patient-hospital interaction employing the blockchain framework, and this work extends the same.

We make certain that all parties must safely and reliably communicate with one another. The system has a few established protocols or regulations that must be adhered to by all parties involved. The protocol suite is written using functions in smart contracts. We have the following smart contracts in our system.

– Smart Contract between Policyholder (P), Insurance Company (IC) and Database Owner (DBO): **SC_P_IC_DBO**

– Smart Contract between Patient (P) and Database Owner (DBO): **SC_P_DBO**

Additionally, we have an auxiliary smart contract named **SC_Registration**, in which system entities should register themselves prior to participating in the protocol. Therefore, the system consists of three smart contracts. In the following subsections, we will examine how these various system actors interact with one another. Before we continue, let us lay out a few fundamental truths or presumptions of our system.

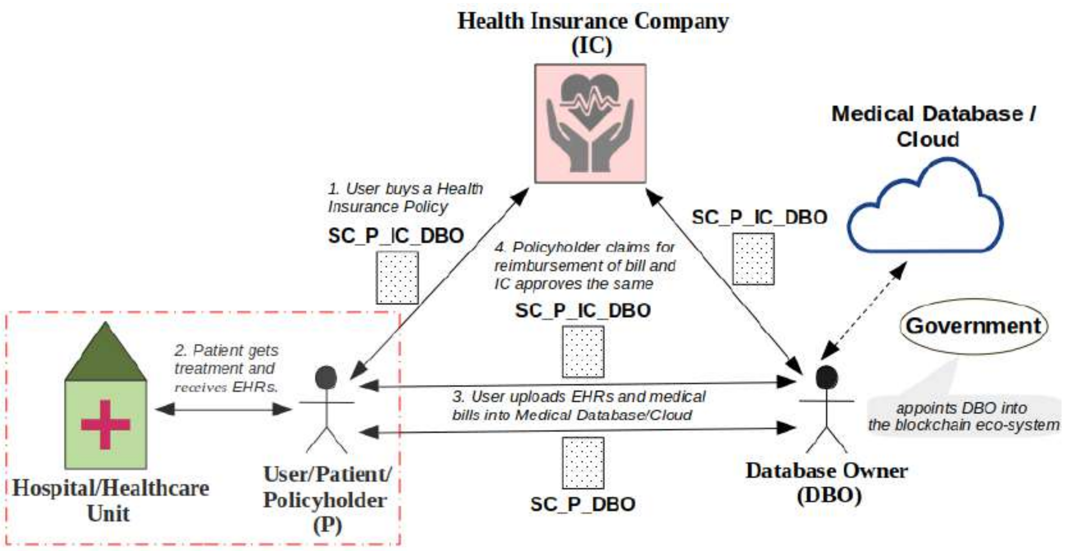

Health Insurance Company (IC)
Medical Database / Cloud
1. User buys a Health Insurance Policy
SC_P_IC_DBO
SC_P_IC_DBO
4. Policyholder claims for reimbursement of bill and IC approves the same
SC_P_IC_DBO
SC_P_IC_DBO
Government
2. Patient gets treatment and receives EHRs.
3. User uploads EHRs and medical bills into Medical Database/Cloud
appoints DBO into the blockchain eco-system
Hospital/Healthcare Unit
User/Patient/ Policyholder (P)
SC_P_DBO
Database Owner (DBO)

FIGURE 4.4: *System Model*

4.3.2 Assumptions

1. Every entity must register first before participating in our system. Every party has a unique ID generated by the smart contract during registration. One unique ID maps to a particular *PublicKey*.

2. Our system considers the Medical Data Repository Owner or the Database Owner (DBO) semi-trusted. The government selects DBOs only after

checking their credibility. The norms of each country stipulate that DBOs must meet specific prerequisite standards. Even though they are registered and trusted by the government, Our system maintains the activity log of DBOs in the blockchain to hold them accountable for their actions, and they are required to adhere to the system protocols; hence, they are *semi-trusted*. A DBO may operate multiple nodes on the blockchain network in order to increase the network's accessibility and decentralization.

3. The users or policyholders must maintain their Electronic Health Records (EHRs) in some medical data repository to claim reimbursement against treatment costs from the insurer. In this regard, a two-party fair exchange protocol (e.g. Fairswap [45]) is executed between the user and DBO. Our earlier chapter [39], can be consulted for a deeper understanding and linkability with this study.

4. In our system, a user can buy a policy by paying the policy price decided by the insurer as a one-time payment, unlike most policy schemes where the user needs to pay a premium in a periodical manner.

5. Health insurance claims are primarily two types - Cashless Claims and Reimbursement Claims. A cashless facility means the insurer pays directly to the network hospital for the treatment of the insured as per the policy conditions. In contrast, a reimbursement claim means settling the hospital bill out-of-pocket and then applying for reimbursement from the insurance company. Out of the two, here we implemented the system considering the second one.

4.3.3 Communication Protocol between Policyholder and Health Insurance Company

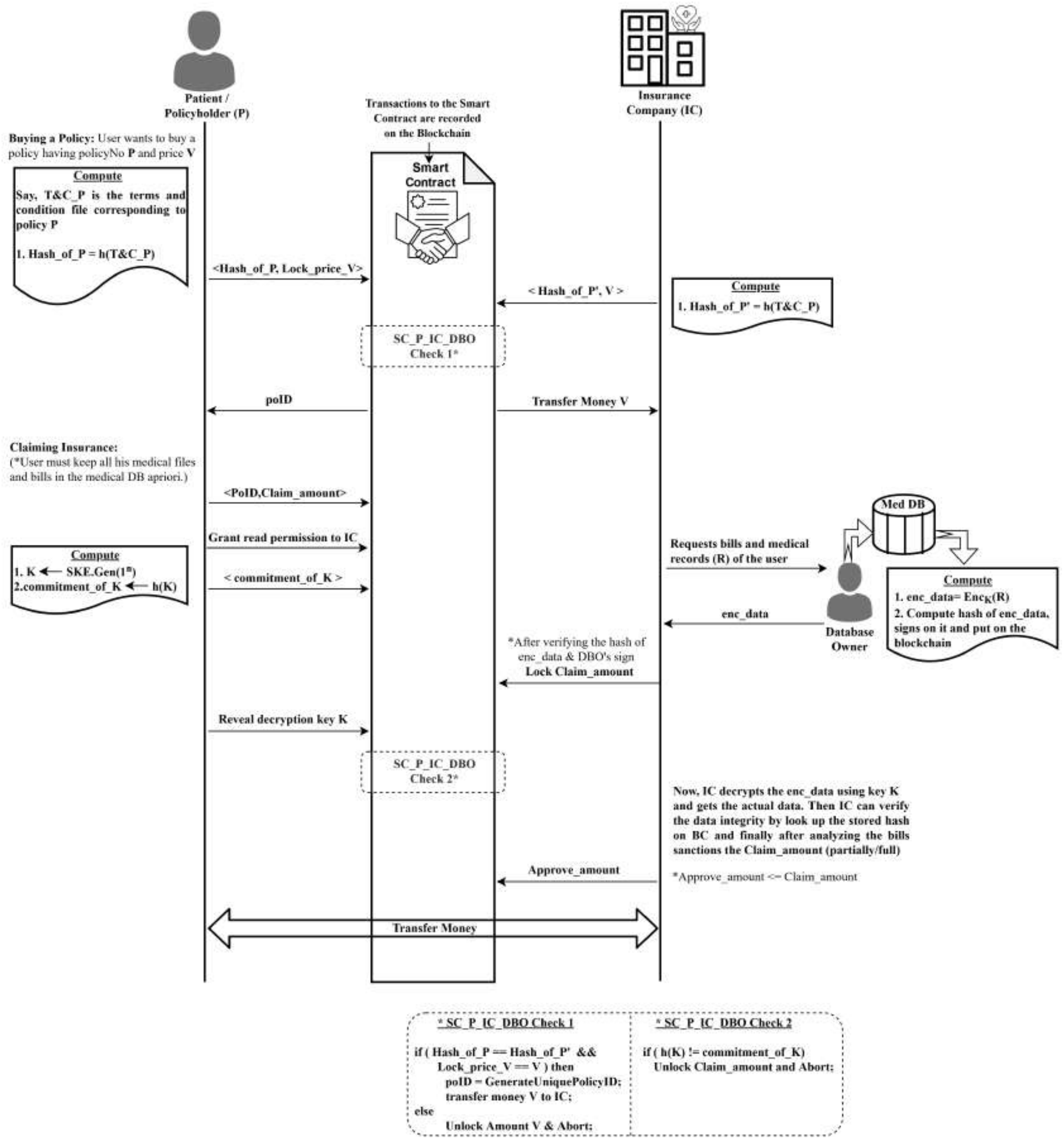
We assume that every *Health Insurance Company* maintains a local database containing all the policy-related information. This database is openly readable to the public. For the sake of simplicity, we can presume that the insurance company holds a relational table with the schema definition *Policy (policyNo, termsAndcondition, price)*, having *policyNo* as the primary key attribute.

For policy purchasing or claiming processes, policyholders and insurance companies would engage in a sequence of blockchain-based transactions for future accountability purposes, thereby obtaining fairness to the system. *policyNo*, *termsAndConditions*, and *price* are denoted as P, $T\&C\_P$, and V, respectively.

Purchasing an Insurance Policy

It is a two-phase process -

In ***phase one***, the buyer expresses their desire to the insurance company for a specific policy (P). By locking the policy's price (V) in the smart contract and also recording the hash of the terms and conditions file ($h(T\&C\_P)$) into the blockchain, the purchaser agrees to the policy's terms and conditions.


Patient / Policyholder (P)
Transactions to the Smart Contract are recorded on the Blockchain
Insurance Company (IC)
Buying a Policy: User wants to buy a policy having policyNo P and price V
Compute
Say, T&C_P is the terms and condition file corresponding to policy P
1. Hash_of_P = h(T&C_P)
Smart Contract
<Hash_of_P, Lock_price_V>
< Hash_of_P', V >
Compute
1. Hash_of_P' = h(T&C_P)
SC_P_IC_DBO Check 1*
poID
Transfer Money V
Claiming Insurance:
(*User must keep all his medical files and bills in the medical DB apriori.)
<PoID,Claim_amount>
Grant read permission to IC
Compute
1. K ⟵ SKE.Gen(1ⁿ)
2.commitment_of_K ⟵ h(K)
< commitment_of_K >
Requests bills and medical records (R) of the user
Med DB
Compute
1. enc_data= Enc_K(R)
2. Compute hash of enc_data, signs on it and put on the blockchain
enc_data
Database Owner
*After verifying the hash of enc_data & DBO's sign
Lock Claim_amount
Reveal decryption key K
SC_P_IC_DBO Check 2*
Now, IC decrypts the enc_data using key K and gets the actual data. Then IC can verify the data integrity by look up the stored hash on BC and finally after analyzing the bills sanctions the Claim_amount (partially/full)
Approve_amount
*Approve_amount <= Claim_amount
Transfer Money
* SC_P_IC_DBO Check 1
* SC_P_IC_DBO Check 2

```
if ( Hash_of_P == Hash_of_P'  &&
      Lock_price_V == V ) then
        poID = GenerateUniquePolicyID;
        transfer money V to IC;
else
        Unlock Amount V & Abort;
```

```
if ( h(K) != commitment_of_K)
  Unlock Claim_amount and Abort;
```

FIGURE 4.5: *Interaction between Policyholder and Health Insurance Company*

In ***phase two***, the insurance provider must also acknowledge by specifying the hash of the terms and conditions file and the policy price within a predetermined time frame. Assume the insurance company responds within the given time frame by specifying the $h(T\&C\_P)$ and policy price on the blockchain. In this example, the smart contract evaluates if these two hash values are identical (i.e., the hash value provided by the buyer in step one and the hash value provided by the insurance company in phase two). In addition, the contract assesses whether the policy price locked by the user differs from the price listed by the insurance company. If they match, the policy price (V) is paid to the insurance company's account, and the system assigns the user a unique policy ID ($poID$) for future usage. The blockchain stores all pertinent information about this $poID$ (for example, buyer and seller information, the hash of $T\&C\_P$, the purchase timestamp, and so on). The policyholder is now eligible to receive the policy's benefits upon fulfilling the terms and conditions. The user may withdraw their locked funds from the system if the two hashed values do not match or if the insurance provider does not respond within the specified time frame.

Claiming for Reimbursement

We have developed a blockchain-based system that enables an insurance company to confirm the legitimacy of a user's (or patient's) medical bill reimbursement claims. When a user submits a claim against their medical policy, the insurance company requests authorization to read the information relating to the medical expenditure from the medical repository, presuming that the claimant has already uploaded all medical files to the database beforehand.

The patient's medical information is categorized into three types -

1. Personal Information (like name, age, address, etc.)
2. Medical Treatment Related Information (like prescriptions, lab test reports, etc.)
3. Medical Expenditure Related Information (like invoices, medical bills, etc.)

Categorizing medical information into granular tiers facilitates sharing information with a third party in a restricted manner. The policyholder grants the appropriate authorization to the insurance company to access their medical records from the repository (or cloud). The insurance company requests the relevant user's files or documents from the Database Owner (DBO) after obtaining appropriate access permission from the user. First, the DBO checks that the insurance company has legal authorization to view these files through the smart contract function and then provides them. After receiving the required file(s), the insurance company determines whether the policy's terms and conditions have been met and then approves the fund. However, it is not a good idea to transmit the data to the insurance company at this point. It might simply steal the data without reimbursing the patient. A solution is proposed to prevent such an attempt. First, the user will generate a unique one-time secret key K, communicate it to the DBO, and put the key's commitment on the blockchain. When the insurance company requests the patient's data from the DBO, the DBO encrypts the data with the key K, delivers the encrypted data to the insurance company (offline), and stores the hash of the encrypted data and digital signature on the blockchain. After receiving the encrypted file, the insurance company seeks the user for the decryption key K. The user only discloses the key if the insurance provider has already locked the claimed amount in the smart contract. After receiving the user's key K, the insurance company decrypts the encrypted data. The insurance company then verifies the accuracy of the information using blockchain and authorizes the sanctioned amount within a predetermined time frame. By retrieving the hash value from the blockchain, the insurance provider may validate the authenticity of the information. The insurance company can also verify the legitimacy of medical bills by examining the digital signatures (of hospital authority and patient) that are also kept on the blockchain.

An insurance company might decide not to react to a user's claim request. However, the insurance company actively participated in the policy-selling process in order to gain the user's money. This is undeniably malicious action on the part of the insurance company, which attempts to defraud the policyholder. We alleviate this risk by permitting insurance companies to maintain a security deposit of a specific amount in the system by locking the same in the smart contract. The security deposit should be greater than the cost of the company's most expensive policy (i.e., Security Deposit $>=$ Costliest Policy Price). An insurance firm may deposit or withdraw its security funds at any time. However, the company must always retain the minimum threshold amount locked. The security deposit partially compensates the buyer. Suppose a legitimate user's claimed amount is not reimbursed due to the mal-functionalities of the insurance company. In such a scenario, the system will return at least the policy premium to the purchaser, debiting from the locked security deposit of the insurance firm. Even if the premium cannot be repaid, the insurance company is immediately deregistered from the system and permanently banned from doing business further. Smart contract facilitates these functionalities in our system. Figure 4.5 depicts our protocol at a high level.

4.4 Implementation and Technical Details

In this section, we will present the algorithms or methods that underpin the primary functionalities of our smart contracts. These smart contracts are written in Solidity programming language.

4.4.1 Terminology

Before detailing the algorithms, let us review the terminology and notation we have used in our implementation (Table 4.2). Other acronyms in the chapter are self-explanatory. Table 4.3 lists the structure definitions used in the Algorithms.

4.4.2 Algorithms

The algorithms are classified into three groups:

1. **Algorithms for Entity Registration**
2. **Algorithm for Data Upload and Access Control**
3. **Algorithm for Data Access and Incentives**

1. Algorithms for Entity Registration: Every entity possesses a $< SK, PK >$ pair, where PK and SK serve as the Public Key (aka, address) and Secret Key (aka, authentication factor) of the entity, respectively. A user

TABLE 4.2: Terminology and Notation used in our Scheme

Abbreviation	Interpretation
P	User/Patient/Policyholder
IC	Insurance Company
DBO	Database Owner
pAddr	User/Patient/Policyholder Address
icAddr	Insurance Company Address
dboAddr	Database Owner Address
pID	User/Patient/Policyholder ID
icID	Insurance Company ID
dboID	Database Owner ID
poID	Policy ID
cID	Claim ID
asID	Application for Storing ID
T_{Event}	Timestamp, when the Event occurs
MR_File	Merkle Tree Root Hash of File
CST	Current System Time
TTL	Threshold Time Limit

TABLE 4.3: Structure Definition

Structure Name	Member Variables
ApplicationForStoring	asID, pAddr, dboAddr, key, MR_File, MR_EncFile, $T_{Application}$, $T_{VerificationMR}$, $T_{KeyReveal}$, $T_{Complain}$, $T_{Approval}$, $T_{UnlockingByP}$, $T_{UnlockingByDBO}$
PolicyDetails	poID, buyerID, icID, $T_{BuyingPolicy}$, terms&ConFileHash, claimIDs[ ]
ClaimDetails	cID, ebID, $T_{GeneratingClaimByP}$, claimedAmount, approvedAmount, $comm_K$, K, $T_{RevealKey}$, $T_{LockingByIC}$, $T_{UnlockingByIC}$, $T_{Approval}$

can register themselves into the system by invoking the function ***UserRegistration*** as specified in Algorithm 1.

Algorithm 1: User/Policyholder Registration

```
Function UserRegistration($PK_P$, $hash\_of\_Personal\_Info$)        ▷ Caller: P
    if ( caller $\neq$ pAddr OR $PK_P$ is already registered ) then
        Exit
    Generate: a unique $pID$
    Create: a mapping entry from $PK_P$ to the associated $pID$
    Record: $hash\_of\_Personal\_Info$ corresponding to $pID$ on BC
end
```

Likewise, other actors may also register with the system and acquire a unique identifier upon successful registration -
IC: $icID \leftarrow$ ICRegistration$(PK_{IC}, name)$
DBO: $dboID \leftarrow$ DBORegistration$(PK_{DBO}, name)$
DBO cannot register themselves. Instead, a trusted third party, or TTP (here, we consider the Government to be TTP), is in charge of bringing a DBO into

the system. The TTP will invoke the function ***DBORegistration***.

2. Algorithm for Data Upload and Access Control: When treatment is over, patients can store their medical records in a database or medical repository, and DBO charges a fee for providing the storage space. If a patient wishes to submit an insurance claim to reimburse medical expenses, s/he must maintain their medical records in the database. Any fair exchange protocol between two parties can be applied to serve the purpose. Several articles and protocols exist in the literature specifically for sharing medical data (e.g. [96], [129], [164], [166], [174]). In this protocol, we use the Fairswap [45] protocol (SC_P_DBO) to make sure that P and DBO exchange information fairly. The DBO will not provide storage and abort if it receives an invalid file. Structure *ApplicationForStoring* member variables are populated during protocol invocations to instantiate the two-party exchange protocol. When the patient stores the EHR generated by the hospital and runs SC_P_DBO, an ID ($asID$) for *ApplicationForStoring* is created. The user controls access to their medical data in the medical repository. The blockchain stores *Access Control Policy* details. The DBO must comply with this access control policy before transferring a patient's EHR to a third party. Readers may refer to the original work[45] for more details.

3. Algorithm for Data Access and Incentives: We can further divide these algorithms into four subcategories - Algorithm 2 describes the fair procedure for **Purchasing an Insurance Policy** from an IC. Procedures for **Claiming an Insurance** and **Processing a Claim** are described in Algorithm 3 and Algorithm 4, respectively. The **Claim Approval** process is specified in Algorithm 5.
Functions for Buying a Policy (Algorithm 2):

- Consider, a user holding ID - pID wishes to purchase a policy from an insurance company having ID - $icID$. The policy costs V units. The user or policy buyer reads the policy's terms and conditions ($T\&C\_P$). Then the user computes the hash of the terms and conditions file locally as $hashOfTermsAndCon \leftarrow h(T\&C\_P)$ and calls ***BuyPolicyPhaseOne*** at time t1.

- First, the user stores the hash of the policy's terms & conditions on the BC and locks the policy price V in the SC. The IC should respond to the buyer's desire to purchase an insurance policy within a specific time frame (TTL). The IC calls ***BuyPolicyPhaseTwo*** (at time $t2$). IC must specify the policy price and the file hash on-chain. The smart contract determines if the values provided by both parties (policy buyer or user in phase one and insurance provider in phase two) are consistent.

- If the IC does not respond within TTL, i.e. $(t2 - t1) > TTL$ after the first phase of buying, the user can unlock their money (i.e. policy price) from the system by calling ***WithdrawLockedPolicyBuyingMoney***.

Algorithm 2: Buying Insurance Policy

```
Function BuyPolicyPhaseOne($icID$, $hashOfTermsAndCon$)                    ▷ Caller: P
    Record: $hashOfTermsAndCon$, $icID$ on BC
    Lock: the Policy Price $V$ in SC
    Record: the CST (say, $t1$) on BC
end
Function BuyPolicyPhaseTwo($pID$, $price$, $hashOfTermsAndCon$)             ▷ Caller: IC
    if ( $caller \neq icID$ ) then
        Exit
    if ( $price > securityMoney_{icID}$ ) then
        De-register the IC from the System and Exit
    $t2 \leftarrow$ CST
    if ( $(t2 - t1) > TTL$ ) then
        Exit
    Retrieve: the amount locked by buyer $pID$ from BC and matches with $price$
    Retrieve: the hash of terms and condition file mentioned by buyer $pID$ from BC and
      matches with $hashOfTermsAndCon$
    if ( any of the matches are found wrong ) then
        Exit
    Generate: a unique $poID$
    Instantiate: a structure $PolicyDetails$ $pd$ corresponding to $poID$ and store on BC
    return ($poID$)
end
Function WithdrawLockedPolicyBuyingMoney($icID$)                           ▷ Caller: P
    Retrieve: the timestamp($t1$) of locking the money from BC
    if (CST - $t1$) > TTL) AND the seller IC does not respond then
        Unlock: the money from SC and Credit to the Buyer's Account
end
```

Algorithm 3: Claiming Insurance

```
Function ClaimMoney($poID$, $asID$, $billID$, $claimedAmount$, $comm\_K$)     ▷ Caller: P
    Retrieve: the structures $PolicyDetails$ $pd$, and $ApplicationForStoring$ $as$ corresponding to
      $poID$, and $asID$ from BC resp.
    Parse: $pd$ and $as$ and Verify for the correctness of the necessary field values
    Obtain: Medical $invoice$ using $billID$
    Verify: Multi-Signatures of hospital and patient on the $invoice$
    Check: ( caller $==$ $pd.buyerID$ $==$ $as.pID$ )
    Check: ( $as.timestamp\_approval \neq 0$ )
    Check: ( $claimedAmount \leq invoiceAmount$ )
    Incorporate the business logic to ensure that the user has not previously claimed for the same
      bill.
    if ( any of the above checks is FALSE ) then
        Exit
    Generate: a unique $cID$
    Instantiate: a structure $ClaimDetails$ $cd$
    $cd.comm_K \leftarrow comm\_K$
    $cd.T_{GeneratingClaimByP} \leftarrow CST$
    Record: $cd$ on BC
    Create: an mapping entry - $cIDTopoID[cID] \leftarrow poID$
    Grant: Read Access to $pd.icID$
end
```

Functions for Claiming (Algorithm 3):

– The policyholder (P) can claim reimbursement from insurer (IC) by calling function ClaimMoney. P generates a temporary key K and calculates K's commitment offline.
$K \leftarrow SKE.Gen(1^n)$.
$comm\_K \leftarrow h(K)$.

Algorithm 4: Processing a Claim

```
Function KeepSigOnHashOfEncFile($cID$, $sign\_DBO$)                    ▷ Caller: DBO
    Record: <$sign\_DBO$ and $CST$> corresponding to $cID$ on BC
    Mark: the key $K$ as expired
end
Function LockClaimedMoney($cID$, $amount$)                              ▷ Caller: IC
    Obtain: $poID \leftarrow cIDTopoID[cID]$
    Retrieve: the structures PolicyDetails pd, ClaimDetails cd corresponding to $poID$ and $cID$
      from BC resp.
    if ( $caller \neq pd.icID$ ) then
        Exit
    if ( $amount == cd.claimedAmount$ ) then
        Lock: the $amount$ in SC
        $cd.T_{LockingByIC} \leftarrow CST$
end
Function RevealSecretKey($cID$, $K$)                                     ▷ Caller: P
    Obtain: $poID \leftarrow cIDTopoID[cID]$
    Retrieve: the structures PolicyDetails pd, ClaimDetails cd corresponding to $poID$ and $cID$
      from BC resp.
    if ( $caller \neq pd.buyerID$ OR
    $h(K) \neq cd.comm_K$ OR
    $(CST - cd.T_{LockingByIC}) > TTL$ ) then
        Exit
    $cd.K \leftarrow K$
    $cd.T_{RevealKey} \leftarrow CST$
end
Function WithdrawLockedClaimedMoney($cID$)                              ▷ Caller: IC
    Obtain: $poID \leftarrow cIDTopoID[cID]$
    Retrieve: the structures PolicyDetails pd, ClaimDetails cd corresponding to $poID$ and $cID$
      from BC resp.
    if ( $caller \neq pd.icID$ ) then
        Exit
    if ( $cd.T_{RevealKey} == 0$ AND $cd.T_{LockingByIC} \neq 0$ AND $(CST - cd.T_{LockingByIC}) >$
      $TTL$ ) then
        Unlock: the $claimedAmount$ from SC and Credit to the IC's Account.
end
```

P delivers the key K to the DBO (one-to-one offline communication) and includes the $comm\_K$ as function parameters. Once the transaction is mined, $comm\_K$ goes to BC. Using BC, this function validates important data (member variables of various structures), verifies signatures, and looks for timestamps of crucial operations. The algorithm ensures that the policyholder does not file a duplicate claim for the same medical bill. If all the conditions are met, a unique cID is generated and communicated to the policyholder for further reference. The function instantiates and stores a $ClaimDetails$ structure in BC. P must also provide the IC read access to their medical files/bills in the medical repository.

Functions for Processing a Claim (Algorithm 4):

– The DBO obtains the Key K from P and checks to verify that the $comm\_K$ agrees with the received key K. If it matches, then DBO encrypts the user's file R (say)
  $enc\_data \leftarrow SKE.Enc_K(R)$
  The DBO must invalidate K after encryption. Invalidating K prevents security risks. DBO computes the hash of the encrypted data,

digitally signs on it, and stores it in BC for future accountability (i.e., $Sign_{SK_{DBO}}(h(enc\_data))$). DBO calls the function ***KeepSigOnHashOfEncFile*** and transfers the encrypted file to the IC offline.

– Next, the IC locks the claimed *amount* in the **SC** and invokes the function ***LockClaimedMoney***.

– Once IC locks the claimed amount; the P must produce key K within TTL so IC can decrypt the encrypted files by calling ***RevealKey***.

– If P does not submit the key K within the stated time window, the IC can withdraw the locked money by invoking ***WithdrawLockedClaimedMoney***

Algorithm 5: Approving a Claim

```
Function ApproveClaim($cID$, $approvedAmount$)                         ▷ Caller: IC
    Obtain: $poID \leftarrow cIDTopoID[cID]$
    Retrieve: the structures PolicyDetails pd, ClaimDetails cd corresponding to $poID$ and $cID$
      from BC resp.
    if ( $caller \neq pd.icID$ OR $cd.T_{RevealKey} == 0$ OR $cd.T_{Approval} \neq 0$ OR ($CST$ -
      $cd.T_{RevealKey}$)>$TTL$ ) then
        Exit
    $cd.approvedAmount \leftarrow approvedAmount$
    $cd.T_{Approval} \leftarrow CST$
    Unlock: the approvedAmount from SC and Credit to the Policyholder's Account
    $remainingAmount \leftarrow (cd.claimedAmount - cd.approvedAmount)$
    if ( $remainingAmount > 0$ ) then
        Unlock: the remainingAmount from SC and Credit to the IC's Account
end
Function SelfApproveClaim($cID$, $icID$)                               ▷ Caller: P
    Obtain: $poID \leftarrow cIDTopoID[cID]$
    Retrieve: the structures PolicyDetails pd, ClaimDetails cd corresponding to $poID$ and $cID$
      from BC resp.
    if ( $caller \neq pd.buyerID$ ) then
        Exit
    if ( $cd.T_{RevealKey} \neq 0$ AND $cd.T_{Approval} == 0$ AND ($CST$ - $cd.T_{RevealKey}$) > $TTL$) then
        $cd.approvedAmount \leftarrow cd.claimedAmount$
        $cd.T_{Approval} \leftarrow CST$
        $cd.isSelfApproved \leftarrow TRUE$
        Unlock: the claimedAmount from SC and Credit to the Policyholder's Account.
end
```

Functions for Approving a Claim (Algorithm 5):

– The IC decrypts the encrypted file using the user's key K and verifies the integrity. The IC approves the claim based on the policy's Terms and Conditions. Sometimes a claim is only partially granted. The IC must follow all the terms in the $T\&C\_P$ file. The file's hash is already in the blockchain; therefore, there is no way to defy rules or norms. After the user reveals the key, the IC should call ***ApproveClaim*** within TTL.

– If the IC does not call the procedure above within the set time limit (even if the user revealed the key on time), the user can validate the claim amount themself. The policyholder can unlock the IC's locked amount by invoking ***SelfApproveClaim***. It ensures that the policyholder is treated fairly.

The functional sequence diagram among policyholder, insurance company and database owner (specified in the smart contract SC_P_IC_DBO) is shown in Figure 4.6.

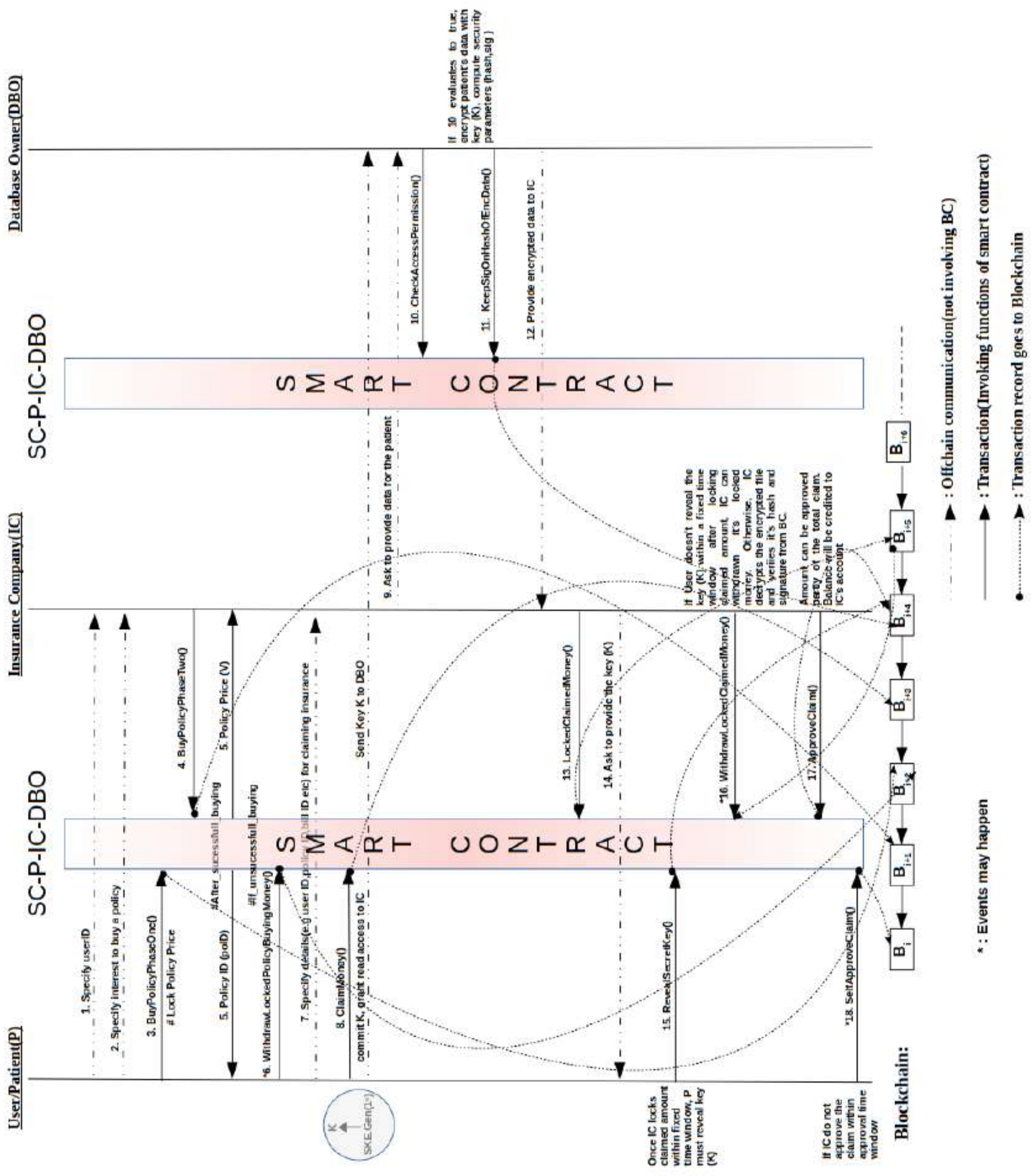

User/Patient(P)
SC-P-IC-DBO
Insurance Company(IC)
SC-P-IC-DBO
Database Owner(DBO)
SMART CONTRACT
SMART CONTRACT
1. Specify userID
2. Specify interest to buy a policy
3. BuyPolicyPhaseOne()
# Lock Policy Price
4. BuyPolicyPhaseTwo()
5. Policy ID (polD)
5. Policy Price (V)
*6. WithdrawLockedPolicyBuyingMoney()
#After_sucessfull_buying
#If_unsucessfull_buying
7. Specify details(e.g user ID, policy ID, bill ID etc) for claiming insurance
8. ClaimMoney()
commit K, grant read access to IC
Send Key K to DBO
9. Ask to provide data for the patient
10. CheckAccessPermission()
11. KeepSigOnHashOfEncData()
12. Provide encrypted data to IC
If 10 evaluates to true, encrypt patient's data with key (K), compute security parameters (hash,sig)
13. LockedClaimedMoney()
14. Ask to provide the key (K)
15. RevealSecretKey()
*16. WithdrawLockedClaimedMoney()
17. ApproveClaim()
*18. SelfApproveClaim()
If User doesn't reveal the key (K) within a fixed time window after locking claimed amount, IC can withdrawn its locked money. Otherwise, IC decrypts the encrypted file and verifies it's hash and signature from BC.
Amount can be approved partly of the total claim. Balance will be credited to IC's account
Once IC locks claimed amount within fixed time window, P must reveal key (K)
If IC do not approve the claim within approval time window
K
SKE.Gen(1^n)
Blockchain:
B_i
B_i+1
B_i+2
B_i+3
B_i+4
B_i+5
B_i+6
: Offchain communication(not involving BC)
: Transaction(Invoking functions of smart contract)
: Transaction record goes to Blockchain
* : Events may happen

FIGURE 4.6: *Functional Sequence Diagram of SC_P_IC_DBO*

4.5 Security Analysis

Blockchain technology makes use of several basic cryptographic concepts (such as the hash function and digital signature). The blockchain is secure as long as the fundamental cryptographic building blocks are safeguarded. And so, our blockchain-based system also stands protected. The money held on the blockchain is safe, assuming the blockchain is secure, and as a result, the payments performed via the system are likewise safe. We assert that our method addresses significant security concerns and assures fairness for all system users.

4.5.1 Fairness

We discuss about whether patients and insurance companies are both being treated fairly. Even if one of the parties engages in dishonest behavior and attempts to defraud, the malicious party is punished, and the honest party is compensated with the money.

Proposition 5. *(Policyholder's Fairness) Considering that the owner of the Medical Data Repository is semi-trusted and the underlying blockchain is secure, the honest user or policyholder must not lose money or receive inferior service, regardless of how maliciously the other party (i.e., the insurance company) behaves.*

Proof. We shall demonstrate the notion when an honest patient's interests might be jeopardized. In particular, we will evaluate the security based on pricing, data, and responsiveness.

When Insurance Company (IC) is malicious:

<u>IC defrauds policyholder by rejecting claim</u>

The blockchain stores the hash of the file containing the policy's terms and conditions, and the user confirms this hash upon purchasing the policy. Therefore, the insurer cannot later refuse the claim. Additionally, when a policyholder submits a claim for reimbursement of their medical expenses, a transaction is formed, and the related data is recorded to the blockchain. The transaction must be resolved within a specified time frame after the claim; otherwise, the insurer will face a penalty. In a distributed system like blockchain, the insurance company's reputation takes a hit, which is not desirable. If the IC partially sanctions a claim and the policyholder is dissatisfied with the approval, the policyholder may pursue legal recourse and file a complaint against the IC. With the aid of blockchain technology, the judge can promptly resolve the disagreement (as the hash of the terms and conditions file is kept in BC). If the IC fails to react to a policyholder's claim, the system might refund the premium paid by the user. If it is impossible to repay the money (due to insufficient security funds), the IC's access to the system would be blocked permanently, and the IC would also be banned from the further business.

<u>IC steals patient's medical data</u>

If we examine the claim procedure, we will realize that the policyholder must provide the IC authorization to view the EHRs in the medical database so that the insurer may check the legitimacy of claims made by the policyholder. Nevertheless, if the Database Owner (DBO) delivers the medical report, the insurance company can obtain the data and cease all communication with the customer. To counteract this, the DBO transmits encrypted data to the IC (*enc_data*

in Figure 4.5). Encryption is performed using a user-generated temporary key, K. This key must be used only once. The IC locks the claimed amount in the smart contract upon receiving the encrypted data and requests for the user's key (K). The user will not reveal the key unless the insurance locks the money into the contract.
Patient/user data can be classified into three categories: (i) Personal Information, (ii) Medical Reports, and (iii) Invoices/Bills pertaining to Medical Expenses. If confidentiality is required, users can provide access to only the medical expenses segment of the data and suppress the rest. Therefore, it also addresses data privacy concerns.

IC becomes unresponsive in the middle of the protocol

Our system has a set time frame for each consecutive function call in the smart contract. If the IC fails to respond in a timely manner, the user can take appropriate action. For example, if the IC does not respond in Phase Two of the policy-buying process, the user can unlock the funds for purchasing a policy. In the same way, the user can take the necessary measures when submitting a claim for reimbursement of medical expenses to avoid losing money if the IC becomes inactive. □

Proposition 6. *(Insurance Company's Fairness) An honest insurance company will obtain the precise policy price or premium from the policyholders at the time of policy offers, and it also should not be suffered from any financial loss by reimbursement of invalid claims of fraudulent users under the premise that the owner of the Medical Data Repository is semi-trusted and the underlying blockchain is secure.*

Proof. **When Policyholder (P) is malicious:**

P manages to get insurance coverage without paying a policy premium

During the initial phase of policy purchase, the user declares a desire, keeping the money (i.e., policy price V) locked in the smart contract. The user also uploads the hash of the file containing the terms and conditions of the policy to the blockchain. Next, the IC validates the same information during the second phase of the policy purchasing process. Finally, if everything is in order, the smart contract deposits the policy price (V) to the IC's account and provides the buyer with a unique policy ID. Therefore, a user can not have insurance coverage without paying a policy premium.

P submits a false claim

When a policyholder submits a claim for reimbursement of medical expenses, the system first determines whether or not the claim has already been processed in the past. If the claim is fresh, the system then assesses the validity of all medical expenditure records by matching their hash values and verifying the digital signatures of hospital authorities and patients stored in the blockchain. Finally, the system validates that the claim complies with the terms and conditions of the insurance policy. After these verifications, the claim is either partially or entirely approved. Thus, the insurance company's fairness is safeguarded, and policyholders cannot defraud an insurance company in our system.

P becomes unresponsive in the middle of the protocol

The policyholder has no fundamental reason for being unresponsive (as it is the user whose money has been locked in the system). However, a policyholder may be inactive during the claim process. For example, the policyholder may delay providing the decryption key after submitting an insurance claim when the IC needs it to decipher the encrypted content. If this occurs and the time window for revealing the key expires, the IC can quit the protocol without approving any money claimed by the policyholder and release the IC's locked funds from the smart contract. Therefore, if the policyholder becomes unresponsive, the IC will not be tricked. □

4.5.2 Privacy

Medical information about a user or patient is sensitive data.. If a person's medical history is made public, they could feel humiliated and might encounter prejudice daily. As a result, it is crucial to guarantee that access to patient information is only allowed with the patient's consent.

Proposition 7. *(User/Patient Privacy) The medical data of a user cannot be accessed by any of the entities in our proposed system without the user's permission. The user's individually identifiable information is also kept concealed from the public.*

Proof. While enrolling on the blockchain, a user does not disclose information such as name, age, address, phone number, etc. Rather, the hash of these properties is stored in the blockchain during registration. Due to the user-centric nature of our system model, no one can view the user's medical records without their permission. Thus, any unauthorised access to medical records is prevented. The user specifies the identities that have access to their data in the access control matrix. The patient can update the matrix periodically; s/he can grant access permission to a third party or revoke access permission from a third party. In the blockchain, the access control matrix is stored. □

4.5.3 Data Security

Altering a patient's medical records could have disastrous effects. The changes in medication dosages or modifications to the condition for which a diagnosis has been made may put lives in danger. In our context, users can attempt to falsify their medical records to fraudulently submit claims for payment to their insurance provider.

Proposition 8. *(Data Security) No one, not even the patient/user, can tamper with the medical data in our envisioned system and misuse it.*

Proof. Since we have constructed our protocol with blockchain serving as the underlying framework, the immutability of blockchain, which is a fundamental characteristic of the technology, ensures the safety of medical data. □

4.5.4 Liveness

Liveness is critical in distributed systems because it guarantees that the system remains active and responsive even in the presence of failures, delays, or malicious behavior. A blockchain-based system inherently supports liveness through decentralized consensus, ensuring continuous progress and transaction finality. The fault-tolerant nature of the network enables uninterrupted operation even during node failures or malicious behavior, contributing to implicit liveness.

Proposition 9. *(Liveness) The proposed blockchain-based system keeps progressing and continues to process and finalize transactions.*

Proof. While blockchain-based systems inherently possess liveness as an implicit property, ensuring the smooth and continuous functioning of the underlying network, application-level liveness becomes essential for scenarios where specific actions must be taken within defined time frames. In certain instances, situations might arise where the non-execution of one function can potentially block the execution of subsequent functions, creating a need for timely and autonomous actions to maintain system responsiveness and application-level liveness. For example, in a proposed smart contract for an insurance claim process, the function ***ApproveClaim()*** must be executed by the insurance company within a specific time window after the policyholder submits a claim. However, if the insurance company fails to act within the fixed time, the subsequent claim processing may be blocked, impeding the system's overall liveness. To address this, our system introduces a supplementary safeguard. Upon the expiration of the time window, the policyholder is granted the capability to invoke the ***SelfApproveClaim()*** function independently. This measure empowers the policyholder to settle the claim autonomously, maintaining the system's liveness and ensuring timely execution. By integrating these features into the smart contract design, we enhance the application-level liveness of the system, offering a reliable and dynamic environment for all stakeholders involved. □

4.6 Result and Discussion

Implementation Setup: Our model has been implemented on the Ethereum test networks using a computer system having an Intel Core i7-6700HQ processor running Linux Mint 18.04 19.1 (Tessa), a 64-bit operating system, and 16.00 GiB of RAM. We have employed an Infura endpoint and the Ropsten test network. On GitHub[†], the source code is made available. The feasibility of any blockchain model depends on the implementation cost and time required. The specifics of Ethereum transaction cost and latency are discussed in Chapter 2, Section 2.6.10.

