Thesis Paper

Advanced Cryptographic Techniques for Privacy in Blockchain Smart Contracts.

Submitted by:

Md. Sifatullah

ID: 201-35-617

Batch: 31

Department of Software Engineering

Supervised by:

Mr. Esraq Humayun

Lecturer

Department of Software Engineering

Daffodil International University

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Software Engineering

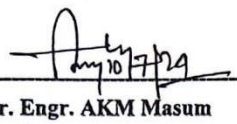
Spring 2024

Approval

APPROVAL

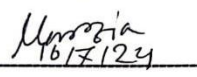
This thesis titled on **"Advanced Cryptographic Techniques for Privacy in Blockchain Smart Contracts."**, submitted by **Md. Sifatullah (ID: 201-35-617)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS



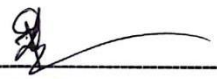
Chairman

Dr. Engr. AKM Masum
Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



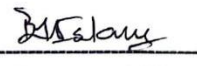
Internal Examiner 1

Dr. Marzia Ahmed
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 2

Md. Rajib Mia
Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



External Examiner

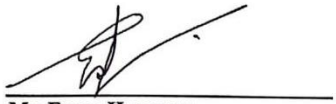
Dr. Md. Manowarul Islam
Associate Professor
Dept. of Computer Science and Engineering
Jagannath University

Declaration

DECLARATION

I hereby declare that, this thesis report is done by me under the supervision of Mr. Esraq Humayun, Lecturer, Department of Software Engineering, Daffodil International University, in partial fulfillment my original work. I am also declaring that neither this thesis nor any part therefore has been submitted else here for the award of Bachelor or any degree.

Supervised by:



Mr. Esraq Humayun

Lecturer

Department of Software Engineering

Daffodil International University

Submitted by:

Md. Sifatullah

ID: 201-35-617

Department of Software Engineering

Daffodil International University

TABLE OF CONTENT

Chapter 1

Introduction

1.1 Motivation

In Germany and the Western world is where patients visit private or public health facilities when ill. Doctors may prescribe medication, which patients purchase from pharmacies. Prescriptions are written on special paper, stamped with a unique stamp, and verified by pharmacists. Patients must keep track of their prescriptions, and the doctor must ensure no harmful interactions with other medications.

Iatrogenic illness refers to conditions arising from medical treatment, including medication errors and adverse drug events. Medication errors are a main reason for harmful things in hospitalized patients. Prescription errors are the most common preventable medication errors.

Statistics on prescription errors: A study found 1111 prescribing errors in 17,808 medications in one week. Errors were classified as "likely to cause need for monitoring" (214).

Modernization with e-prescription: In the USA, April 2014, there are about 90% of pharmacies that accept electronic prescriptions where 70% of doctors were using them. E-prescription usage increased significantly from 2008 to 2014.

The E-Health Act in Germany, adopted in January 2015, mandates electronic medication plans by 2018 for patients on three or more medications. It aims at reducing medication interactions. A 2015 study found that only 6.5% of medication plans were error-free.

1.2 Purpose of research questions

The need for a unified and error-resistant system: Increased digital connectivity introduces new vulnerabilities like identity theft and hacking. Systems require immutability, identity verification, and redundancy. Standard databases partially meet these needs; alternative technologies should be considered.

Blockchain technology was introduced with Bitcoin in 2008, offering an unbending, cryptographically synchronized secured of a distributed database management system. Blockchain applications have extended beyond cryptocurrency, especially in the financial sector. Zyskind et al. suggest using blockchain with secure multiparty computing for managing sensitive data.

Blockchain in healthcare: Investigated for electronic patient records (EPRs) and other applications. Current solutions either depend on third-party trust or are inefficient due to consensus processes. Van Dijk and Juels argue that cryptography alone is insufficient for privacy-preserving cloud

computing, requiring tamper-proof hardware that distributes the computing of a complex trusted ecosystem.

1.3 Limitations

Research inquiry criteria related to storing prescriptions of the patient's profile. Doctor identities and mainly pharmacy profiles on a system of blockchain. The design of the architecture on blockchain applications to ensure data privacy transfer but untrusted parties. Development of a blockchain application for prescription management with strict access controls.