The addresses of the deployed contracts are listed in Table 4.4. Figure 4.7 and Figure 4.8, respectively, illustrate the transaction cost and time required for each contract deployment. At the time of deployment, the cost of gas was 18.9 Gwei, and ether was 2,300.54 USD. The cost of deployment varies depending

[b]https://doi.org/10.5281/zenodo.8232704

TABLE 4.4: Deployment Addresses of Smart Contracts

Smart Contract	Address
SC_Registration	0x5a818296705cC24Feec4CfEAF1DfdaE056fEf037
SC_P_IC_DBO	0xD519535972d006DD72AbBd60453Ae78747065B5e
SC_P_DBO	0xC062E1eF5EdB815bcF5B93C6BaD497ABCA407f31

TABLE 4.5: Deployment Cost of Smart Contracts

Smart Contract	Deployment Cost(Ether)
SC_Registration	0.0353157
SC_P_IC_DBO	0.0516384
SC_P_DBO	0.04783231

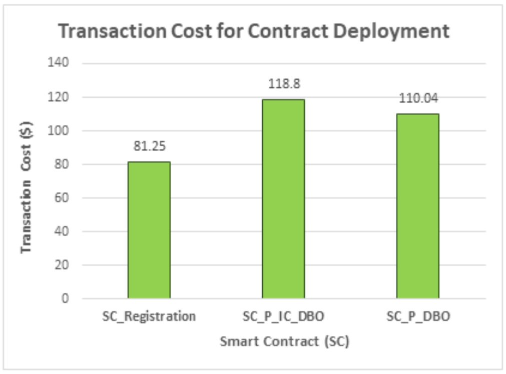


FIGURE 4.7: *Transaction Cost for Contract Deployment*

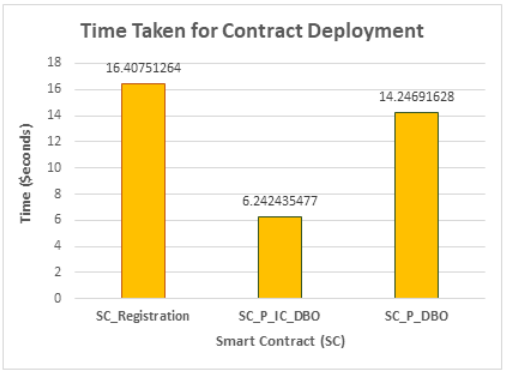


FIGURE 4.8: *Time Taken for Contract Deployment*

on the size of the contracts (Table 4.5). The contract deployment occurs only once, so these are one-time expenses. Once the protocol is in place, we can reap the benefits as long as it is used.

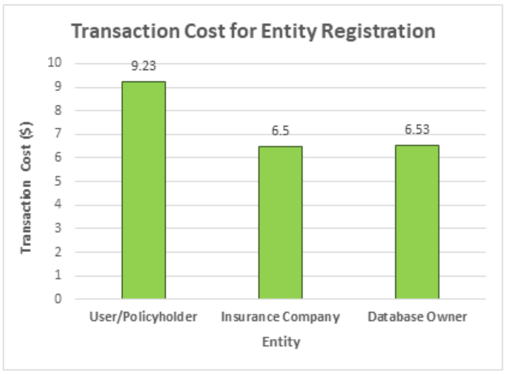


FIGURE 4.9: *Transaction Cost for Entity Registration*

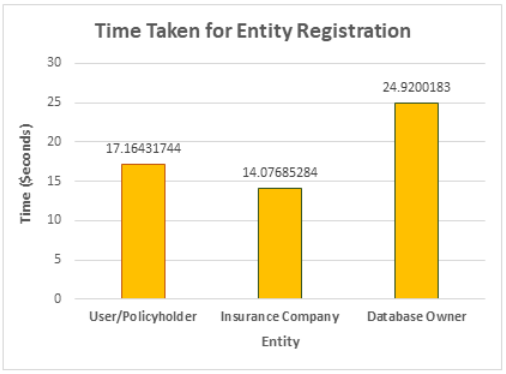


FIGURE 4.10: *Time Taken for Entity Registration*

Figure 4.9 and Figure 4.10 represent the transaction costs and time involved in the entities' registration procedure. These are similar in magnitude for different parties except for the policyholders, who have a few more variables involved in the registration. The transaction cost for specific protocol functionalities is related to the size of the medical file. The file size may vary based on the medical treatment administered. For the experiment, we constructed the base file arbitrarily. The file is segmented into several parts. The variables, referred to as the number of input gates and the gate's buffer size, represent the number and size of the split chunks.

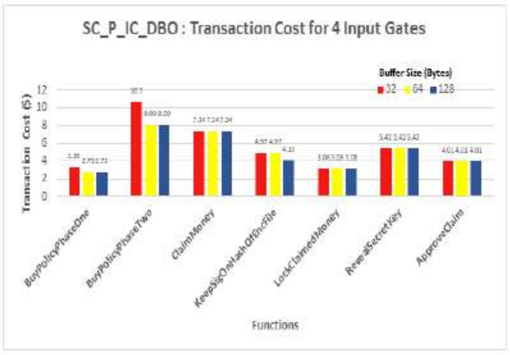


FIGURE 4.11: *SC_P_IC_DBO Transaction Cost for 4 Input Gates*

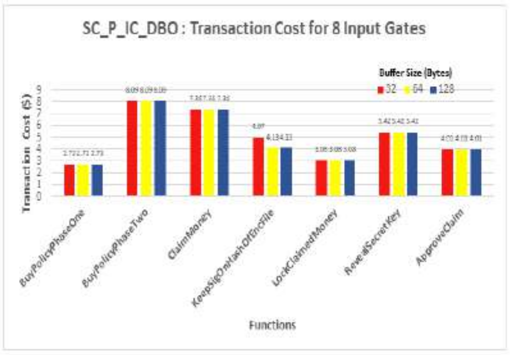


FIGURE 4.12: *SC_P_IC_DBO Transaction Cost for 8 Input Gates*

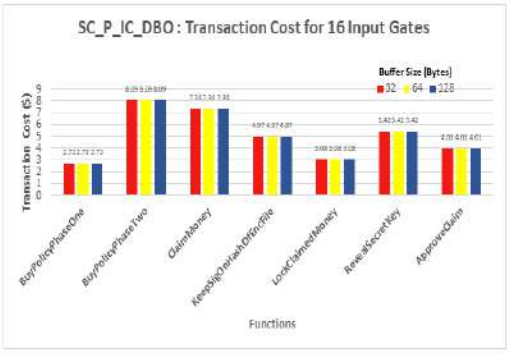


FIGURE 4.13: *SC_P_IC_DBO Transaction Cost for 16 Input Gates*

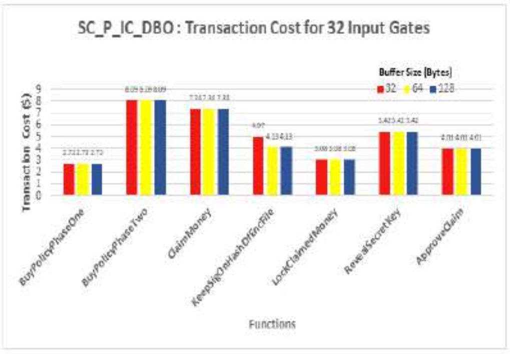


FIGURE 4.14: *SC_P_IC_DBO Transaction Cost for 32 Input Gates*

With changing numbers of input gates and buffer sizes, we have illustrated the costs associated with different function calls for the contract SC_P_IC_DBO (Figure 4.11, 4.12, 4.13, 4.14). The range of input gates is 4, 8, 16, and 32, while the usual buffer sizes are 32, 64, and 128. The number of input gates and the buffer size could be multiplied to get the file size. As a result, a file with 4 input gates and a 32-byte buffer would be 128 bytes in size.

The SC_P_IC_DBO graphs (Figure 4.11, 4.12, 4.13, 4.14) reveal that they exhibit a similar pattern despite having different numbers of input gates. While maintaining a fixed number of input gates and altering the buffer size, the transaction cost for various functions minimally changes. The graphs demonstrate that some functions need a higher cost because of their complex functionality. It is economically feasible since the functions' benefits surpass their costs.

4.7 Conclusion

In this work, we have provided a novel *Health Insurance Processing System* that ensures the fairness of all parties without relying on mutual trust. Smart contracts securely enable online correspondence between policyholders and insurers, increasing transparency and automation. Additionally, it aids in speeding up the insurance company's key processing stages, from policy offering to claim verification and settlement. The use of blockchain technology and cryptographic primitives assures against fraudulent conduct. We prototyped and deployed our health insurance processing system in private and Ropsten test networks. The experimental analysis demonstrates that multiple performance indicators produce satisfactory results. Our protocol exhibits blockchain's utility and significance in the insurance industry—specifically, health insurance. It establishes that it may be the next ground-breaking technology to replace the nation's current insurance processing infrastructure.

The unprecedented situations faced by global society during the COVID-19 pandemic gave rise to new challenges in public healthcare systems worldwide. A critical instance is the development of a secure and robust vaccine passport system, which provides documentary proof of vaccination against certain infectious diseases like COVID-19, Ebola, and flu. That is taken up in the next chapter.

5

Blockchain-Enabled Secure Vaccine Passport System

"Misinformation or distrust of vaccines can be like a contagion that can spread as fast as measles."

— **Theresa Tam**

A vaccine passport, also known as an immunity passport, serves as documentary proof, implying a person has been vaccinated against certain infectious diseases. It can be digital, like a phone app, or physical, like a small paper card. People can carry it with them and show it whenever required, like before entering the office, boarding an airplane, or visiting a restaurant, movie theatre, or gym. In the recent past, we have witnessed the global impact of the COVID-19 pandemic [5], underscoring the critical importance of vaccine passports in managing and mitigating the spread of infectious diseases. However, the relevance of vaccine passports extends beyond the COVID-19 pandemic to encompass a broader spectrum of contagious illnesses that pose significant public health risks. Historical outbreaks such as the Spanish flu in 1918 [117], the H1N1 influenza pandemic in 2009 [118], and the Ebola outbreak in West Africa in 2014 [56] highlight the devastating consequences of infectious diseases on global populations. These events serve as stark reminders of the urgent need for robust public health measures, including vaccination and disease containment strategies. As the COVID-19 pandemic continues to persist in recent memory, the implementation of a vaccine passport system emerges as a crucial tool in safeguarding public health and promoting safe mobility. The prospect of fully reopening businesses, facilitating international travel, and reviving economies hinges on the adoption of vaccine passports as a means to verify vaccination status and mitigate the transmission of contagious diseases. As international passenger traffic gradually rebounds from the impacts of the COVID-19 pandemic, the role of vaccine passports becomes increasingly prominent in facilitating safe movement across borders. By providing individuals with proof of vaccination, vaccine passports not only enable access to various venues and activities but also contribute to broader public health objectives by arresting the spread of infectious diseases.

Challenges of the Vaccine Passport System

Vaccine passports play a crucial role in managing infectious diseases, yet they also present several challenges.

1. **Authentication Concerns:** The issuance of vaccine certificates by healthcare clinics or third parties introduces the risk of counterfeit or fraudulent passports. Instances of counterfeit vaccine certificates have been reported in various disease outbreaks, such as the Ebola outbreak in West Africa. For example, during the COVID-19 pandemic, there were reports of individuals purchasing fake vaccine certificates on the black market to bypass travel restrictions or gain access to venues. Such incidents underscore the importance of robust authentication mechanisms to prevent forgery and ensure the integrity of vaccine passports.

2. **Regulatory Ambiguity:** The proliferation of various vaccines introduces complexity in determining their efficacy, dosage requirements, and international recognition. Lack of standardization across different diseases complicates travel restrictions and entry requirements based on vaccine type. For instance, certain vaccines may not be universally accepted for international travel during outbreaks of diseases such as yellow fever or Zika virus. In the recent past, during the COVID-19 outbreak, we have seen that the European Union does not allow Indian citizens vaccinated with Covishield to enter their countries [73]; likewise, the US has not approved Sputnik-V and Co-vaxin. Such inconsistencies highlight the need for harmonization of vaccine regulations to facilitate global mobility and ensure equitable access to travel opportunities.

3. **Equity Issues:** Vaccine distribution disparities worsen existing inequalities worldwide. Limited vaccine availability, along with unequal distribution, means some privileged people receive doses while others do not. Unequal access to healthcare, coupled with restrictions on the import and export of medical supplies, exacerbates global health inequalities. For example, during the H1N1 influenza pandemic, wealthier nations stockpiled vaccines, leaving developing countries with limited access. Additionally, concerns about fake vaccine certificates and unethical practices at vaccination centers undermine fairness and trust in the vaccination process.

4. **Privacy Considerations:** Privacy concerns emerge as individuals may be reluctant to disclose personal information beyond vaccination status. A vaccine passport should solely serve the purpose of verifying vaccination without compromising individuals' privacy by revealing unnecessary personal details. Furthermore, individuals with medical exemptions may hesitate to disclose their health conditions due to fears of social stigma or discrimination.

By acknowledging and addressing these challenges, the development and implementation of vaccine passport systems can better serve their intended purpose while safeguarding public health and individual rights across various infectious diseases.

Limitations of Existing Legacy Technology

1. **Data Tampering Vulnerabilities:** Traditional centralized databases are vulnerable to unauthorized data changes, which can occur due to insider manipulation or external hacking attempts. These systems often lack robust auditing mechanisms, making it difficult to detect and prevent fraudulent alterations. As a result, data integrity is compromised, leading to potential inaccuracies in critical information such as vaccination records.

2. **Lack of Transparency and Auditability:** Centralized systems operate as opaque entities managed by a single trusted authority. This lack of transparency means that users must place blind trust in the administrators maintaining these systems. Without open mechanisms for independent verification and auditing of data integrity, concerns arise regarding the accuracy and reliability of the information stored within these databases.

3. **Centralized Trust Model:** Current solutions rely on a centralized trust model, wherein a single entity controls the entire system. While this may simplify management, it introduces significant vulnerabilities, such as single points of failure and susceptibility to insider threats or external attacks. Moreover, the lack of decentralized data integrity undermines the overall reliability of these systems.

4. **Data Silos and Lack of Interoperability:** Data fragmentation across various institutions, organizations, or nations creates silos that hinder global information sharing. Incompatible data formats, standards, and policies further exacerbate this issue, impeding efforts to establish seamless interoperability. Without standardized and universally accessible platforms for managing immunization records, the efficient exchange of critical data remains a challenge.

5. **Privacy Concerns:** Centralized databases containing sensitive healthcare information raise significant privacy concerns. The potential for data leakage or misuse poses risks to individuals' privacy rights and confidentiality. Inadequate privacy-preserving controls and access policies further compound these concerns, highlighting the need for enhanced safeguards to protect users' personal information while still enabling efficient verification of vaccination status.

Objectives

In the Vaccine Passport System, we want the following criteria to be satisfied.

– Various countries are involved that allow their foreign guest to enter only if the traveler produces a valid vaccine certificate. Here, the basic assumption is that the verifier (i.e., the entity that verifies the vaccine passport) does not trust the user or the traveler.

– At the same time, the involved parties in the system would like to take a common decision on whether the travelers should produce a valid vaccine certificate, and they can not forge the certificate in any way.

– The users or travelers do not want to reveal their personal information due to privacy concerns - they intend to prove that they are vaccinated and nothing more.

– Also, from the global perspective, we want an equitable distribution of vaccine doses across all countries to guarantee fairness.

Blockchain integrated with smart contracts can be a perfect fit to satisfy the criteria mentioned above.

1. **Immutability :** Storing the records of vaccine passports in a blockchain helps to protect these from any unauthorized modifications of data. The immutable property of blockchain guarantees that once the data is stored inside a blockchain, it persists permanently - no one can tamper with the data.

2. **Decentralized Distributed System :** Since blockchain is a distributed system, it does not solely depend on a single centralized authority for the verification of transactions or records. Here, multiple nodes of different countries located across the globe would take part to validate a transaction. So, the probability of forging a vaccine passport by a single entity is almost negligible. Any certain specific country also can not be able to function maliciously in such a distributed network spanning across the entire world.

3. **Privacy :** In a public blockchain system, as data exists across the various nodes, so one might be concerned about privacy issues. However, we can cleverly store the record in a blockchain so that we can easily verify that a person is vaccinated without revealing any details about his/her private information.

4. **Fairness :** Smart Contracts can be written in such a way that we can obtain statistics about how many people of the total population of a country have been vaccinated. This information essentially helps to identify which countries are lagging in the vaccination process, and in turn, it helps the Governments of so many countries take necessary actions immediately. Also, a smart contract ensures that no one can forge a vaccine certificate colluding with the vaccination center, and a malicious vaccination center cannot put wrong records regarding the vaccination status inside the blockchain, intending to sell vaccine doses to a third party for money.

5.0.1 Contributions

In this work, we propose a vaccination passport system using a blockchain framework to ensure the following -

- *Fairness:*
    1. An individual can not create a fake vaccine certificate itself or collude with the vaccination center.
    2. A vaccination center can not record any false information into the blockchain that a person has been vaccinated, which is not the case. Also, it can not sell vaccine doses to third parties for extra money.
    3. Global perspective - Obtaining statistics regarding the proportion of the citizens who got vaccinated out of the total population for a specific country and also to get the figure about the number of available vaccine doses in different centers. It, in turn, allows distributing the vaccine doses where the supply is insufficient - ensuring every part across the globe should get a uniform/fair share of the vaccine doses.
- *Immutability:* Record stored in the blockchain regarding the vaccination status of a person can not be altered by any malicious or unauthorized access.
- *Privacy:* We will store information in the blockchain in such a way that it does not compromise the private information of a person, but still, a person can verify to others that s/he has been vaccinated. Also, we incorporate an access control policy; it is up to the person to decide who can verify the vaccination record of the person. The person can grant (or revoke) access permission to (or from) other parties occasionally.

Although our proposed work solves most of the crucial problems, the problem caused by a lack of regulation persists. We believe this can not be solved unless and until the WHO specifies and standardizes the set of valid vaccines.

5.0.2 Organization

The rest of the chapter is structured as follows -
Section 5.1 briefly discusses the current state of the art. Section 5.2 describes our system model. Here, we have discussed our system components, security goal, adversarial model, assumptions, major procedures, and other technical details. We addressed our security claims in Section 5.3. The outcomes of our proposed system are shown in Section 5.4. Finally, Section 5.5 concludes the chapter and showcases some future directions.

5.1 Related Work

After the World Health Organization (WHO) declared COVID-19 a global pandemic in March 2020, governments, NGOs, and corporations worldwide made concerted efforts to combat the spread of the disease. One potential solution that gained significant attention was the use of immunity or vaccine passports to help manage the spread of infectious diseases. This approach contrasts sharply with contact tracing apps, which focus on tracking and isolating infected individuals during outbreaks, specifically in the pre-vaccination era when no proven treatments or vaccines were available, making isolation one of the most effective strategies to control outbreaks [2], [3], [27], [154]. The shift from isolating infected individuals to enabling safe interactions among vaccinated individuals in the post-vaccination context underscores a fundamental change in strategy. Although the concept of vaccine passports has existed for some time, the COVID-19 pandemic brought it to the forefront as a means to facilitate a return to normalcy. This renewed interest has highlighted the broader applicability of vaccine passports beyond the COVID-19 context for managing public health and enabling safe travel and access to services.

One of the earliest proposals for a blockchain-enabled vaccine passport system was the Vaccine Credential Initiative (VCI) [161] established in 2020 by the Argonaut Project. The VCI aims to provide individuals with secure, privacy-preserving, and interoperable access to their immunization information. The VCI uses a combination of blockchain and secure enclaves to create a tamper-proof record of an individual's vaccine status.

Numerous non-governmental organizations, professional associations, and private companies have been developing health and identity documents to address the COVID-19 pandemic. For example, the World Economic Forum and the Commons Project worked on an online platform called "CommonPass" [157] that documents an individual's COVID-19 status, including vaccinations, PCR tests, and health declarations. IBM has also created a "Digital Health Pass" [74] for health verification of employees, customers, or visitors. Meanwhile, the International Air Transport Association launched the "IATA Travel Pass Initiative" [75] smartphone app to inform airline staff and passengers about testing and vaccination requirements.

In addition, the European Commission has proposed the creation of a "Digital Green Certificate" [47] to facilitate safe travel within the European Union. The "Digital Green Certificate" would use blockchain technology to provide a secure and tamper-proof record of an individual's COVID-19 vaccine status and test results. In a similar direction, various other countries have developed their own vaccine passport systems, including Denmark's "Corona Pass" [60] and Israel's "Green Pass" [61].

Governments and private organizations rushed to finish these projects to combat the pandemic, resulting in shoddy applications with significant issues. Furthermore, these applications were only tested in a limited environment. For example, "CommonPass" was trialled by a handful of airlines and governments, and "Health Pass" by a few organizations. In addition, the "Digital Green Certificate" only applied within the European Union, while Denmark's "Corona Pass" and Israel's "Green Pass" were only valid within their respective countries. Furthermore, these "Corona Pass" and "Green Pass" systems rely on a centralized authority to manage and verify information, creating the potential for further problems.

The downsides of these project initiatives are listed in Table 5.1. However,

Table 5.1: Drawbacks of Various Vaccine Passport Projects Initiated by Governments and Private Organizations

System	**Drawbacks**
Vaccine Credential Initiative (VCI)	– Requires healthcare providers to integrate with the VCI system, which may be a barrier to adoption
CommonPass	– Limited adoption, currently being trialed by a small number of airlines and governments
Health Pass by IBM	– Limited adoption, currently being used by a small number of organizations
Digital Green Certificate	– Only applicable within the European Union – Limited adoption, still under development
Denmark's Corona Pass	– Requires a centralized authority to manage and verify the information – Limited adoption, only applicable within Denmark
Israel's Green Pass	– Requires a centralized authority to manage and verify the information – Limited adoption, only applicable within Israel

these proposals demonstrate the increasing interest in using secure vaccine passport systems across multiple geographical regions. So, researchers have gradually come up with their proposals.

The paper [68] proposes a blockchain-based solution for COVID-19 management using digital immunity certificates. It highlights the benefits of data security, privacy, and cost-efficiency. However, the paper does not thoroughly address challenges like implementation difficulties, scalability, and real-world effectiveness. Further research is needed before this solution can be practically implemented.

The authors of the paper [13] proposed a privacy-preserving distributed platform for COVID-19 vaccine passports using blockchain and smart contracts. This platform securely creates, stores, and verifies digital vaccine certificates.

The author [67] proposed a blockchain-based COVID-19 vaccine passport system called VacciFi. It offers benefits such as integrity and verifiability. However, it faces significant challenges, including privacy concerns, potential inequality and exclusion, technical implementation issues, and hurdles in achieving widespread adoption.

The paper [140] introduces a blockchain-based system for secure vaccination records using smart contracts and IPFS storage. This system enhances data integrity and prevents forgery with unique hash values for each certificate.

The paper [1] discusses the challenges faced by Africa in procuring COVID-19 vaccines and authenticating certification. The authors suggest a blockchain-based system called BLOCOVID to secure and verify vaccination certificates using distributed ledger technology. Vaccine serial numbers and certificates are stored on the blockchain as hash values to ensure that they cannot be altered and are authentic.

The paper [23] addresses privacy issues related to the personal data of users by proposing a two-factor authentication system. One part of the system relies on information that the user possesses, such as biometrics like retina scans and fingerprints. The other part relies on information that the user knows, such as personal details like date of birth, gender, and country.

In the paper [165], the authors propose modifying the process of vaccine user verification by validating the user's physical license with the information provided by the user's QR code.

Another interesting work [109] demonstrates the use of priority-based vaccine distribution in areas with higher positive test results.

The authors [143] suggest a Hyperledger Fabric-based consortium blockchain solution to create digital vaccine passports (DVPs) for combating counterfeit paper passports during the COVID-19 pandemic. They recommend federated identity management for secure verification across different trust realms.

The paper [126] introduces a blockchain-based solution called Block-HPCT that incorporates smart contracts for digital health passports and contact tracing using proof of location. It utilizes trusted oracles, IPFS, and Hyperledger Fabric for secure data storage, aiming for transparency in COVID-19 information management.

In the paper [119], the authors introduce a protocol for managing digital COVID-19 certificates. It allows users to control data sharing in a hierarchical system using proxy re-encryption and blockchain.

The study [26] presents a blockchain-based vaccine passport system using a dual-chain framework: a public blockchain with IoT for transparent cold-chain logistics and a consortium blockchain for privacy and auditing. It employs distributed threshold signatures to prevent collusion in vaccine qualification and cryptographic tools to protect user privacy during customs checks.

The paper [54] introduces a system named UniVAC for verifying universal vaccine passports. It uses ciphertext policy attribute-based encryption and blockchain to provide secure access control to COVID-19 vaccine data. This setup ensures that transactions are transparently recorded and data indexing is reliable.

TABLE 5.2: A Comparative Analysis with Current State-of-the-Art

Article	Algorithm	Implementation	Fairness	Privacy	Access Control Policy	Additional Security Features				
						Prevention of Passport Forgery	Prohibition of Black Marketing of Vaccine Vials	Validation of Vaccine vial Authenticity	Use of Distributed Storage System - IPFS	Reward/Penalty based System for honest/ malicious entities
[68]	✔	✔	✘	✔	✔	✔	✘	✘	✔	✘
[13]	✘	✔	✘	✔	✔	✔	✘	✘	✔	✘
[67]	✘	✘	✘	✔	✔	✔	✘	✘	✘	✘
[140]	✘	✔	✘	✔	✘	✘	✘	✘	✔	✘
[1]	✘	✔	✘	✔	✘	✔	✘	✘	✘	✘
[23]	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘
[165]	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘
[109]	✔	✔	✔	✘	✘	✘	✘	✘	✘	✘
[143]	✔	✘	✘	✔	✘	✘	✘	✘	✘	✘
[126]	✘	✔	✘	✔	✔	✘	✘	✘	✔	✘
[119]	✔	✔	✘	✔	✘	✔	✘	✘	✔	✘
[26]	✘	✘	✔	✔	✘	✔	✘	✘	✘	✘
[54]	✔	✔	✘	✔	✔	✘	✘	✘	✔	✘
[85]	✘	✔	✘	✘	✔	✔	✔	✘	✘	✘
[103]	✔	✔	✘	✔	✔	✘	✘	✘	✘	✘
Proposed Model	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

The paper [85] introduces Vacchain, a blockchain-based system to improve the security and traceability of vaccine distribution. It presents a SYS-MAN mechanism for role verification, mutual agreement protocols for ownership transfer, and blockchain-based vaccine passports. The aim is to enhance data reliability and prevent counterfeiting.

The study [103] builds a blockchain application using Solidity smart contracts to enhance vaccine traceability and certificate reliability. Tested on various networks, it shows high performance and successful deployment.

The existing vaccine passport systems have notable limitations and research gaps. Most systems rely on a central authority, which requires trust in that authority. Not much research has been done on using blockchain technology to remove this centralization. The papers on blockchain-based solutions for COVID-19 management highlight significant challenges that impede practical implementation. Several studies, including those by [1], [13], [68], [119], [140], [143], emphasize issues related to scalability, integration with diverse healthcare systems, and regulatory compliance [23], [26], [54], [67], [85], [126]. Privacy concerns and the need for robust security measures are recurrent themes in the research by [23], [26], [54], [67], [85], [126]. Additionally, several studies, such as those by [109], [165], highlight the challenges of ensuring data integrity and preventing forgery. The need for widespread adoption, comprehensive data management, and user consent is also noted in the works by [54], [103]. There is significant untapped potential in using smart contracts integrated with blockchain. Critical issues such as fairness among involved parties, verification of vaccine authenticity, and prevention of black-market trading remain inadequately addressed. Our comparison with state-of-the-art systems, as found in Table 5.2, highlights these gaps and emphasizes the need for further research in this area.

In this chapter, we propose a smart contract-powered blockchain system that addresses various security aspects. The proposed vaccine passport system offers enhanced security and decentralization compared to existing solutions such as the EU Digital COVID Certificate (DCC), which relies on centralized storage, and IBM's Digital Health Pass, which depends on external verification processes. By leveraging blockchain, our system ensures tamper-proof records and improved user control over data. Instead of storing users' vaccination information directly on the blockchain or in a centralized database, we store it in decentralized IPFS storage. This approach provides greater security and privacy for users and reduces costs associated with storing large amounts of data on the blockchain. Our system uses smart contracts to ensure fairness among parties and verifies the authenticity of vaccine doses, thus addressing key security concerns.

5.2 System Model

Remark 3. *The problem is to develop a robust framework for vaccine passport systems that address critical issues such as forgery, privacy breaches, and inefficiencies in centralized solutions. Existing systems often fail to ensure data*

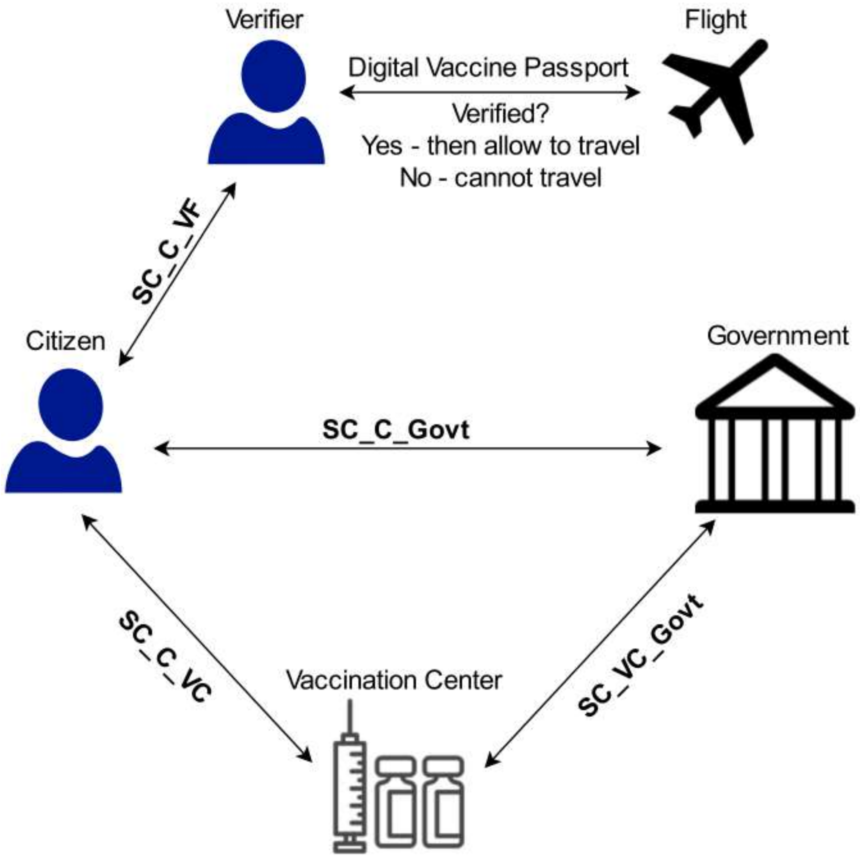


FIGURE 5.1: *Architecture of the vaccine passport system, illustrating the connections between the blockchain, government, vaccination centers, verifiers, and users.*

integrity, leaving records susceptible to tampering and unauthorized access. Additionally, these systems lack sufficient measures to protect user privacy, as sensitive vaccination information is typically stored in centralized databases that are vulnerable to breaches. Manual verification processes are also slow, prone to errors, and difficult to scale during large-scale verification scenarios. The proposed framework, leveraging blockchain technology, offers a decentralized, tamper-proof, and transparent system that ensures authenticity, protects user privacy, and streamlines verification processes. This approach provides a secure, and efficient solution for managing vaccine passports, addressing current challenges while preparing for future needs.

5.2.1 Components

In our system, we have the following entities/parties (Figure 5.1):

1. **Government (*Govt*):** It is one of the major entities in our system and has several significant functionalities.
   - Govt is responsible for registering VCs in the system after verifying their credentials.

  - Govt validates citizens' identity before generating $tokenID$ for taking vaccine dose.
  - Govt locks money in the SCs that gets transferred to the VCs for their service charges as per the policy.
  - Govt also ensures the adequate supply of vaccine doses to various VCs so that the entire process can run smoothly.

2. **Vaccination Center (VC):** It actively involved in the vaccination program. Once registered into the system, VC can start their job and obtain fees for the service offered.
3. **Citizen/User/Traveller (C):** It is the user (sometimes also referred to as $Citizen$ or $Traveller$) who takes the vaccine dose from the VC and then obtains the vaccine passport.
4. **Verifier (VF):** It is another entity that verifies if the citizens' vaccine passports are valid.
5. **Blockchain (BC):** Our proposed model employs a public blockchain e.g. Ethereum, which is a permissionless blockchain that allows anybody to join the network. BC is used as a tamper-proof log of records distributed across multiple nodes.
6. **Smart Contract (SC):** These are globally accessible executable pieces of code that regulate the key operations within the blockchain.
7. **InterPlanetary File System ($IPFS$):** We have used IPFS to store the citizens' vaccine passports in a distributed manner.

5.2.2 Security Goal

We state the security properties which must be realized by our proposed protocol:

- *Fairness*: To ensure that
  - A person/citizen can not forge a vaccine certificate.
  - A person should be able to validate the authenticity of the vaccine vial.
  - Vaccination centers can not misuse the vaccine doses for their own profit. Black marketing is prevented.
  - Global statistics of the vaccination process can be obtained.
- *Privacy*: To ensure that a person's private information, like name, address, etc, would not be compromised or leaked to the outside world.
- *Data Security*: To ensure that no one can tamper with the vaccine data or records.

5.2.3 Adversarial Model

- **User/Party/Player:** A user/party/player is said to be **honest** if they adhere to the system protocol. Otherwise, It is said to be **malicious** (i.e. when a user performs sporadically or deviates arbitrarily).

- **Adversary:** In the context of security, an adversary is a polynomial time algorithm that can compromise any user at any given point. This algorithm has an upper bound. The adversary is dynamic, meaning it can coordinate attacks through message exchanges on behalf of malicious users. On the other hand, the adversary cannot interfere with honest users' message exchanges, nor can it break cryptographic primitives like encryption schemes, digital signatures or hash functions except with negligible probability. Additionally, the adversary is limited in computational power and storage capabilities. Finally, we assume that all participants in our system are **rational**. By **rational**, we mean that when an adversary wants to exploit certain flaws in the underlying system, its primary and sole objective is to gain sufficient benefits - either monetary or useful information. Other sorts of malevolent functionality are not considered adversarial conduct here (e.g. wasting system resources - CPU, memory, time and so on).

5.2.4 Assumption

1. Every entity in our system has a unique $< SK, PK >$ pair where PK and SK represent the entity's address and authentication factor, respectively.

2. The Government of every country is responsible for distributing the vaccine doses to its country's vaccination centers. How the Govt gets the vaccine doses from the manufacturer (i.e. supply-chain system) is out of the scope of this work.

3. A Vaccination center (VC) must satisfy certain prerequisite conditions (e.g. it should be a hospital or healthcare unit) and appeal to the respective Government expressing their interest. The requirements or criteria may vary for different Governments of various countries. If they satisfy all the necessary measures, the Government introduces VC into the system by providing a unique ID.

4. To ensure that every citizen has access to the vaccine, we are not charging any fees for taking vaccine doses. However, Governments pay vaccination centers for their services on a per-vaccine basis.

5. Our system primarily uses the blockchain to store hashes of records, timestamps of significant events, issuing authority signatures, and other context-specific auxiliary information to ensure accountability, integrity, authenticity, and fairness. Citizen vaccination information is stored in IPFS, a decentralized peer-to-peer storage system.

6. Information about registered vaccination centers, such as their location and contact details, is displayed on a public dashboard. Any updates to this information are available to the public in real-time.

7. To simplify the problem caused by regulatory issues, we do not specify any particular vaccines by name. Instead, we assume that a person must take a dose of the vaccine. In reality, citizens must receive multiple vaccine doses periodically.

5.2.5 Protocol Design

TABLE 5.3: Terminology & Notation used in our Scheme

Abbreviation	**Interpretation**
V	Vaccine Dose/Vial
VP	Vaccine Passport
Govt	Government
VC	Vaccination Center
C	Citizen
VF	Vaccine Passport Verifier
$< SK_{Entity}, PK_{Entity} >$	Key pair used by an *Entity*, $Entity \in \{Govt, VC, C, VF\}$
vID	Vaccine Vial ID
$vcID$	Vaccination Center ID
$tokenID$	Token ID
$applID$	Application ID
cID	Content Identifier in IPFS System
T_{Event}	Timestamp, when the Event occurs

Our proposed scheme is divided into 6 major modules, which are discussed below.

Module 1: Registration of Vaccination Centers (VCs)
Before participating in the vaccination program, VCs must acquire a licence. And they must meet certain pre-requisite conditions imposed by the individual country's government to obtain a licence. After validating the required credentials, the government registers an entity as VC in the system, and the VC is assigned a unique vaccination center ID ($vcID$) generated by the smart contract. Figure 5.2 depicts the VC registration process.

I. At first, the VC sends an application (*Appl*) to the Govt, furnishing all the required details in off-chain communication and then registers the timestamp of the application (T_{Appl}) on-chain through SC.

II. Govt, in turn, creates a digest of the received application and puts it on the BC.

III. Once VC agrees to the hash value, SC generates an Application ID ($ApplID$) for future reference. On the contrary, if the VC does not consent to the hash value, the protocol terminates, and the VC needs to send a new application again.

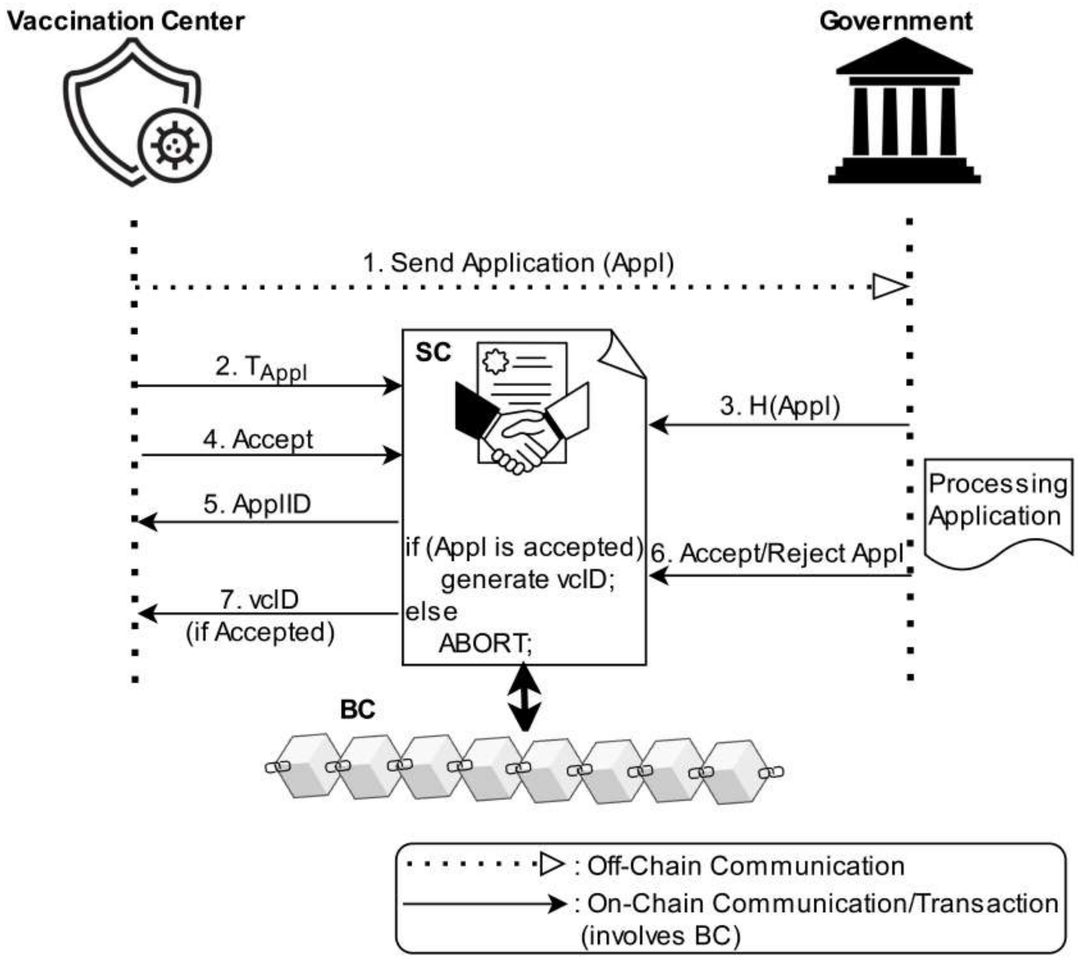

Vaccination Center
Government
1. Send Application (Appl)
SC
2. T_Appl
3. H(Appl)
4. Accept
5. ApplID
Processing Application
if (Appl is accepted) generate vcID; else ABORT;
6. Accept/Reject Appl
7. vcID (if Accepted)
BC
: Off-Chain Communication
: On-Chain Communication/Transaction (involves BC)

FIGURE 5.2: *Registration of Vaccination center*

IV. Next, the Govt processes the application within a fixed period and verifies if the application satisfies the necessary requirements. Accordingly, Govt accepts or rejects the application on-chain. If the application gets accepted, the SC will generate and assign a unique $vcID$ against the applicant.

Module 2: Refill of Vaccine Vials/Doses

Govt distributes the vaccine vials to the registered vaccination centers. Let us say that V is the set of vaccine vial IDs, which will be transferred to the VC with ID $vcID$ upon VC applying for refilling its vaccine stock. The cardinality of the set V is $|V| = n$.

$$V = \{v_i\}, i \in \mathbb{N} \wedge i \in [1, n]$$

I. First, the VC applies to the Govt to refill its vaccine stock. The timestamp of the application ($T_{RefillAppl}$) is recorded on BC.

II. Before transferring the vaccine vials (corresponding to the set V) to the VC, the Govt will compute the MR (i.e. Merkle Tree Root Hash) for the set V, organizing the member elements in ascending order. Figure 5.3 demonstrate the computation of MR for the set V, where $|V| = 8$.

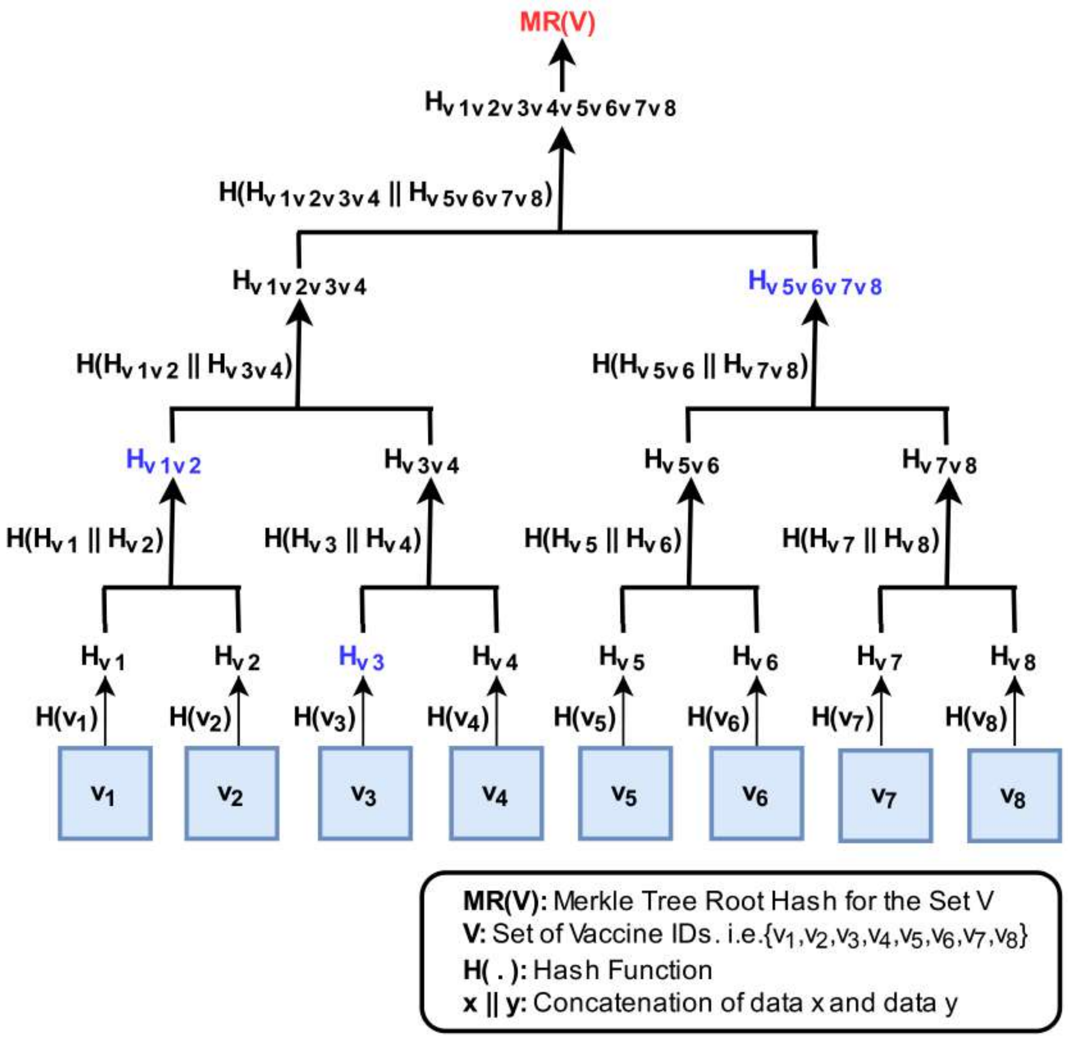

MR(V)
$H_{v1v2v3v4v5v6v7v8}$
$H(H_{v1v2v3v4} \| H_{v5v6v7v8})$
$H_{v1v2v3v4}$
$H_{v5v6v7v8}$
$H(H_{v1v2} \| H_{v3v4})$
$H(H_{v5v6} \| H_{v7v8})$
H_{v1v2}
H_{v3v4}
H_{v5v6}
H_{v7v8}
$H(H_{v1} \| H_{v2})$
$H(H_{v3} \| H_{v4})$
$H(H_{v5} \| H_{v6})$
$H(H_{v7} \| H_{v8})$
H_{v1} H_{v2} H_{v3} H_{v4} H_{v5} H_{v6} H_{v7} H_{v8}
$H(v_1)$ $H(v_2)$ $H(v_3)$ $H(v_4)$ $H(v_5)$ $H(v_6)$ $H(v_7)$ $H(v_8)$
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
MR(V): Merkle Tree Root Hash for the Set V
V: Set of Vaccine IDs. i.e.$\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\}$
H(.): Hash Function
x || y: Concatenation of data x and data y

FIGURE 5.3: *Computation of MR for set V*

Computing MR, Govt puts the value on the BC as the commitment of set V (i.e. $BC \leftarrow MR(V)$) and then sends the corresponding vaccine vials to the VC. Govt also locks the service charge of the VC in the SC apriori.