Here mostly focus on the limitations of smart contracts for operational logic with basic simple permissions due to constraints of time and the discussion of blockchain design issues for potential PoC extensions. Acknowledgement of legal considerations and stakeholder roles, though not extensively covered.

1.4 Related works

There are many advanced technologies. Blockchain is one of the most advanced in this field of technology. We use this technology because this blockchain technology ensures a secure and reliable system to operate. Our government, healthcare, and industrial sectors, especially the banking sector, are widely using this technology. A technological solution that is based on blockchain technology is called ENIGMA. It ensured that the technique could perform multiparty computations simultaneously. In general, they recommend an implementation of blockchains for the administration of permissions and the storage of highly created pointers for the purpose of encrypting the data. While a reliable service is responsible for hosting the real data. The presentation was made by Zyskind and colleagues (2015). Rather than recording financial transactions in plain text, Hawk is a decentralized smart contract system that was developed with the intention of protecting the confidentiality of blockchain-based transactions. It leverages zero-knowledge proofs and other cryptographic techniques to obfuscate transaction details. The system allows programmers to write private contracts easily with a compiler that generates secure cryptographic protocols automatically. The authors also introduced a formal blockchain model of cryptography to rigorously define and reason about security. Hawk addresses privacy and usability challenges, enhancing secure decentralized applications (Kosba, Miller, Shi, Wen, & Papamanthou, 2015).

CHAPTER 2

2.1 Cryptographic Foundations:

Cryptography involves techniques to transform data to prevent unintended recipients from accessing or altering it. It addresses confidentiality (preventing unauthorized access), integrity (ensuring data hasn't been tampered with), and non-repudiation (ensuring a sender cannot deny sending data).

Types of Encryption:

Unkeyed encryption involves functions like hashing and permutations that do not require a key. These are used primarily for creating fixed-size representations of data (hashes) or rearranging data (permutations).

Symmetric Key Encryption: Always using the right key for both encrypted and decrypted data. This meant the sending and receiving have to the same secret, which is the key—to communicate securely. Examples include AES (Advanced Encryption Standard) algorithms.

Asymmetric-Key Encrypt: Using the same pair of keys.

1. A public key for encrypt and a private key for decrypt. The key is mathematically related. While they cannot be derived from each other. This enables secure communication without both parties needing to share a secret key beforehand. Examples include RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography).

2.1.1 Key Concepts and Terminology

Consider MMM formed from a specified alphabet. This alphabet could be binary ({0,1}\{0, 1\}{0,1}), English, or hexadecimal ({0,1,…,9,A,B,…,F}\{0, 1, \dots, 9, A, B, \dots, F\}{0,1,…,9,A,B,…,F}). Any element m∈Mm \in Mm∈M is referred to as a plaintext message, as it is assumed to be written in plain text and openly visible. Similarly, the ciphertext space is represented by the set CCC, which contains elements from an alphabet that may differ from the alphabet of MMM.

Serves as a bijection between MMM and CCC, uniquely determined by a key e∈Ke \in Ke∈K from the key space. This bijection is denoted by EeE_eEe. Similarly, a decryption function is a bijection from CCC back to MMM, defined by a decryption key d∈Kd \in Kd∈K and is represented as DdD_dDd.

Encryption and decryption processes, we require the encryption set {Ee:e∈K}\{E_e: e \in K\}{Ee :e∈K} and the decryption set {Dd:d∈K}\{D_d: d \in K\}{Dd:d∈K}, such that for every e∈Ke \in Ke∈K, there exists a corresponding unique d∈Kd \in Kd∈K, where Dd=Ee−1D_d = E_e^{-1}Dd =Ee−1. This ensures that the decryption of a message can be written as:

quad \forall m \in M.Dd(Ee(m))=m∀m∈M.

In other words, applying the decryption function to the encrypted message recovers the original plaintext.

It is important to note that the key pair {e,d}\{e, d\}{e,d} may either consist of the same key or different keys. The sets MMM, CCC, KKK, and {Ee:e∈K}\{E_e: e \in K\}{Ee:e∈K} are considered public information, {e, d\}{e,d}.

2.1.2 Private and Public Key Cryptography and Hash Functions

Within the scope of this thesis, hashing is a crucial primitive that always plays a magnificent part, comprehending blockchains and privacy. The design of hash functions transforms any length of fixed length of binary string in a one way manner. We refer to this function as a one-way function because we can easily calculate it with minimal computational resources, but retrieving the inverse of the function requires an immense, practically impossible effort. Output length of the hash-function already is an important characteristic, typically set at either 256 or 512 bits. It is crucial because a longer hash increases the number of potential outputs. One important aspect of a hash function is its ability to minimize or eliminate collisions, where different inputs produce that the same output which is, h(x) = h(y). It is crucial for the hash-function to be deterministic completely, meaning it consistently produces same output for a given input. People widely regard SHA-3 as the most secure hash function due to its extreme difficulty in reversing or tampering with the contents. This hash function meets the requirements of an optimal hashing-function. It also resistant to length extension attacks. Discovery of collisions for SHA-3's predecessor, SHA1, occurred in late February 2017. This implies that one could manipulate a PDF to generate a specific hash value. It required a significant number of computations, with a combination of CPU and GPU computation time spanning over several millennia. We achieved the attack proof by using an identical-prefix collision attack,. In a stStevens, Bursztein, Karpman, Albertini, and Markov (2017) conducted a study. Most password verification systems commonly use hashes. The database compares the password input to a stored hash, ensuring secure storage without storing passwords in plain text. We will explain the concept of proof-of-work in section /refconsensus, which requires an understanding of hash functions. Let's dive into symmetric-key cryptography by looking at a scenario where two parties, Alice or Bob, want to reach over a channel that is not secure (like a public chat or a loudhailer). Simply hashing and sending messages is insufficient, as it is extremely hard to determine the original message from its hash. Firstly, they are agreed on a scheme of encryption that includes a message space or key space, encryption and decryption

functions, and ciphertext space. When it comes to private-key cryptography, they establish shared keys in a such way that De(Ee(m)) = m. After encrypting her plaintext message, Alice sends the encrypted message to Bob. Bob proceeds to decrypt the message, resulting in the original plaintext. There seems to be an issue with the initial agreement for the shared private key. Despite the common belief that the encryption scheme is well-known or easily discoverable, it's important to acknowledge that the number of possible encryption transformations is limited. This necessitates Alice and Bob establishing communication through a secure channel, which can often be impractical. In 1976, researchers resolved the key distribution problem, widely regarded as a pivotal moment in the evolution of modern types of cryptography.

The solution to this problem of distribution formed the foundation of the asymmetric key of cryptography. Ralph C. Merkle was the first to suggest sharing a key across an unsecure channel (1978, Merkle). when participants of two Alice and Bob want to share their conversation:

1. Bob and Alice decide on the transformation of encryption.
2. Each of them generates two keys, pkA, skA, and pkB, skB.
3. The public has access to pkA and pkB, but SkA and SkB remain confidential.
4. Alice uses the agreed-upon encryption algorithm and public key to encrypt Bob.
5. Bob receives a message from Alice via any medium.
6. Using his secret key, Bob decrypts DskB (c) = m.

2.1.3 Digital Signature Mechanisms

The purpose of Digital signatures are used to achieve authentication and non-repudiation. They ensure that a unique human or machine has sent a specific message. This is analogous to a handwritten signature or a special seal used in traditional letters. Where the functionality of a digital The signature verifies the message, prevents the sender from denying they sent it, and ensures authenticity verifies the sender's identity with greater certainty than a handwritten signature.

Types of Digital Signature Algorithms:

Signature with Message Recovery: These algorithms do not need the original message for verification, as it can be recovered directly from the signature. They are particularly useful for sending short messages.

Signature with Appendix: These algorithms require the original message for verification. They depend on hashing algorithms and are more commonly used due to their lower susceptibility to existential forgery attacks.

The significance of Merkle trees lies in their ability to allow efficient and secure data verification. Verifying that a leaf node is part of a tree is a logarithmic-time computational problem, which is significantly faster than linear-time verification in a list structure. This efficiency is crucial for blockchain technology, where verifying large amounts of data quickly and securely is necessary.

2.2 Blockchain Technology Fundamentals

There is no universally accepted, formal definition of blockchain technology. Most definitions use Bitcoin, the first application of blockchain, as a starting point, but this does not encompass all blockchain systems.