III. Receiving the vaccine vials, VC also computes the MR based on the vial IDs it got and then checks whether the calculated value matches the one that the Govt stored on BC. If the values match, VC accepts the vials; otherwise, it refuses the delivery and returns to the Govt again.

IV. If VC accepts the vaccine stock, SC generates a unique $stockID$ and assigns the stock to the VC. Otherwise, SC unlocks the locked money and transfers it to the Govt.

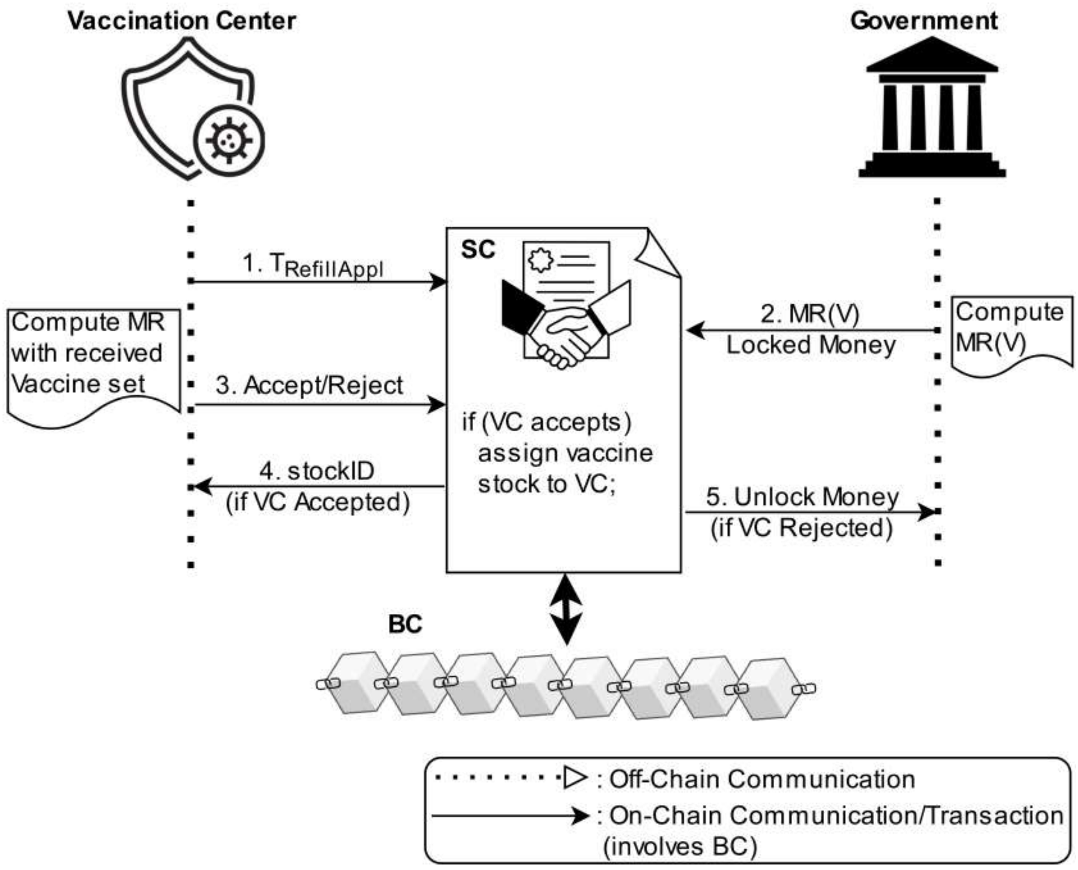

Vaccination Center
Government
SC
1. $T_{RefillAppl}$
Compute MR with received Vaccine set
3. Accept/Reject
4. stockID (if VC Accepted)
if (VC accepts) assign vaccine stock to VC;
2. MR(V) Locked Money
Compute MR(V)
5. Unlock Money (if VC Rejected)
BC
: Off-Chain Communication
: On-Chain Communication/Transaction (involves BC)

FIGURE 5.4: *Refilling Vaccine Stock*

Figure 5.4 illustrates the process of refilling vaccine stock.

Module 3: Obtaining *TokenID*
If citizens want to receive the vaccine, they must obtain a unique token ID from the Govt. The process typically involves three steps.

I. The citizen must contact the appropriate Govt authority and provide valid proof of citizenship to express their interest in being vaccinated. This communication usually occurs offline.

II. Simultaneously, the citizen should also generate a message digest of their private information, which includes their name, address, date of birth, and citizen ID, and record it on the *BC*. Since the hash is stored on *BC*, it preserves data privacy.

$$\begin{aligned} m &\leftarrow (Name||Addr||DOB||CitizenID) \\ commit_m &\leftarrow H(m) \\ BC &\leftarrow commit_m \end{aligned}$$

III. Once the Govt verifies that the citizen has properly recorded the message digest on the *BC* and that it matches the information provided offline, the Govt will issue the citizen a unique *tokenID* through the *SC*.

Figure 5.5 illustrates the process of *tokenID* generation.

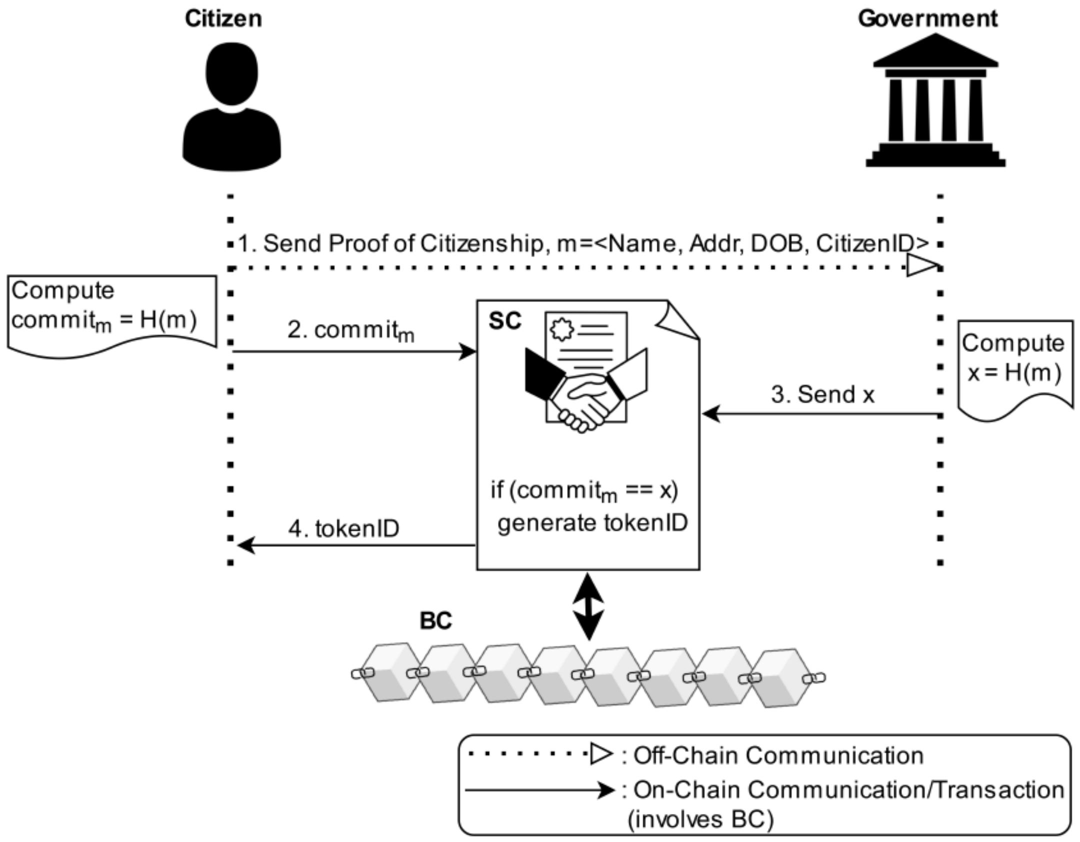

Citizen
Government
1. Send Proof of Citizenship, m=<Name, Addr, DOB, CitizenID>
Compute commit$_m$ = H(m)
2. commit$_m$
SC
Compute x = H(m)
3. Send x
if (commit$_m$ == x) generate tokenID
4. tokenID
BC
: Off-Chain Communication
: On-Chain Communication/Transaction (involves BC)

FIGURE 5.5: *Obtaining TokenID*

Module 4: Injecting Vaccine to Citizen by Vaccination Center
Once a citizen has received a unique *tokenID*, they are eligible to receive a vaccine at a convenient VC. Information about the VCs, such as their address, contact details, and vaccine availability, is publicly available. When a citizen goes to the VC to receive the vaccine, a protocol runs between the citizen and the VC to ensure that the vaccine is administered correctly.

I. First, the citizen reveals their *tokenID* to VC. Then VC obtains information regarding the vaccination status of the citizen from the *BC*. If the citizen is not vaccinated, the protocol proceeds further; otherwise, abort.

II. In the next stage, both the parties, i.e. VC and citizen, lock a certain amount into the *SC*. Locking money ensures fairness. If the parties behave maliciously later in the protocol, they will be penalized by deducting their locked money. Honest parties will eventually recover their locked money at later stages.

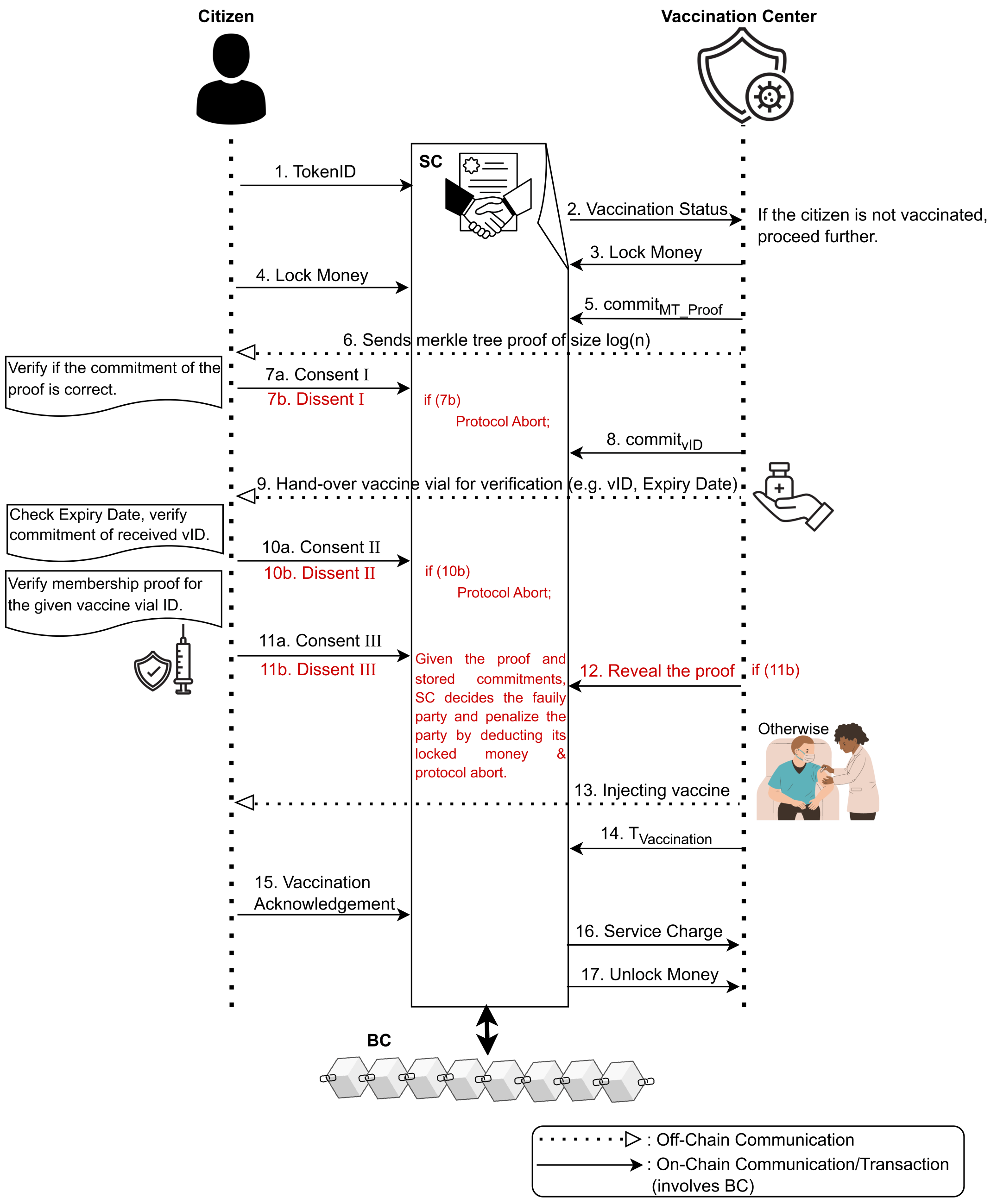

Citizen
Vaccination Center
SC
1. TokenID
2. Vaccination Status
If the citizen is not vaccinated, proceed further.
3. Lock Money
4. Lock Money
5. commit$_{MT\_Proof}$
6. Sends merkle tree proof of size log(n)
Verify if the commitment of the proof is correct.
7a. Consent I
7b. Dissent I
if (7b)
Protocol Abort;
8. commit$_{vID}$
9. Hand-over vaccine vial for verification (e.g. vID, Expiry Date)
Check Expiry Date, verify commitment of received vID.
10a. Consent II
10b. Dissent II
if (10b)
Protocol Abort;
Verify membership proof for the given vaccine vial ID.
11a. Consent III
11b. Dissent III
Given the proof and stored commitments, SC decides the fauily party and penalize the party by deducting its locked money & protocol abort.
12. Reveal the proof
if (11b)
Otherwise
13. Injecting vaccine
14. T$_{Vaccination}$
15. Vaccination Acknowledgement
16. Service Charge
17. Unlock Money
BC
: Off-Chain Communication
: On-Chain Communication/Transaction (involves BC)

FIGURE 5.6: *Injecting Vaccine to Citizen by Vaccination Center*

III. Once the money gets locked, VC proceeds further to inject the vaccine dose. VC picks the vaccine vial. Next, VC proves towards the citizen that the chosen vial is authentic and received from the Govt. Essentially, VC convinces this fact by providing a set membership proof corresponding to the selected vaccine vial ID. VC generates a Merkle Tree Proof of $\log(n)$ in size. VC sends the $\log(n)$ size proof to the citizen in offline mode and puts the hash of the proof on *BC* as a commitment of the proof. Without loss of generality, let's say VC picks the vaccine vial with vID - $v4$ to be injected (Figure 5.3). In this case, the proof consists of $< H_{v3}, H_{v1v2}, H_{v5v6v7v8} >$ (Blue colored internal nodes in Figure 5.3).

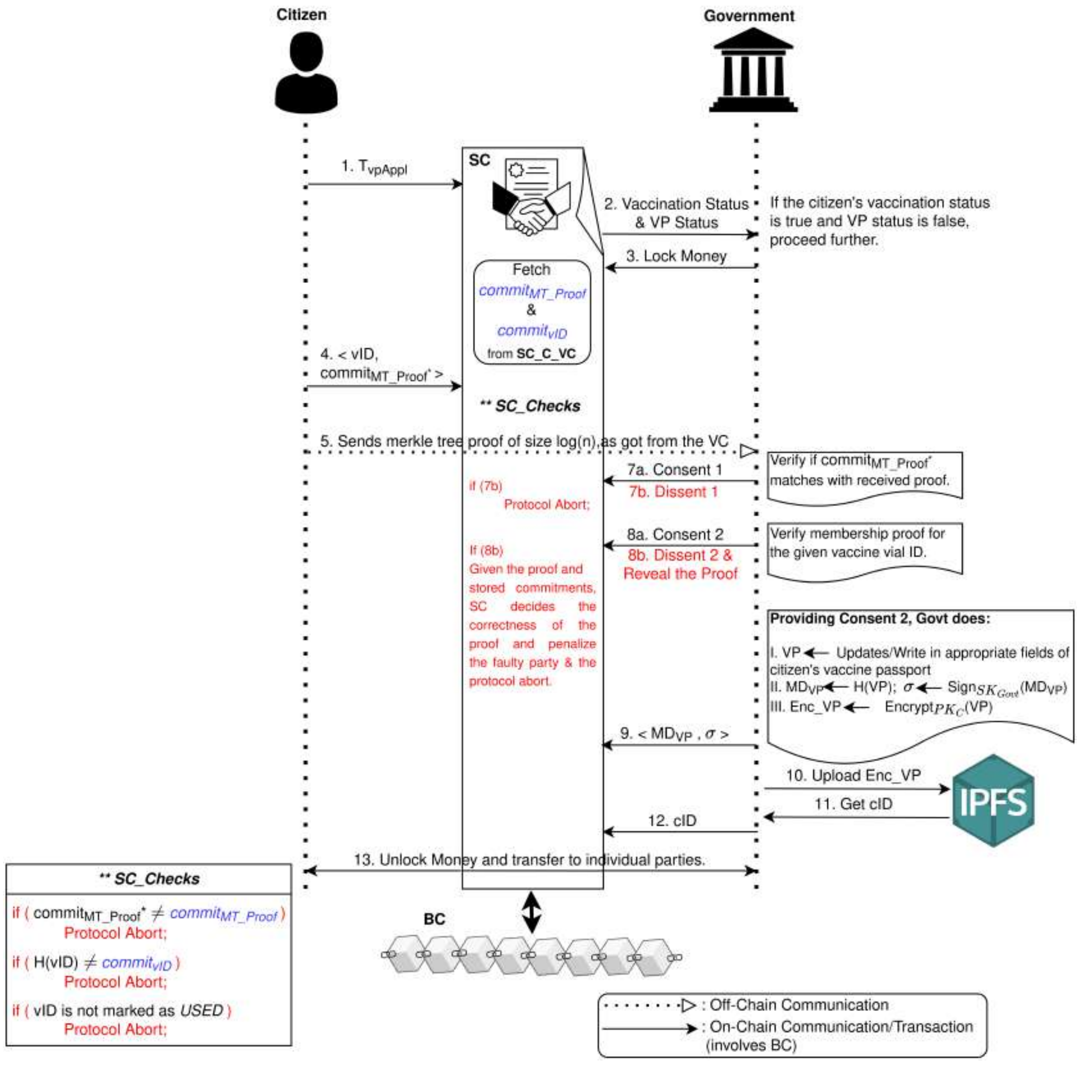
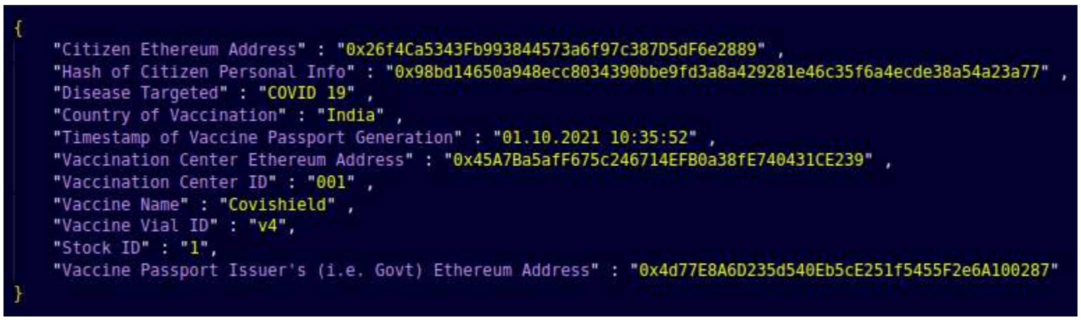

Citizen
Government
SC
1. T_{vpAppl}
2. Vaccination Status & VP Status
If the citizen's vaccination status is true and VP status is false, proceed further.
3. Lock Money
Fetch $commit_{MT\_Proof}$ & $commit_{vID}$ from SC_C_VC
4. < vID, $commit_{MT\_Proof^*}$ >
** SC_Checks
5. Sends merkle tree proof of size log(n) as got from the VC
Verify if $commit_{MT\_Proof^*}$ matches with received proof.
7a. Consent 1
7b. Dissent 1
if (7b) Protocol Abort;
8a. Consent 2
8b. Dissent 2 & Reveal the Proof
Verify membership proof for the given vaccine vial ID.
If (8b) Given the proof and stored commitments, SC decides the correctness of the proof and penalize the faulty party & the protocol abort.
Providing Consent 2, Govt does:
I. VP ← Updates/Write in appropriate fields of citizen's vaccine passport
II. MD_{VP} ← H(VP); σ ← $Sign_{SK_{Govt}}(MD_{VP})$
III. Enc_VP ← $Encrypt_{PK_C}(VP)$
9. < MD_{VP} , σ >
10. Upload Enc_VP
11. Get cID
IPFS
12. cID
13. Unlock Money and transfer to individual parties.
** SC_Checks
if ($commit_{MT\_Proof^*} \neq commit_{MT\_Proof}$) Protocol Abort;
if (H(vID) $\neq commit_{vID}$) Protocol Abort;
if (vID is not marked as USED) Protocol Abort;
BC
: Off-Chain Communication
: On-Chain Communication/Transaction (involves BC)

FIGURE 5.7: *Generating and Storing Citizen's Vaccine Passport on IPFS*



```
{
    "Citizen Ethereum Address" : "0x26f4Ca5343Fb993844573a6f97c387D5dF6e2889" ,
    "Hash of Citizen Personal Info" : "0x98bd14650a948ecc8034390bbe9fd3a8a429281e46c35f6a4ecde38a54a23a77" ,
    "Disease Targeted" : "COVID 19" ,
    "Country of Vaccination" : "India" ,
    "Timestamp of Vaccine Passport Generation" : "01.10.2021 10:35:52" ,
    "Vaccination Center Ethereum Address" : "0x45A7Ba5afF675c246714EFB0a38fE740431CE239" ,
    "Vaccination Center ID" : "001" ,
    "Vaccine Name" : "Covishield" ,
    "Vaccine Vial ID" : "v4",
    "Stock ID" : "1",
    "Vaccine Passport Issuer's (i.e. Govt) Ethereum Address" : "0x4d77E8A6D235d540Eb5cE251f5455F2e6A100287"
}
```

FIGURE 5.8: *Vaccine Passport as a JSON File*

$$MT\_Proof \leftarrow < H_{v3}, H_{v1v2}, H_{v5v6v7v8} >$$
$$commit_{MT\_Proof} \leftarrow H(H_{v3}||H_{v1v2}||H_{v5v6v7v8})$$
$$BC \leftarrow commit_{MT\_Proof}$$

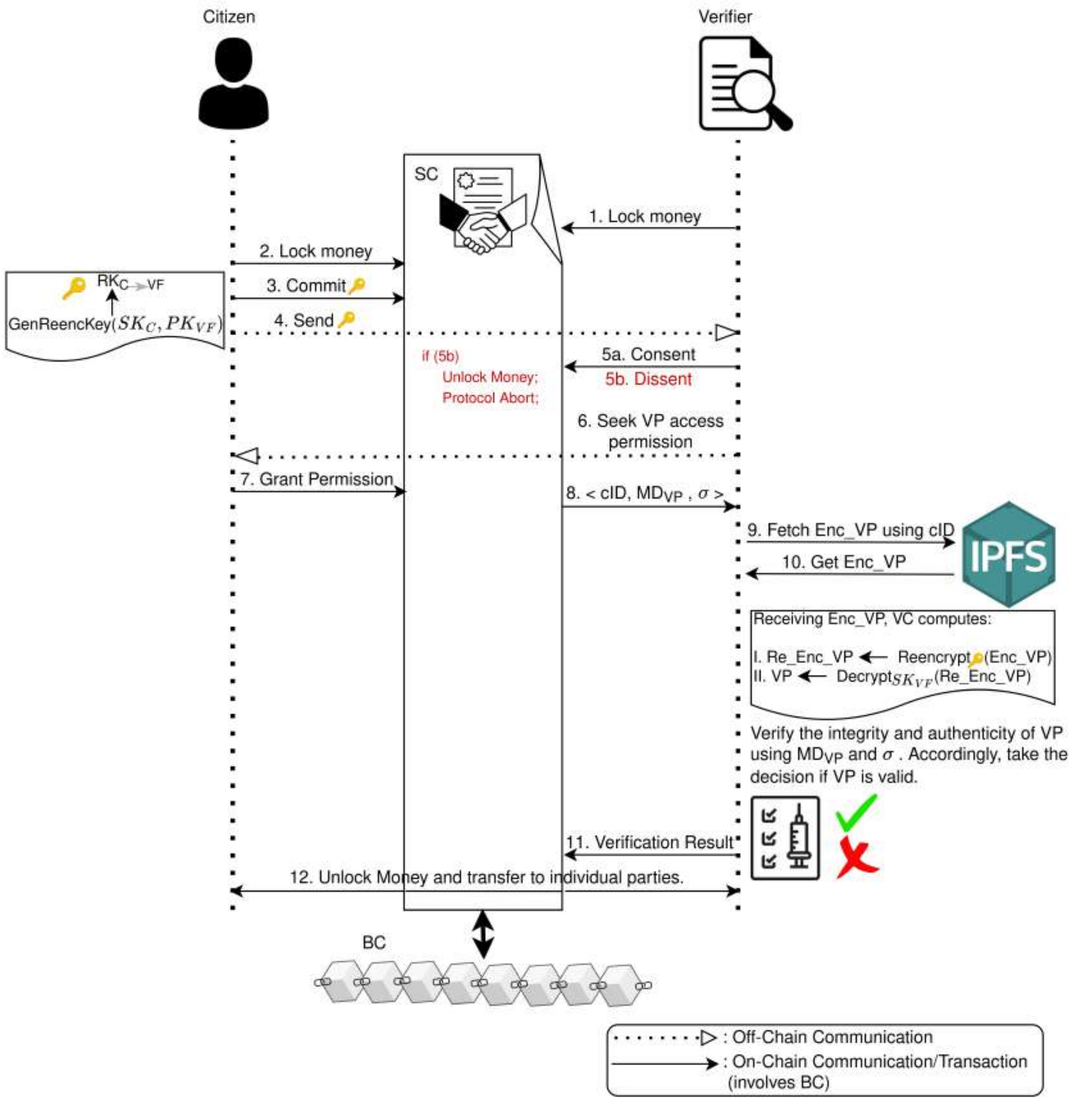

Citizen
Verifier
SC
1. Lock money
2. Lock money
3. Commit
4. Send
RK C→VF
GenReencKey(SK_C, PK_{VF})
5a. Consent
5b. Dissent
if (5b)
Unlock Money;
Protocol Abort;
6. Seek VP access permission
7. Grant Permission
8. < cID, MD_VP , σ >
9. Fetch Enc_VP using cID
10. Get Enc_VP
IPFS
Receiving Enc_VP, VC computes:
I. Re_Enc_VP ← Reencrypt(Enc_VP)
II. VP ← Decrypt$_{SK_{VF}}$(Re_Enc_VP)
Verify the integrity and authenticity of VP using MD_VP and σ . Accordingly, take the decision if VP is valid.
11. Verification Result
12. Unlock Money and transfer to individual parties.
BC
: Off-Chain Communication
: On-Chain Communication/Transaction (involves BC)

FIGURE 5.9: *Verification Process of Citizen's VP*

VC sends the MT_Proof for vial ID - v_4 to the citizen and keeps the commitment of the proof $commit_{MT\_Proof}$ on the BC.

IV. Receiving MT_Proof, the citizen verifies if it matches the $commit_{MT\_Proof}$ stored on BC. If so, citizen provides their consent (**Consent I** as mentioned in Figure 5.6); else, protocol aborts.

V. Next, VC hands over the vaccine vial to the citizen, keeping the commitment of vaccine vial ID on BC. Subsequently, SC conducts a thorough check to confirm the freshness of the vaccine vial associated with the given commitment, ensuring it has not been used previously. In our example, VC does the following:

$$commit_{vID} \leftarrow H(v4)$$
$$BC \leftarrow commit_{vID}$$

VI. Then citizen verifies if the given vaccine vial satisfies the $commit_{vID}$. Also, citizen checks the expiry date and other important information printed on the vial. If the citizen finds everything is right, s/he provides their consent (**Consent II** as mentioned in Figure 5.6); else, protocol aborts.

VII. Next, the citizen again provides their consent for the third time (**Consent III** as mentioned in Figure 5.6). This time citizen verifies whether the given MT_Proof for the specified vial with vID matches the Merkle Tree Root hash, i.e., MR. Notably, Govt stored this MR on the BC while delivering the vaccine vials to the VC.

On the contrary, if the citizen finds that the MR does not match the given proof for the vial, then the citizen complaints to the SC. If citizen dissents, VC must reveal the proof to the SC within a specific time window. SC verifies the correctness of the complaint and judges the faulty party. Consequently, the malicious party gets penalized and it will lose its locked money.

VIII. Upon receiving **Consent III** from the citizen, the VC administers the vaccine dose and records the vaccination timestamp on the BC.

IX. Following vaccination, the citizen is required to acknowledge receipt within a specified time window. In the event of a negative acknowledgement, where the citizen denies receiving the vaccine despite the VC registering a timestamp of vaccination, legal intervention may be necessary to address the discrepancy. Although real-time image capture during vaccination could potentially resolve this issue using cameras/IoT devices, it falls outside the scope of this work.

X. Upon receiving a positive acknowledgement from the citizen, the corresponding *vID* is marked as *USED*. Simultaneously, the VC is granted its service charge, which had been previously locked in the SC by the Govt during the initial dispatch of the vaccine stock. Additionally, the SC unlocks the VC's security deposit. However, the locked amount from the citizen is not immediately released for security reasons. It will be released after the receipt of the Vaccine Passport (VP) as part of the subsequent protocol between the citizen and the Govt.

Module 5: Generating and Storing of Citizen's Vaccine Passport
Following the successful vaccine administration, the citizen's vaccination status is promptly updated on the BC. Subsequently, citizens are required to apply for a VP from the government, encompassing crucial details such as the vaccination date, time, vaccine vial ID, and VC information. For a visual representation of a typical VP's contents, please refer to Figure 5.8.

In our system, VP is securely stored off-chain through the IPFS system. A citizen must initiate the VP application process to unlock the funds held during the upfront vaccination protocol. Moreover, from a security perspective, this protocol assumes significance as it empowers the SC to verify the accuracy of information provided by the citizen. The process of VP application, generation, and storage generally involves the following steps:

I. A citizen applies for a VP from the government by locking a certain amount on the SC and recording the application timestamp on the BC.

II. If the SC verifies the applicant's vaccination status as true and confirms the absence of the VP, the government proceeds to the next stage by locking a certain amount on the SC.

III. Subsequently, the citizen needs to substantiate the truth of their vaccination by disclosing the vaccine vial ID (*vID*) and committing the Merkle Tree Proof ($commit_{MT\_Proof^*}$) on the SC for vID membership verification. Simultaneously, the citizen sends the proof to the government offline.

IV. The SC retrieves the stored values for vial ID commitment and Merkle tree proof commitment, previously shared by the VC, upon which the citizen provided its Consent I and Consent II (Figure 5.6). If the commitment values match those shared by the citizen, the protocol proceeds; otherwise, it terminates. The SC also checks if the vial with *vID* is marked as *USED*.

V. Upon receiving the Merkle tree proof offline, the government verifies its match with the commitment. Consequently, the government provides its initial response - **Consent 1** or **Dissent 1**.

VI. Once Consent 1 is given, the government verifies if the Merkle tree proof validates the vial's membership with *vID*. Subsequently, the government provides its **Consent 2** or **Dissent 2**. If Dissent 2 is given, the government must submit the proof to the SC.

VII. With Consent 2 granted, the government initiates the creation of the citizen's VP by performing the following operations:

  1) Creates the citizen's VP.
  2) Computes the message digest of the VP (i.e., MD_{VP}).
  3) Signs digitally on the MD_{VP}.
  4) Keeps the MD_{VP} and its signature on the BC.
  5) Encrypts the VP using PK_C.
  6) Uploads the encrypted VP of the citizen to IPFS.
  7) Retrieves the cID from IPFS.
  8) Records the cID on the BC.

$$\begin{aligned}
MD_{VP} &\leftarrow H(VP)\\
\sigma &\leftarrow SIG_{SK_{Govt}}(MD_{VP})\\
BC &\leftarrow < MD_{VP}, \sigma >\\
Enc\_VP &\leftarrow Encrypt_{PK_C}(VP)\\
cID &\leftarrow \ Uploads\ Enc\_VP\ to\ IPFS\\
BC &\leftarrow cID
\end{aligned}$$

VIII. Once the citizen's VP is generated and cID is recorded on the BC, the SC unlocks the security deposits for both parties. It is important to note that, at this time, the citizen receives the money not only for this particular protocol but also for the preceding protocol conducted between the citizen and the vaccination center.

Refer to Figure 5.7 for an illustration of the process of creating and storing a vaccine passport for a citizen.

Module 6: Verification of Vaccine Passport
When a verifier proceeds to check a user's VP, s/he must pass through a protocol as shown in Figure 5.9.

I. VF seeks permission from the citizen to check their VP. Once the citizen grants so, VF obtains the cID through SC interaction. Obtaining the cID, VF can fetch the citizen's encrypted VP from $IPFS$.

II. Since VP was encrypted under the citizen's public key (i.e. PK_C), VF needs the citizen's secret key (i.e. SK_C) to decrypt it. However, sharing SK compromises system security. So, instead of sharing a secret key, we are introducing the Proxy Re-encryption technique here. However, the proxy is absent here; the proxy's job is delegated to the end parties.

III. The citizen generates a re-encryption key and then shares the key with the VF, while keeping the key's commitment on the BC.

$$\begin{aligned}
RK_{C\rightarrow VF} &\leftarrow GenReencKey(SK_C, PK_{VF})\\
commit_{RK} &\leftarrow H(RK_{C\rightarrow VF})\\
BC &\leftarrow commit_{RK}
\end{aligned}$$

IV. Receiving the $RK_{C\rightarrow VF}$, VF will re-encrypt the encrypted file. This re-encryption enables the VF to decrypt the file using its own secret key (SK_{VF}).

V. After the decryption, VF fetches the message digest of the citizen's VP (i.e., MD_{VP}) and checks if it complies with the decrypted file. VF also verifies the issuing VC's signature σ from BC. And then, the VF decides if the citizen's VP is valid. A record containing the VF's details, timestamp, and verification result is put on the BC.

5.2.6 Implementation & Technical Details

As per the protocols described in Subsection 5.2.5, the entire system can be divided into six main modules. Each module facilitates the interaction between multiple parties and records the transactions in the blockchain. These six main modules serve the following purposes:

1. **Registration of New Vaccination Centers**
2. **Refilling Vaccine Stock at Vaccination Centers**
3. **Obtaining TokenID by Citizen**
4. **Injecting Vaccine to Citizen by Vaccination Centers**
5. **Generating & Storing Vaccine Passport by Government**
6. **Verifying Vaccine Passport by Verifier**

Each module is written as an algorithm, converted into smart contract codes, and deployed on the Sepolia Test Network. Each smart contract consists of a set of *structures*, *mappings*, and *methods*. We have documented *timestamps* corresponding to various events or instances when an entity invokes a function, and these are detailed in Table 5.4. Readers can refer to Table 5.5 and Table 5.6, respectively, for the necessary structures and mapping definitions used in our algorithms.

1. **Algorithm for Registration of New Vaccination Centers:** The algorithm 4 overseeing the *VC* registration process (as discussed in *Subsection 5.2.5 Module 1*) comprises specific methods that are invoked in sequence by the alternating parties (*VC* and *Govt*) at specific time intervals.

   This algorithm 4 allows new vaccination centers *(VC)* to register themselves onto the blockchain network. Once registered, they are provided with a unique ID *(vcID)* and can start administering vaccines.
   Sequence of methods in algorithm 4:

   – *timestampRegAppl* $\rightarrow$ *VC* creates *RegAppl* and sets T_{Appl}.
   – *regApplHash* $\rightarrow$ *Govt* submits the hash of the application.
   – *decideOnAcceptanceHash* $\rightarrow$ *VC* accepts/rejects the hash.
   – *decideOnAcceptanceRegAppl* $\rightarrow$ *Govt* accepts/rejects application.

2. **Algorithm for Refilling Vaccine Stock at Vaccination Centers:** The algorithm 5 facilitates the replenishment of vaccine vials or doses at vaccination centers, with crucial information securely stored on the blockchain (as discussed in *Subsection 5.2.5 Module 2*). This ensures transparency and security in the vaccine distribution process. The algorithm comprises specific functions that are alternately invoked by *VC* and *Govt* in a timely manner.
   Sequence of methods in algorithm 5:

Table 5.4: Timestamp Definitions Across Various Modules of our Vaccine Passport System

Module 1: *Registration of Vaccination Centers*	
Abbreviation	**Interpretation**
$T_{regAppl}$	Timestamp when *VC* submits the registration application
$T_{hashAppl}$	Timestamp when *Govt* submits hash of the received application
$T_{decideOnHash}$	Timestamp when *VC* consents/descents on the hash value
$T_{decideOnAppl}$	Timestamp when *Govt* accepts/rejects the registration application
Module 2: *Refilling of Vaccine Stock*	
Abbreviation	**Interpretation**
$T_{refillAppl}$	Timestamp when *VC* submits the refill application
$T_{commitment}$	Timestamp when *Govt* commits MR of the vaccine set
Module 3: *Obtaining TokenID*	
Abbreviation	**Interpretation**
T_{Appl}	Timestamp when *C* submits token application
$T_{Verification}$	Timestamp when *Govt* provides result verifying application
Module 4: *Injecting Vaccine*	
Abbreviation	**Interpretation**
$T_{protocolBegins}$	Timestamp when *C* initiates the protocol
$T_{lockMoneyByVC}$	Timestamp when *VC* locks money
$T_{lockMoneyByC}$	Timestamp when *C* locks money
$T_{commit_{MT\_Proof}}$	Timestamp when *VC* commits *MT_Proof*
$T_{consent1}$	Timestamp when *C* sends consent1/descent1
$T_{commit_{vID}}$	Timestamp when *VC* commits to *vID*
$T_{consent2}$	Timestamp when *C* sends consent2/descent2
$T_{consent3}$	Timestamp when *C* sends consent3/descent3
$T_{moneyReceivedByC}$	Timestamp when *C* receives it's locked money
$T_{moneyReceivedByVC}$	Timestamp when *VC* receives it's locked money
$T_{protocolEnds}$	Timestamp when *VC* ends the protocol successfully
Module 5: *Generating and Storing Vaccine Passport*	
Abbreviation	**Interpretation**
$T_{lockMoneyByC}$	Timestamp when *C* locks money
$T_{lockMoneyByGovt}$	Timestamp when *Govt* locks money
$T_{provideVaccinationProof}$	Timestamp when *C* sends the vaccination proof
$T_{consent1}$	Timestamp when *Govt* sends consent1/descent1
$T_{consent2}$	Timestamp when *Govt* sends consent2/descent2
$T_{issueVP}$	Timestamp when *Govt* issues *VP*
$T_{moneyReceivedByC}$	Timestamp when *C* receives it's locked money
$T_{moneyReceivedByGovt}$	Timestamp when *Govt* receives it's locked money
Module 6: *Verifying Vaccine Passport*	
Abbreviation	**Interpretation**
$T_{lockMoneyByVF}$	Timestamp when *VF* locks money
$T_{lockMoneyAndCommitRkByC}$	Timestamp when *C* locks money and commits Re-encryption Key
$T_{provideConsent}$	Timestamp when *VF* provides consent
$T_{grantAccessByC}$	Timestamp when *C* grants access to *VF*
$T_{fetchVPInfo}$	Timestamp when *VF* fetches *VP* information
$T_{verificationResult}$	Timestamp when *VF* sends verification result
$T_{unlockMoney}$	Timestamp when security money gets unlocked

– *refillStockAppl* $\rightarrow$ *VC* submits *ReStockAppl* to *Govt*.

– *commitVaccineSet* $\rightarrow$ *Govt* commits the MR of the vaccine set to be delivered and also locks the service charge for *VC* on *SC*.

– *decideOnAcceptanceVaccineSet* $\rightarrow$ *VC* provides its consent if the MR matches with the received vaccine vials set; otherwise, it declines.

– *takeAwayLockedMoney* $\rightarrow$ *Govt* can withdraw the locked amount if *VC* denies accepting the vaccine set or becomes unresponsive.