Buterin, the founder of Ethereum, describes blockchain as a "magic computer" where anyone can upload self-executing programs. The system ensures that the current and all previous states of every program are publicly visible and guarantees that programs will execute exactly as defined by the blockchain protocol. This definition captures many characteristics of blockchain systems but is not very rigorous or technical.

ISO/TC 307 records increasing trust through its transparency and removing the need for intermediaries. IBM's Definition ighlighting transparency and security.

Various attempts to classify different blockchain technologies, such as the one by Okada, Yamasaki, and Bracamonte (2017), have been made but do not provide satisfactory definitions. An ISO technical committee (ISO/TC 307) has been formed to define areas for standardization in blockchain and electronic distributed ledger technologies.

2.2.1 Introduction to Bitcoin and Early Blockchain Models

BT was first outlined in the seminal white paper by Satoshi Nakamoto in 2008, which described the construction of Bitcoin, the first cryptocurrency. In electronic money systems, double-spending refers to the risk of using the same digital coin multiple times. Normally addressed by a central authority like a bank. Transactions are recorded chronologically, preventing double-spending. Proof-of-Work (PoW) Algorithm Establishes consensus on the correct blockchain, making very costly fake a transaction than the gain of potentail. Users are incentivized to correctly validate transactions, ensuring the integrity of the blockchain. Without a proper consensus algorithm, trust

in the Bitcoin blockchain would be impossible as anyone could potentially alter the transaction history.

Bitcoin and Blockchain relation:

Initially, there was no distinction between Bitcoin and blockchain since Bitcoin was the first and only application of blockchain technology at the time. Most early applications of blockchain were within cryptocurrencies and financial processes. The advent of Ethereum introduced smart contracts, expanding blockchain applications beyond financial processes to other areas by leveraging the disintermediation of trust. The true identity of Satoshi Nakamoto remains unknown, despite various claims by researchers and cryptocurrency enthusiasts.

2.2.2 Post-Bitcoin blockchains

What is currently known as distributed ledger technology, or blockchain technology, is reconstructed using some of the characteristics of Bitcoin. New blockchains frequently allow for more operations and have more versatile applications, all while preserving the essential features of Bitcoin. The development of this technology continues, regularly releasing new methods and uses. These are typically published as white papers by startups or corporate research teams. The fundamentals of blockchain technology remain unchanged:

• **Distributed:** Every node has a complete copy of the database's history, making them all equal in that regard. Less equal nodes, also known as lightweight nodes, are another possibility. These nodes just have a few of the most recent blocks saved locally. Privatekey cryptography is typically used over the Internet for node-to-node communication.

• **Time-stamped:** The further back in time that the present block is, the harder it is to rewrite history because every transactions hashed into all of the block that come before it. This blockchain turns into a verifiably accurate auditing instrument.

• **Consensus:** Using a consensus technique, nodes determine the single truth regarding which version of the database is accurate. This helps to verify. Design and function of the blockchain have a significant impact on the consensus algorithm type that is being employed.

2.2.3 Specialization and Permissions

As blockchain technology gets advanced beyond Bitcoin the two clear cut options for including participants in network validation and observation emerged. The distinction lies between permissioned and permissionless blockchains, with the possibility of hybrid solutions in certain scenarios. A blockchain that is openly accessible on the internet is referred to as permissionless. Well-known examples blockchains include Bitcoin and Ethereum. As the number of actions allowed increases, so does the potential for hacking the blockchain. This incident occurred during the well-known DAO hack, in which around USD 50 million was illicitly taken from an ether fund. (Buterin, 2016). In order to maintain privacy, it is crucial to ensure that the data on the blockchain remains completely anonymous. This is a challenge that Bitcoin, Vasek, and Moore attempted to address in 2015. Permissioned blockchains were developed to address situations where it may not be possible or desirable to anonymize all data or allow everyone to participate in a network. The fundamental idea behind permissioned blockchains is that network participation and membership are controlled. An alliance of businesses, governmental bodies, or other organizations may accomplish this by setting a predetermined set of criteria or by extending an invitation to new members on a one-to-one basis. Along with the added anonymity, there are other advantages, including speedier transactions, improved scalability, and more flexibility in network adaptation. Permissioned blockchains can sometimes be more susceptible to unintentional modifications to their history, depending on the consensus mechanism used. In other words, immutability and censorship resistance are occasionally traded for speed, privacy, and scalability. According to Mattila (2016), this allows for easier concurrency on a permissioned blockchain, which can use a PoW-consensus algorithm with lower resource requirements.

Additionally, blockchains can be designed to be more or less specific in terms of what behaviors are allowed on them. A highly specialized chain with a single goal is the safe transmission of the cryptocurrency's tokens, as seen with Bitcoin and most other coins. Conversely, Ethereum boasts an integrated virtual machine that allows for the seamless deployment of smart contracts. Decentralised applications, or DApps, are what Ethereum was designed specifically to facilitate. Unlike a specialty blockchain, a generic blockchain is not geared for carrying out a single, unique purpose. Ethereum and Bitcoin are both permissionless; however, there are two options on the permissioned spectrum: the specialist Mul-Tichain platform and the multipurpose eris-db from Monax. A permissions layer, an EVM implementation, and Tendermint consensus by default are features of the blockchain client erisdb, though they can be changed. Using a Proof-of-Validation (PoV) mechanism, Because Multichain is designed to execute transactions, it is specialised, whereas Monax is intended to provide a wide range of services. It simply means that it is made simpler by design. This does not, however, imply that it is impossible to create high-performance payment systems on Monax or bespoke services on Multichain. It employs an algorithm that limits the most votes a miner can cast in a certain period of time rather than using proof-of-work. (“Ethereum Homestead Documentation - What is Ethereum,” 2016).

2.2.4 Ethereum Misnomer and Smart Contract Platforms

The term "smart contracts" is misleading because they are neither intelligent nor traditional contracts. In the context of blockchain, smart contracts are pieces of logic published on the blockchain that can receive or perform transactions like any address and function as immutable agreements. Nick Szabo described smart contracts as computerized transaction protocols that execute the terms of a contract. The idea is to incorporate parts of contracts into software so that breaching them becomes either costly or impossible.

Characteristics (According to Szabo):

Online Enforceability: Ensures that contract terms are fulfilled through proactive measures (making breaches technically impossible or allowing parties to exit on valid breaches) and reactive measures (deterring malicious behavior through reputation or enforcement and recovering assets after breaches).

Verifiability: Contracts should be auditable in case of disputes.

Privity: Ensures that only necessary participants have knowledge and control over the data involved in the contract.

There is a tension between privity (minimizing external access) and visibility. The goal is to optimize the balance, giving minimal information and control to external parties while maintaining the ability to verify, observe, and enforce the contract. Before blockchain technology and advances in cryptographic techniques, Szabo suggested trusting an intermediary, like an auditor, to solve the optimization problem.

Ethereum Platform:

General Blockchain: Ethereum is a general-purpose blockchain with a virtual machine (Ethereum Virtual Machine, EVM) to run smart contracts.

Isolation: Smart contracts on Ethereum are isolated from the network, file system, or other processes on the node machines.

Solidity: A high-level, Turing-complete language created for writing smart contracts on Ethereum. Solidity has become the standard language for other platforms with smart contract capabilities, similar in syntax to JavaScript but written in a different style.

Deployment: Solidity contracts are compiled into EVM bytecode and deployed to specific Ethereum addresses.

2.2.5 Consensus algorithms

Because Bitcoin's goal was to transfer value into an uncontrolled, distrustful environment where a reliable method of authenticating transactions was essential, consensus algorithms are highly relevant to blockchain technology. The purpose of a consensus algorithm is to ensure there is only one transaction history, free of conflicting transactions. For example, it ensures that no account can engage in "double-spending," or using the same token multiple times. Several consensus

algorithms are briefly introduced here; for more details, refer to Back (1997), Nakamoto (2008), Fischer (1983), and Tendermint (2017).