Table 5.5: Structs Used in our Implementation

Module 1: *Registration of Vaccine Centers*	
Struct Name	**Members**
VC	*vcID*, *currentStockID*, *vialsInStock*, *moneyEarned*
RegAppl	*underReview*, $T_{regAppl}$, $T_{hashAppl}$, *hash*, $T_{decideOnHash}$, *decision*, $T_{decideOnAppl}$, *regApplID*
Module 2: *Refilling of Vaccine Stock*	
Struct Name	**Members**
ReStockAppl	*refillApplID*, $T_{refillAppl}$, *underProcess*, *vialsCount*, *commitment*, $T_{commitment}$
$VaccineStock$	*stockID*, *owner*, *vialsCount*, *stockMR*
Module 3: *Obtaining TokenID*	
Struct Name	**Parameters**
Citizen	*citizenInfoDigest*, *tokenID*, *vaccinationStatus*, *vpStatus*, *cID*
$TokenAppl$	*tokenApplID*, *citizenInfoDigest*, *underReview*, $T_{tokenAppl}$, *result*, T_{result}
Module 4: *Injecting Vaccine*	
Struct Name	**Parameters**
$InjectingProtocol$	*protocolID*, *underProcess*, *tokenID*, *vcID*, $T_{protocolBegins}$, $T_{lockMoneyByVC}$, $T_{lockMoneyByC}$, $commit_{MT\_Proof}$, $T_{commit_{MT\_Proof}}$, *consent1*, $T_{consent1}$, $commit_{vID}$, $T_{commit_{vID}}$, *consent2*, $T_{consent2}$, *consent3*, $T_{consent3}$, $T_{vaccination}$, $T_{moneyReceivedByC}$, $T_{moneyReceivedByVC}$, *acknowledgement*, $T_{acknowledgement}$
Module 5: *Generating & Storing Vaccine Passport*	
Struct Name	**Parameters**
VP	MD_{VP}, σ, *cID*
$VPAppl$	*vpApplID*, *applicantTokenID*, $T_{lockMoneyByC}$, $T_{lockMoneyByGovt}$, $T_{provideVaccinationProof}$, *consent1*, $T_{consent1}$, *consent2*, $T_{consent2}$, $T_{issueVP}$, $T_{moneyReceivedByC}$, $T_{moneyReceivedByGovt}$
Module 6: *Verifying Vaccine Passport*	
Struct Name	**Parameters**
$VerificationProtocol$	*vfProtocolID*, *underExecution*, *tokenID*, *vfAddr*, $T_{lockMoneyByVF}$, $T_{lockMoneyAndCommitRkByC}$, *consent*, $T_{provideConsent}$, $T_{grantAccessByC}$, $T_{fetchVPInfo}$, *verificationResult*, $T_{verificationResult}$, $T_{unlockMoney}$

3. **Algorithm for Obtaining TokenID by Citizen:** The algorithm 6 enables citizens to obtain a unique TokenID, a prerequisite for accessing vaccination services. As mentioned in *Subsection 5.2.5 Module 3*), the process begins with a citizen (C) applying for the *TokenID* from the *Govt*. To protect privacy, the citizen's private information is initially transmitted to the *Govt* through an off-chain mode. After this, all subsequent transactions take place on the blockchain.
   Sequence of methods in algorithm 6:

TABLE 5.6: Mappings Used in our Implementation

Module 1: *Registration of Vaccine Centers*	
Mapping Name	**Relations**
currentRegistrationAppl	Maps *RegAppl* ← *VC* Address
RegApplBelongsTo	Maps *VC* Address ← *regApplID*
vcAddrTovcID	Maps *vcID* ← *VC* Address
vcIDToVCDetails	Maps *VC* ← *vcID*
Module 2: *Refilling Vaccine Stock*	
Mapping Name	**Relations**
currentRefillAppl	Maps *ReStockAppl* ← *VC* Address
lockedServiceCharge	Maps Locked Amount ← *VC* Address
vaccineStockDetails	Maps *VaccineStock* ← *stockID*
Module 3: *Obtaining TokenID*	
Mapping Name	**Relations**
currentTokenAppl	Maps *TokenAppl* ← *citizenInfoDigest*
tokenAppl	Maps *TokenAppl* ← *tokenApplID*
citizenAddrTocitizenInfoDigest	Maps *citizenInfoDigest* ← *Citizen* Address
citizenInfoDigestToTokenID	Maps *tokenID* ← *citizenInfoDigest*
tokenIDToCitizenDetails	Maps *Citizen* ← *tokenID*
Module 4: *Injecting Vaccine*	
Mapping Name	**Relations**
currentInjectingProtocol	Maps *InjectingProtocol* ← *C* Address
informationAboutVP	Maps *VP* ← *tokenID*
vialState	Maps {*"Used"*, *"Reserved"*, *"Unused"*} ← *vial ID commitment*
Module 5: *Generating and Storing Vaccine Passport*	
Mapping Name	**Relations**
currentVPAppl	Maps *VPAppl* ← *C* Address
Module 6: *Verifying Vaccine Passport*	
Mapping Name	**Relations**
verificationProtocolDetails	Maps *VerificationProtocol* ← *vfProtocolID*
accessControl	Maps *Boolean (true/false)* ← *tokenID* × *vfAddr*

- *applForTokenID* → *C* applies for a *TokenID*. The function parameter *citizenInfoDigest* represents the commitment of personal data (mentioned as $commit_m$ in Figure 5.5).
- *verifyAppl* → *Govt* decides whether to accept or reject the application based on the provided information.

4. **Algorithm for Injecting Vaccine to Citizen:** The Algorithm 7 outlines the secure and transparent process through which a *VC* administers a vaccine dose to a *C*. The details of the protocol have been discussed in *Subsection 5.2.5 Module 4*.
   Sequence of methods in algorithm 7:
   - *beginProtocol* → *C* begins the protocol mentioning its desired $vcID$.
   - *lockMoneyByVC* → VC locks security money.
   - *lockMoneyByC* → *C* locks security money.
   - *commitMTProof* → VC commits MT_Proof.
   - *provideConsent1* → *C* provides its first consent.
   - *commitVialID* → VC commits the vial ID (*vID*) to ensure traceability.

Algorithm 4 Algorithm for VC Registration

```
Function timestampRegAppl()                                        ▷ Caller: VC
    Fetch: Current RegAppl of VC
    Check: If VC not yet registered
    Check: If $RegAppl.underReview == false$
    $RegAppl \leftarrow$ new $RegAppl$
    $RegAppl.T_{regAppl} \leftarrow block.timestamp$
    $RegAppl.underReview \leftarrow$ true
    Update: Mapping entries of currentRegistrationAppl
    Store: RegAppl
end

Function regApplHash(vcAddr, hashAppl)                             ▷ Caller: Govt
    Fetch: Current RegAppl of VC having address vcAddr
    Check: If VC not yet registered
    Check: If $RegAppl.underReview == true$
    Check: If $RegAppl.T_{regAppl} \neq 0$
    Check: If $(block.timestamp - RegAppl.T_{regAppl}) \leq$ timeout
    $RegAppl.T_{hashAppl} \leftarrow block.timestamp$
    $RegAppl.hash \leftarrow hashAppl$
    Update: RegAppl
end

Function decideOnAcceptanceHash(decision)                          ▷ Caller: VC
    Fetch: Current RegAppl of VC
    Check: If VC not yet registered
    Check: If $RegAppl.underReview == true$
    Check: If $RegAppl.T_{hashAppl} \neq 0$
    Check: If $(block.timestamp - RegAppl.T_{hashAppl}) \leq$ timeout
    if $decision == true$ then
        Generate: a unique regApplID
        $RegAppl.regApplID \leftarrow regApplID$
        Update: Mapping entries of regApplBelongsTo
    else
        $RegAppl.underReview \leftarrow$ false
    $RegAppl.T_{decideOnHash} \leftarrow block.timestamp$
    Update: RegAppl
end

Function decideOnAcceptanceRegAppl(regApplID, decision)            ▷ Caller: Govt
    Check: If regApplID valid
    $vcAddr \leftarrow regApplBelongsTo[regApplID]$
    Fetch: Current RegAppl of VC having address vcAddr
    Check: If VC not yet registered
    Check: If $RegAppl.underReview == true$
    Check: If $RegAppl.T_{decideOnHash} \neq 0$
    Check: If $(block.timestamp - RegAppl.T_{decideOnHash}) \leq$ timeout
    if $decision == true$ then
        Generate: a unique vcID
        Assign the vcID to the VC
        $VC \leftarrow$ new $VC$
        $VC.vcID \leftarrow vcID$; $VC.vialsInStock \leftarrow 0$
        Update: VC
        Update: Mapping entries of vcAddrTovcID & vcIDToVCDetails
    $RegAppl.decision \leftarrow decision$
    $RegAppl.T_{decideOnAppl} \leftarrow block.timestamp$
    $RegAppl.underReview \leftarrow$ false
    Update: RegAppl
end
```

- *provideConsent2* $\rightarrow$ C provides its second consent.
- *provideConsent3* $\rightarrow$ C provides its third consent, verifying if the given vID satisfies the membership proof.
- *registerVaxTimestamp* $\rightarrow$ VC registers the timestamp of vaccination on the BC.

Algorithm 5 Algorithm for Refilling Vaccine Stock

```
Function refillStockAppl()                                          ▷ Caller: $VC$
    Check: If $VC$ is registered
    Check: If $VC.vialsInStock == 0$
    Fetch: Current $ReStockAppl$ of $VC$
    Check: If $ReStockAppl.underProcess ==$ false
    $ReStockAppl \leftarrow$ new $ReStockAppl$
    Generate: a unique $refillApplID$
    $ReStockAppl.refillApplID \leftarrow refillApplID$
    $ReStockAppl.T_{refillAppl} \leftarrow block.timestamp$
    $ReStockAppl.underProcess \leftarrow$ true
    Update: $ReStockAppl$
    Update: Mapping entries of $currentRefillAppl$
end

Function commitVaccineSet($vialsCount, MR, vcAddr$)                 ▷ Caller: $Govt$
    Check: If $VC$ having address $vcAddr$ is registered
    Check: If $vialsCount > 0$
    Fetch: Current $ReStockAppl$ of $VC$.
    Check: If $ReStockAppl.underProcess ==$ true
    Check: $ReStockAppl.T_{refillAppl} \neq 0$
    Check: If $(block.timestamp - ReStockAppl.T_{refillAppl}) \leq$ timeout
    Check: If correct amount (as $serviceCharge$ of $VC$) is locked
    Update: Mapping entries of $lockedServiceCharge$
    $ReStockAppl.vialsCount \leftarrow vialsCount$
    $ReStockAppl.commitment \leftarrow MR$
    $ReStockAppl.T_{commitment} \leftarrow block.timestamp$
    Update: $ReStockAppl$
end

Function decideOnAcceptanceVaccineSet($decision$)                   ▷ Caller: $VC$
    Check: If $VC$ is registered.
    Fetch: Current $ReStockAppl$ of $VC$
    Check: If $ReStockAppl.underProcess ==$ true
    Check: If $ReStockAppl.T_{commitment} \neq 0$
    Check: If $(block.timestamp - ReStockAppl.T_{commitment}) \leq$ timeout
    if $decision == true$ then
        Generate: a unique $stockID$
        Instantiate new $VaccineStock$ and populate the members
        $VC.currentStockID \leftarrow stockID$
        Update: $VC$
        Update: Mapping entries of $vaccineStockDetails$
    else
        Transfer: locked money to $Govt$
        Update: Mapping entries of $lockedServiceCharge$
    $ReStockAppl.T_{acceptVaccineSet} \leftarrow block.timestamp$
    $ReStockAppl.underProcess \leftarrow$ false
    Update: $ReStockAppl$
end

Function takeAwayLockedMoney($vcAddr$)                              ▷ Caller: $Govt$
    Check: If $VC$ having address $vcAddr$ is registered
    Fetch: Current $ReStockAppl$ of $VC$.
    Check: If $ReStockAppl.underProcess ==$ true
    Check: If $ReStockAppl.T_{commitment} \neq 0$
    Check: If $ReStockAppl.T_{acceptVaccineSet} == 0$
    Check: If $(block.timestamp - ReStockAppl.T_{commitment}) >$ timeout
    $ReStockAppl.underProcess \leftarrow$ false
    Transfer: locked money to $Govt$
    Update: Mapping entries of $lockedServiceCharge$
end
```

– *acknowledgeVaccination* $\rightarrow C$ acknowledges the vaccination, completing the process.

5. **Algorithm for Generating and Storing Vaccine Passport of Citizen:**

Algorithm 6 Algorithm for Obtaining Citizen Token

```
Function applForTokenID(citizenInfoDigest)                                  ▷ Caller: C
    Check: C with given citizenInfoDigest not yet received tokenID
    Fetch: Current TokenAppl of C
    Check: If TokenAppl.underReview == false
    TokenAppl ← new TokenAppl
    Generate: a unique tokenApplID
    TokenAppl.tokenApplID ← tokenApplID
    TokenAppl.citizenInfoDigest ← citizenInfoDigest
    TokenAppl.underReview ← true
    $TokenAppl.T_{tokenAppl}$ ← block.timestamp
    Update: TokenAppl
    Update: Mapping entries of currentTokenAppl, tokenAppl & citizenAddrTocitizenInfoDigest
end

Function verifyAppl(tokenApplID, decision)                                  ▷ Caller: Govt
    Fetch: TokenAppl corresponding to tokenApplID
    Check: If TokenAppl.underReview == true
    Check: If $TokenAppl.T_{tokenAppl} \neq 0$
    Check: If $TokenAppl.T_{result} == 0$
    Check: If $(block.timestamp - TokenAppl.T_{tokenAppl}) \leq$ timeout
    Check: If C with TokenAppl.citizenInfoDigest not yet received tokenID
    if decision == true then
        Generate: a unique tokenID
        Citizen ← new Citizen
        Citizen.citizenInfoDigest ← TokenAppl.citizenInfoDigest
        Citizen.tokenID ← tokenID
        Citizen.vaccinationStatus ← false
        Citizen.cID ← NULL
        Update: Citizen
        Update: Mapping entries of citizenInfoDigestToTokenID & tokenIDToCitizenDetails
    TokenAppl.result ← decision
    $TokenAppl.T_{result}$ ← block.timestamp
    TokenAppl.underReview ← false
    Update: TokenAppl
end
```

After successfully receiving the vaccine, the *C* must apply for the *VP* to the *Govt*. The vaccine passport includes information about the vaccine name, target disease, timestamp of vaccination, and other relevant details. Due to the large size of the file, it is not stored directly on the *BC*. Instead, *Govt* uploads the encrypted vaccine passport (*VP*) to IPFS and then stores the essential security information on the *BC* invoking *SC* functions. The Algorithm 8 corresponding to the protocol *Subsection 5.2.5 Module 5* depicts the entire process of how the *Govt* issues and stores *VP* of a citizen who received the vaccine.

Sequence of methods in algorithm 8:

– *initiateVPApplAndLockMoney* → *C* initiates the *VP* application by locking a certain amount on *SC*.

– *lockMoneyByGovt* → *Govt* also locks the same amount on *SC*.

– *sendVaccinationProof* → *C* submits vaccination proof, specifies vial ID - *vID* and commits *MT_proof* on-chain.

– *sendConsent1* → Upon offline verification of the Merkle tree proof against the on-chain commitment, the *Govt* issues its initial consent.

– *sendConsent2* → *Govt* provides its second consent if the *vID* satisfies the given membership proof.

Algorithm 7 Algorithm for Injecting Vaccine

```
Function beginProtocol($vcID$)                                    ▷ Caller: $C$
    Check: If $vcID$ is valid
    Check: If $C$ has a valid $tokenID$ and not yet vaccinated
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == false
    $InjectingProtocol \leftarrow$ new $InjectingProtocol$
    Generate: a unique $protocolID$
    $InjectingProtocol.protocolID \leftarrow protocolID$
    $InjectingProtocol.underProcess \leftarrow$ true
    $InjectingProtocol.tokenID \leftarrow tokenID$
    $InjectingProtocol.vcID \leftarrow vcID$
    $InjectingProtocol.T_{protocolBegins} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
    Update: Mapping entries of $currentInjectingProtocol$
end

Function lockMoneyByVC($cAddr$)                                   ▷ Caller: $VC$
    Check: If $VC$ has a valid $vcID$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is not vaccinated
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{protocolBegins} \neq 0$
    Check: If $InjectingProtocol.T_{lockMoneyByVC} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{protocolBegins}) \leq$ timeout
    Check: If correct amount is locked
    $InjectingProtocol.T_{lockMoneyByVC} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end

Function lockMoneyByC($vcID$)                                     ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is not vaccinated
    Check: If $vcID$ is valid
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{lockMoneyByVC} \neq 0$
    Check: If $InjectingProtocol.T_{lockMoneyByC} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{lockMoneyByVC}) \leq$ timeout
    Check: If correct amount is locked
    $InjectingProtocol.T_{lockMoneyByC} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end

Function commitMTProof($cAddr$, $commit_{MT\_Proof}$)              ▷ Caller: $VC$
    Check: If $VC$ has a valid $vcID$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is not vaccinated
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{lockMoneyByC} \neq 0$
    Check: If $InjectingProtocol.T_{commit_{MT\_Proof}} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{lockMoneyByC}) \leq$ timeout
    $InjectingProtocol.commit_{MT\_Proof} \leftarrow commit_{MT\_Proof}$
    $InjectingProtocol.T_{commit_{MT\_Proof}} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end
```

– *uploadVPInfo* → Finally, *Govt* uploads the encrypted *VP* on IPFS and uploads essential security parameters onchain by calling this function. At the same time, *SC* releases the locked amount to both *Govt* and *C*.

Algorithm 7 Algorithm for Injecting Vaccine (Contd.)

```
Function provideConsent1($vcID$, $consent1$)                                    ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is not vaccinated
    Check: If $vcID$ is valid
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{commit_{MT\_Proof}} \neq 0$
    Check: If $InjectingProtocol.T_{consent1} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{commit_{MT\_Proof}}) \leq$ timeout
    if $consent1 == false$ then
        Transfer Money: $C$ and $VC$ get back their locking amount
        $T_{moneyReceivedByC} = T_{moneyReceivedByVC} \leftarrow block.timestamp$
        $InjectingProtocol.underProcess \leftarrow$ false
    $InjectingProtocol.consent1 \leftarrow consent1$
    $InjectingProtocol.T_{consent1} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end

Function commitVialID($cAddr$, $commit_{vID}$)                                  ▷ Caller: $VC$
    Check: If $VC$ has a valid $vcID$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is not vaccinated
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{consent1} \neq 0$
    Check: If $InjectingProtocol.T_{commit_{vID}} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{consent1}) \leq$ timeout
    Check: If $vialState$ of the vial with $commit_{vID}$ is "Unused"
    $InjectingProtocol.commit_{vID} \leftarrow commit_{vID}$
    $InjectingProtocol.T_{commit_{vID}} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
    Update: Mapping entries of $vialState$ (change vial state to "Reserved")
end

Function provideConsent2($vcID$, $consent2$)                                    ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is not vaccinated
    Check: If $vcID$ is valid
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{commit_{vID}} \neq 0$
    Check: If $InjectingProtocol.T_{consent2} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{commit_{vID}}) \leq$ timeout
    if $consent2 == false$ then
        Transfer Money: $C$ and $VC$ get back their locking amount
        $T_{moneyReceivedByC} = T_{moneyReceivedByVC} \leftarrow block.timestamp$
        $InjectingProtocol.underProcess \leftarrow$ false
    $InjectingProtocol.consent2 \leftarrow consent2$
    $InjectingProtocol.T_{consent2} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end
```

6. **Algorithm for Verifying Vaccine Passport:** This algorithm (corresponding to the protocol described in *Subsection 5.2.5 Module 6*) allows healthcare providers and other authorized parties to verify the authenticity of a citizen's vaccine passport. This helps prevent fraud and ensures that only vaccinated individuals are granted access to certain services.
   Sequence of methods in algorithm 9:

   – *lockMoneyByVF* → *VF* locks money on *SC* specifying the *C*'s address

Algorithm 7 Algorithm for Injecting Vaccine (Contd.)

```
Function provideConsent3($vcID$, $consent3$)                                ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is not vaccinated
    Check: If $vcID$ is valid
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{consent2} \neq 0$
    Check: If $InjectingProtocol.T_{consent3} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{consent2}) \leq$ timeout
    $InjectingProtocol.consent3 \leftarrow consent3$
    $InjectingProtocol.T_{consent3} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end

Function registerVaxTimestamp($cAddr$)                                      ▷ Caller: $VC$
    Check: If $VC$ has a valid $vcID$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is not vaccinated
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{consent3} \neq 0$
    Check: If $InjectingProtocol.consent3$ == true
    Check: If $InjectingProtocol.T_{vaccination} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{consent3}) \leq$ timeout
    $InjectingProtocol.T_{vaccination} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end

Function acknowledgeVaccination($vcID$, $ack$)                              ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is not vaccinated
    Check: If $vcID$ is valid
    Fetch: Current $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.underProcess$ == true
    Check: If $InjectingProtocol.vcID == vcID$
    Check: If $InjectingProtocol.tokenID == tokenID$
    Check: If $InjectingProtocol.T_{vaccination} \neq 0$
    Check: If $InjectingProtocol.T_{acknowledgement} == 0$
    Check: If $(block.timestamp - InjectingProtocol.T_{vaccination}) \leq$ timeout
    if $ack == true$ then
        Transfer Money: $VC$ gets its locking amount and also the $serviceCharge$
        $T_{moneyReceivedByVC} \leftarrow block.timestamp$
        $C.vaccinationStatus \leftarrow$ true
        $VC.vialsInStock \leftarrow VC.vialsInStock - 1$
        $VC.moneyEarned \leftarrow VC.moneyEarned + serviceCharge$
        $InjectingProtocol.underProcess \leftarrow$ false
        Update: $VC$, $C$
        Update: Mapping entries of $vialState$ (change vial state to "Used")
    $InjectingProtocol.acknowledgment \leftarrow ack$
    $InjectingProtocol.T_{acknowledgement} \leftarrow block.timestamp$
    Update: $InjectingProtocol$
end
```

for which passport verification is sought, and it marks the start of the verification protocol.

– *lockMoneyAndCommitRK* $\rightarrow$ *C* also locks the same amount on *SC* and commits re-encryption key - *RK*.

– *provideConsent* $\rightarrow$ *VF* provides its consent if the offline received *RK* matches with its commitment.

– *grantAccessPermission* $\rightarrow$ *C* provides access permission to *VF* to fetch *cID* for encrypted *VP*.

Algorithm 8 Algorithm for Generating and Storing Vaccine Passport

```
Function initiateVPApplAndLockMoney()                                ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is vaccinated
    Check: If $C$ has not obtained $VP$
    Check: If correct amount is locked
    Fetch: Current $VPAppl$ of $C$
    Check: If $VPAppl.underprocess$ == false
    $VPAppl \leftarrow$ new $VPAppl$
    Generate: a unique $vpApplID$
    $VPAppl.vpApplID \leftarrow vpApplID$
    $VPAppl.applicantTokenID \leftarrow tokenID$
    $VPAppl.T_{lockMoneyByC} \leftarrow block.timestamp$
    Update: $VPAppl$
    Update: Mapping entries of $currentVPAppl$
end

Function lockMoneyByGovt($cAddr$)                                    ▷ Caller: $Govt$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is vaccinated
    Check: If $C$ has not obtained $VP$
    Check: If correct amount is locked
    Fetch: Current $VPAppl$ of $C$
    Check: If $VPAppl.underprocess$ == true
    Check: If $VPAppl.applicantTokenID == tokenID$
    Check: If $VPAppl.T_{lockMoneyByC} \neq 0$
    Check: If $VPAppl.T_{lockMoneyByGovt} == 0$
    Check: If $(block.timestamp - VPAppl.T_{lockMoneyByC}) \leq$ timeout
    $VPAppl.T_{lockMoneyByGovt} \leftarrow block.timestamp$
    Update: $VPAppl$
end

Function sendVaccinationProof($vID$, $commit_{MT\_Proof}$)            ▷ Caller: $C$
    Check: If $C$ has a valid $tokenID$ and is vaccinated
    Check: If $C$ has not obtained $VP$
    Fetch: Current $VPAppl$ of $C$
    Check: If $VPAppl.underprocess$ == true
    Check: If $VPAppl.applicantTokenID == tokenID$
    Check: If $VPAppl.T_{lockMoneyByGovt} \neq 0$
    Check: If $VPAppl.T_{provideVaccinationProof} == 0$
    Check: If $(block.timestamp - VPAppl.T_{lockMoneyByGovt}) \leq$timeout
    Fetch: Latest $InjectingProtocol$ of $C$
    Check: If $InjectingProtocol.commit_{MT\_Proof} == commit_{MT\_Proof}$
    Check: If $InjectingProtocol.commit_{vID} == h(vID)$
    Check: If vial with ID - $vID$ is "Used", employing Map $vialState$
    $VPAppl.T_{provideVaccinationProof} \leftarrow block.timestamp$
    Update: $VPAppl$
end

Function sendConsent1($cAddr$, $consent1$)                           ▷ Caller: $Govt$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is vaccinated
    Check: If $C$ has not obtained $VP$
    Fetch: Current $VPAppl$ of $C$
    Check: If $VPAppl.underprocess$ == true
    Check: If $VPAppl.applicantTokenID == tokenID$
    Check: If $VPAppl.T_{provideVaccinationProof} \neq 0$
    Check: If $VPAppl.T_{consent1} == 0$
    Check: If $(block.timestamp - VPAppl.T_{provideVaccinationProof}) \leq$ timeout
    if $consent1 == false$ then
        Transfer Money: $Govt$ and $C$ get back their locking amount
        $VPAppl.T_{moneyReceivedByGovt} \leftarrow block.timestamp$
        $VPAppl.T_{moneyReceivedByC} \leftarrow block.timestamp$
        $VPAppl.underProcess \leftarrow$ false
    $VPAppl.consent1 \leftarrow consent1$
    $VPAppl.T_{consent1} \leftarrow block.timestamp$
    Update: $VPAppl$
end
```

– *fetchVPInfo* $\rightarrow$ *VF* retrieves *C*'s encrypted *VP* details.

Algorithm 8 Algorithm for Generating and Storing Vaccine Passport (Contd.)

```
Function sendConsent2($cAddr$, $consent2$)                                  ▷ Caller: $Govt$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is vaccinated
    Check: If $C$ has not obtained $VP$
    Fetch: Current $VPAppl$ of $C$
    Check: If $VPAppl.underprocess ==$ true
    Check: If $VPAppl.applicantTokenID == tokenID$
    Check: If $VPAppl.T_{consent1} \neq 0$
    Check: If $VPAppl.T_{consent2} == 0$
    Check: If $(block.timestamp - VPAppl.T_{consent1}) \leq$ timeout
    $VPAppl.consent2 \leftarrow consent2$
    $VPAppl.T_{consent2} \leftarrow block.timestamp$
    Update: $VPAppl$
end

Function uploadVPInfoAndGetPayment($cAddr$, $MD_{VP}$, $sign$, $cID$)      ▷ Caller: $Govt$
    Check: If $C$ with $cAddr$ has a valid $tokenID$ and is vaccinated
    Check: If $C$ has not obtained $VP$
    Fetch: Current $VPAppl$ of $C$
    Check: If $VPAppl.underprocess ==$ true
    Check: If $VPAppl.applicantTokenID == tokenID$
    Check: If $VPAppl.T_{consent2} \neq 0$
    Check: If $VPAppl.T_{issueVP} == 0$
    Check: If $(block.timestamp - VPAppl.T_{consent2}) \leq$ timeout
    Check: If $VPAppl.consent2 ==$ true
    Transfer Money: $Govt$ and $C$ get back their locking amount
    Transfer Money: $C$ also gets back it's locking amount for $InjectingProtocol$
    $VPAppl.T_{moneyReceivedByGovt} \leftarrow block.timestamp$
    $VPAppl.T_{moneyReceivedByC} \leftarrow block.timestamp$
    $VPAppl.underProcess \leftarrow$ false
    Update: $VPAppl$
    $VP \leftarrow$ new $VP$
    $VP.MD_{VP} \leftarrow MD_{VP}$
    $VP.\sigma \leftarrow sign$
    $VP.cID \leftarrow cID$
    Update: $VP$
    Assign: $VP$ to $C$
    Update: Mapping entries of $informationAboutVP$
end
```

– *verificationResult* → *VF* sends the verification result.

5.3 Security Analysis

Subsection 5.2.2 covers the system's security goals. Now, we will elaborate on how each goal is achieved using the blockchain platform and cryptographic techniques.

Blockchain technology incorporates various fundamental cryptographic primitives, such as hash functions and digital signatures. The security of the blockchain relies on the protection of these basic cryptographic elements. We assume that the fundamental cryptographic primitives are secure, which implies that our underlying blockchain platform is also protected. Consequently, the money held on the blockchain is safe, and therefore, the payments performed via the system are likewise secure. Our system employs ECC (Elliptic Curve Cryptography) to secure vaccination data and uses SHA3 hashed identifiers to anonymize personal details. All sensitive information is encrypted before being stored in the Ethereum blockchain or shared with authorized parties.

Algorithm 9 Algorithm for Verification of Vaccine Passport

```
Function lockMoneyByVF($cAddr$)                                    ▷ Caller: $VF$
    Check: If $C$ with $cAddr$ has a valid $tokenID$
    Check: If $C$ is vaccinated and holds a $VP$
    Check: If correct amount is locked
    Fetch: Latest $VerificationProtocol$ between $C$ and $VF$
    Check: If $VerificationProtocol.underExecution ==$ false
    $VerificationProtocol \leftarrow$ new $VerificationProtocol$
    Generate: a unique $vfProtocolID$
    $VerificationProtocol.vfProtocolID \leftarrow vfProtocolID$
    $VerificationProtocol.underExecution \leftarrow$ true
    $VerificationProtocol.tokenID \leftarrow tokenID$
    $VerificationProtocol.vfAddr \leftarrow msg.sender$
    $VerificationProtocol.T_{lockMoneyByVF} \leftarrow block.timestamp$
    Update: $VerificationProtocol$
    Update: Mapping entries of $verificationProtocolDetails$
end

Function lockMoneyAndCommitRK($vfProtocolID, commitRK$)            ▷ Caller: $C$
    Check: If correct amount is locked
    Fetch: Latest $VerificationProtocol$ corresponding to $vfProtocolID$
    Check: If $VerificationProtocol.underExecution ==$ true
    Check: If $VerificationProtocol.tokenID == C$'s $tokenID$
    Check: If $VerificationProtocol.T_{lockMoneyByVF}$ != 0
    Check: If $VerificationProtocol.T_{lockMoneyAndCommitRkByC} == 0$
    Check: If $(block.timestamp - VPAppl.T_{lockMoneyByVF}) \leq$ timeout
    $VerificationProtocol.commitRK \leftarrow commitRK$
    $VerificationProtocol.T_{lockMoneyAndCommitRkByC} \leftarrow block.timestamp$
    Update: $VerificationProtocol$
end

Function provideConsent($vfProtocolID, decision$)                  ▷ Caller: $VF$
    Fetch: Latest $VerificationProtocol$ corresponding to $vfProtocolID$
    Check: If $VerificationProtocol.underExecution ==$ true
    Check: If $VerificationProtocol.vfAddr == msg.sender$
    Check: If $VerificationProtocol.T_{lockMoneyAndCommitRkByC}$ != 0
    Check: If $VerificationProtocol.T_{provideConsent} == 0$
    Check: If $(block.timestamp - VPAppl.T_{lockMoneyAndCommitRkByC}) \leq$ timeout
    $VerificationProtocol.consent \leftarrow decision$
    $VerificationProtocol.T_{provideConsent} \leftarrow block.timestamp$
    if $decision == false$ then
        Transfer Money: $VF$ and $C$ get back their locking amount
        $VerificationProtocol.T_{unlockMoney} \leftarrow block.timestamp$
        $VerificationProtocol.underExecution \leftarrow$ false
    Update: $VerificationProtocol$
end

Function grantAccessPermission($vfProtocolID$)                     ▷ Caller: $C$
    Fetch: Latest $VerificationProtocol$ corresponding to $vfProtocolID$
    Check: If $VerificationProtocol.underExecution ==$ true
    Check: If $VerificationProtocol.tokenID == C$'s $tokenID$
    Check: If $VerificationProtocol.T_{provideConsent}$ != 0
    Check: If $VerificationProtocol.T_{grantPermission} == 0$
    Check: If $(block.timestamp - VPAppl.T_{provideConsent}) \leq$ timeout
    Check: If $VerificationProtocol.consent ==$ true
    $VerificationProtocol.T_{grantAccessByC} \leftarrow block.timestamp$
    Update: $VerificationProtocol$
    Update: Mapping entries of $accessControl$
end
```

With these considerations in mind, let us delve into the details of the security analysis of our system.

1. ***Fairness***

   – *Unforgeability.* The VP containing the vaccination status of an individual is stored in the IPFS by the vaccination center. In order to successfully

Algorithm 9 Algorithm for Verification of Vaccine Passport (Contd.)

```
Function fetchVPInfo($vfProtocolID$)                                    ▷ Caller: $VF$
    Fetch: Latest $VerificationProtocol$ corresponding to $vfProtocolID$
    Check: If $VerificationProtocol.underExecution == \text{true}$
    Check: If $VerificationProtocol.vfAddr == msg.sender$
    Check: If $VerificationProtocol.T_{grantAccessByC} \mathrel{!}= 0$
    Check: If $VerificationProtocol.T_{fetchVPInfo} == 0$
    Check: If $(block.timestamp - VPAppl.T_{grantAccessByC}) \leq \text{timeout}$
    $VerificationProtocol.T_{fetchVPInfo} \leftarrow block.timestamp$
    Update: $VerificationProtocol$
    Return: $VP$ details of $C$ corresponding to $tokenID$ equals $VerificationProtocol.tokenID$
end

Function verificationResult($vfProtocolID$)                             ▷ Caller: $VF$
    Fetch: Latest $VerificationProtocol$ corresponding to $vfProtocolID$
    Check: If $VerificationProtocol.underExecution == \text{true}$
    Check: If $VerificationProtocol.vfAddr == msg.sender$
    Check: If $VerificationProtocol.T_{fetchVPInfo} \mathrel{!}= 0$
    Check: If $VerificationProtocol.T_{verificationResult} == 0$
    Check: If $(block.timestamp - VPAppl.T_{fetchVPInfo}) \leq \text{timeout}$
    $VerificationProtocol.verificationResult \leftarrow result$
    $VerificationProtocol.T_{verificationResult} \leftarrow block.timestamp$
    Transfer Money: $VF$ and $C$ get back their locking amount
    $VerificationProtocol.T_{unlockMoney} \leftarrow block.timestamp$
    $VerificationProtocol.underExecution \leftarrow \text{false}$
    Update: $VerificationProtocol$
end
```

forge a vaccine passport, an adversary has to use a token-ID of a vaccinated citizen (say citizen C) to pass the verification protocol (say for some verifier VF). However, the usage of a proxy re-encryption mechanism would require the adversary to successfully generate the re-encryption key $RK_{C \rightarrow VF}$. But the adversary has no knowledge of the secret key SK_C of citizen C.
Therefore, the protection against unforgeability is ensured by the following security properties of the proxy re-encryption scheme PRE, which is assumed to be secure

(a) The encryption scheme underlying PRE is CPA- secure. Any adversary $\mathcal{A}$ that is able to compute $RK_{C \rightarrow VF}$ for any VF and without the knowledge of SK_C with high probability can be used to build an adversary $\mathcal{B}$ that breaks the CPA security of the encryption scheme underlying the PRE scheme. This can be illustrated using the following security game as depicted in figure 5.10:
$\mathcal{B}$ wins if $b = b'$. So the winning probability of $\mathcal{B}$ is at least as much as that of $\mathcal{A}$.

(b) PRE is secure against collusion. Several colluding proxies cannot gain any information about the secret key of C by using the re-encryption keys issued to them by C to be able to re-delegate decryption rights on behalf of C. That is, a set of re-encryption keys $RK_{C \rightarrow VF_1}, \ldots, RK_{C \rightarrow VF_k}$ leak no information about SK_C. But the adversary with access to token-ID of C and having seen $RK_{C \rightarrow VF}$ can certainly re-use it to pass verification for VF as no other information about the citizen is demanded at the verification

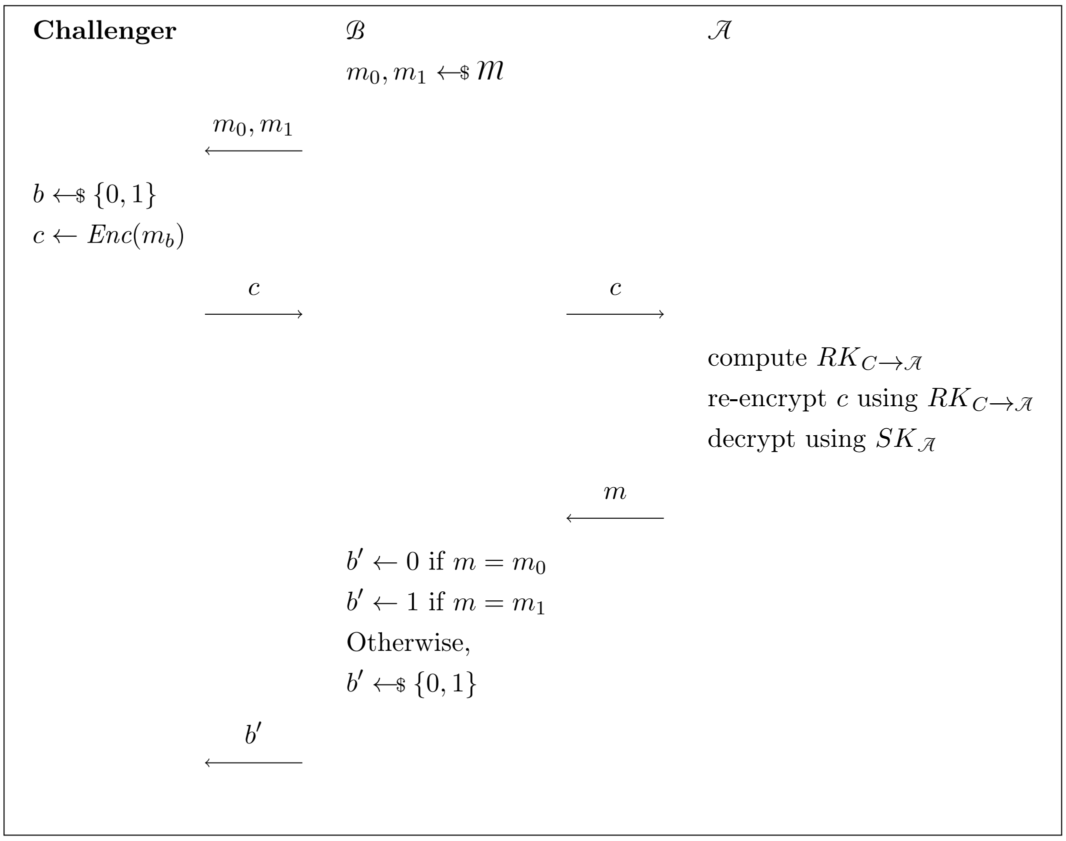

Challenger
$\mathcal{B}$
$\mathcal{A}$
$m_0, m_1 \leftarrow\$ \mathcal{M}$
m_0, m_1
$b \leftarrow\$ \{0,1\}$
$c \leftarrow Enc(m_b)$
c
c
compute $RK_{C \rightarrow \mathcal{A}}$
re-encrypt c using $RK_{C \rightarrow \mathcal{A}}$
decrypt using $SK_{\mathcal{A}}$
m
$b' \leftarrow 0$ if $m = m_0$
$b' \leftarrow 1$ if $m = m_1$
Otherwise,
$b' \leftarrow\$ \{0,1\}$
b'

FIGURE 5.10: Vaccine Passport Forgery Security Game

stage. To prevent this, the re-encryption key can be shared over a secure channel instead of being sent over the public blockchain network.

The system ensures unforgeability through robust cryptographic measures. Proxy re-encryption mechanisms and secure key management protocols are employed to prevent unauthorized access and tampering. Each vaccine passport is associated with a cryptographic signature, making it infeasible to forge without the private key of the issuing authority. Additionally, blockchain immutability ensures that once a record is created, it cannot be altered without detection.

Here are the specific techniques detailed and their implications for ensuring authenticity:

– **Immutable Records**: Blockchain ensures that data cannot be altered once recorded. Cryptographic hashes act as digital fingerprints, instantly detecting any unauthorized modifications.
– **Digital Signatures**: Each vaccine passport is digitally signed by authorized healthcare providers or institutions, verifying the authenticity of the issuer and ensuring data integrity.
– **Smart Contracts**: Verification processes are automated through smart contracts, which enforce predefined rules to validate passport authenticity.

- **Decentralized Storage**: Vaccine data references are distributed across multiple blockchain nodes, preventing a single point of failure or unauthorized modifications.
- **Tamper-Proof Anchoring**: Instead of storing sensitive vaccine data directly on the blockchain, a cryptographic hash of the data is anchored to the chain, ensuring secure off-chain validation.

By combining these measures, the proposed system guarantees the unforgeability of vaccine passports, ensuring their security, authenticity, and resilience against tampering or fraudulent claims.

– *Prevention of misuse.* Illegal distribution and misuse of vaccine doses are prevented by mandatory registration and application procedures, details of which are registered in the blockchain and thus publicly verifiable. Primitives like *Merkle tree* and *collision-resistant hash functions* are used to facilitate validity checking of each dose.

Additionally, locking amounts in the smart contracts at the start of a protocol as and when required further enforces fairness and penalizes malicious behavior.

2. ***Privacy***

– While registering on the blockchain, a citizen does not disclose personally identifiable information (PII) such as name, age, address, phone number, etc. Instead, the citizen sends the hash of their PII to the smart contract during the token generation process. The pre-image resistance property of a cryptographic hash function ensures that personal details are computationally infeasible to retrieve from the hash.

Once the token is generated for a citizen, it is bound to their public key. In subsequent protocols, only this token ID is used, and no other personal information is required, therefore preserving privacy.

Citizens' vaccine passports contain only vaccination administration details and are stored on IPFS in an encrypted format. IPFS provides decentralized storage, while encryption ensures that only the citizen possesses the decryption keys, maintaining data confidentiality. For others to verify a citizen's vaccine passport, they must undergo a proxy re-encryption scheme.

In the verification process, verifiers must obtain explicit permission from the citizen. Without consent, verifiers cannot proceed further. Citizens retain control over access permissions through an access control matrix stored on the blockchain. This matrix enables citizens to manage permissions for third parties, granting or revoking access as needed through interactions with smart contracts.

In summary, these measures ensure robust privacy protection for citizens across all aspects of our blockchain-based vaccine passport system.

3. ***Data Security***

- Altering a citizen's vaccine passport records could have disastrous effects, potentially accelerating the spread of contagious diseases and endangering lives. In our context, citizens might attempt to falsify their vaccine passports to gain benefits reserved for vaccinated individuals without actually receiving the vaccine. Through meticulous system implementation, we ensure that no one, not even the citizen themselves, can forge or tamper with their stored vaccine passport record in our envisioned system and exploit it.

  Our protocol utilizes blockchain as the underlying framework, leveraging its inherent immutability to uphold the integrity of citizens' vaccine passport records. We store various cryptographic computation results, such as hash values, commitment values, and digital signatures, on the blockchain whenever necessary to protect sensitive data.

4. ***Liveness***

   - Liveness is critical in distributed systems because it guarantees that the system remains active and responsive even in the presence of failures, delays, or malicious behavior. A blockchain-based system inherently supports liveness through decentralized consensus, ensuring continuous progress and transaction finality. The fault-tolerant nature of the network enables uninterrupted operation even during node failures or malicious behavior, contributing to implicit liveness.

     We ensure that our proposed blockchain-based system keeps progressing and continues to process and finalize transactions.

     While blockchain-based systems inherently possess liveness as an implicit property, ensuring the smooth and continuous functioning of the underlying network, application-level liveness becomes essential for scenarios where specific actions must be taken within defined time frames. In certain instances, the non-execution of one function can potentially block the execution of subsequent functions, making the system stagnant for an indefinite period of time. This creates a need for timely and autonomous actions to maintain system responsiveness and application-level liveness.

     In our system, we have implemented an additional security measure. Each function must be called within a specific, predetermined time frame. If a function is not called within this time frame, the responsible party will be penalized by having their locked stake in the smart contract deducted and transferred to the other party as compensation. This action will result in the termination of the protocol. We have outlined and documented all potential exit functions to provide these features.

     By integrating these features into the smart contract design, we enhance the application-level liveness of the system, offering a reliable, autonomous, and dynamic environment for all stakeholders involved and ensuring the timely execution of various operations.

5.4 Results and Discussions

Implementation Setup: We successfully implemented the proposed *Vaccine Passport System* on a system running Linux Ubuntu 22.04.2 LTS with a 12 Gen Intel(R) Core(TM) i7-1255U and 16.0 GiB of RAM. We deployed the smart contracts, written in Solidity, on the Ethereum Sepolia test network, utilizing the MetaMask crypto wallet for account creation and transaction initiation. The source code of our smart contracts is available on the GitHub repository*. The deployed contract addresses and deployment gas costs are detailed in Table 5.7.

Table 5.7: Deployment Addresses and Cost of Smart Contracts

Smart Contract	Deployment Address	Deployment Gas Cost
SC_VC_Govt	0x98e8e9c40d7feab2d0b5a373c694e71de4310c6c	1971190
SC_C_Govt_1	0x2af1b8f42985cf9156cd15bb6ba539fe62e3aa25	1435723
SC_C_Govt_2	0x33638204d0712448ca5490c66708091dceb87f4e	3249238
SC_C_VC_1	0x7c63c824f2d50cb492949b6f293e5c2eda2031ac	5104937
SC_C_VC_2	0x9b5bf13df8b35ada1eb073d268cd57b9400f2066	2388173
SC_C_VF	0x98aeb13345fc1250024543d9171dabf7bcd77c25	1847275
SC_Requirements_Check	0x605cfc9c7f146810cf5ed9c91ac51eaf3c051889	666361

As outlined in Table 5.7, two contracts, namely *SC_ C_ VC* and *SC_ C_ Govt*, are split into two parts due to their length. Modules 1 and 2 (i.e., registration of VC and refilling of vaccine stock, respectively) belong to the smart contract *SC_ VC_ Govt*. Module 3 (i.e., obtaining token ID) is part of *SC_ C_ Govt_ 1*, while Module 4 (i.e., injecting vaccine) is managed by *SC_ C_ VC*. Module 5 (generating and storing vaccine passports) belongs to *SC_ C_ Govt_ 2*, and Module 6 (i.e., verifying a vaccine passport) is handled by *SC_ C_ VF*. Additionally, the required check statements are managed using a separate contract named *SC_ Requirements_ Check*.