Bitcoin resolved the consensus issue by requiring each new block to meet a target through the addition of a variable nonce, the hash of the previous block, and the current block. Due to the dispersed nature of the hashing function's output, it is impossible to construct a block that easily meets the target. Thus, mining machines across the network compete to find the correct number. Once a block meets the target, users verify the transactions. If enough validating nodes confirm the transactions, they agree to append the block to the chain. This process is known as proof-of-work (PoW). The goal is to prevent any single person or group from gaining too much power, so a limited resource is chosen for voting and approval of blocks. The difficulty of the hashing problem increases with the frequency of previous solutions because processing power is becoming cheaper due to Moore's Law and cloud computing. However, a common criticism of PoW is the significant energy wastage. Some miners only operate in winter, using farms to heat their homes. This centralizes the decentralized network as miners pool resources into a few massive Bitcoin farms. Additionally, Bitcoin's fixed block size (approximately 1 MB) and block duration (about 10 minutes) result in low transaction throughput. Alternatives focus on improving throughput and reducing energy waste. one-third Byzantine validators.

The attacker's fork would succeed with just 1% of the total stake ("Proof of Stake FAQ: Ethereum GitHub Wiki," 2017).

Chapter 3

Implementation

This chapter proposes a blockchain-based smart contract approach for a proof-of-concept electronic medicine plan (EMP). It begins by explaining the three distinct user archetypes and their user stories to provide functional specifications for the proof of concept.Additional details about the Proof of Concept, including its quality attributes and how to integrate it with a blockchain, are provided.

After that, a diagram illustrating the standard exchanges on Blockchain is displayed, with exchanges in two of the smart contracts. Permissioning in smart contracts and the decentralized, unsecured, and immutable characteristics of blockchain technology are used in the proposed solution to the problem as mentioned. It is noteworthy, therefore, that this approach has not eliminated any security or privacy risks that are not related to the blockchain. The conversation includes a few mentions of the more significant off-chain problems.

3.1 Requirements and User Stories

This thesis describes an implementation usable only by three archetypical users: patients, physicians, and pharmacies. User stories were created to outline the different functional requirements for the application, which are presented first. Following that, each user type is explained in further depth. The user stories and requirements were minimized to maintain a workable level of security and usability for the proof of concept.

Patients:

View Prescriptions: Patients want to see their list of prescriptions to choose which medication to take.

Secure Access: They need to identify themselves securely to access their personal data on the **blockchain.**

Communication: Patients want to communicate with doctors and pharmacies about their prescription plans.

Doctors:

View and Modify Prescriptions: Doctors need to view and change their patients' prescriptions to provide proper treatment and prevent drug mistakes.

Confirm Patient Identity: Before making changes, doctors must confirm the patient's identity through their account.

Monitor Prescriptions: Doctors need to view prescriptions given by other doctors to ensure patient safety.

Secure Access: Doctors must authenticate themselves securely to access patient data on the blockchain.

Pharmacies:

Confirm Prescriptions: Pharmacies need to confirm a patient's prescription to ensure they aren't trying to buy unauthorized pharmaceuticals.

Secure Access: Pharmacies must authenticate themselves securely to access patient data on the blockchain.

System Requirements

Privacy:

Prescriptions cannot be linked to the identification of the patients, doctors, or pharmacy in the absence of user permission for non-admin accounts.

Network Access: Only authorized users should able to join the network.

Immutable Traceability:

The system must maintain an immutable record of prescription changes, including:

1. The prescriber of the medication.
2. Whether the medication was sold following the prescription.
3. The location of the sale.

3.2 PoC Design

This section outlines the design, shaped by the requirements and user stories discussed in the previous section. It begins with an explanation of the smart contracts that underpin the PoC logic, followed by a proposal for blockchain implementation, accompanied by an explanation of the features that would be available. From now on, contract and function names will be written in this font to make reading easier.

3.2.1 Design the Overview

This thesis presents a high-level overview of the EMP's design. At the top ("software system level"), the abstraction is higher, whereas at the bottom ("contract level"), it is lower. "Smart Contracts" component concerning the code developed for the Proof of Concept.

1.2.2 Smart Contracts System

The proposed architecture for the smart contract system is built on the concept that different contract types should handle various classes of tasks. the "Five Types Model" is employed to categorize the contracts ("Monax: Solidity Explainer: The Five-Types Model," 2016). The contract categories in the model are:

1. **Database Contracts:** These contracts manage data storage and include fundamental read, write, and get functions. They may also handle permission verification.
2. **Controller Contracts:** One level higher in abstraction, these contracts manage database contracts to execute batch read/.
3. **Managing Contracts:** The contracts regulate and oversee the behavior of other contracts and manage communication between contracts.
4. **Application Logic Contracts:** These contracts use controllers to execute application-specific tasks.