The two main factors determining any blockchain model's feasibility are the cost of implementation and the time taken. Likewise, we considered these two main factors during the evaluation of our model. A detailed discussion of Ethereum transaction cost and latency can be found in Chapter 2, Section 2.6.10.

Transaction Cost: The average gas price in our case was 3.042282375575E-08 ETH, as determined by the MetaMask wallet. At the time of evaluating the results, the exchange rate was 1 ETH = 3831.140006 USD. The average gas consumption for various transactions related to different modules, including transactions for vaccine registration, refilling vaccine stock, obtaining token IDs, administering vaccines, generating and storing vaccine passports, and verifying vaccine passports, is illustrated in Figures 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, respectively.

The bar charts depicting gas consumption for the transactions related to various modules of the blockchain-enabled vaccine passport system highlight significant differences based on the complexity and security requirements of each

*https://github.com/Debendranath-Das/Blockchain-Enabled-Secure-Vaccine-Passport-System/tree/

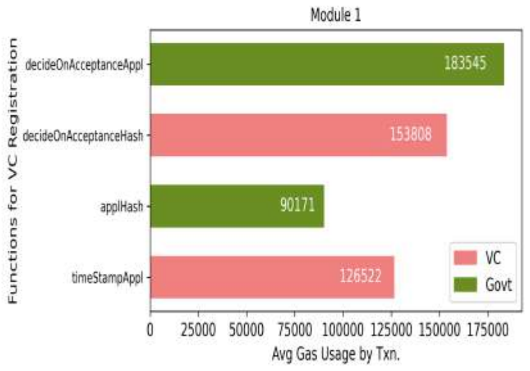


FIGURE 5.11: Gas Consumption of Transactions for VC Registration (Module 1)

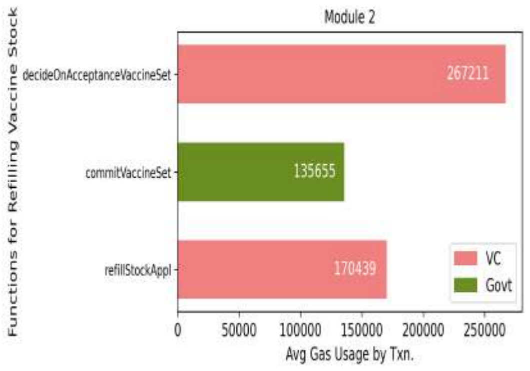


FIGURE 5.12: Gas Consumption of Transactions for Refilling Vaccine Stock (Module 2)

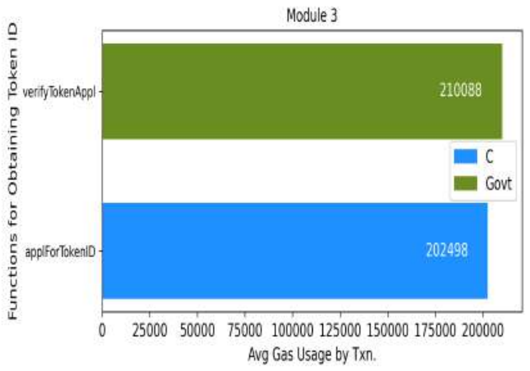


FIGURE 5.13: Gas Consumption of Transactions for Obtaining Token ID (Module 3)

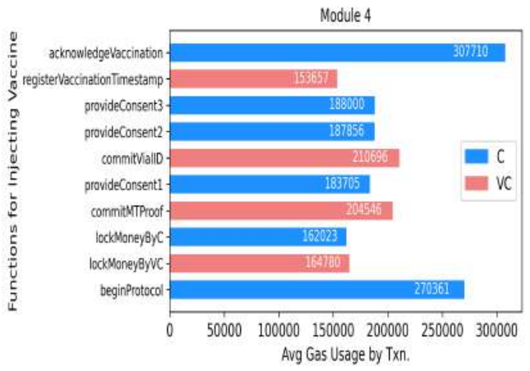


FIGURE 5.14: Gas Consumption of Transactions for Injecting Vaccine (Module 4)

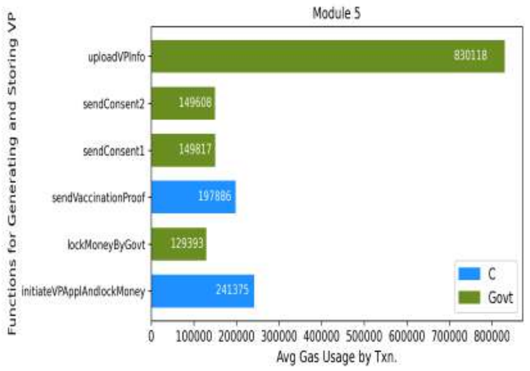


FIGURE 5.15: Gas Consumption of Transactions for Generating and Storing Vaccine Passport (Module 5)

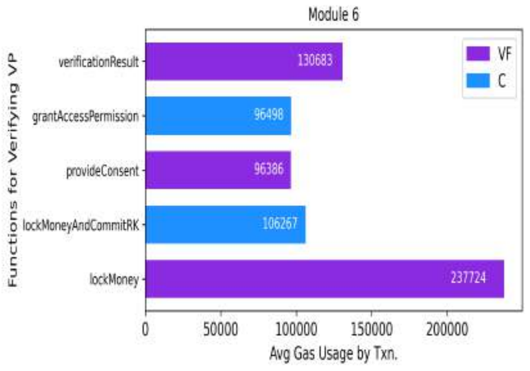


FIGURE 5.16: Gas Consumption of Transactions for Verifying Vaccine Passport (Module 6)

operation. Certain functions incur comparatively higher gas costs due to their more complex functionalities. The *VC Registration* module exhibits moderate

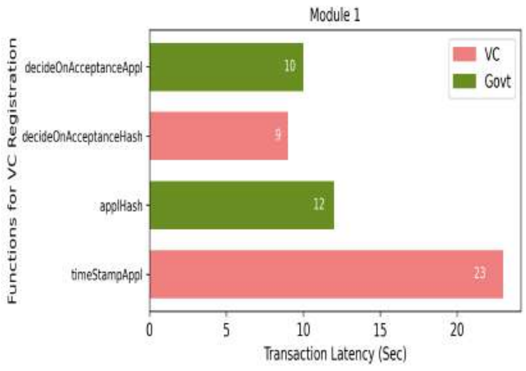


FIGURE 5.17: Transactions Latency for VC Registration (Module 1)

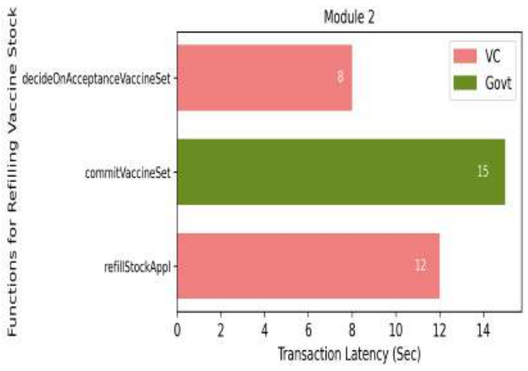


FIGURE 5.18: Transactions Latency for Refilling Vaccine Stock (Module 2)

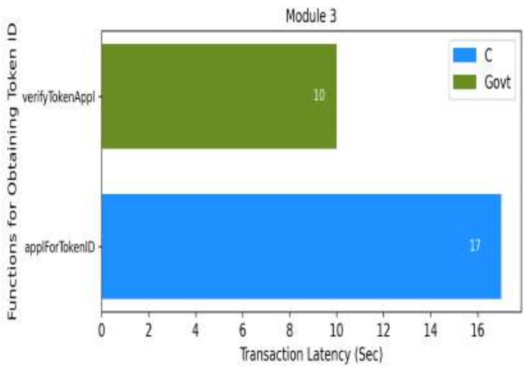


FIGURE 5.19: Transactions Latency for Obtaining Token ID (Module 3)

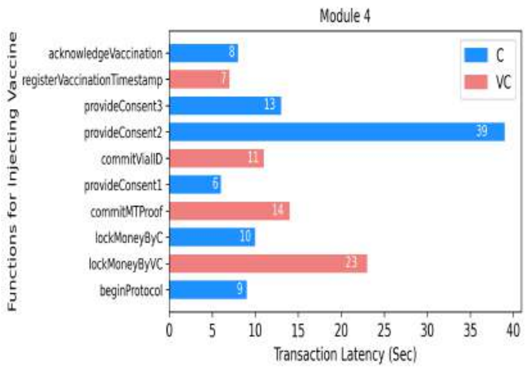


FIGURE 5.20: Transactions Latency for Injecting Vaccine (Module 4)

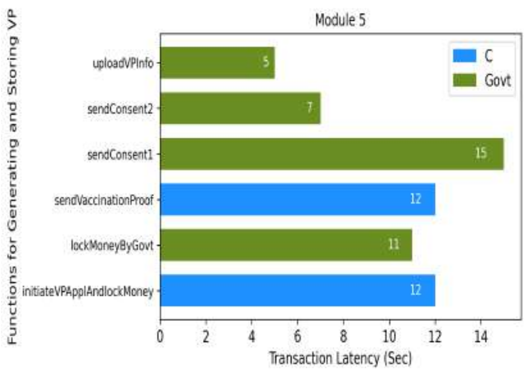


FIGURE 5.21: Transactions Latency for Generating and Storing Vaccine Passport (Module 5)

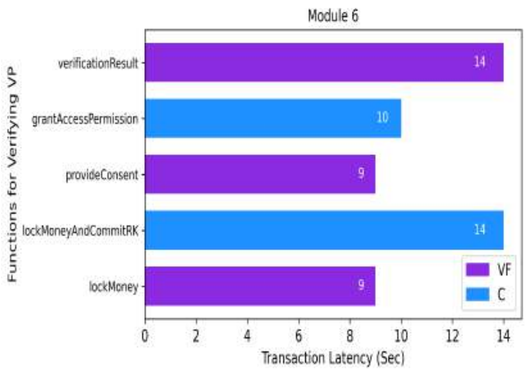


FIGURE 5.22: Transactions Latency for Verifying Vaccine Passport (Module 6)

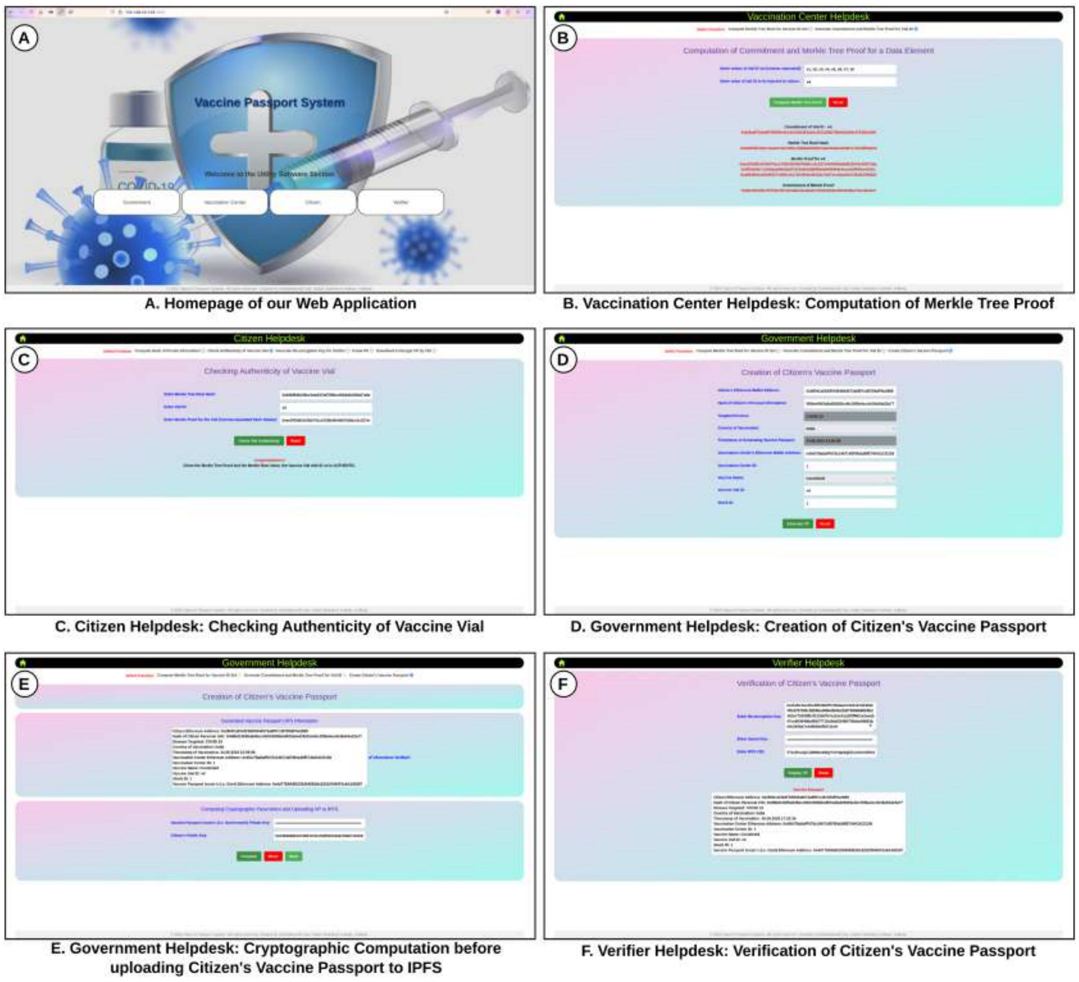
gas consumption, indicating a balanced approach to ensuring secure registration without excessive costs. *Refilling Vaccine Stock*, on the other hand, shows relatively high gas consumption, reflecting the extensive validation processes necessary to maintain accurate stock records and prevent discrepancies. *Obtaining a Token ID* is efficient, with lower gas costs, making it a cost-effective process for users to secure their unique identification for vaccine-related activities. *Injecting Vaccines* demonstrates increased gas usage due to multiple verification steps that ensure the integrity and accuracy of vaccination records. The process of *Generating and Storing Vaccine Passports* consumes significant gas, attributed to the intensive cryptographic operations and data storage requirements. Conversely, *Verifying Vaccine Passports* is optimized for efficiency, with relatively low gas consumption, ensuring quick and cost-effective validation of vaccination status.

Transaction Latency: We have measured the average latency across a substantial number of successful transactions for our system. The transaction latency for different modules is illustrated in Figures 5.17, 5.18, 5.19, 5.20, 5.21, 5.22. These measurements provide insights into the performance and reliability of our system under varying network conditions.

We have developed a web-based application to facilitate various offline cryptographic computations required for individual entities (Government, Citizen, Vaccination Center, and Verifier) in our system. These computations include the generation of Merkle tree root hashes using Keccak-256 (SHA-3), the creation of Merkle tree proofs, the commitment of personal information and vaccine vials, and the implementation of a re-encryption protocol based on proxy re-encryption. The application has been developed using industry-standard web technologies: HTML5, CSS3, JavaScript (ES6+), and Node.js (v20.11.0) for the backend. We've implemented robust input validation and error handling to ensure the security and integrity of cryptographic operations. The user interface is designed to be intuitive, guiding users through complex processes with step-by-step instructions and real-time feedback. For data storage, we've integrated an IPFS daemon hosted locally to store the encrypted vaccine passports. The web server is hosted locally. The entire source code for the application is available on GitHub repository[†] Figure 5.23 depicts several screens of our web application, showcasing its clean interface.

Challenges Faced during Implementation and Mitigation of These Challenges: The implementation of the vaccine passport system faced several challenges, including scalability issues, which were addressed by minimizing on-chain data storage and optimizing contract efficiency; data privacy concerns, mitigated through off-chain storage with on-chain cryptographic hashes; interoperability challenges, resolved using middleware and APIs; fraud prevention, ensured through smart contracts and digital signature validation; high transaction costs, reduced with gas-efficient coding and selective data anchoring; and user accessibility, improved by designing a user-friendly, multilingual interface. These measures collectively ensured a secure, efficient, and practical solution.

[†]https://doi.org/10.5281/zenodo.12533359



A. Homepage of our Web Application

B. Vaccination Center Helpdesk: Computation of Merkle Tree Proof

C. Citizen Helpdesk: Checking Authenticity of Vaccine Vial

D. Government Helpdesk: Creation of Citizen's Vaccine Passport

E. Government Helpdesk: Cryptographic Computation before uploading Citizen's Vaccine Passport to IPFS

F. Verifier Helpdesk: Verification of Citizen's Vaccine Passport

FIGURE 5.23: A few screenshots demonstrating the functionalities of our Web Application

5.5 Conclusion

Our proposed *Blockchain-Enabled Secure Vaccine Passport System* has successfully achieved the initial goals of ensuring secure, transparent, and efficient vaccine administration, creating vaccine passports, and verifying the same. Our system leverages blockchain's immutable ledger and integrates smart contracts to automate various processes, thereby minimizing fraud. While implementing the system, we also ensured that user privacy would not be compromised. Through comprehensive experimental evaluation, we have demonstrated that the system is robust and functions correctly, providing reliable and tamper-proof vaccination records. The use of IPFS for off-chain data storage further enhances security while maintaining accessibility. These results validate our approach and indicate the potential for widespread adoption across various jurisdictions, ultimately contributing to the modernization and security of public health infrastructure.

Moving forward, it is essential to focus on optimizing the scalability and performance of the system to handle increasing adoption and transaction volumes. Ensuring interoperability with existing health information systems and other blockchain networks for seamless integration and data exchange will be crucial. Additionally, the system should be adaptable to comply with diverse regulatory requirements across different regions, enhancing its global applicability. Improving the user experience, particularly for non-technical users, will be essential for widespread acceptance. By addressing these areas, the blockchain-enabled secure vaccine passport system can develop into a comprehensive solution, significantly advancing global public health infrastructure and vaccine management. With such a system in place, we can effectively manage global pandemics like COVID-19 without resorting to nationwide lockdowns and severe economic disruptions.

Having explored the potential benefits of integrating blockchain technology within the healthcare sector and associated industries, we now shift our focus to the financial domain. The subsequent two chapters (Chapter 6 and Chapter 7) will explore the impact of blockchain technology on the banking and finance sectors.

6 Blockchain-Enabled Secure Payment Card Tokenization System

"Social security, bank account, and credit card numbers aren't just data. In the wrong hands, they can wipe out someone's life savings, wreck their credit and cause financial ruin."

— **Melissa Bean**

The unprecedented surge in e-commerce transactions, accelerated by the COVID-19 pandemic, has been facilitated by the widespread adoption of popular e-wallets like Paytm, PhonePe, GPay, and Amazon Pay. These platforms streamline online purchases by allowing users to store and utilize credit or debit card details, including the Primary Account Number (PAN), expiration date, and Card Verification Value (CVV). However, storing sensitive data on merchant websites poses significant risks, as cybercriminals can exploit vulnerabilities to gain unauthorized access, compromising financial security. Despite two-factor authentication, card details alone can often initiate unauthorized transactions, highlighting the persistent risks of traditional storage methods. Figure 6.1 depicts the sensitive information associated with a payment card.

Recognizing these threats, the Reserve Bank of India has mandated that payment aggregators, wallets, and online merchants cease retaining sensitive consumer card data from October 1, 2022, underscoring the urgent need for

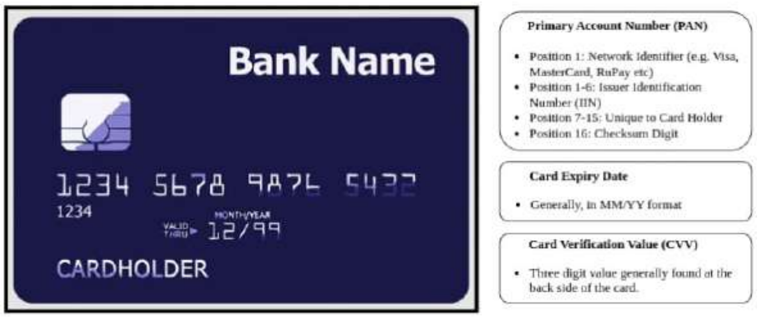

Bank Name
1234 5678 9876 5432
1234
MONTH/YEAR
VALID THRU
12/99
CARDHOLDER
Primary Account Number (PAN)
Position 1: Network Identifier (e.g. Visa, MasterCard, RuPay etc)
Position 1-6: Issuer Identification Number (IIN)
Position 7-15: Unique to Card Holder
Position 16: Checksum Digit
Card Expiry Date
Generally, in MM/YY format
Card Verification Value (CVV)
Three digit value generally found at the back side of the card.

FIGURE 6.1: Sensitive Values of a Card

secure online transaction solutions[128].

In light of these developments, traditional methods of storing and transmitting sensitive card data are proving increasingly susceptible to exploitation, underscoring the pressing need for innovative solutions to mitigate risks and safeguard consumer interests.

Card tokenization emerges as a pivotal mechanism to address these challenges. By substituting sensitive card information with non-sensitive unique tokens, tokenization mitigates the risks associated with storing and transmitting card data, thereby enhancing data security and privacy in digital transactions. Complementing this process, de-tokenization facilitates the reverse mapping of tokens back to their original, sensitive form when required, establishing a comprehensive framework for secure digital transactions.

6.1 Objectives

In the realm of tokenization, Token Service Providers (TSPs) like Visa, Rupay, and MasterCard play a crucial role. As registered entities, they are responsible for token generation, assurance, issuance, and provisioning, facilitating the secure exchange of sensitive card data for tokens. However, the current tokenization systems face two major challenges:

- **Centralized TSP Model:** Reliance on centralized TSPs poses risks of single point of failure, leading to widespread data breaches and vulnerability to targeted attacks.
- **Indirect Access to Tokens:** Cardholders typically interact with intermediaries, known as token requestors (e.g., payment processors or financial institutions), to initiate the tokenization process. This indirect access introduces delays, complexity, and heightened risk of data exposure as sensitive information traverses through the middleman.

Therefore, the objective is to resolve these challenges with the aid of advanced technology.

6.1.1 Contributions

This work introduces a novel blockchain-based decentralized tokenization system to overcome centralized limitations and streamline tokenization for better security and efficiency. Key features include:

- **Decentralized Token Service Provider (TSP) Model:** Token generation and management are distributed across a blockchain network, enhancing security, transparency, and resilience. Decentralizing the tokenization process enhances security by removing single points of failure, making the system resilient to targeted attacks. Unlike centralized systems like Visa's Token Service Providers (TSP), blockchain-based tokenization relies on consensus mechanisms and cryptographic proof, ensuring data integrity and transparency.

- **Smart Contracts as TSPs:** Automated smart contracts serve as TSPs, eliminating intermediaries and promoting decentralization, automation, and transparency.

- **Direct Token Access for Cardholders:** Cardholders interact directly with smart contracts for token generation, reducing bottlenecks and offering greater control over data.

Our system represents a paradigm shift in digital transactions, offering enhanced security, efficiency, and user empowerment through decentralization and smart contract utilization.

6.1.2 Organization

The rest of the chapter is structured as follows - Section 6.2 briefly discusses the current state of the art. In Section 6.3, we discuss the token eco-system in detail. Section 6.4 describes our system model. Section 6.5 shows the results of our proposed system. Finally, we have concluded the chapter in Section 6.6.

6.2 Related Work

The use of blockchain technology has been increasingly explored in various applications, including tokenization systems. The growing reliance on card payments and the necessity for organizations was highlighted to safeguard sensitive credit card data, leading to the emergence of tokenization as a solution to mitigate payment card industry (PCI) compliance burdens [40]. “Indy528” was introduced, which is an innovative platform employing NFT tokens to represent federated machine learning (FML) models, pioneering the representation of machine learning models as NFTs [11]. By leveraging blockchain technology and NFTs, the research aims to enhance transparency and accountability in federated learning, offering a decentralized marketplace for trading and sharing machine learning models with detailed model card information. Emphasis has been given to the mechanisms through which blockchain technology fosters trust in exchanges, contrasting them with traditional exchange models [156]. By conducting interviews with experienced blockchain practitioners, the research sheds light on how mathematics, cryptography, and digital escrows contribute to trust in the exchange, offering insights valuable to marketing scholars and practitioners navigating online trust issues. The urgent need was addressed for resilient identity management systems in the face of data breaches and identity theft, leveraging blockchain technology to introduce self-sovereign identity (SSI) management [151]. Through a focus on the public transportation sector, the research demonstrates the potential of blockchain-based decentralized identity management systems to enhance security and transparency across multiple operators and countries, offering passengers standardized travel credentials and greater control over their identities. The complexity of access control (AC) in

Space Situation Awareness (SSA) systems was addressed, proposing a decentralized authentication mechanism called BlendCAC, inspired by blockchain technology [175]. Through leveraging smart contracts and blockchain for identity authentication and capability-based access control, BlendCAC offers a robust and scalable solution for protecting devices, services, and information within SSA networks, demonstrated through a proof-of-concept implementation on both resource-constrained and more powerful computing devices. The rising trend of online voting and the cautious approach due to potential vulnerabilities were explored, highlighting blockchain technology as a promising solution for secure, decentralized electronic voting systems [77]. By examining the current landscape of blockchain-based voting research, the study aims to address concerns such as privacy protection and transaction speed to ensure the viability and scalability of blockchain-enabled electronic voting systems. The integration of blockchain, non-fungible tokens (NFTs), and model cards within a 5G/6G network slice broker and marketplace was also explored, aiming to enhance network resource management and allocation efficiency [12]. It addresses emerging challenges in network slicing and proposes innovative solutions leveraging blockchain technology and NFTs for improved transparency, security, and scalability in network resource management. Overall, blockchain technology has demonstrated its capabilities in enabling secure, transparent, and efficient systems across various industries. By leveraging distributed ledger technology and tokenization, blockchain-enabled systems have the potential to revolutionize traditional processes and create new opportunities for innovation and growth.

6.3 Token Eco-System

Before delving into the details of the Token Eco-system [127], it is crucial to understand how typical transaction processing takes place through a card prior to Tokenization. The parties involved in a typical card transaction processing system include:

1. **Cardholder:** Transaction Initiator, Card Owner
2. **Merchant/Terminal Machine:** Payment Terminal e.g., Point of Sale or e-Wallet
3. **Acquiring Bank:** Merchant's Bank
4. **Interconnect Network:** Data Transmission Bridge, Network Interface
5. **Card Issuing Bank:** Card Provider Bank, Account Holder Bank
6. **Intermediary Financial Institute:** Middleman Bank, Financial Mediator

6.3.1 Card Transaction Processing Prior to Tokenization

As depicted in Figure 6.2, initially, the cardholder initiates a transaction at a merchant's establishment. They interact with the merchant's payment terminal,

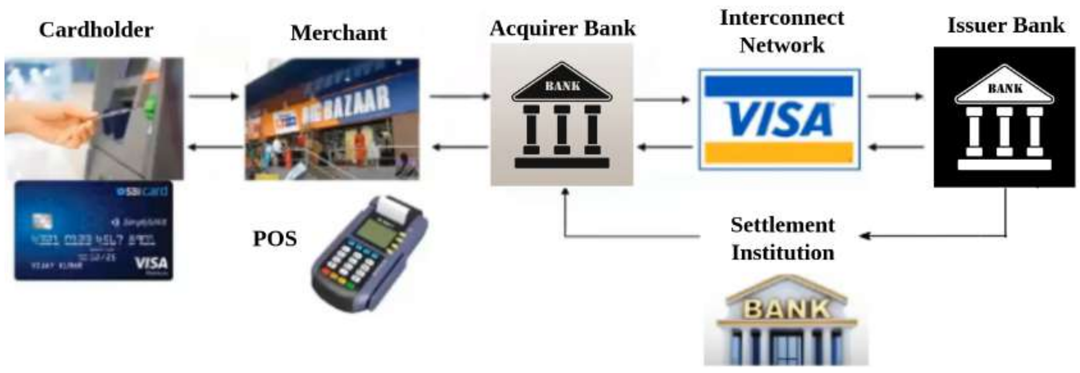


FIGURE 6.2: Card Transaction Processing Prior to Tokenization

such as a Point of Sale (POS) system or an e-wallet. The terminal is linked with the merchant's bank account, referred to here as the Acquiring Bank. If the card and the Acquiring Bank are incompatible, the transaction data is routed through interconnect networks like VISA, Mastercard, or Rupay. These networks identify the issuing bank associated with the card.

Subsequently, the issuing bank verifies the transaction's authenticity and the cardholder's legitimacy. It also checks the cardholder's account balance in the case of debit cards or the available credit limit for credit cards. If all checks pass, the issuing bank approves the transaction, and the message propagates back to the merchant's level, signaling approval on the POS terminal.

The settlement process involves determining the payment flow between banks, facilitated by networks like VISA. Banks that issue a particular type of card must maintain accounts with designated settlement institutions, which are specified by the respective interconnect networks.

If the Acquiring and Issuing Banks are identical, transaction data doesn't need to traverse the entire network, expediting settlement.

6.3.2 Card Transaction Processing in Token Eco-System

Card tokenization maps sensitive card information with surrogate non-sensitive values, known as tokens. Tokens in payment systems have key features:

i) A single card number can generate various token values for different uses.

ii) Tokens can be specific to merchants, channels, or devices.

iii) If hacked, tokens have limited value since they are tied to specific situations like a particular device or merchant.

iv) Cardholders can have multiple tokens for one card number, each with its own usage rules (aka. Token Domain Restriction).

As mentioned earlier, the Token Ecosystem involves additional entities such as **Token Service Providers (TSPs)** and **Token Requestors**, each playing distinct roles in the transaction process.

TSPs are registered with EMVCo (EMVCo, founded by Europay, Mastercard, and Visa, oversees the technical standard for secure card payments across smart payment cards, terminals, and ATMs.), which provides them with a unique 3-digit TSP Code. They are responsible for:

i) Generating, assuring, issuing, and provisioning tokens.

ii) Implementing token domain restriction control, limiting token usage based on specific criteria such as channel, merchant, consumer, device, or initiator.

iii) Facilitating token processing, including de-tokenization.

iv) Integrating with both token requestors and card issuers.

Token Requestors, identified by an 11-digit Token Requestor ID, are registered entities with TSPs. They request tokens from TSPs on behalf of cardholders, such as Google Pay.

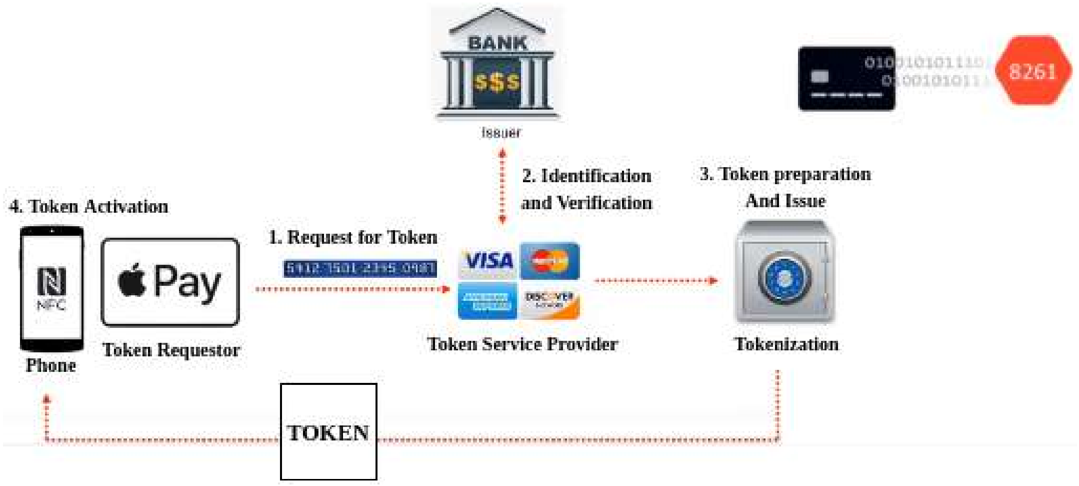


FIGURE 6.3: Token Issuance in Token Eco-System

As shown in Figure 6.3, when a token requestor such as Apple Pay wants to obtain a token on behalf of a cardholder, it sends a request to a TSP. The card values, including the PAN, remain unchanged during the transaction. The request is then sent to the issuing bank for authentication. Once the bank validates the request, the TSP generates and issues a token, which is securely linked to the PAN. This token is then provisioned onto the requesting device, thereby activating it through a process called token provisioning.

After token issuance, the phone transmits the token instead of the PAN during transaction initiation, as depicted in Figure 6.4. The token value transfers to the merchant, then to the TSP for detokenization into the PAN. The issuer bank approves transactions against the PAN, authenticating the cardholder and verifying details. If valid, the issuer approves; otherwise, rejects the transaction. Approved transactions and the PAN return to the vault for re-tokenization. The TSP transfers the token back to the acquirer, reaching the merchant's terminal to display the transaction status. This process ensures security by transmitting only tokens, with a limited section of the system retaining knowledge of the actual PAN values.

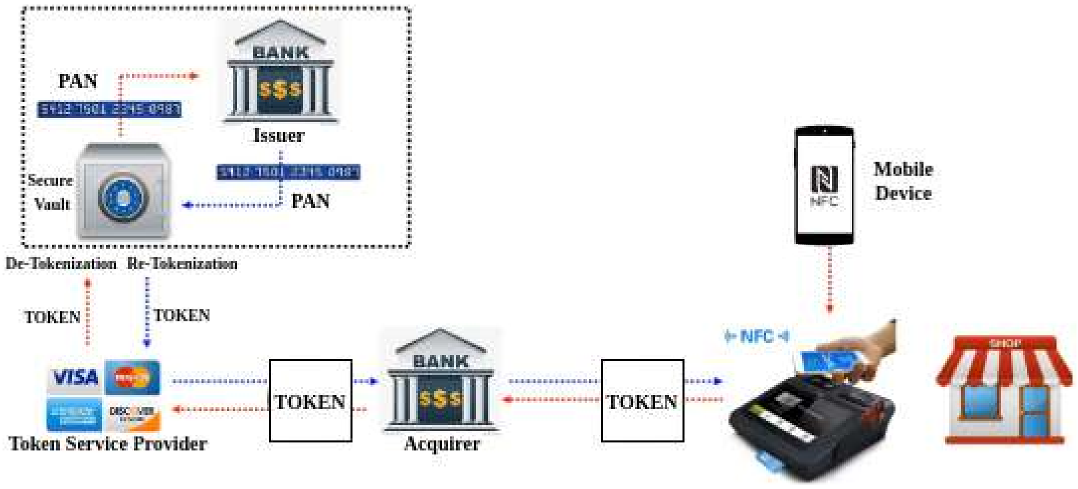


FIGURE 6.4: Transaction Processing through Token in Token Eco-System

6.4 Proposed Model

Remark 4. *The reliance on traditional payment card systems exposes users and financial institutions to significant risks, including data breaches, fraud, and unauthorized transactions. Sensitive card details are often stored in centralized databases, making them attractive targets for cyberattacks. Additionally, existing tokenization methods rely on centralized entities, which introduces vulnerabilities and a single point of failure. The problem is to develop a decentralized, secure, and efficient payment card tokenization system that eliminates the risks of centralized control, ensures transaction integrity, and protects sensitive user information. Leveraging blockchain technology, this problem addresses these challenges by creating a tamper-proof, privacy-preserving framework for payment card tokenization.*

6.4.1 Components

Our system comprises the following key entities:

1. **Customers:** Serve as payers/senders (or payees/receivers) in transactions.
2. **Banks:** Act as issuing banks that generate tokens as well as acquiring banks that redeem tokens.
3. **Regulatory Body:** An authorized governing entity that validates and approves bank registrations, ensuring regulatory compliance.
4. **Smart Contracts:** Handle the entire functionality of the token ecosystem, including entity registration, token issuance, and token execution. Three smart contracts are employed:

   i) SC_Bank_Registration for bank registration, *ii) SC_ Customer_ Registration* for customer registration, and

iii) SC_ Tokenization for tokenization processes.

5. **Blockchain:** An immutable ledger maintained by a peer-to-peer distributed network, securely recording all transaction data.

6.4.2 Assumptions

1. Customers and banks are pre-existing real-world entities that are onboarded to the blockchain ecosystem.
2. All the actors hold an Ethereum wallet and possess an Ethereum address to facilitate communication.
3. Smart contract-generated tokens have inherent domain restrictions, allowing redemption only by the designated payee within a certain period after issuance.

6.4.3 Protocol Design

I. **Bank Registration:** The smart contract (*SC_ Bank_ Registration*) facilitates the protocol for bank registration. The sequential steps, as depicted in Figure 6.5, are as follows:

1. The bank initiates the process by locking a security deposit in the smart contract via on-chain communication over the blockchain network.
2. Similarly, the regulatory body locks a security deposit in the smart contract through an on-chain transaction.
3. The bank sends its proof of banking license (PoBL) off-chain to the regulatory body, while committing the PoBL on-chain to the smart contract.
4. The regulatory body verifies the received PoBL against its committed value, ensuring data integrity and authenticity.
5. The regulatory body communicates the verification result (approval or rejection) to the smart contract via an on-chain transaction.
6. Upon successful verification, the smart contract notifies the bank regarding its registration approval and unlocks both security deposits through on-chain transactions.

II. **Customer Registration:** The customer registration process, facilitated by the smart contract (*SC_ Customer_ Registration*), mirrors bank registration, as depicted in Figure 6.6. The customer deposits security funds in a smart contract alongside the bank. Then, the customer provides off-chain proof of account ownership to the bank, keeping the commitment on-chain. The bank verifies the provided proof against the committed value, ensuring authenticity. Upon successful verification, the bank communicates the result to the smart

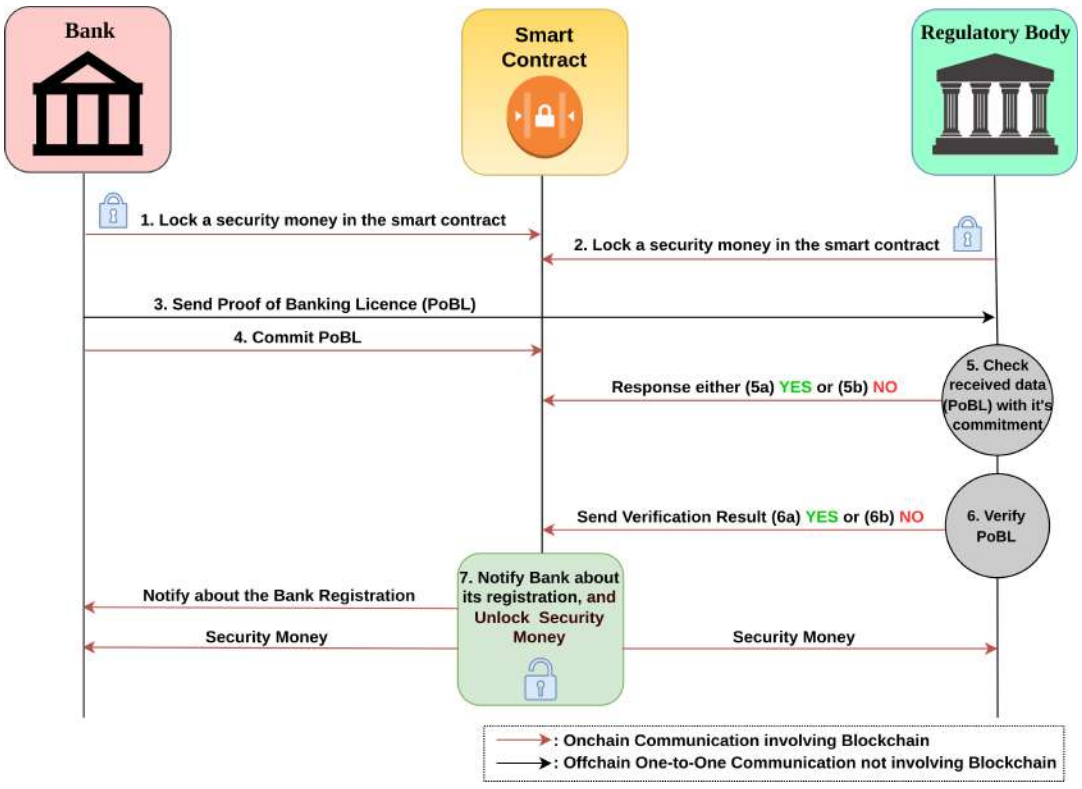

Bank
Smart Contract
Regulatory Body
1. Lock a security money in the smart contract
2. Lock a security money in the smart contract
3. Send Proof of Banking Licence (PoBL)
4. Commit PoBL
Response either (5a) YES or (5b) NO
5. Check received data (PoBL) with it's commitment
Send Verification Result (6a) YES or (6b) NO
6. Verify PoBL
Notify about the Bank Registration
Security Money
7. Notify Bank about its registration, and Unlock Security Money
Security Money
: Onchain Communication involving Blockchain
: Offchain One-to-One Communication not involving Blockchain

FIGURE 6.5: Bank Registration

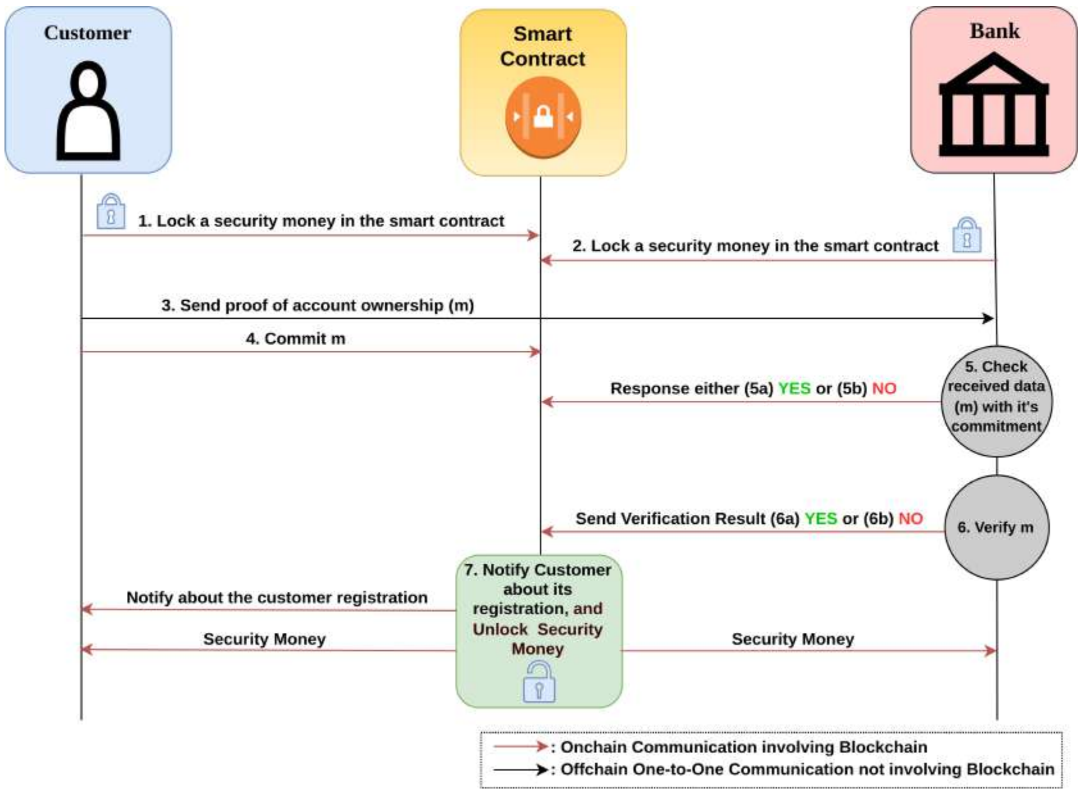

Customer
Smart Contract
Bank
1. Lock a security money in the smart contract
2. Lock a security money in the smart contract
3. Send proof of account ownership (m)
4. Commit m
Response either (5a) YES or (5b) NO
5. Check received data (m) with it's commitment
Send Verification Result (6a) YES or (6b) NO
6. Verify m
Notify about the customer registration
Security Money
7. Notify Customer about its registration, and Unlock Security Money
Security Money
: Onchain Communication involving Blockchain
: Offchain One-to-One Communication not involving Blockchain

FIGURE 6.6: Customer Registration

contract through an on-chain transaction. The smart contract, in turn, notifies the customer of their approved registration and unlocks both security deposits.

Once customers and banks register, they become part of our distributed token ecosystem. Only registered entities can participate in the card tokenization process.

III. **Token Issuance:**

1. The process begins with the customer locking a security deposit in the smart contract, followed by the bank.
2. Subsequently, the customer provides their card details to the bank off-chain and commits this information on-chain to the smart contract.
3. The bank conducts a series of verifications: firstly, it checks if the received card information matches the committed data and responds accordingly. If successful, it proceeds to verify the card's validity, including the account number, CVV, and expiry date against its off-chain database. Additionally, it verifies if the card is currently active and sends a third response.
4. Upon receiving positive responses from the bank, the customer requests the issuance of an on-chain token with the specified receiver's address and amount. However, if any of the verifications fail, the protocol aborts, and the security deposits are unlocked.
5. The bank confirms if the customer's balance is adequate to cover the requested token amount and notifies the contract accordingly.
6. Upon receiving a positive notification from the bank, the smart contract issues a $tokenID$ to the customer on-chain. Each token is uniquely identified and includes a timestamp for traceability. Finally, upon successful token issuance, the smart contract unlocks both security deposits.

IV. **Token Execution at Sender/Payer Side:**

1. The process begins with the payer locking a security deposit in the smart contract, followed by the issuer bank.
2. The payer then requests token execution from the issuer bank, specifying the token ID on-chain. If the request is illegitimate (e.g., payer mentions invalid *token ID*), the protocol aborts, and deposits are unlocked.
3. Upon successful token execution, the issuer bank debits the token amount from the payer's account, updates the balance off-chain, and sends a confirmation message on-chain. The smart contract notifies the payee to collect the token amount and unlocks deposits through on-chain transactions.

IV. **Token Execution at Receiver/Payee Side:** After the payer completes the token execution process, the designated payee or receiver must collaborate with the acquirer bank to redeem the token amount within a specific time frame following the payer's token execution. This process is similar to the token execution process conducted by the payer.

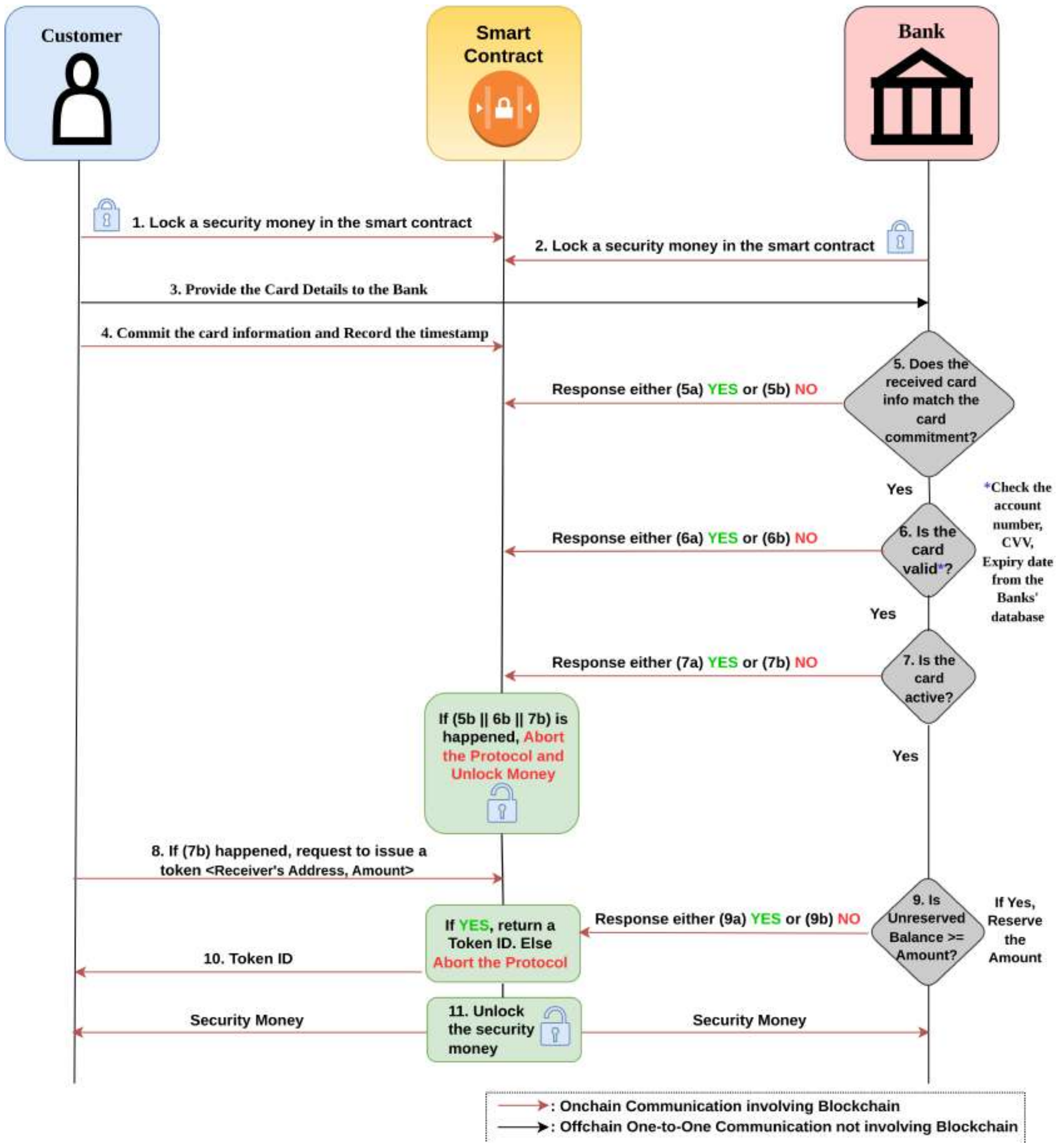

Customer
Smart Contract
Bank
1. Lock a security money in the smart contract
2. Lock a security money in the smart contract
3. Provide the Card Details to the Bank
4. Commit the card information and Record the timestamp
Response either (5a) YES or (5b) NO
5. Does the received card info match the card commitment?
Yes
Response either (6a) YES or (6b) NO
6. Is the card valid*?
*Check the account number, CVV, Expiry date from the Banks' database
Yes
Response either (7a) YES or (7b) NO
7. Is the card active?
If (5b || 6b || 7b) is happened, Abort the Protocol and Unlock Money
Yes
8. If (7b) happened, request to issue a token <Receiver's Address, Amount>
If YES, return a Token ID. Else Abort the Protocol
Response either (9a) YES or (9b) NO
9. Is Unreserved Balance >= Amount?
If Yes, Reserve the Amount
10. Token ID
Security Money
11. Unlock the security money
Security Money
: Onchain Communication involving Blockchain
: Offchain One-to-One Communication not involving Blockchain

FIGURE 6.7: Token Issuance

Figures 6.7, 6.8, and 6.9 illustrate the processes of token issuance, token execution at the payer side, and token execution at the payee side, respectively. These processes are governed by the smart contract ($SC\_Tokenization$).

The smart contracts encode protocols as a series of functions. To prevent stagnation, each subsequent function invocation must occur within a predefined time limit. Failure to comply designates the corresponding party as malicious, leading to a penalty deduction from their locked security deposit. This time-bound execution ensures protocol progression and discourages undesirable behavior from any involved entity. Throughout the process, user privacy is safeguarded by encrypting sensitive transaction data before token generation or execution operations. Hashing techniques anonymize user identities, ensuring no personally identifiable information is stored on-chain, thereby adhering to privacy standards like GDPR.

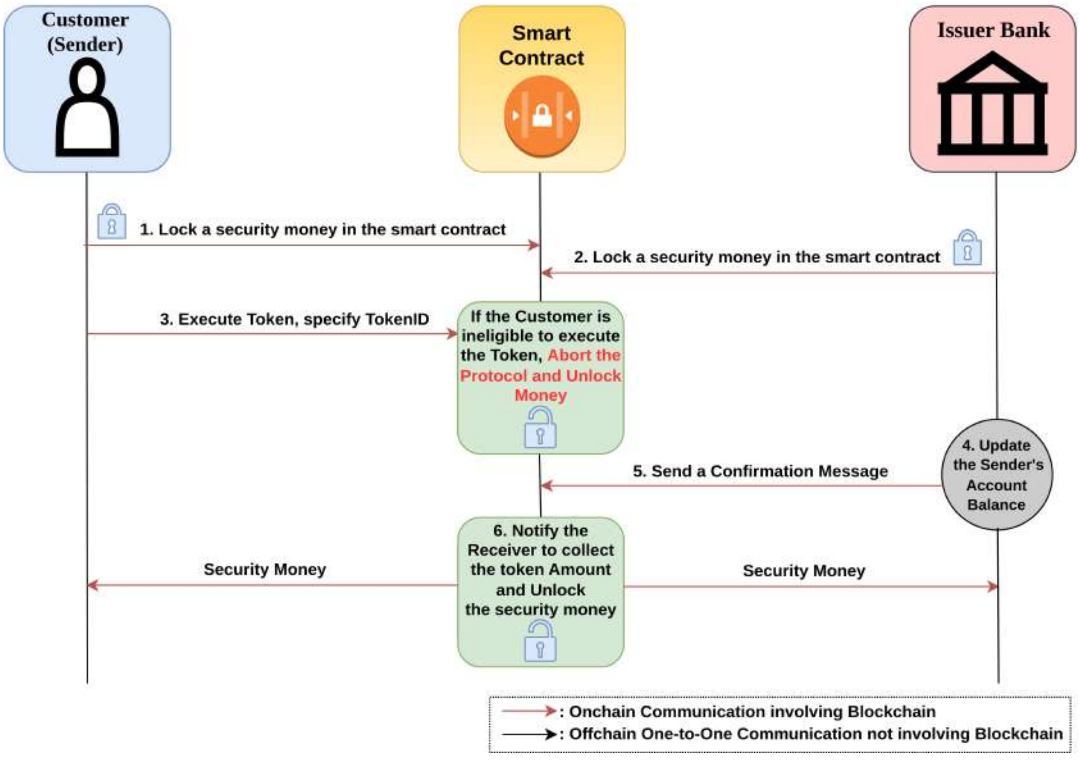

Customer (Sender)
Smart Contract
Issuer Bank
1. Lock a security money in the smart contract
2. Lock a security money in the smart contract
3. Execute Token, specify TokenID
If the Customer is ineligible to execute the Token, Abort the Protocol and Unlock Money
4. Update the Sender's Account Balance
5. Send a Confirmation Message
6. Notify the Receiver to collect the token Amount and Unlock the security money
Security Money
Security Money
: Onchain Communication involving Blockchain
: Offchain One-to-One Communication not involving Blockchain

FIGURE 6.8: Token Execution at Payer (or Sender) Side

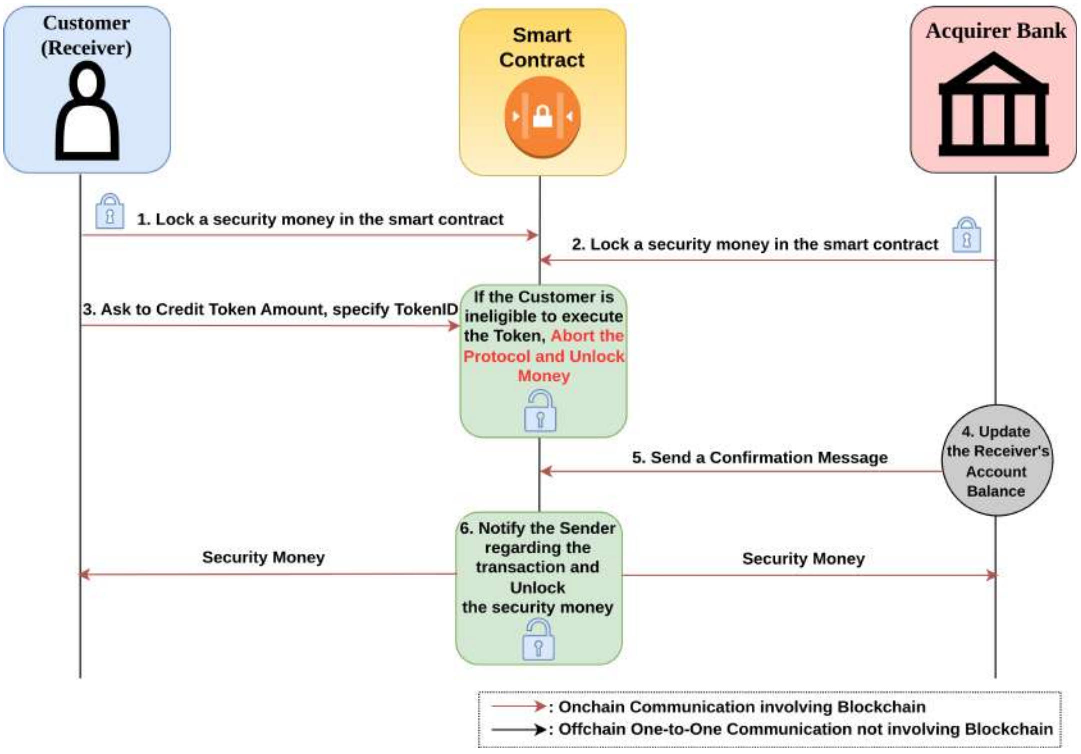

Customer (Receiver)
Smart Contract
Acquirer Bank
1. Lock a security money in the smart contract
2. Lock a security money in the smart contract
3. Ask to Credit Token Amount, specify TokenID
If the Customer is ineligible to execute the Token, Abort the Protocol and Unlock Money
4. Update the Receiver's Account Balance
5. Send a Confirmation Message
6. Notify the Sender regarding the transaction and Unlock the security money
Security Money
Security Money
: Onchain Communication involving Blockchain
: Offchain One-to-One Communication not involving Blockchain

FIGURE 6.9: Token Execution at Payee (or Receiver) Side

6.4.4 Implementation Details

As outlined in the preceding section, the system is supported by three Solidity smart contracts:

1. **SC_Bank_Registration:** This smart contract manages the registration of new banks, aiding the onboarding process of the legacy banking systems onto the blockchain platform. It consists of the following functions:

    i. *lockMoneyByBank*: A bank calls this function to initiate the registration process by locking a specified amount of money in the smart contract.

    ii. *lockMoneyByRegualatoryBody*: After the bank locks the money, the regulatory body also locks an equal amount as a security deposit to the smart contract by invoking the function to ensure fairness.

    iii. *commitPoBL*: Following the regulatory body's action, the bank sends proof of its banking license offline and calls this function to commit this information on the blockchain. This step involves the bank submitting a cryptographic hash of the proof of banking license, ensuring that the provided data is securely recorded and can be verified later.

    iv. *checkIfCommitmentMatches*: The regulatory body verifies the bank's committed proof of banking license with this function, ensuring the commitment value matches the received information.

    v. *sendVerificationResult*: The regulatory body invokes this function to record the final verification result on the blockchain, which confirms whether the bank has passed the registration verification process. At this time the security amount is unlocked and transferred to the respective parties.

2. **SC_Customer_Registration:** Similar to bank registration, the customers also need to register for the utilization of blockchain-provided benefits. This smart contract handles the customer registration process through the following functions, mirroring the functions of the bank registration:

    i. *lockMoneyByCustomer*: The customer invokes this function to begin the registration process, where they lock a designated amount of money in the smart contract as a security deposit.

    ii. *lockMoneyByBank*: After the customer's deposit, the bank also locks the security deposit to the smart contract by calling this function.

    iii. *commitAccountInfo*: Following the bank's security lock, the customer provides their account information offline and calls this function to record this information on the blockchain, utilizing a cryptographic hash to ensure data integrity.

    iv. *checkIfCommitmentMatches*: The bank employs this function to confirm that the customer's committed account information corresponds with the data received offline, verifying the accuracy of the submission.

    v. *sendVerificationResult*: Finally, the bank invokes this function to log the verification results on the blockchain. During this process, the locked security amount is unlocked and transferred to the respective parties, completing the registration and confirming the customer's eligibility for blockchain benefits.

3. **SC_Tokenization:** This smart contract governs the process of token issuance and execution of the token involving the payer and payee. We can break this contract into 3 sub-modules - 1) Process of Token Issuance, 2) Process of Token Execution by Payer and 3) Process of Token Execution by Payee.

   (a) **Token Issuance**: Token issuance involves the creation and issuance of tokens to the requester. It consists of the following functions:

   i. *lockSecurityMoneyByPayer*: This function allows the payer to initiate the application for the token issuance process by locking their money in the smart contract.

   ii. *lockSecurityMoneyByBank*: Following the payer's action, the function allows the bank to lock its money in the smart contract to ensure fairness and mutual commitment to the process.

   iii. *commitCardDetails*: After both parties have locked their money, this function enables the payer to submit a commitment of their credit/debit card details on-chain while sharing the actual card information with the bank offline in a secure one-to-one communication.

   iv. *checkIfCommitmentMatches*: The function allows the bank to verify that the card details received offline match the on-chain commitment provided by the payer, ensuring data consistency.

   v. *checkIfCardValid*: This function enables the bank to check the validity of the payer's card details and provide a response indicating whether the card information is valid.

   vi. *checkIfCardActive*: The function allows the bank to verify further if the payer's card is currently active and provide a response accordingly.

   vii. *requestToken*: Suppose the bank's response indicates that the card is active. In that case, this function enables the payer to request the issuance of a token by specifying the payee's Ethereum address and the token amount.

   viii. *checkBalanceAndIssueToken*: The function allows the bank to check the payer's unreserved account balance against the requested token amount. If the balance is sufficient, the bank issues a token with a designated token ID to the requester. Also, the smart contract releases the locked funds to both parties at this time.

   (b) **Token Execution by Payer**: Once the token has been issued successfully, the payer must execute the token within a specific time window; otherwise, its validity will expire. This process consists of the following functions:

   i. *initiateTokenExecutionByPayer*: This function allows the payer to initiate the token execution process by locking their money in the smart contract. It verifies the validity of the token ID and ensures that the token execution process with this token ID is not already underway.

ii. *lockSecurityMoneyByIssuerBank*: This function allows the issuer bank to lock its money in the smart contract to ensure fairness.

iii. *askToExecuteTokenByPayer*: Next, the payer invokes this function to request the token execution.

iv. *sendConfirmationMessageByIssuerBank* This function allows the issuer bank to send a confirmation message about the token execution status. At this time, the issuer bank debits the payer's account by the token amount. The smart contract also releases the locked money to the respective parties.

(c) **Token Execution by Payee**: once the payer executes the token and the amount has been debited from the payer's account, the payee must initiate the token's execution within a certain time limit to credit the balance. Failure to do so will result in the expiration of the token. This process consists of the following functions:

i. *initiateTokenExecutionByPayee*: This function enables the intended recipient or payee to initiate the token execution process by locking their funds in the smart contract. The function verifies the validity of the token ID and ensures that the recipient has not already initiated the execution process for this token ID.

ii. *lockSecurityMoneyByAcquirerBank*: This function enables the acquirer bank to lock its funds in the smart contract, ensuring fairness and mutual commitment.

iii. *askToExecuteTokenByPayee*: In this step, the payee calls this function to request the execution of the token.

iv. *sendConfirmationMessageByAcquirerBank*: This function permits the acquirer bank to send a confirmation regarding the token execution status. At this point, the acquiring bank deposits the token amount into the payee's account, and the smart contract releases the security funds to the relevant parties.

if the payee fails to execute the token within the stipulated time, the payer needs to invoke an additional function in the smart contract to request a refund of the token amount.

The timestamp of each function invocation is securely recorded on the blockchain. To prevent unnecessary stagnation, each subsequent function must be invoked within the stipulated time frame. At the beginning of each protocol, parties must lock a specific amount as a security deposit in the smart contract. This precautionary measure enables the system to penalize any party found at fault by deducting their locked funds.

The source code is available on GitHub repository[*].

[*] https://doi.org/10.5281/zenodo.12580470

6.4.5 Security and Privacy Implications of Decentralizing the Tokenization Process

Security Implications: Decentralizing the tokenization process using blockchain introduces significant security advantages over centralized systems:

- **Elimination of Single Points of Failure:** Centralized systems rely on a single entity to manage tokenized data, making them vulnerable to cyberattacks. Decentralization distributes data and processes across the blockchain network, significantly reducing the risk of catastrophic breaches.
- **Data Immutability:** Blockchain's immutable ledger ensures that tokenized data cannot be altered once recorded, safeguarding against tampering or unauthorized modifications.
- **Transparency and Verifiability:** Each transaction is permanently recorded on the blockchain, creating an auditable trail that enhances accountability and trust.
- **Resilience:** The distributed nature of blockchain ensures high availability and fault tolerance, even if some nodes are compromised or offline.

Privacy Implications:

- Sensitive data is not stored directly on-chain; instead, cryptographic hashes or references are recorded on the blockchain to ensure privacy.
- Tokenization replaces sensitive data with unique, meaningless tokens, making the original data inaccessible without authorization.
- Smart contracts and access control mechanisms enforce strict authorization policies, ensuring that only authorized entities can access or use the tokenized data.
- Encryption secures data during transmission, protecting it from interception or unauthorized access.
- Blockchain's immutability ensures that recorded data cannot be altered or tampered with, maintaining integrity.
- Digital signatures validate the authenticity of transactions and entities involved, making the tokenization process tamper-resistant and secure.

Considerations and Challenges: While decentralization enhances security, it introduces specific challenges that require careful attention:

- **Smart Contract Vulnerabilities:** Improperly implemented smart contracts can be exploited, leading to unauthorized access or tokenization failures. Rigorous testing and formal verification are crucial to mitigate these risks.

- **Scalability Constraints:** Decentralized networks can struggle with high-frequency transactions, resulting in latency and increased costs. Efficient network optimizations are essential to address these limitations.

- **Consensus Mechanism Risks:** Delays or reordering of transactions due to blockchain consensus mechanisms can impact real-time tokenization processes, requiring protocols that minimize such risks.

Decentralized tokenization systems, despite these challenges, provide a robust and reliable alternative to centralized systems by addressing long-standing vulnerabilities and enhancing overall security and privacy.

6.5 Results

Implementation Setup: We successfully implemented the proposed *Card Tokenization System* on a system running Linux Ubuntu 22.04.2 LTS with an Intel(R) Core(TM) i5-8250U and 8.0 GiB of RAM. We deployed the smart contracts, written in Solidity, on the Ethereum Sepolia test network, utilizing the MetaMask crypto wallet for account creation and transaction initiation. The deployed contract addresses and deployment gas costs are detailed in Table 6.1.

TABLE 6.1: Deployment Addresses and Cost of Smart Contracts

Smart Contract	Deployment Address	Deployment Gas Cost
SC_Bank_Registration	0xCD56Ae845370f70742089880f731ee71152c3EEb	2267900
SC_Customer_Registration	0xce9C06B7634e14571ee99234A6b6Bf21b01907C1	2721206
SC_Tokenization	0x33F1176F893d434a454689BFB77ece8A464bF23D	4830243

Our main goal was to determine the feasibility of our blockchain model through various results. The cost and time were the two main factors we considered. Chapter 2, Section 2.6.10 contains a comprehensive discussion on Ethereum transaction costs and latency.

Transaction Cost: Figure 6.10, 6.11 illustrate the gas costs of various transactions related to entity registration and tokenization processes, respectively.

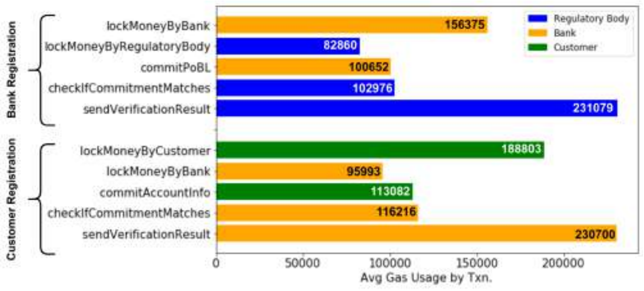


FIGURE 6.10: Gas Consumption of Transactions for Entity Registration

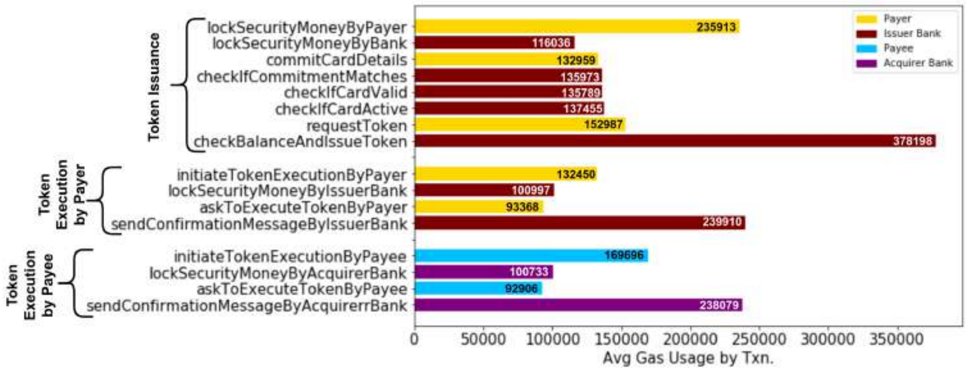


FIGURE 6.11: Gas Consumption of Transactions for Card Tokenization

Transaction Latency: We measured the average latency over a significant number of successful transactions for our system, depicted in Figure 6.12, 6.13.

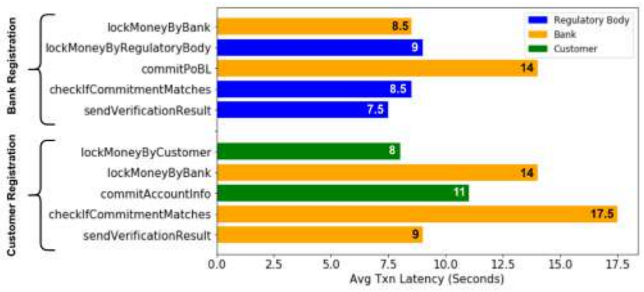


FIGURE 6.12: Transactions Latency for Entity Registration

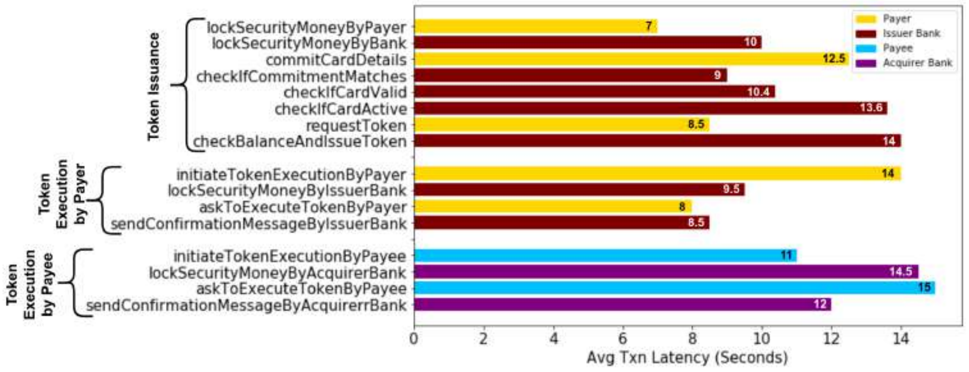


FIGURE 6.13: Transactions Latency for Card Tokenization

Gas Cost Optimization: One of the key challenges in a blockchain-enabled tokenization system is minimizing gas costs during the tokenization process, especially for high-frequency transactions. To address this, the following optimizations were implemented:

- **Off-Chain Computations:** Computationally intensive processes, such as token generation and validation, are handled off-chain, with only essential cryptographic references and transaction metadata stored on-chain. This approach reduces the on-chain computational burden and gas costs.
- **Efficient Smart Contract Design:** Smart contracts were optimized to use minimal storage and computational resources by employing efficient data structures and avoiding unnecessary operations, thereby lowering the gas consumed per transaction.
- **Selective On-Chain Storage:** Instead of storing full token details, cryptographic hashes or references are stored on-chain. This significantly reduces storage costs, which form a major component of gas fees.

To further enhance cost efficiency and scalability, the following techniques are planned for future integration:

- **Batching Transactions:** Consolidating multiple transactions into a single batch for processing reduces the number of on-chain operations, effectively lowering the overall gas expenditure per transaction.
- **Layer-2 Solutions:** Integrating layer-2 scaling solutions, such as rollups or sidechains, offers a viable strategy to further reduce gas costs and improve scalability for high-frequency transactions.

These optimizations ensure that the proposed tokenization system remains both cost-effective and scalable, even under high transaction volumes.

6.6 Conclusion

The proposed solution addresses the critical limitations of the current tokenization system by introducing decentralization facilitated by smart contracts acting as autonomous TSPs. Cardholders do not need to rely on token requestors, enabling them to apply directly to obtain a token. This innovative approach offers greater resilience and security, reducing dependency on centralized entities and enhancing the overall efficiency of the token ecosystem. Experimental results demonstrate the effectiveness of this solution, highlighting its potential to significantly improve transaction processing and security in card tokenization systems.

The proposed system can integrate with existing payment networks by leveraging APIs provided by Visa and Mastercard, requiring only minor compatibility adjustments. Future research and development efforts should focus on:

i) Challenges such as standardizing token formats and ensuring compatibility with existing infrastructure must be addressed. Future work could explore cross-network token interoperability to facilitate seamless adoption.

ii) Optimizing smart contract integration as tokenized security providers for seamless interoperability.

iii) Enhancing scalability and throughput to handle increasing transaction volumes.

iv) Identifying and mitigating security vulnerabilities in decentralized tokenization systems. Collaborative initiatives across stakeholders are key to driving adoption and standardization.

7 Blockchain-Enabled Multicurrency Supported Distributed e-Banking System

“Banking is necessary, banks are not.”

— **Bill Gates**

The emergence of blockchain technology has brought about a paradigm shift in the financial landscape, presenting new opportunities for innovation and disruption. One of the most promising applications of blockchain technology is in the banking and finance sector [93], giving rise to Decentralized Finance (aka DeFi).

Traditional banking systems operate on a centralized model, where banks and financial institutions serve as trusted intermediaries to manage transactions, maintain records, and ensure security. While effective in many scenarios, this model can face challenges, including higher operational costs, inefficiencies in certain processes, limited transparency, and potential vulnerabilities due to centralized control. Moreover, the most significant disadvantage is the susceptibility to fraudulent activities [162].

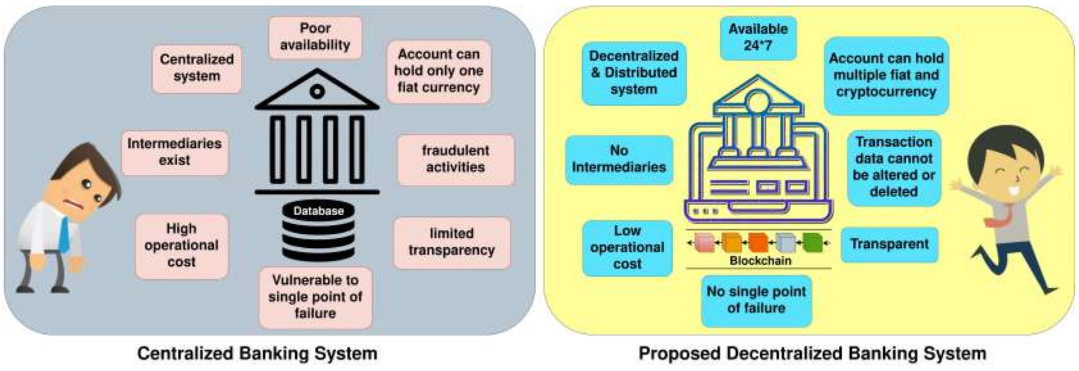

Centralized system
Poor availability
Account can hold only one fiat currency
Intermediaries exist
fraudulent activities
Database
High operational cost
limited transparency
Vulnerable to single point of failure
Centralized Banking System
Decentralized & Distributed system
Available 24*7
Account can hold multiple fiat and cryptocurrency
No Intermediaries
Transaction data cannot be altered or deleted
Low operational cost
Blockchain
Transparent
No single point of failure
Proposed Decentralized Banking System

FIGURE 7.1: Centralized Banking System Vs Decentralized Banking System

In contrast, as portrayed in Figure 7.1, a blockchain-enabled decentralized banking system addresses the challenges of traditional banking by removing the intermediaries and creating a trustless, transparent, and secure environment for financial transactions [142]. At its core, blockchain is a distributed and immutable ledger, ensuring the integrity of financial records as once data is recorded, it cannot be altered or deleted. In such a system, transactions are verified and added to the blockchain through a distributed consensus mechanism involving a network of nodes. This eliminates the need for a central authority and enables faster and more efficient cross-border payments while reducing transaction fees [44].

Additionally, with the advent of Ethereum [25], blockchain technology enables the creation of programmable smart contracts, self-executing agreements with predefined conditions. Without intermediaries, these smart contracts can automate various banking processes, such as loan approvals, asset transfers, and interest payments. This not only streamlines operations but also reduces the risk of human error and increases the overall efficiency of the banking system.

However, like any transformative technology, blockchain-enabled decentralized banking systems also face challenges, including scalability, regulatory compliance, and user adoption [33]. The adoption of cryptocurrencies [49] in countries has been controversial, with public key addresses being the only identifiable information. To create a robust and sustainable ecosystem, overcoming these hurdles requires collaboration between technology innovators, financial institutions, and regulatory bodies. In light of these intricate dynamics, current research is actively delving into the question of whether a decentralized banking system based on blockchain technology can cater to users who prefer traditional fiat currencies like the INR or Dollar instead of cryptocurrencies. Such a feature is essential for a gradual and adaptive transition from conventional fiat currency systems to cryptocurrency systems. Significantly, the current landscape lacks an existing system that comprehensively supports the simultaneous holding of both fiat currencies and cryptocurrencies, making this a key topic of ongoing investigation and exploration. This unique dual capability has the potential to bridge the gap between the familiarity of conventional currencies and the novelty of cryptocurrencies in the context of shifting financial paradigms.

To illustrate this notion, let us consider an example of a hypothetical distributed marketplace with sellers and buyers. Some sellers accept payments in fiat currencies such as the Indian Rupee, the United States Dollar, or the Euro, while others prefer cryptocurrencies. This scenario poses challenges for buyers, as they must pay in the currency chosen by the seller. If a buyer only has a cryptocurrency wallet, they can only transact with vendors who accept cryptocurrencies, not those who accept fiat currencies. Likewise, buyers with fiat currency can only buy from vendors who accept fiat currency. However, this restriction can be overcome if a wallet or bank account allows holding both fiat currency and cryptocurrency. In such a scenario, users hold a bank account with multiple currencies (e.g. Ether, INR, etc.) and can transact using any agreed-upon currency between the buyer and seller. As of our current knowledge, no such system exists in practice.

7.0.1 Contributions

We have proposed an innovative idea of having a bank account capable of simultaneously holding both fiat currencies and cryptocurrencies. These accounts can support having multiple fiat currencies, e.g. INR, Dollar, and Euro. We have created a prototype for the model on the Ethereum platform and launched it as a web application with a user-friendly graphical interface. We highlight our system's distinctive features:

1. **Simultaneous Fiat-Crypto Holding:** This model innovates by enabling a bank account to handle both fiat currencies and cryptocurrencies concurrently, distinguishing it from traditional banking structures.

2. **Multi-Fiat Compatibility:** Beyond individual fiat support, the model allows various fiat currencies like INR, Dollar, and Euro to coexist within one account, showcasing adaptability.

3. **Implementation on Ethereum Prototype:** A working prototype of the proposed banking system has been created using smart contracts on the Ethereum public blockchain platform.

4. **Decentralized with Zero Downtime:** Built on the blockchain, the model's decentralized, distributed nature ensures continuous operation without downtime, unlike traditional centralized banking systems.

5. **User-Friendly Web Application:** The prototype is accessible via a user-friendly web application with a graphical interface, making it more approachable for non-tech-savvy users.

6. **Comprehensive Spectrum of Banking Functionality:** It encompasses account opening, deposits, withdrawals, fund transfers, real-time currency exchanges as per the conversion rate, and account closure, offering a holistic financial solution uniting fiat and crypto in one dynamic framework.

By amalgamating these unique features, the proposed model introduces a novel solution to the challenge of integrating conventional fiat currencies and cryptocurrencies within a single financial framework. This model has the potential to bridge the gap between the two currency paradigms and offer users a more seamless and versatile financial experience.

7.0.2 Organization

The rest of the chapter is structured as follows - We have discussed the state-of-the-art in Section 7.1. Then, Section 7.2 provides a quick overview of the basic building blocks. Section 7.3 presents the system architecture and implementation details. Next, we have addressed our security assertions in Section 7.4. After that, Section 7.5 shows the results of our proposed system and also discusses the outcome. Finally, we have concluded the chapter in Section 7.6.

7.1 Related Work

In this section, we provide an overview of relevant research and studies that explore the applications of blockchain technology within the financial industry.

Eyal et al. [49] conducted a comprehensive investigation into the potential implications of blockchain technology within the financial realm and its relevance to overseeing the banking sector. The primary objective of their study was to uncover the value that blockchain could bring to the field of finance and account management. They delved into historical data analysis and highlighted the role of Bitcoin in addressing persistent challenges within the financial sector.

Popova et al. [120] focused on the application of blockchain technology in safeguarding financial transaction information without the use of tokens. Their study thoroughly examined the security aspects of decentralized databases and introduced a novel solution to the challenge of maintaining data integrity within these databases through tokenless blockchain technology.

Cocco et al. [33] delved into the challenges and prospects associated with the adoption of blockchain technology within the banking sector. Their research sheds light on the unique challenges and opportunities that blockchain presents in the context of traditional banking operations.

Wu et al. [171] provided insights into how blockchain technology benefits commercial banks. Their study focused on three key areas: transaction fees, cross-border payments, and asset securitization conducted by commercial banks. Wu et al. emphasized the potential for blockchain to reduce transaction costs and enhance operational efficiency in the commercial banking sector.

Dozier et al. [44] scrutinized blockchain innovation as a groundbreaking development within the financial management sector. Through an evaluation involving 12 financial management providers, the authors observed that financial service organizations often perceive blockchain advancement as a lower priority due to the need for a clear value proposition. They conclude that as industries confront new technologies and advancements, such as blockchain, it becomes imperative to adopt efficient methodologies for assessment, determining whether they can reap benefits from the innovation's adoption or not.

Li et al. [93] demonstrated that blockchain technology has the potential to disrupt existing business models and to examine the potential mechanisms through which this disruption might occur. In conclusion, the paper recommends that individuals closely monitor developments in this field to prepare for potential disruptions in their businesses.

Arantes et al. [9] proposed leveraging blockchain technology to streamline the processes of lending, monitoring, and evaluating development projects within the Brazilian Development Bank. The proposal aims to streamline public fund allocations, enhance manual tasks, lower operational costs, and provide data to support comprehensive analyses of the bank's loans.

Sharma et al. [142] addressed the issue of financial fraud within the Indian Banking System. They explored the potential of blockchain technology to decentralize the system and mitigate the problem. The paper briefly discussed key challenges within the Indian Banking System, such as issues with double taxation and subpar documentation practices.

Kuzmina et al. [88] examined the unique aspects of introducing and utilizing central bank digital currencies (CBDCs) in various countries worldwide. Their research encompassed a review of how CBDCs are defined, identifying the primary reasons and objectives for creating such a tool, and outlining its key attributes. Special emphasis was placed on analyzing specific projects from individual countries as examples to gauge the effectiveness and viability of incorporating central bank digital currencies into monetary systems.

Building on the insights gained from previous research in the financial industry, our exploration now shifts to the realm of cryptocurrency wallets. While the prior part illuminated the potential of blockchain in financial applications, this part delves into the practical tools that facilitate cryptocurrency management and transactions, offering a comprehensive overview of their capabilities and integration with fiat currencies. The majority of the wallets are tailored for the management of cryptocurrencies rather than traditional fiat currencies. These wallets excel at functions like storing, sending, receiving, and exchanging a diverse range of cryptocurrencies. Nevertheless, a select few do offer limited features for fiat currency support and integration with third-party services. Notably, Coinbase Wallet [34] and Binance Trust Wallet [19] are associated with major cryptocurrency exchanges, allowing users to engage in fiat-to-crypto transactions. At the same time, Exodus Wallet [48] provides fiat-to-crypto onramps through partner services. Edge Wallet[46], on the other hand, facilitates direct cryptocurrency purchases using fiat through bank account linking. However, it is essential to bear in mind that these wallets primarily serve as conduits for cryptocurrencies. For exclusive fiat transactions, conventional banking apps or financial service platforms like PayPal are generally preferred. When it comes to acquiring cryptocurrencies using fiat, cryptocurrency exchanges often bridge the gap, with many of these wallets harmoniously integrating with these platforms to streamline the process.

We have summarized the features explored in our proposed system with respect to the existing literature in Table 7.1.

TABLE 7.1: Contrasting the Existing State-of-the-Art with the Proposed System

Article	Distributed	Multicurrency		Implementation details	User interface for application	Data security	Availability
		Fiat	Crypto				
[49]	✔	✘	✔	✘	✘	✔	✔
[44]	✔	✘	✔	✘	✘	✔	✘
[9]	✔	✘	✔	✔	✘	✔	✘
[142]	✔	✘	✔	✘	✘	✔	✘
Proposed	✔	✔	✔	✔	✔	✔	✔

7.2 Background

In this section, we offer a succinct overview of various useful technologies and tools that we use in our application.

– **Metamask Wallet:** MetaMask is a popular cryptocurrency wallet and gateway to decentralized applications [90]. It functions as a browser extension or a mobile app, allowing users to manage their cryptocurrency holdings, interact with decentralized applications, and execute transactions on the Ethereum blockchain (Mainnet and various Testnets like Goerli, Sepolia, etc.). MetaMask acts as a bridge between the users and the blockchain, providing a user-friendly interface for managing digital assets and interacting with the decentralized web.

– **Web 3.0:** Web 3.0, often referred to as the "decentralized web", is an evolution of the traditional web (Web 2.0) [69]. It aims to create a more open, user-centric, and decentralized internet experience. Web 3.0 envisions a world where users have greater control over their data and online interactions, facilitated by blockchain technology and decentralized protocols. Instead of relying solely on central servers, Web 3.0 applications use peer-to-peer networks and smart contracts to create more trustless and autonomous online interactions [131].

7.3 System Model

In this section, we will delve into the architecture of the proposed banking system and furnish implementation details.

Remark 5. *Traditional e-banking systems face significant challenges, including reliance on centralized architectures that introduce vulnerabilities such as single points of failure, data breaches, and inefficiencies in managing accounts that hold multiple currencies—both fiat and cryptocurrencies. These systems often lack the transparency, trust, and flexibility required for seamless global transactions, fail to support real-time currency conversion effectively, and are limited by operational downtime. The problem is to design a blockchain-enabled, decentralized e-banking system that securely manages multicurrency accounts, supports real-time conversion between currencies, ensures transparent and tamper-proof transaction handling, provides robust user authentication, and operates* 24×7*. Additionally, a GUI-based web application is essential to enable seamless interactions and enhance user accessibility and experience.*

7.3.1 System Components

The integral components of the system are as follows -

– **Customers:** These are the end-users of the banking system, encompassing individuals and entities seeking to leverage the hybrid capabilities of the account for seamless management of both fiat currencies and cryptocurrencies.

– **Smart Contract:** Smart contracts form the core of the system, automating transactions like deposits, withdrawals, fund transfers, and currency

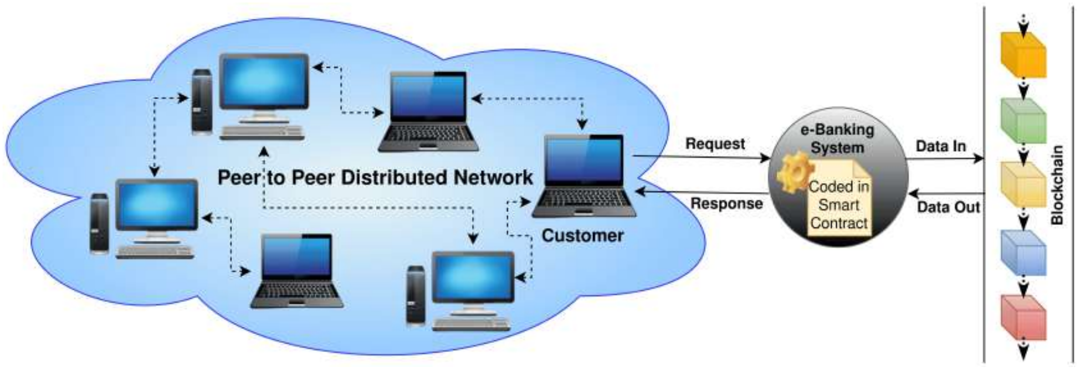


FIGURE 7.2: Context Diagram: Blockchain-enabled Distributed e-Banking System

exchanges. This streamlines operations boost accuracy and removes intermediaries.

- **Blockchain:** Serving as the foundation, the blockchain ensures the immutability and security of transaction records, enabling transparency and trust within the system through its decentralized ledger structure.

- **Peer-to-Peer Network:** The system operates on a decentralized peer-to-peer network, enabling the execution of transactions without centralized authority and operationalizing the system 24×7.

Figure 7.2, as shown in the diagram, illustrates the context of our model.

7.3.2 Functionalities Offered by Our e-Banking System

Pre-requisite: An account in the Metamask web-based wallet connected to Sepolia Testnet with a nonzero balance (test ethers) and an active internet connection.

The Metamask Wallet contains the user's balance (in ETH only). However, the user's bank account (A/C) contains balances in different currencies (crypto and fiat). Let $\boldsymbol{C}$ be the set of different currency units that the user can hold in their bank account. In this case, the set is as follows:

$$\boldsymbol{C = \{ETH, INR, Dollar, Euro\}}$$

1. **Open Bank Account:**
   - Generate a unique A/C number for a new user.
   - Establish a one-to-one correspondence between the user's address (i.e., Public Key) and their Bank A/C Number.

2. **Deposit:**
   - User enters the value to be deposited and selects one unit from the set $\boldsymbol{C}$.

  - If ETH is the desired unit, the amount is directly deducted from Metamask wallet balance and deposited to the user's Bank Account.
  - For other units (fiat currencies), deduct an equivalent value in ETH from the Metamask wallet balance and store in the smart contract as the same unit chosen by the user.

3. **Withdraw:**
   - User enters the value to be withdrawn and selects one unit from the set $\boldsymbol{C}$.
   - Check the smart contract to ensure sufficient balance in the selected unit for withdrawal from the bank account. If the balance is sufficient, allow the user to withdraw the specified amount in the chosen unit. Consequently, debit the user's bank A/C balance (in the selected unit) and credit an equivalent amount in ETH to the user's Metamask wallet.
4. **Fund Transfer:**
   - User specifies the receiver A/C number (or public key address), the transfer amount, and selects one unit from the set $\boldsymbol{C}$.
   - Validate that both the sender and receiver have bank accounts.
   - Confirm the sender has sufficient A/C balance (in the selected unit) for the transfer. If so, enable the user to transfer funds to the recipient's bank A/C. Consequently, debit the sender's bank A/C by the transferred amount (in the chosen unit), and credit an equivalent amount in the recipient's bank A/C (in the same unit).
   - Metamask Wallet balances (in ETH) for both sender and receiver remain unchanged, except for a slight deduction in the sender's Metamask wallet balance to cover Ethereum transaction fees.
5. **Currency Exchange:**
   - Users can exchange their currency from one unit to another. The user specifies the amount, source, and destination currency units from the set $\boldsymbol{C}$.
   - Validate the user's balance (in the chosen source currency unit) for conversion. If sufficient, convert the amount to the desired currency unit based on the currency exchange rate, updating the user's A/C balance accordingly.
6. **Check Bank A/C Specifications and Balance:**
   - Display information as follows:
     A/C Holder Name: Anonymous
     A/C Number: 001
     Ethereum Address (Public Key): 0X1234abD384...(20 bytes)
     Balance: 'w' ETH, 'x' INR, 'y' Dollar, 'z' Euro

7. **Close Bank Account:**

   – Transfer the entire bank A/C balance to the user's Metamask Wallet upon closing the account. For example, if the user's bank account contains 'w' ETH, 'x' INR, 'y' Dollar, and 'z' Euro at closure, the Metamask wallet will be credited with $\boldsymbol{(w + ToEther(x, INR) + ToEther(y, Dollar) + ToEther(z, INR))}$ ETH. Here, ***ToEther()*** is a utility function converting currency values to their equivalent Ether counterparts.

   – Deactivate the bank account in the smart contract.

For all transactions involving currency conversion, the applicable currency exchange rate will be used during the execution of the transaction.

7.3.3 Managing Multi-Currency Transactions in the System

The e-banking system manages multiple fiat currencies by leveraging a single Ethereum ledger, where account balances for different currencies are stored as state variables in ETH units. When displaying balances to users, real-time conversion rates are applied to present the equivalent fiat currency amounts, avoiding direct currency exchanges and ensuring simplicity and consistency. Users can deposit, withdraw, or transfer funds in multiple fiat or crypto units through an external cryptocurrency wallet, such as MetaMask.

During deposits, the equivalent amount in ETH is deducted from the user's wallet and credited to their bank account, represented as a smart contract on the Ethereum ledger, updating the state variable for the desired currency unit. For withdrawals, the requested amount is converted to its ETH equivalent using the real-time conversion rate and credited back to the user's cryptocurrency wallet as ETH. For fund transfers, the user can specify the desired currency unit, and the sender's account is debited in that unit while the receiver's account is credited with the same amount.

Notably, every banking transaction—whether deposit, withdrawal, or fund transfer—requires updating the state variables in the smart contract, necessitating interaction with the underlying blockchain. As a result, the transaction initiator incurs a transaction fee, introducing an additional overhead cost associated with blockchain-based banking systems. This trade-off highlights the balance between the benefits of decentralization and the operational costs of blockchain infrastructure.

7.3.4 Algorithm

The workflow of our e-banking System is presented in Algorithm 1. In order to simplify comprehension, we have encapsulated the functions of the smart contract as a black box. Nevertheless, The source code is available on GitHub repository*.

*https://doi.org/10.5281/zenodo.11079932

Algorithm 6: Work Flow of our e-Banking System

```
Connect the Web Application to Metamask Wallet Account
if !(Bank account exists) then                                          ▷ Create a new bank account
    Prompt: username, initial_deposit_value, currency_unit
    Validate: user input.
    if currency_unit $\neq$ ETH then
        Convert: ToEther(initial_deposit_value, currency_unit)
    Account_number $\leftarrow$ Open_account()                 ▷ Execute back-end smart contract
    Return: Account_number                          ▷ User gets unique Account_number
while true do
    Display: Banking Home Page with options - {1. Deposit, 2. Withdraw, 3. Fund
      Transfer, 4. Currency Exchange and 5. Check Account Balance, 6. Close
      Account}
    Read: user choice
    if user's choice is 1 then                                                    ▷ Deposit funds
        Prompt: deposit_amount, currency_unit
        Validate: user input
        if currency_unit $\neq$ ETH then
            Convert: ToEther(deposit_amount, currency_unit)
        Deposit()                ▷ Execute smart contract & Update account balance
    else if user's choice is 2 then                                              ▷ Withdraw funds
        Prompt: withdrawal_amount, currency_unit
        Validate: user input
        if withdrawal_amount $\leq$ Account Balance (in specified currency_unit) then
            if currency_unit $\neq$ ETH then
                Convert: ToEther(withdrawal_amount, currency_unit)
            Withdraw()              ▷ Execute smart contract & Update account balance
    else if user's choice is 3 then                                              ▷ Transfer funds
        Prompt: recipient_account_number, transfer_amount, currency_unit
        Validate: user input
        if recipient_account_number is valid and transfer_amount $\leq$ Sender's Account
          Balance (in specified currency_unit) then
            if currency_unit $\neq$ ETH then
                Convert: ToEther(transfer_amount, currency_unit)
            Fund_transfer()
    else if user's choice is 4 then                                         ▷ Exchange Currency
        Prompt: exchange_amount, currency_unit_from, currency_unit_to
        Validate: user input
        if currency_unit_from $\neq$ currency_unit_to and exchange_amount $\leq$ Account Balance
          (in specified currency_unit_from) then
            Currency_exchange()       ▷ Execute smart contract & Update account balance
    else if user's choice is 5 then                                   ▷ Display Account Balance
        Display the account balances under every currency unit
    else if user's choice is 6 then                                              ▷ Close account
        Close_account()  ▷ Execute smart contract, deactivate Account_number & Transfer the
          cumulative ETH balances of all currency units to Metamask wallet
        Exit and terminate the program.
```

One crucial aspect we emphasize is our approach to handling currency conversion. The smart contract uses predefined conversion formulas based on real-time exchange rates obtained through external Oracle API. For transaction management, the contract validates user inputs, calculates the required amount post-conversion, and logs the transaction on the blockchain for transparency and auditability. Each currency unit has a dedicated state variable within the smart contract, responsible for storing A/C balances in WEI (1 ETH = 10^{18} WEI). To illustrate this concept, consider an example where a user opens an account with an initial deposit of 'X' INR. First, we convert this amount to its WEI equivalent (say, 'X' INR = 'Y' WEI) using the prevailing currency exchange rate off-chain. Subsequently, 'Y' WEI is deducted from the user's Metamask wallet balance and recorded as the INR balance within the smart contract. When the user queries their account balance, the smart contract returns 'Y' WEI as the 'INR' balance, while other balances remain at '0', assuming no intermediate transactions or fund inflows. This 'Y' WEI immediately gets converted to equivalent INR as per the applicable exchange rate at that instant and is conveyed to the user on the screen. Due to dynamic real-time exchange rates, the displayed balance might not precisely match 'X' INR.

7.3.5 Implementation Details

We provide a detailed analysis of how our e-banking system is implemented, revealing the essential technologies and architectural components that work together to bring our vision into reality.

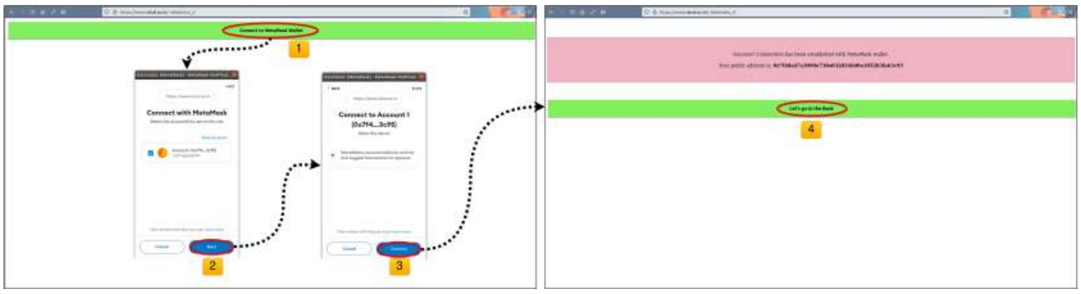


FIGURE 7.3: Connecting to Metamask Wallet

At the same time, we offer a visual narrative of our system in action for ease of understanding. Through meticulously captured screenshots, we illustrate operational scenarios, aligning them with their respective functionalities. These visual aids, arranged in a logical sequence, illuminate the control flow and highlight key transaction updates through distinct boundary markings. The primary operations of our e-banking system encompass establishing a connection to a Metamask wallet, opening a new bank account, depositing funds, withdrawing money, transferring funds to other accounts, exchanging currency, and closing a bank account are depicted in Figure 7.3, 7.4, 7.5, 7.6, 7.7, 7.8 and 7.9 respectively.

1. **Development of Smart Contract**
The heart of our e-banking system is an intricately designed smart contract coded in the Solidity programming language. This smart contract serves as the bedrock for our vital banking operations, orchestrating the entire spectrum of functionalities, including account creation, deposits, withdrawals, secure fund transfers, balance inquiries, currency exchange, and account closure. Solidity's robust capabilities empower us to safeguard the paramount attributes of security, immutability, and transparency that the banking sector demands.

2. **Deployment of Smart contract**
Upon completing the coding phase of our smart contract, we initiated a rigorous

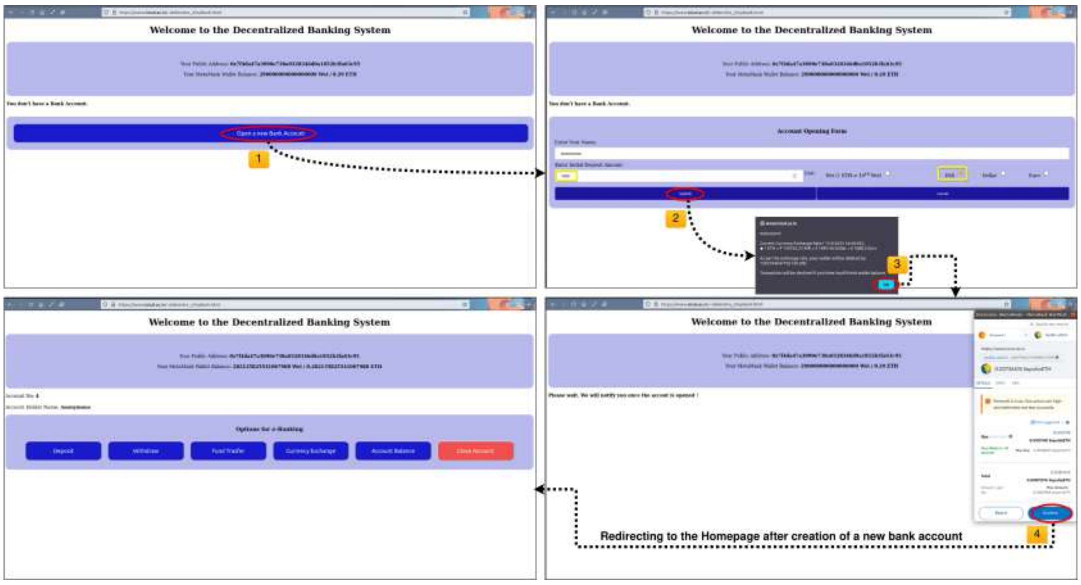


FIGURE 7.4: Opening a New Bank Account

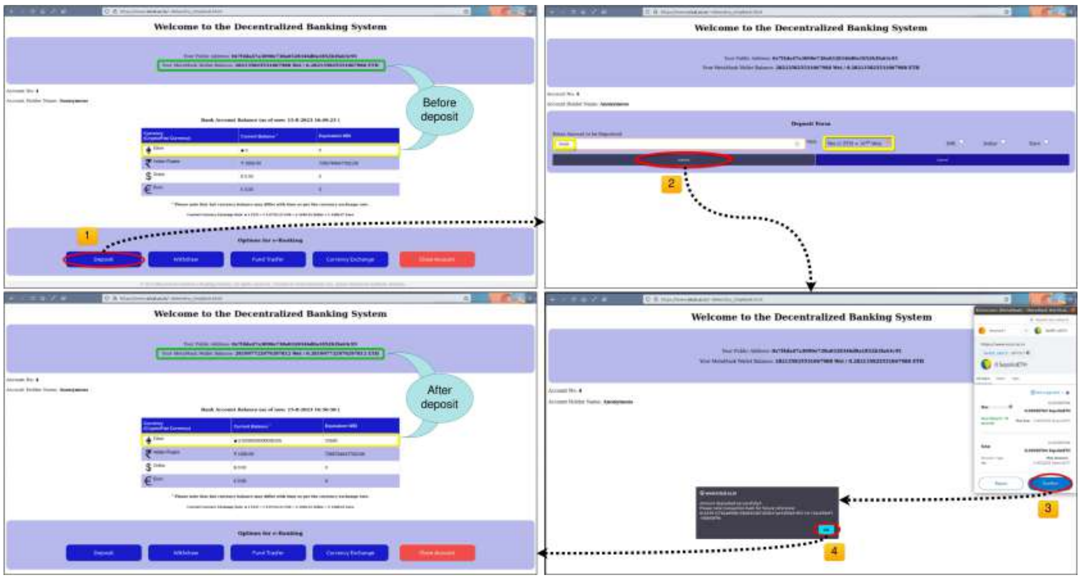


FIGURE 7.5: Depositing Money to Bank Account

testing phase within the private Ganache blockchain network. This meticulous examination confirmed the functionality and reliability of our codebase. With a boosted confidence level, we embarked on the deployment journey, making a significant stride by deploying our contract on the Sepolia Testnet. This enabled us to have a controlled environment that facilitated us to conduct rigorous testing and validation.

3. **Designing of Front-end Interface**
We artfully designed an array of front-end webpages to harmoniously interlace users with the intricate blockchain back-end. We used industry-standard web technologies like HTML, CSS, and JavaScript to craft an intuitive and user-centric interface. Our digital platform allows users to easily interact with blockchain operations, making it simpler to execute smart contract functions.

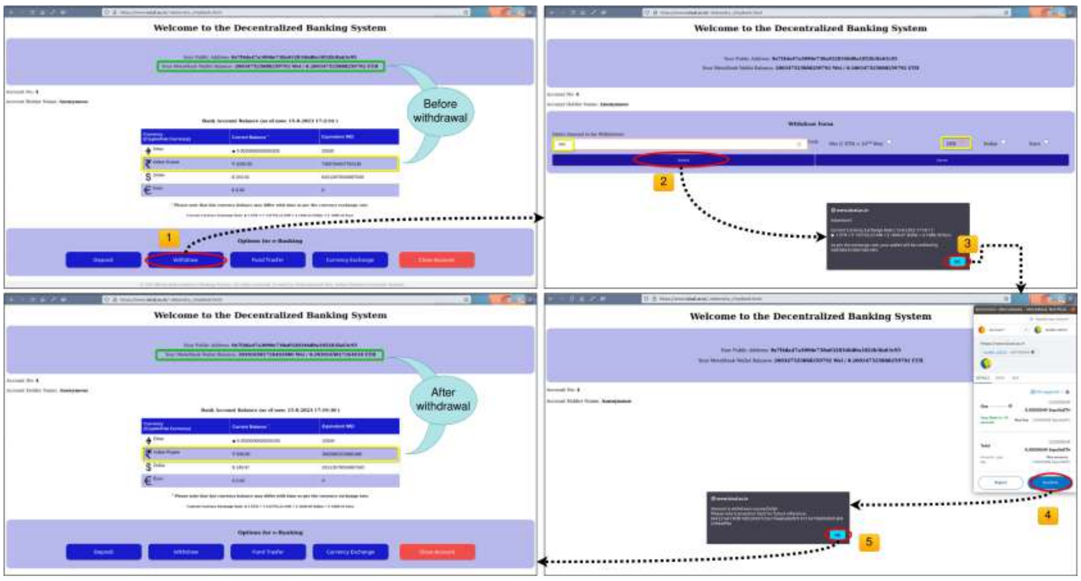


FIGURE 7.6: Withdrawing Money from Bank Account

To ensure seamless interactions, we integrate the Web 3.0 framework. This allows our front-end to communicate with the back-end blockchain, perform transactions, and provide important account-related information.

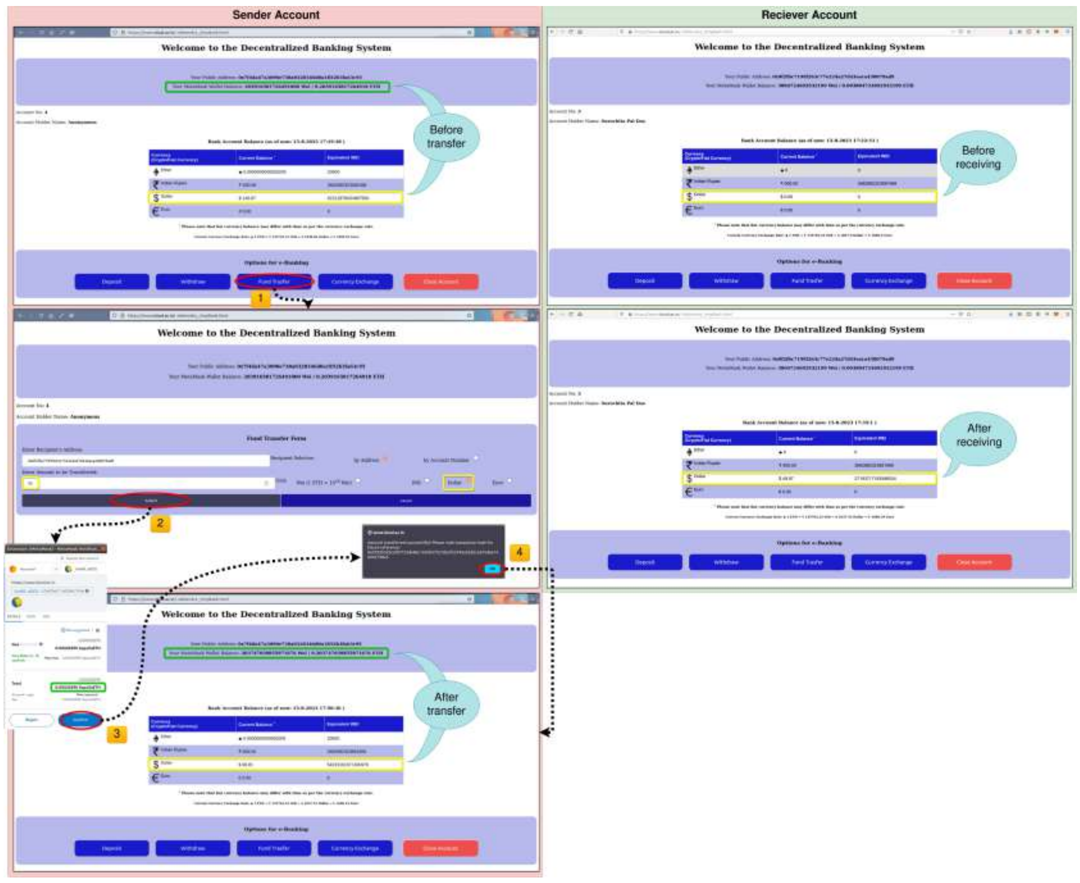


FIGURE 7.7: Transferring Money between Two Bank Accounts

4. **Secure Communication between Front-end and Back-end**
The symphony between our front-end and the smart contract unfolds via a secure communication pipe. All users' input and requests undergo careful validation before embarking on their journey to the smart contract. In parallel, responses originating from the smart contract retrace the same channel, arriving at the front-end with unwavering integrity. This holistic approach ensures the sanctity of data throughout the e-banking process, underpinning consistency and precision.

We have incorporated an external API to obtain up-to-date currency conversion rates for transactions that involve unit conversion. The data received is processed on the client side before being sent to the back-end smart contract, resulting in a smooth currency exchange process. Similarly, when retrieving user account balances from the back-end smart contract, the values are automatically converted to fiat currency units if necessary, using the currency conversion rates provided by the API. This seamless integration ensures that balances are accurately displayed on the user interface.

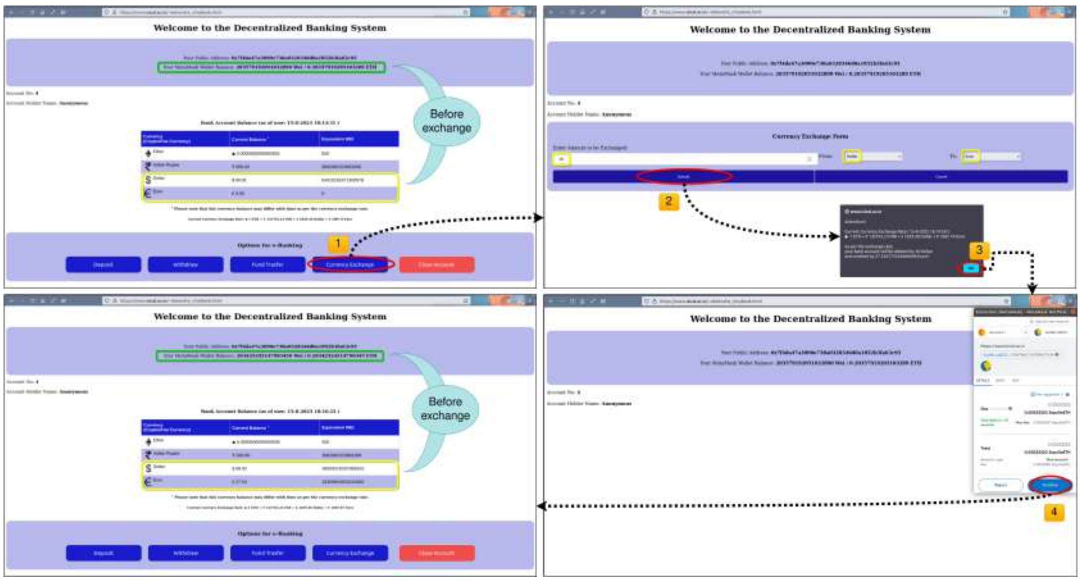

Welcome to the Decentralized Banking System
Before exchange
1
2
3
4
Welcome to the Decentralized Banking System
Currency Exchange Form
Welcome to the Decentralized Banking System
Before exchange
Options for e-Banking
Deposit
Withdraw
Fund Transfer
Currency Exchange
Close account
Welcome to the Decentralized Banking System

FIGURE 7.8: Currency Exchange

7.4 Security Features

Blockchain technology relies on cryptographic primitives, such as hash functions and digital signatures. The security of the blockchain and, consequently, our entire system hinges on the robustness of these foundational cryptographic elements. Assuming the foundational technologies are adequately secured, let us delve into the security features offered by our application, many of which stem from its underlying blockchain architecture.

1. **Immutable Transaction Records:** Our system ensures that once transactions are recorded on the blockchain, they become permanent and cannot

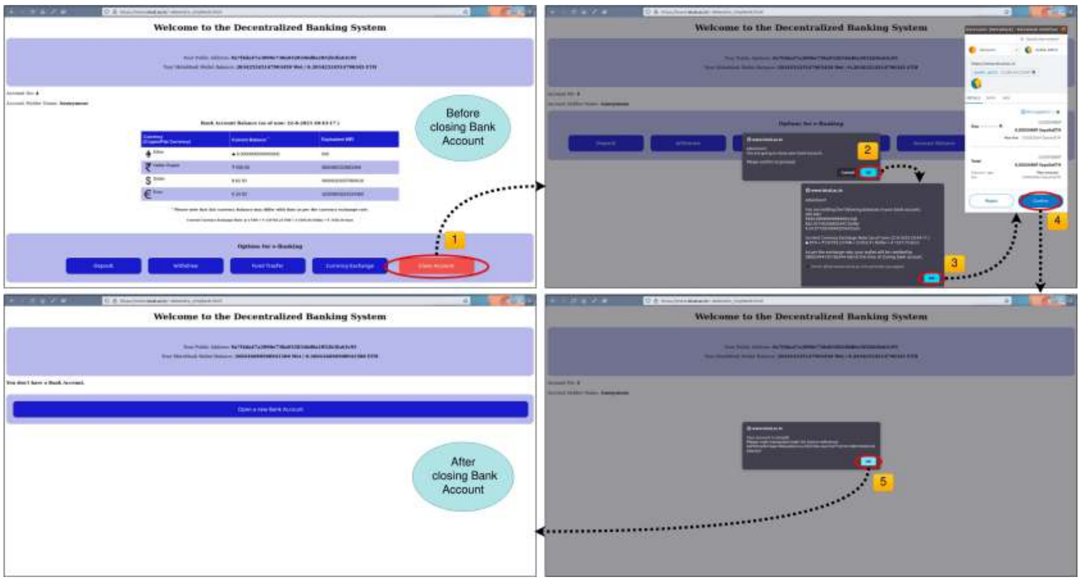

Welcome to the Decentralized Banking System
Before closing Bank Account
1
2
3
4
5
Options for e-Banking
After closing Bank Account
Open a new Bank Account

FIGURE 7.9: Closing a Bank Account

be altered. This guarantees the accuracy and reliability of financial histories, preventing any manipulation or unauthorized changes.

2. **Decentralized Data Storage:** User data is distributed across the network, eliminating the risk of failures (e.g., server crashes). It enhances system resilience and security by reducing vulnerability to attacks targeting centralized data repositories.

3. **Tamper-Resistant Smart Contracts:** Our banking operations, encompassing Deposits, Withdrawals, Fund Transfers, and Currency Exchanges, are meticulously encoded within smart contract code, ensuring precision and transparency. These smart contracts are committed to the blockchain in an unalterable form, effectively shielding them from tampering. This guarantees that the execution of each transaction adheres to its predefined terms and conditions, eliminating any room for manipulation or ambiguity.

4. **Cryptography-Backed User Authentication:** The e-banking system leverages advanced cryptographic methods, specifically the Elliptic Curve Digital Signature Algorithm (ECDSA), to ensure secure user authentication and protect user identities. During registration, users generate a cryptographic key pair, consisting of a public key (PK) and a private key (SK). The public key (PK) serves as the user's unique identifier within the system, while the private key (SK) is securely stored and managed by the user. All interactions with the e-banking system, including account access and transaction authorizations, require cryptographic signatures created using the private key. These signatures are validated against the user's public key to ensure the authenticity of the actions. This approach eliminates the reliance on traditional usernames and passwords, which are susceptible to phishing and brute-force attacks, thereby enhancing security.

Moreover, the decentralized nature of the authentication mechanism aligns with the principles of blockchain, ensuring that user credentials are not stored on centralized servers, reducing the risk of breaches. The system ensures non-repudiation, as users cannot deny their actions once cryptographically signed, further strengthening trust and accountability within the e-banking system.

This cryptographic authentication mechanism provides a robust, tamper-proof, and privacy-preserving foundation for user interactions, ensuring only legitimate users can access and operate within the system.

5. **Protected Private Key Management:** Users have sole ownership and control over their private keys in the Metamask wallet. It minimizes the risk of unauthorized access, ensuring user assets remain secure from external threats.

6. **Built-In Network Resilience:** Our decentralized network design reduces dependency on centralized systems, enhancing resilience against service disruptions caused by localized failures or targeted attacks and ensuring higher service availability

7. **Continuous Monitoring and Rapid Response:** Our application is already deployed and operational for user access; it benefits from real-world exposure, making it a live testing ground for identifying potential vulnerabilities or shortcomings.

By leveraging these existing security features, our blockchain-enabled distributed e-banking system guarantees a fortified platform that users can trust for managing their financial assets and transactions with utmost confidence. This framework prioritizes financial well-being through advanced blockchain security.

7.5 Result and Discussion

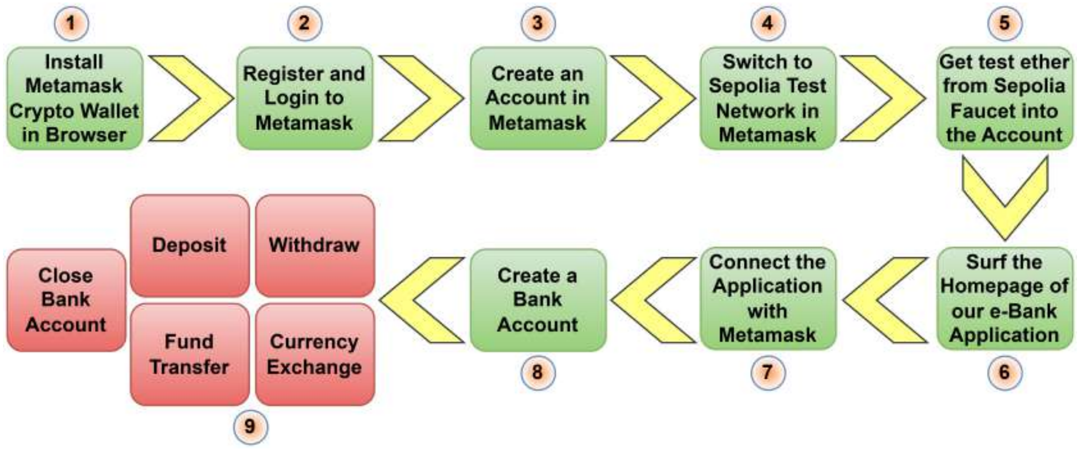

1
Install Metamask Crypto Wallet in Browser
2
Register and Login to Metamask
3
Create an Account in Metamask
4
Switch to Sepolia Test Network in Metamask
5
Get test ether from Sepolia Faucet into the Account
6
Surf the Homepage of our e-Bank Application
7
Connect the Application with Metamask
8
Create a Bank Account
9
Deposit
Withdraw
Fund Transfer
Currency Exchange
Close Bank Account

FIGURE 7.10: Steps to access our e-Banking DApp

Implementation Setup: We have successfully implemented the proposed e-Banking System on a system running Linux Ubuntu 22.04.2 LTS with an Intel(R) Core(TM) i5-8250U and 8.0 GiB of RAM. We deployed the smart contract on the Ethereum Sepolia test network. Metamask crypto wallet is used for account creation and transaction initiation. Our back-end smart contracts were written in Solidity, while the front-end used HTML, CSS, and JavaScript. To ensure seamless interaction between the front-end webpage and the back-end blockchain, we utilized the Web 3.0 library. To access our application, please refer to the instructions outlined in Figure 7.10. The smart contract is reachable through the deployment address - 0x459b5ec8c6c4273ed131934cdf0f2064c563bdc0 on the Sepolia testnet. The contract creation transaction consumed 3900576 units of gas.

Our main goal was to determine the feasibility of our blockchain model through various results. The cost and time were the two main factors we considered. The detailed discussion can be found in Chapter 2, Section 2.6.10.

Transaction Cost: Figure 7.11 illustrates the gas cost of various transactions of our e-banking system. It can be seen from Figure 7.11 that the gas cost for Open_account is high; however, it is a viable expense since this function will be executed only once per user. This initial cost is a reasonable trade-off for ensuring the security and reliability of the Open_account process. Furthermore, it's essential to highlight that the remaining transaction costs are optimized and efficient, which contributes to the overall efficiency of our system.

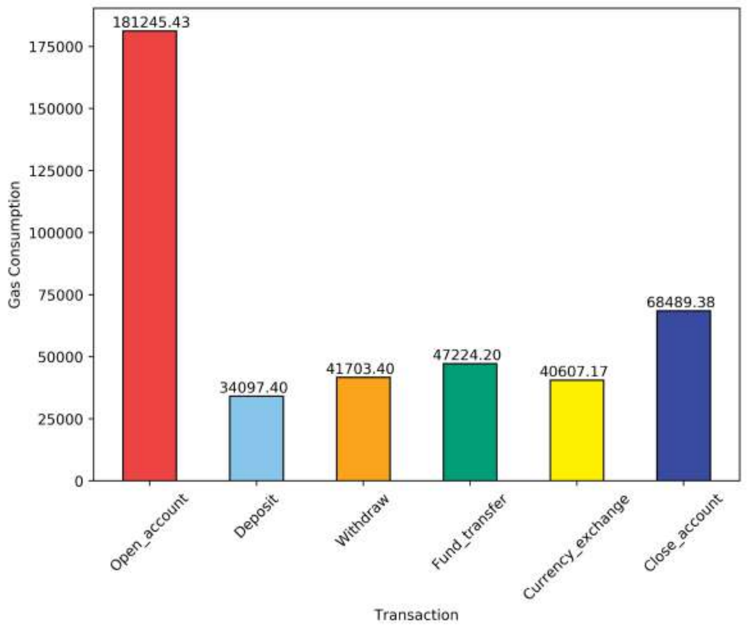


FIGURE 7.11: Gas Consumption of Transactions

Transaction Latency: To determine transaction latency, we measure the maximum, minimum, and average time taken for 100 successful transactions to be submitted to the network and included in a block. The latency for various transactions for our system is depicted in Figure 7.12. Upon closer examination of Figure 7.12, it becomes evident that the average transaction latency for each of the transactions is at an optimum level. This observation underscores the robustness of our implementation. The consistently low transaction latency is a testament to the scalability and performance of our system, ensuring a seamless and responsive user experience.

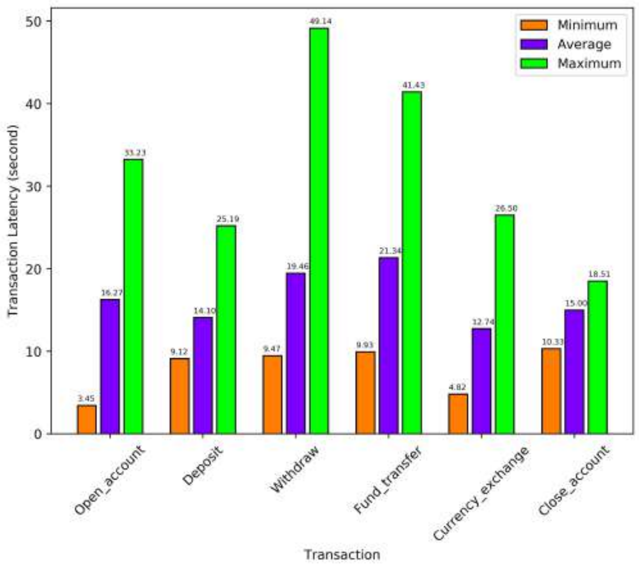


FIGURE 7.12: Transaction Latency

7.6 Conclusion

We explore the potential of blockchain technology within the banking sector. The emergence of blockchain has glorified a new era of decentralized finance (DeFi), challenging the traditional centralized banking model with its inefficiencies and vulnerabilities. Through a carefully crafted model, we have addressed the limitations of conventional banking systems by introducing an innovative dual-capability bank account that can simultaneously accommodate both fiat currencies and cryptocurrencies. The approach facilitates the development of a financial landscape that is more inclusive and flexible, accommodating users who have a preference for traditional fiat currencies as well as the emerging domain of cryptocurrencies. By leveraging smart contracts and decentralized blockchain technology, we have built a functional e-Banking system that holds multiple fiat currencies and seamlessly integrates cryptocurrency transactions. This innovative approach overcomes the significant obstacle of combining traditional and digital currencies by connecting traditional practices with technological advancements. The end result is a robust, decentralized, and user-friendly web application offering diverse banking capabilities. Experimental result shows the satisfactory outcome of various performance metrics. In conclusion, our work advances the vision of a decentralized banking system that accommodates users' diverse preferences, disrupts traditional financial paradigms, and unlocks the potential of blockchain technology to reshape the future of banking and finance. In the future, our e-Banking system holds promising directions for growth and improvement:

1. Developing a dedicated mobile application to enhance user accessibility and convenience.
2. Implementing advanced security measures, including biometric and multi-factor authentication, will strengthen the system's security framework.
3. Expanding a broader range of banking services, such as investment options, insurance services, and wealth management, will transform the e-Banking system into a comprehensive financial hub.
4. Ensuring regulatory compliance and adherence to evolving financial standards remains a key priority to ensure a stable and compliant ecosystem. We aim to integrate KYC (Know Your Customer) and AML (Anti-Money Laundering) compliance measures into our model. This enhancement will enable us to track and prevent fraudulent activities efficiently.

The e-banking system currently operates within the transaction throughput limitations of the underlying Ethereum blockchain, relying on its native capabilities to handle user interactions and transaction processing. While this ensures seamless integration with Ethereum's robust infrastructure, enhancing the system's scalability remains a promising area for future work. Potential improvements include the integration of layer-2 scaling solutions such as rollups or state channels to increase transaction throughput and reduce costs. Additionally, upcoming advancements in Ethereum, such as sharding in Ethereum 2.0, could be leveraged to further improve scalability. These enhancements would enable the system to support a significantly larger user base and higher transaction volumes, making it more robust and adaptable for widespread adoption in the e-banking domain.

While blockchain and smart contract-based applications offer promising opportunities for enhancing transparency, security, and efficiency in banking systems, their implementation must be approached with caution. Smart contracts, though innovative, are not immune to vulnerabilities. Incidents such as reentrancy attacks, logic flaws, and exploits in poorly written code highlight the risks associated with inadequate security measures. For instance, the recent WazirX attack [159] underscored the potential for vulnerabilities in smart contract implementations, emphasizing the need for rigorous testing, secure coding practices, and regular audits. Addressing these challenges is critical to ensuring the reliability and safety of smart contract-based banking solutions, paving the way for their wider adoption and trust in the financial sector.

8 Conclusion

"I think there is always room for improvement, and every day is a chance to get better."

— **Jennifer Brady**

In this concluding chapter, we take a comprehensive look at the previous chapters, summarizing and drawing conclusions from the various aspects explored in this thesis. This research has primarily focused on designing blockchain-enabled secure real-life applications, concentrating on the healthcare and banking sectors. Here, we highlight our key contributions, improvements, and extensions to existing methods. Furthermore, we delve into the potential directions for future research and identify the open problems that lie ahead in the field of blockchain technology.

Chapter 1 functions as the thesis introduction, while **Chapter 2** offers a foundational understanding of various cryptographic concepts and blockchain technology, setting the stage for readers to engage with the thesis effectively. Our first three contributory chapters focus on blockchain's impact in the healthcare domain, while the latter two chapters concentrate on its impact on the banking sector.

8.1 Summary of Technical Contributions

Chapter 3: Blockchain-Enabled Secure Healthcare System
We proposed a novel, Secure, and Smart Healthcare System utilizing blockchain to ensure tamper-proof Electronic Health Records (EHRs) and preserve patient privacy. The system's deployment on private and Ropsten test networks confirmed its practical feasibility and effectiveness in enhancing healthcare data management. Additionally, the system addresses interoperability challenges, enabling seamless data sharing among healthcare providers while maintaining robust security standards.

Chapter 4: Blockchain-Enabled Secure Health Insurance System
This work introduced a Health Insurance Processing System that employs smart contracts to automate interactions between policyholders and insurers. The system achieves increased transparency, expedites key processes, and provides better safeguards against fraudulent activities, demonstrating blockchain's utility in modernizing health insurance infrastructure. Furthermore, the system incorporates mechanisms to enforce fairness, ensuring honest parties are protected against malicious behavior during claim processing.

Chapter 5: Blockchain-Enabled Secure Vaccine Passport System
We developed a Blockchain-Enabled Secure Vaccine Passport System to ensure secure, transparent, and efficient vaccine administration and verification. By leveraging the blockchain's immutable ledger and integrating IPFS for off-chain data storage, the system provides tamper-proof vaccination records while maintaining user privacy. The implementation supports real-time verification of vaccination status, enabling its use in scenarios such as travel or access to public venues.

Chapter 6: Blockchain-Enabled Secure Payment Card Tokenization System
The proposed solution introduced decentralization in payment card tokenization through smart contracts, reducing reliance on centralized entities. This approach enhanced security and efficiency, as confirmed by experimental results, indicating its potential to improve current tokenization systems significantly. Additionally, the system ensures privacy by tokenizing sensitive card data and protecting it from unauthorized access or tampering through blockchain's inherent immutability.

Chapter 7: Blockchain-Enabled Multicurrency Supported Distributed e-Banking System
We presented a dual-capability e-Banking system that integrates both fiat and cryptocurrencies. Utilizing smart contracts and decentralized blockchain technology, the system offers a robust, user-friendly platform for diverse banking needs, bridging traditional finance with emerging digital currencies. The system also enables real-time conversion and multicurrency management, demonstrating its adaptability to evolving financial landscapes.

A unified blockchain platform could serve as the foundation to integrate the proposed healthcare, insurance, and e-banking systems, creating a seamless and interconnected ecosystem. By securely linking health records with insurance claims and payment processing, the platform would enable users to access and manage their healthcare and financial needs through a cohesive interface. For instance, a patient's medical records stored in the healthcare system could be directly referenced during an insurance claim, automating the verification process and reducing administrative delays. If payments are made using credit or debit cards, the tokenization system ensures that sensitive card details are replaced with secure tokens, safeguarding user data while processing transactions. Additionally, the integrated e-banking system could facilitate instant payments for medical expenses, whether using fiat, cryptocurrencies, or tokenized credit card transactions, leveraging smart contracts to ensure transparency and trust. This unified approach enhances operational efficiency, eliminates redundant processes, and maintains robust privacy and security through blockchain's decentralized and tamper-proof architecture. The interoperability of these systems offers a streamlined user experience, bridging healthcare, insurance, and banking while addressing the complexities of modern digital ecosystems.

8.2 Future Scope

While our research has demonstrated promising results, several challenges and areas for future development remain:

1. **Enhanced Privacy and Data Protection Mechanisms**: While our systems ensure user privacy, further research into advanced cryptographic techniques, such as zero-knowledge proofs, can enhance privacy and security in blockchain applications. Balancing the immutable nature of blockchain with data protection regulations like the General Data Protection Regulation (GDPR) of the European Union, particularly the "right to be forgotten" [163], presents an ongoing challenge that requires innovative solutions.

2. **Scalability Solutions**: Addressing scalability challenges is crucial for the widespread adoption of blockchain-based systems. Current blockchain networks often face limitations in handling large transaction volumes due to the constraints of on-chain processing. Future work could explore the implementation of layer 2 solutions, such as state channels and sidechains, to offload transactions from the main blockchain, thereby improving transaction throughput and reducing latency [99], [105]. State channels enable off-chain interactions between participants, requiring minimal on-chain operations, which significantly reduces congestion. Similarly, sidechains operate as independent blockchains connected to the main chain, allowing parallel processing of transactions and custom optimizations tailored to specific use cases. These solutions not only enhance scalability but also lower transaction costs, making the systems more accessible for broader adoption.

3. **Interoperability**: Developing interoperability standards for different blockchain networks can facilitate seamless data sharing and collaboration across various platforms, enhancing the functionality and adoption of blockchain-enabled applications.

4. **Regulatory Compliance**: As blockchain technology evolves, ensuring compliance with regulatory frameworks becomes essential. Future research can explore mechanisms for embedding compliance checks within smart contracts to meet legal and regulatory requirements.

5. **Integration with Emerging Technologies**: Combining blockchain with other emerging technologies, such as artificial intelligence and the Internet of Things (IoT) [43], [136], [145], can unlock new possibilities for secure and intelligent real-life applications in various sectors.

6. **User-Centric Design**: Focusing on user experience and accessibility can drive broader adoption of blockchain applications. Future work could involve developing more intuitive interfaces and enhancing usability to cater to a diverse user base.

7. **Performance Optimization of Smart Contracts**: Optimizing the performance of smart contracts is essential for efficient blockchain operations. Future research could focus on minimizing gas consumption and improving execution speed to make smart contracts more practical for large-scale applications.

8. **Energy Efficiency and Carbon Footprint Reduction**: There is an urgent need to address the high energy consumption and carbon footprint associated with blockchain technologies, particularly cryptocurrencies. Studies have shown that popular cryptocurrencies like Bitcoin consume energy on the scale of entire countries such as Sweden and Thailand. The carbon footprint of Bitcoin has been estimated to be close to that of Greece and Oman [84]. Future research should focus on developing and implementing more energy-efficient consensus mechanisms and blockchain architectures. Case studies on networks like Ethereum 2.0 and Pi Network, which aim to solve some of these challenges, can provide valuable insights. This aligns with global sustainability goals and the urgent need to control human-caused global warming.

9. **Quantum Secured Blockchain**: Quantum computing presents both challenges and opportunities for blockchain technology. While quantum computers could potentially break the current cryptographic methods used in blockchains, they also offer ways to enhance security and performance. The blockchain community is actively researching quantum-resistant cryptographic algorithms to safeguard against future quantum threats. Simultaneously, concepts like quantum blockchains [122] are being explored to leverage quantum principles for improved security. As both fields evolve, their intersection may lead to more secure and efficient blockchain systems, potentially enabling more sophisticated smart contracts and applications.

While our research has focused on the healthcare and banking sectors, blockchain technology has vast potential across various other important sectors, including:

– **Supply chain management**: Enhancing traceability, reducing fraud, and improving efficiency in global supply chains [22], [101], [178].

– **Decentralized data marketplaces**: Enabling secure and fair data sharing and monetization [42], [102], [137].

- **Pharmaceutical industry**: Ensuring drug authenticity and improving clinical trial management [55].
- **Decentralized identity management**: Providing secure, self-sovereign digital identities [41], [58].
- **e-Voting systems**: Increasing transparency and security in electoral processes [104], [106].
- **Real estate**: Streamlining property transactions and record-keeping [147].
- **Energy sector**: Facilitating peer-to-peer energy trading and grid management [94].
- **Intellectual property and copyright protection**: Ensuring proper attribution and managing royalties [65].
- **Government and public services**: Improving transparency in public spending and service delivery [114].
- **Education**: Verifying academic credentials and managing lifelong learning records [28], [98].
- **Charity and non-profit organizations**: Enhancing transparency in fundraising and fund allocation [51].
- **Digital content creation and distribution**: Ensuring fair compensation for creators and reducing piracy [70].
- **Automotive industry**: Authenticating spare parts, securing vehicle history records, enabling car-to-car communication, enhancing cybersecurity in connected vehicles, and managing driverless car operations [53].
- **Logistics and transportation**: Optimizing route planning and cargo tracking [92].
- **Agriculture**: Improving food traceability and managing sustainable farming practices [17].
- **Crime investigation**: Maintaining secure and tamper-proof chain of custody for evidence, enhancing the integrity of forensic processes [86], [95], [116].

Completing this thesis has been a transformative journey, enriching my understanding of blockchain technology and its potential to revolutionize critical sectors. This experience underscored the value of collaboration and adaptability in tackling research challenges. The proposed applications aim to foster a more transparent and efficient world by combating corruption, enhancing trust, and improving various societal and industrial aspects. In conclusion, this research highlights blockchain's potential to enhance security, transparency, and efficiency in healthcare and banking. By addressing current limitations and exploring new avenues, we can further advance the adoption and impact of blockchain-enabled solutions in real-world applications.

Bibliography

[1] P. Agbedanu, F. U. Bawah, V. Akoto-Adjepong, N. Awarayi, I. Nti, S. Boateng, P. Nimbe, and O. Nyarko-Boateng, “Blocovid: A blockchain-based covid-19 digital vaccination certificate verification system,” in *2022 International Conference on Engineering and Emerging Technologies (ICEET)*, IEEE, 2022, pp. 1–6. DOI: 10.1109/ICEET56468.2022.10007366.

[2] N. Ahmed, R. A. Michelin, W. Xue, G. D. Putra, S. Ruj, S. S. Kanhere, and S. Jha, “Dimy: Enabling privacy-preserving contact tracing,” *Journal of Network and Computer Applications*, vol. 202, p. 103 356, 2022. DOI: 10.1016/j.jnca.2022.103356.

[3] N. Ahmed, R. A. Michelin, W. Xue, G. D. Putra, W. Song, S. Ruj, S. S. Kanhere, and S. Jha, “Towards privacy-preserving digital contact tracing,” in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2021, pp. 1–3. DOI: 10.1109/ICBC51069.2021.9461052.

[4] R. Ahuja, “Health insurance for the poor,” *Economic and Political Weekly*, pp. 3171–3178, 2004. [Online]. Available: http://www.jstor.org/stable/4415637.

[5] I. Ali and O. M. Alharbi, “Covid-19: Disease, management, treatment, and social impact,” *Science of the total Environment*, vol. 728, p. 138 861, 2020. DOI: 10.1016/j.scitotenv.2020.138861.

[6] A. A. Amponsah, A. F. Adekoya, and B. A. Weyori, “Improving the financial security of national health insurance using cloud-based blockchain technology application,” *International Journal of Information Management Data Insights*, vol. 2, no. 1, p. 100 081, 2022. DOI: 10.1016/j.jjimei.2022.100081.

[7] R. Annon, *Ethereum Gas and Fees*, https://ethereum.org/en/developers/docs/gas/, 2024.

[8] A. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies.* O'Reilly Media, 2014, ISBN: 9781491902646. [Online]. Available: https://books.google.co.in/books?id=IXmrBQAAQBAJ.

[9] G. M. Arantes, J. N. D'Almeida, M. T. Onodera, S. M. D. B. M. Moreno, and V. d. R. S. Almeida, "Improving the process of lending, monitoring and evaluating through blockchain technologies: An application of blockchain in the brazilian development bank (bndes)," in *2018 IEEE International Conference on Internet of Things*, IEEE, 2018, pp. 1181–1188. DOI: 10.1109/Cybermatics_2018.2018.00211.

[10] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack: Analysis and mitigation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1967–1978, 2016. DOI: 10.1109/TIFS.2016.2623588.

[11] E. Bandara, X. Liang, *et al.*, "Indy528—federated learning model tokenization with non-fungible tokens (nft) and model cards," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, IEEE, 2022, pp. 195–201. DOI: 10.1109/MASS56207.2022.00033.

[12] E. Bandara, S. Shetty, *et al.*, "Kaputa-blockchain, non-fungible token and model card integrated 5g/6g network slice broker and marketplace," in *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, IEEE, 2022, pp. 559–564. DOI: 10.1109/MILCOM55135.2022.10017900.

[13] M. Barati, W. J. Buchanan, O. Lo, and O. Rana, "A privacy-preserving distributed platform for covid-19 vaccine passports," in *Proceedings of the 14th IEEE/ACM international conference on utility and cloud computing companion*, 2021, pp. 1–6. DOI: 10.1145/3492323.349562.

[14] M. V. Baysal, Ö. Özcan-Top, and A. Betin-Can, "Blockchain technology applications in the health domain: A multivocal literature review," *The Journal of Supercomputing*, pp. 1–45, 2022. DOI: 10.1007/s11227-022-04772-1.

[15] BBC, *TSB lacked common sense before IT meltdown, says report*, https://www.bbc.com/news/business-50471919, 2019.

[16] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014. DOI: 10.48550/arXiv.1407.3561.

[17] O. Bermeo-Almeida, M. Cardenas-Rodriguez, T. Samaniego-Cobo, E. Ferruzola-Gómez, R. Cabezas-Cabezas, and W. Bazán-Vera, "Blockchain in agriculture: A systematic literature review," in *Technologies and Innovation: 4th International Conference, CITI 2018, Guayaquil, Ecuador, November 6-9, 2018, Proceedings 4*, Springer, 2018, pp. 44–56. DOI: 10.1007/978-3-030-00940-3_4.

[18] D. J. Bernstein, "The salsa20 family of stream ciphers," in *New stream cipher designs: the eSTREAM finalists*, Springer, 2008, pp. 84–97. DOI: 10.1007/978-3-540-68351-3_8.

[19] Binance, *Bianace Trust Wallet*, https://www.binance.com/, 2024.

[20] D. Boneh and V. Shoup, “A graduate course in applied cryptography,” *Draft 0.5*, 2020.

[21] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, “Elliptic curve cryptography in practice,” in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, Springer, 2014, pp. 157–175. DOI: 10.1007/978-3-662-45472-5_11.

[22] S. Bose, M. Raikwar, D. Mukhopadhyay, A. Chattopadhyay, and K.-Y. Lam, “Blic: A blockchain protocol for manufacturing and supply chain management of ics,” in *2018 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, IEEE, 2018, pp. 1326–1335. DOI: 10.1109/Cybermatics_2018.2018.00229.

[23] P. Bradish, S. Chaudhari, M. Clear, and H. Tewari, “Covichain: A blockchain based covid-19 vaccination passport,” in *Future of Information and Communication Conference*, Springer, 2023, pp. 195–206. DOI: 10.1007/978-3-031-28076-4_17.

[24] V. Buterin *et al.*, “A next-generation smart contract and decentralized application platform,” *white paper*, vol. 3, no. 37, 2014.

[25] V. Buterin, “Ethereum: Platform review,” *Opportunities and Challenges for Private and Consortium Blockchains*, vol. 45, 2016.

[26] Y. Cao, J. Chen, and Y. Cao, “Blockchain-based privacy-preserving vaccine passport system,” *Security and Communication Networks*, vol. 2022, no. 1, p. 4 769 187, 2022. DOI: 10.1155/2022/47691.

[27] P. Chakraborty, S. Maitra, M. Nandi, and S. Talnikar, “Contact tracing in post-covid world,” *Indian Statistical Institute Series*, 2020. DOI: 10.1007/978-981-15-9727-5.

[28] S. Chakraborty, K. Dutta, and D. J. Berndt, “Blockchain based resource management system,” *Available at SSRN 3104351*, 2017. DOI: 10.2139/ssrn.3104351.

[29] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, “A comparative survey of symmetric and asymmetric key cryptography,” in *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, 2014. DOI: 10.1109/ICECCE.2014.7086640.

[30] R. N. Charette, *Health Net Data Breach Affects 1.9 Million People Second major breach for Health Net in two years*, https://bit.ly/Health_Net_Data_Breach, 2011.

[31] H. Chaudhari and M. Crane, “Cross-correlation dynamics and community structures of cryptocurrencies,” *Journal of Computational Science*, vol. 44, p. 101 130, Jul. 2020, ISSN: 1877-7503. DOI: 10.1016/J.JOCS.2020.101130.

[32] V. Chinnasamy, *1.5 Million Customers Impacted By US Bank Data Breach – Possible Lessons Learned*, https://www.indusface.com/blog/1-5-million-customers-impacted-by-us-bank-data-breach-possible-lessons-learned, 2022.

[33] L. Cocco, A. Pinna, and M. Marchesi, "Banking on blockchain: Costs savings thanks to the blockchain technology," *Future Internet*, vol. 9, no. 3, p. 25, 2017. DOI: 10.3390/fi9030025.

[34] Coinbase, *Coinbase Wallet*, https://www.coinbase.com/wallet, 2022.

[35] M. Conti, A. Gangwal, C. Lal, and S. Ruj, "Bitcoin blockchain system: An overview of security and privacy aspects," *Blockchains: A Handbook on Fundamentals, Platforms and Applications*, pp. 75–108, 2023. DOI: 10.1007/978-3-031-32146-7_3.

[36] D. Coppersmith, D. B. Johnson, and S. M. Matyas, "A proposed mode for triple-des encryption," *IBM Journal of Research and Development*, vol. 40, no. 2, pp. 253–262, 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:18377483.

[37] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," *The Rijndael Block Cipher*, 1999.

[38] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, Springer, 1990, pp. 416–427. DOI: 10.1007/0-387-34805-0_39.

[39] D. Das, A. Muthaiah, and S. Ruj, "Blockchain-enabled secure and smart healthcare system," in *International Conference on Design Science Research in Information Systems and Technology*, Springer, 2022, pp. 97–109. DOI: 10.1007/978-3-031-06516-3_8.

[40] S. Díaz-Santiago, L. M. Rodríguez-Henríquez, and D. Chakraborty, "A cryptographic study of tokenization systems," *International Journal of Information Security*, vol. 15, pp. 413–432, 2016. DOI: 10.1007/s10207-015-0313-x.

[41] A. Dixit, W. Asif, and M. Rajarajan, "Smart-contract enabled decentralized identity management framework for industry 4.0," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2020, pp. 2221–2227. DOI: 10.1109/IECON43393.2020.9254545.

[42] A. Dixit, A. Singh, Y. Rahulamathavan, and M. Rajarajan, "Fast data: A fair, secure, and trusted decentralized iiot data marketplace enabled by blockchain," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 2934–2944, 2021. DOI: 10.1109/JIOT.2021.3120640.

[43] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, IEEE, 2017, pp. 618–623. DOI: 10.1109/PERCOMW.2017.7917634.

[44] P. D. Dozier and T. A. Montgomery, "Banking on blockchain: An evaluation of innovation decision making," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1129–1141, 2019. DOI: 10.1109/TEM.2019.2948142.

[45] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 967–984. DOI: 10.1145/3243734.3243857.

[46] Edge, *Edge Wallet*, https://edge.app/, 2021.

[47] European Commission, *EU Digital COVID Certificate*, https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_en, 2021.

[48] Exodus, *Exodus Wallet*, https://www.exodus.com/, 2024.

[49] I. Eyal, "Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities," *Computer*, vol. 50, no. 9, pp. 38–49, 2017. DOI: 10.1109/MC.2017.3571042.

[50] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of medical systems*, vol. 42, no. 8, pp. 1–11, 2018. DOI: 10.1007/s10916-018-0993-7.

[51] M. S. Farooq, M. Khan, and A. Abid, "A framework to make charity collection transparent and auditable using blockchain technology," *Computers & Electrical Engineering*, vol. 83, p. 106 588, 2020. DOI: 10.1016/j.compeleceng.2020.106588.

[52] B. Fernandez and N. Uberoi, *Health Insurance: A Primer* (CRS report for Congress). Congressional Research Service, 2015. [Online]. Available: https://books.google.co.in/books?id=HdcRxQEACAAJ.

[53] P. Fraga-Lamas and T. M. Fernández-Caramés, "A review on blockchain technologies for an advanced and cyber-resilient automotive industry," *IEEE access*, vol. 7, pp. 17 578–17 598, 2019. DOI: 10.1109/ACCESS.2019.2895302.

[54] S. Fugkeaw, "An efficient and scalable vaccine passport verification system based on ciphertext policy attribute-based encryption and blockchain," *Journal of Cloud Computing*, vol. 12, no. 1, p. 111, 2023. DOI: 10.1186/s13677-023-00486-8.

[55] K. Garimella and K. Dutta, "Blockchain architecture for the healthcare ecosystem," in *Blockchain in Healthcare: Analysis, Design and Implementation*, Springer, 2023, pp. 19–46. DOI: 10.1007/978-3-031-45339-7_2.

[56] D. Gatherer, "The 2014 ebola virus disease outbreak in west africa," *Journal of general virology*, vol. 95, no. 8, pp. 1619–1624, 2014. DOI: 10.1099/vir.0.067199-0.

[57] J. Gera, A. R. Palakayala, V. K. K. Rejeti, and T. Anusha, “Blockchain technology for fraudulent practices in insurance claim process,” in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, 2020, pp. 1068–1075. DOI: 10.1109/ICCES48766.2020.9138012.

[58] B. C. Ghosh, V. Ramakrishna, C. Govindarajan, D. Behl, D. Karunamoorthy, E. Abebe, and S. Chakraborty, “Decentralized cross-network identity management for blockchain interoperation,” in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021. DOI: 10.1109/ICBC51069.2021.9461064.

[59] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “Fastfabric: Scaling hyperledger fabric to 20 000 transactions per second,” *International Journal of Network Management*, vol. 30, no. 5, e2099, 2020. DOI: 10.1109/BLOC.2019.8751452.

[60] Governement of Denmark, *Corona Passport*, https://www.sst.dk/en/english/Vaccination-against-influenza-and-covid-19, 2024.

[61] Government of Israel, *Green Pass*, https://corona.health.gov.il/en/green-pass/, 2024.

[62] A. Goyal, A. Elhence, V. Chamola, and B. Sikdar, “A blockchain and machine learning based framework for efficient health insurance management,” in *Proceedings of the 19th ACM conference on embedded networked sensor systems*, 2021, pp. 511–515. DOI: 10.1145/3485730.3493685.

[63] R. Guo, H. Shi, Q. Zhao, and D. Zheng, “Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems,” *IEEE access*, vol. 6, pp. 11 676–11 686, 2018. DOI: 10.1109/ACCESS.2018.2801266.

[64] S. S. Gupta, A. Chattopadhyay, K. Sinha, S. Maitra, and B. P. Sinha, “High-performance hardware implementation for rc4 stream cipher,” *IEEE Transactions on Computers*, vol. 62, no. 4, pp. 730–743, 2012. DOI: 10.1109/TC.2012.19.

[65] G. Gürkaynak, I. Yılmaz, B. Yeşilaltay, and B. Bengi, “Intellectual property law and practice in the blockchain realm,” *Computer law & security review*, vol. 34, no. 4, pp. 847–862, 2018. DOI: 10.1016/j.clsr.2018.05.027.

[66] R. Han, Z. Yan, X. Liang, and L. T. Yang, “How can incentive mechanisms and blockchain benefit with each other? a survey,” *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38, 2022. DOI: 10.1145/3539604.

[67] A. B. Haque, B. Naqvi, A. N. Islam, and S. Hyrynsalmi, “Towards a gdpr-compliant blockchain-based covid vaccination passport,” *Applied Sciences*, vol. 11, no. 13, p. 6132, 2021. DOI: 10.3390/app11136132.

[68] H. R. Hasan, K. Salah, R. Jayaraman, J. Arshad, I. Yaqoob, M. Omar, and S. Ellahham, "Blockchain-based solution for covid-19 digital medical passports and immunity certificates," *Ieee Access*, vol. 8, pp. 222 093–222 108, 2020. DOI: 10.1109/ACCESS.2020.3043350.

[69] J. Hendler, "Web 3.0 emerging," *Computer*, vol. 42, no. 1, pp. 111–113, 2009. DOI: 10.1109/MC.2009.30.

[70] G. Heo, D. Yang, I. Doh, and K. Chae, "Efficient and secure blockchain system for digital content trading," *IEEE Access*, vol. 9, pp. 77 438–77 450, 2021. DOI: 10.1109/ACCESS.2021.3082215.

[71] HIPAA, *Healthcare Data Breach Statistics*, https://www.hipaajournal.com/healthcare-data-breach-statistics/, 2022.

[72] HL7, *Health Level Seven International*, https://www.hl7.org/, 2024.

[73] HT Correspondent, *Covishield may not be eligible for 'vaccine passport' by the EU*, http://tinyurl.com/y8sja52s, 2021.

[74] IBM, *What is a vaccine passport?* https://www.ibm.com/topics/vaccine-passport, 2021.

[75] International Air Transport Association, *IATA Travel Pass Initiative*, https://www.iata.org/en/programs/passenger/, 2024.

[76] L. Ismail and S. Zeadally, "Healthcare insurance frauds: Taxonomy and blockchain-based detection framework (block-hi)," *IT professional*, vol. 23, no. 4, pp. 36–43, 2021. DOI: 10.1109/MITP.2021.3071534.

[77] U. Jafar, M. J. A. Aziz, and Z. Shukur, "Blockchain for electronic voting system—review and open research challenges," *Sensors*, vol. 21, no. 17, p. 5874, 2021. DOI: 10.3390/s21175874.

[78] S. K. Jena, B. Kumar, B. Mohanty, A. Singhal, and R. C. Barik, "An advanced blockchain-based hyperledger fabric solution for tracing fraudulent claims in the healthcare industry," *Decision Analytics Journal*, p. 100 411, 2024. DOI: 10.1016/j.dajour.2024.100411.

[79] S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma, and J. He, "Blochie: A blockchain-based platform for healthcare information exchange," in *2018 ieee international conference on smart computing (smartcomp)*, IEEE, 2018, pp. 49–56. DOI: 10.1109/SMARTCOMP.2018.00073.

[80] S. Jing, X. Zheng, and Z. Chen, "Review and investigation of merkle tree's technical principles and related application fields," in *2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA)*, 2021, pp. 86–90. DOI: 10.1109/CAIBDA53561.2021.00026.

[81] J. Katz, *Digital signatures*. Springer, 2010, vol. 1.

[82] J. Katz and Y. Lindell, "Introduction to modern cryptography," *Book*, 2020.

[83] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-peer Networking and Applications*, vol. 14, no. 5, pp. 2901–2925, 2021. DOI: 10.1007/s12083-021-01127-0.

[84] V. Kohli, S. Chakravarty, V. Chamola, K. S. Sangwan, and S. Zeadally, "An analysis of energy consumption and carbon footprints of cryptocurrencies and possible solutions," *Digital Communications and Networks*, vol. 9, no. 1, pp. 79–89, 2023, ISSN: 2352-8648. DOI: 10.1016/j.dcan.2022.06.017.

[85] A. Koyama, V. C. Tran, M. Fujimoto, V. N. Q. Bao, and T. H. Tran, "A decentralized covid-19 vaccine tracking system using blockchain technology," *Cryptography*, vol. 7, no. 1, p. 13, 2023. DOI: 10.3390/cryptography7010013.

[86] G. Kumar, R. Saha, C. Lal, and M. Conti, "Internet-of-forensic (iof): A blockchain based digital forensics framework for iot applications," *Future Generation Computer Systems*, vol. 120, pp. 13–25, 2021. DOI: 10.1016/j.future.2021.02.016.

[87] R. Kumar and R. Tripathi, "Towards design and implementation of security and privacy framework for internet of medical things (iomt) by leveraging blockchain and ipfs technology," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 7916–7955, 2021. DOI: 10.1007/s11227-020-03570-x.

[88] O. Kuzmina, M. Konovalova, and T. Stepanova, "The global practice of implementation and use of digital currencies of central banks," in *International Conference Ecosystems Without Borders*, Springer, 2023, pp. 206–222. DOI: 10.1007/978-3-031-34329-2_21.

[89] F. Lamberti, V. Gatteschi, C. Demartini, M. Pelissier, A. Gomez, and V. Santamaria, "Blockchains can work for car insurance: Using smart contracts and sensors to provide on-demand coverage," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 72–81, 2018. DOI: 10.1109/MCE.2018.2816247.

[90] W.-M. Lee, "Using the metamask chrome extension," *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript*, pp. 93–126, 2019. DOI: 10.1007/978-1-4842-9271-6.

[91] H. Li, L. Zhu, M. Shen, F. Gao, X. Tao, and S. Liu, "Blockchain-based data preservation system for medical data," *Journal of Medical Systems*, vol. 42, Aug. 2018. DOI: 10.1007/s10916-018-0997-3.

[92] H. Li, D. Han, and M. Tang, "Logisticschain: A blockchain-based secure storage scheme for logistics data," *Mobile Information Systems*, vol. 2021, no. 1, p. 8 840 399, 2021. DOI: 10.1155/2021/8840399.

[93] L. Li, M. Sy, and A. McMurray, "Blockchain innovation and its impact on business banking operations," in *Data Intensive Computing Applications for Big Data*, IOS Press, 2018, pp. 583–598. DOI: 10.3233/978-1-61499-814-3-583.

[94] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6564–6574, 2020. DOI: 10.1109/TII.2020.2974537.

[95] M. Li, C. Lal, M. Conti, and D. Hu, "Lechain: A blockchain-based lawful evidence management scheme for digital forensics," *Future Generation Computer Systems*, vol. 115, pp. 406–420, 2021. DOI: 10.1016/j.future.2020.09.038.

[96] X. Liang, M. Barua, R. Lu, X. Lin, and X. S. Shen, "Healthshare: Achieving secure and privacy-preserving health information sharing through health social networks," *Computer Communications*, vol. 35, no. 15, pp. 1910–1920, 2012. DOI: 10.1016/j.comcom.2012.01.009.

[97] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," in *2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*, IEEE, 2017, pp. 1–5. DOI: 10.1109/PIMRC.2017.8292361.

[98] D. Madala, M. P. Jhanwar, and A. Chattopadhyay, "Certificate transparency using blockchain," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2018, pp. 71–80. DOI: 10.1109/ICDMW.2018.00018.

[99] G. Malavolta, P. A. Moreno-Sánchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," *Proceedings 2019 Network and Distributed System Security Symposium*, 2019. DOI: 10.14722/ndss.2019.23330.

[100] N. Maleki, B. Padmanabhan, and K. Dutta, "The effect of monetary incentives on health care social media content: Study based on topic modeling and sentiment analysis," *Journal of Medical Internet Research*, vol. 25, e44307, 2023. DOI: 10.2196/44307.

[101] S. Malik, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "Trustchain: Trust management in blockchain and iot supported supply chains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 184–193. DOI: 10.1109/Blockchain.2019.00032.

[102] A. Manzoor, M. Liyanage, A. Braeke, S. S. Kanhere, and M. Ylianttila, "Blockchain based proxy re-encryption scheme for secure iot data sharing," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 99–103. DOI: 10.1109/BLOC.2019.8751336.

[103] F. Masood and A. R. Faridi, "Developing a novel blockchain-based vaccine tracking and certificate system: An end-to-end approach," *Peer-to-Peer Networking and Applications*, pp. 1–19, 2024. DOI: 10.1007/s12083-024-01662-6.

[104] P. McCorry, M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "On secure e-voting over blockchain," *Digital Threats: Research and Practice (DTRAP)*, vol. 2, no. 4, pp. 1–13, 2021. DOI: 10.1145/3461461.

[105] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Information Security and Privacy*, Springer International Publishing, 2016, pp. 57–76. DOI: 10.1007/978-3-319-40253-6_4.

[106] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Financial Cryptography and Data Security: 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers 21*, Springer, 2017, pp. 357–375. DOI: 10.1007/978-3-319-70972-7_20.

[107] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*, IEEE, 2018, pp. 1–4. DOI: 10.1109/ICCCNT.2018.8494045.

[108] Nabeel's Blog, *Proxy Re-encryption*, https://mohamednabeel.blogspot.com/2011/03/proxy-re-encryption.html, 2011.

[109] S. S. Nabil, M. S. A. Pran, A. A. Al Haque, N. R. Chakraborty, M. J. M. Chowdhury, and M. S. Ferdous, "Blockchain-based covid vaccination registration and monitoring," *Blockchain: Research and Applications*, vol. 3, no. 4, p. 100 092, 2022. DOI: 10.1016/j.bcra.2022.100092.

[110] S. Nakamoto, "Re: Bitcoin p2p e-cash paper," *The Cryptography Mailing List*, 2008.

[111] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction.* Princeton: Princeton University Press, 2016.

[112] I. Nath, "Data exchange platform to fight insurance fraud on blockchain," in *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, IEEE Computer Society, 2016, pp. 821–825. DOI: 10.1109/ICDMW.2016.0121.

[113] T. Nie and T. Zhang, "A study of des and blowfish encryption algorithm," in *Tencon 2009-2009 IEEE Region 10 Conference*, IEEE, 2009, pp. 1–4. DOI: 10.1109/TENCON.2009.5396115.

[114] A. Nigmatov, A. Pradeep, and N. Musulmonova, "Blockchain technology in improving transparency and efficiency in government operations," in *2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2023, pp. 01–06. DOI: 10.1109/ECAI58194.2023.10194154.

[115] C. Oham, R. Jurdak, S. S. Kanhere, A. Dorri, and S. Jha, “B-fica: Blockchain based framework for auto-insurance claim and adjudication,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, 2018, pp. 1171–1180. DOI: 10.1109/Cybermatics_2018.2018.00210.

[116] C. Oham, R. A. Michelin, R. Jurdak, S. S. Kanhere, and S. Jha, “Wide: A witness-based data priority mechanism for vehicular forensics,” *Blockchain: Research and Applications*, vol. 3, no. 2, p. 100 050, 2022. DOI: 10.1016/j.bcra.2021.100050.

[117] Y. J. Park, J. Farooq, J. Cho, N. Sadanandan, B. Cozene, B. Gonzales-Portillo, M. Saft, M. C. Borlongan, M. C. Borlongan, R. D. Shytle, *et al.*, “Fighting the war against covid-19 via cell-based regenerative medicine: Lessons learned from 1918 spanish flu and other previous pandemics,” *Stem cell reviews and reports*, vol. 17, pp. 9–32, 2021. DOI: 10.1007/s12015-020-10026-5.

[118] M. Patel, A. Dennis, C. Flutter, and Z. Khan, “Pandemic (h1n1) 2009 influenza,” *British journal of anaesthesia*, vol. 104, no. 2, pp. 128–142, 2010. DOI: 10.1093/bja/aep375.

[119] R. Pericàs-Gornals, M. Mut-Puigserver, and M. M. Payeras-Capellà, “Highly private blockchain-based management system for digital covid-19 certificates,” *International Journal of Information Security*, vol. 21, no. 5, pp. 1069–1090, 2022. DOI: 10.1007/s10207-022-00598-3.

[120] N. A. Popova and N. G. Butakova, “Research of a possibility of using blockchain technology without tokens to protect banking transactions,” in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, IEEE, 2019, pp. 1764–1768. DOI: 10.1109/EIConRus.2019.8657279.

[121] F. Pub, “Data encryption standard (des),” *FIPS PUB*, pp. 46–3, 1999.

[122] Quantum Blockchains Inc., *Quantum Blockchains*, https://www.quantumblockchains.io/, 2022.

[123] S. Ragan, *Excellus BCBS discloses breach, 10 million members affected*, https://bit.ly/Excellus_Data_Breach, 2015.

[124] M. Raikwar, S. Mazumdar, S. Ruj, S. S. Gupta, A. Chattopadhyay, and K.-Y. Lam, “A blockchain framework for insurance processes,” in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, IEEE, 2018, pp. 1–4. DOI: 10.1109/NTMS.2018.8328731.

[125] S. Rao, “Health insurance: Concepts, issues and challenges,” *Economic and Political Weekly*, pp. 3835–3844, 2004. [Online]. Available: http://www.jstor.org/stable/4415449.

[126] M. M. Rashid, P. Choi, S.-H. Lee, and K.-R. Kwon, "Block-hpct: Blockchain enabled digital health passports and contact tracing of infectious diseases like covid-19," *Sensors*, vol. 22, no. 11, p. 4256, 2022. DOI: 10.3390/s22114256.

[127] Razorpay, *What is Tokenisation? Types, and How Does It Work*, https://razorpay.com/blog/tokenisation-and-its-impact-on-online-payments/, 2024.

[128] RBI, *FAQs on Device based Tokenisation – Card Transactions*, https://www.rbi.org.in/commonperson/English/Scripts/FAQs.aspx?Id=2917, 2023.

[129] M. Rodriguez-Garcia, M.-A. Sicilia, and J. M. Dodero, "A privacy-preserving design for sharing demand-driven patient datasets over permissioned blockchains and p2p secure transfer," *PeerJ Computer Science*, vol. 7, e568, 2021. DOI: 10.7717/peerj-cs.568.

[130] R. Roy and K. T. George, "Detecting insurance claims fraud using machine learning techniques," in *2017 international conference on circuit, power and computing technologies (ICCPCT)*, IEEE, 2017, pp. 1–6. DOI: 10.1109/ICCPCT.2017.8074258.

[131] R. Rudman and R. Bruwer, "Defining web 3.0: Opportunities and challenges," *The Electronic Library*, vol. 34, no. 1, pp. 132–154, 2016. DOI: 10.1108/EL-08-2014-0140.

[132] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Computer Security - ESORICS 2014*, Springer International Publishing, 2014, pp. 345–364. DOI: 10.1007/978-3-319-11212-1_20.

[133] W. El-Samad, M. Atieh, and M. Adda, "Transforming health insurance claims adjudication with blockchain-based solutions," *Procedia Computer Science*, vol. 224, pp. 147–154, 2023. DOI: 10.1016/j.procs.2023.09.022.

[134] D. Saveetha and G. Maragatham, "A decentralized blockchain based system for secure health record and claims processing," in *2022 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2022, pp. 1–8. DOI: 10.1109/ICCCI54379.2022.9740838.

[135] A. Savelyev, "Contract law 2.0:'smart'contracts as the beginning of the end of classic contract law," *Information & communications technology law*, vol. 26, no. 2, pp. 116–134, 2017. DOI: 10.1080/13600834.2017.1301036.

[136] J. Sengupta, S. Ruj, and S. D. Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot," *Journal of network and computer applications*, vol. 149, p. 102 481, 2020. DOI: 10.1016/j.jnca.2019.102481.

[137] J. Sengupta, S. Ruj, and S. D. Bit, “Fairshare: Blockchain enabled fair, accountable and secure data sharing for industrial iot,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2929–2941, 2023. DOI: 10.1109/TNSM.2023.3239832.

[138] I. P. S. Setiawan and A. Alamsyah, “Enhancing security, privacy, and traceability in indonesia’s national health insurance claims process using blockchain technology,” in *2023 International Conference on Artificial Intelligence, Blockchain, Cloud Computing, and Data Analytics (ICoABCD)*, IEEE, 2023, pp. 77–82. DOI: 10.1109/ICoABCD59879.2023.10390967.

[139] A. Shahani, *Premera Blue Cross Cyberattack Exposed Millions Of Customer Records*, https://bit.ly/Premera_Blue_Cross_Cyberattack, 2015.

[140] R. N. Shaikh, C. G. Jadhav, V. R. Bhogawade, G. Narang, and A. M. Gangadhar, “Block chain based electronic vaccination record storing system,” in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, IEEE, vol. 1, 2022, pp. 272–276. DOI: 10.1109/ICACCS54159.2022.9785292.

[141] M. Shand and J. Vuillemin, “Fast implementations of rsa cryptography,” in *Proceedings of IEEE 11th Symposium on Computer Arithmetic*, IEEE, 1993, pp. 252–259. DOI: 10.1109/ARITH.1993.378085.

[142] A. Sharma and M. Damle, “Blockchain technology: Reinventing the security and efficiency posture of the indian banking system,” in *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, IEEE, 2022, pp. 364–369. DOI: 10.1109/IIHC55949.2022.10060224.

[143] D.-H. Shih, P.-L. Shih, T.-W. Wu, S.-H. Liang, and M.-H. Shih, “An international federal hyperledger fabric verification framework for digital covid-19 vaccine passport,” in *Healthcare*, MDPI, vol. 10, 2022, p. 1950. DOI: 10.1109/IIHC55949.2022.10060224.

[144] D. S. Sippy, *Fraudulent Health Insurance Claims*, https://www.medindia.net/patients/insurance/fraudulent-health-insurance-claims.html, 2013.

[145] N. Sivaselvan, K. V. Bhat, M. Rajarajan, and A. K. Das, “A new scalable and secure access control scheme using blockchain technology for iot,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2957–2974, 2023. DOI: 10.1109/TNSM.2023.3246120.

[146] N. P. Smart *et al.*, *Cryptography: an introduction*. McGraw-Hill New York, 2003, vol. 3.

[147] A. Spielman, “Blockchain: Digitally rebuilding the real estate industry,” Ph.D. dissertation, Massachusetts Institute of Technology, 2016. [Online]. Available: http://hdl.handle.net/1721.1/106753.

[148] N. P. V. Sravan, P. K. Baruah, S. S. Mudigonda, *et al.*, "Use of blockchain technology in integrating heath insurance company and hospital," *Int J Sci Eng Res*, vol. 9, no. 10, pp. 1664–1669, 2018.

[149] W. Stallings, *Cryptography and Network Security: Principles and Practice* (The William Stallings books on computer and data communications technology). Prentice Hall, 1999, ISBN: 9780138690175.

[150] D. R. Stinson and M. Paterson, "Cryptography: Theory and practice," *Book*, 2018.

[151] L. Stockburger, G. Kokosioulis, *et al.*, "Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation," *Blockchain: Research and Applications*, vol. 2, no. 2, p. 100 014, 2021. DOI: 10.1016/j.bcra.2021.100014.

[152] Y. Sun, R. Zhang, X. Wang, K. Gao, and L. Liu, "A decentralizing attribute-based signature for healthcare blockchain," in *2018 27th International conference on computer communication and networks (ICCCN)*, IEEE, 2018, pp. 1–9. DOI: 10.1109/ICCCN.2018.8487349.

[153] M. Szydlo, "Merkle tree traversal in log space and time," in *Eurocrypt*, Springer, vol. 3027, 2004, pp. 541–554. DOI: 10.1007/978-3-540-24676-3_32.

[154] S. Tahir, H. Tahir, A. Sajjad, M. Rajarajan, and F. Khan, "Privacy-preserving covid-19 contact tracing using blockchain," *Journal of Communications and Networks*, vol. 23, no. 5, pp. 360–373, 2021. DOI: 10.23919/JCN.2021.000031.

[155] A. K. Talukder, M. Chaitanya, D. Arnold, and K. Sakurai, "Proof of disease: A blockchain consensus protocol for accurate medical decisions and reducing the disease burden," in *2018 IEEE SmartWorld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, IEEE, 2018, pp. 257–262. DOI: 10.1109/SmartWorld.2018.00079.

[156] T. M. Tan and S. Saraniemi, "Trust in blockchain-enabled exchanges: Future directions in blockchain marketing," *Journal of the Academy of marketing Science*, vol. 51, no. 4, pp. 914–939, 2023. DOI: 10.1007/s11747-022-00889-0.

[157] The Commons Project Foundation, *CommonPass*, https://www.thecommonsproject.org/.

[158] M. Thenmozhi, R. Dhanalakshmi, S. Geetha, and R. Valli, "Implementing blockchain technologies for health insurance claim processing in hospitals," *Materials Today: Proceedings*, 2021. DOI: 10.1016/j.matpr.2021.02.776.

[159] Times of India, *WazirX probe hits wall as suspect flees to B'desh*, https://timesofindia.indiatimes.com/city/delhi/wazirx-probe-hits-wall-as-suspect-flees-to-bdesh/articleshow/116575615.cms, 2024.

[160] Y. Tsiounis and M. Yung, "On the security of elgamal based encryption," in *International Workshop on Public Key Cryptography*, Springer, 1998, pp. 117–134. DOI: 10.1007/BFb0054019.

[161] Vaccination Credential Initiative, *VCI*, https://vci.org/, 2022.

[162] R. M. Visconti, "Microfinance vs. traditional banking in developing countries," *International Journal of Financial Innovation in Banking*, vol. 1, no. 1-2, pp. 43–61, 2016. DOI: 10.1504/IJFIB.2016.076613.

[163] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017. DOI: 10.1007/978-3-319-57959-7.

[164] M. Wang, Y. Guo, C. Zhang, C. Wang, H. Huang, and X. Jia, "Medshare: A privacy-preserving medical data sharing system by using blockchain," *IEEE Transactions on Services Computing*, 2021. DOI: 10.1109/TSC.2021.3114719.

[165] R. Wang, B. Wu, and T. Xia, "A blockchain-based multiple-parties-involved vaccination passport system," in *2022 3rd International Conference on E-commerce and Internet Technology (ECIT 2022)*, Atlantis Press, 2022, pp. 772–785. DOI: 10.2991/978-94-6463-005-3_78.

[166] Z. Wang and S. Guan, "A blockchain-based traceable and secure data-sharing scheme," *PeerJ Computer Science*, vol. 9, e1337, 2023. DOI: 10.7717/peerj-cs.1337.

[167] Wikipedia, *2017 Equifax data breach*, https://en.wikipedia.org/wiki/2017_Equifax_data_breach, 2024.

[168] Wikipedia, *Anthem Medical Data Breach*, https://en.wikipedia.org/wiki/Anthem_medical_data_breach, 2015.

[169] Wikipedia, *Bangladesh Bank robbery*, https://en.wikipedia.org/wiki/Bangladesh_Bank_robbery, 2024.

[170] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[171] B. Wu and T. Duan, "The advantages of blockchain technology in commercial bank operation and management," in *Proceedings of the 2019 4th International Conference on Machine Learning Technologies*, 2019, pp. 83–87. DOI: 10.1145/3340997.3341009.

[172] Q. Xia, E. Sifah, K. Asamoah, J. Gao, X. Du, and M. Guizani, "Medshare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. PP, pp. 1–1, Jul. 2017. DOI: 10.1109/ACCESS.2017.2730843.

[173] Y. Xiao, Y. Liu, Y. Wu, T. Li, X. Xian, and W. Jiang, "Healthchain: A blockchain for electronic health records (preprint)," *Journal of Medical Internet Research*, vol. 23, Jan. 2019. DOI: 10.2196/13556.

[174] G. Xu, C. Qi, W. Dong, L. Gong, S. Liu, S. Chen, J. Liu, and X. Zheng, "A privacy-preserving medical data sharing scheme based on blockchain," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 2, pp. 698–709, 2022. DOI: 10.1109/JBHI.2022.3203577.

[175] R. Xu, Y. Chen, *et al.*, "Exploration of blockchain-enabled decentralized capability-based access control strategy for space situation awareness," *Optical Engineering*, vol. 58, no. 4, pp. 041 609–041 609, 2019. DOI: 10.1117/1.OE.58.4.041609.

[176] G. Yang and C. Li, "A design of blockchain-based architecture for the security of electronic health record (ehr) systems," in *2018 IEEE International conference on cloud computing technology and science (CloudCom)*, IEEE, 2018, pp. 261–265. DOI: 10.1109/CloudCom2018.2018.00058.

[177] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, pp. 1–8, 2016. DOI: 10.1007/s10916-016-0574-6.

[178] H. Zhang and K. Sakurai, "Blockchain for iot-based digital supply chain: A survey," in *Advances in Internet, Data and Web Technologies: The 8th International Conference on Emerging Internet, Data and Web Technologies (EIDWT-2020)*, Springer, 2020, pp. 564–573. DOI: 10.1007/978-3-030-39746-3_57.

[179] J. Zhang, N. Xue, and X. Huang, "A secure system for pervasive social network-based healthcare," *IEEE Access*, vol. 4, pp. 9239–9250, 2016. DOI: 10.1109/ACCESS.2016.2645904.

[180] X. Zhang and S. Poslad, "Blockchain support for flexible queries with granular access control to electronic medical records (emr)," in *2018 IEEE International conference on communications (ICC)*, IEEE, 2018, pp. 1–6. DOI: 10.1109/ICC.2018.8422883.

[181] L. Zhou, L. Wang, and Y. Sun, "Mistore: A blockchain-based medical insurance storage system," *Journal of medical systems*, vol. 42, no. 8, pp. 1–17, 2018. DOI: 10.1007/s10916-018-0996-4.