5. **Utility Contracts:** General-purpose tasks can be offloaded to highly specialized utility contracts, such as those performing specific functions or hashing data.

The PoC contracts are:

- **PatientInfoDb:** Backend database contract that stores patient data, including prescriptions and prescribing doctor. Each patient has a Patient-struct with corresponding data and next/prev attributes to allow iteration using a doubly-linked list.
- **PermissionsDb:** A database contract that stores permissions, which can be modified as follows:

  - Patient permissions (e.g., only allowed to read info related to their own address)

  - Pharmacy permissions (e.g., allowed to read info about patients who have permitted it)

  - Doctor permissions (allowed to add patients, add prescriptions, and read patient info)

  - Consent codes for patients to grant specific doctors or pharmacies the right to prescribe or sell medication.

  - Checks for the existence of the patient-prescription tuple when a doctor attempts

3.2.3 Variables On The Blockchain and Data

In accordance with requirement R1, no unencrypted data that could identify a user may be stored on the blockchain. This prevents anyone using the blockchain from viewing an individual's entire prescription history. If the specificity is reduced and the individual becomes unidentifiable. Therefore, it is considered that prescriptions in plaintext are sc.

3.2 .4 Infrastructure and Governance Support

POC relies not just on smart contracts but also necessitates a blockchain infrastructure and a robust key management system. Moreover, consensus must be established, whether through a distributed or centralized mechanism, regarding the eligibility of physicians and pharmacies to participate in the network. For this proof of concept, a proof-of-stake (PoS), Tendermint, or proof-of-authority (PoA) consensus algorithm on a permissioned blockchain is recommended. The use of a PoW ca in a accessible blockchain is discouraged due to its high cost and inefficiency, unless infrastructure costs are manageable. In such cases, leveraging the Ethereum public blockchain might be

considered, despite potential security risks such as the exposure of doctor credentials and increased contract vulnerabilities.

In scenarios where intrinsic tokens hold value, a proof-of-stake algorithm proves more effective, incentivizing validators to approve blocks. Alternatively, premiums paid by health insurance companies could hinge on the level of validation provided by physicians or pharmacists, possibly overseen by governmental bodies. Delegated proof-of-stake mechanisms could also be explored under similar principles. A proof-of-authority consensus model might be feasible if trusted authorities oversee transaction verification, potentially involving regulated entities like secure data centers or audited insurance firms to ensure compliance with EU regulations.

However, these measures somewhat contradict the initiative's goal of eliminating single points of failure and central authorities. While no single reliable third party is involved, collaborative efforts among multiple parties could suffice. Strictly speaking, additional validation beyond the blockchain's permissioning layer and smart contract controls may not be necessary, but establishing an organizational framework such as a decentralized autonomous organization (DAO) to verify prescriptions and onboard users is recommended. Mandating audits of physicians and pharmacists and ensuring auditable actions and recommendations on the blockchain are also crucial steps.

Most nodes on the blockchain would function as non-validating nodes, alongside patients, physicians, and pharmacies, which might need full nodes. Eris-db by Monax, an open-source blockchain platform utilizing the EVM (Ethereum Virtual Machine), is suitable for this project, offering a robust environment despite its imperfections. Integration with IPFS via Eris-Db's permissioning layer.

Before deploying, compatibility checks within a essential. Setting up keys for the first user and configuring the Genesis JSON file are initial steps, requiring secure communication channels for key transmission. Various cloud instances (e.g., Digital Ocean, AWS) could be configured as Tendermint validators for the EMP PoC, enabling lightweight client nodes for patients, physicians, and pharmacies without the need for dedicated hardware.

Ensuring user integrity involves measures to prevent fraud and comply with Know-Your-Customer (KYC) regulations, such as maintaining encrypted records of account registrations and collaborating with state authorities. Transparency regarding registered doctors and pharmacies might mitigate security risks associated with user registrations, promoting trust in the adoption of modern technologies in healthcare systems.

Chapter 4

Conclusion

4.1 Summary of results

The research design for this thesis was structured to create and evaluate a proof-of-concept (PoC) program for a decentralized Blockchain-based electronic medication management system. The research was conducted in several phases:

1. Literature Review and Requirement Analysis:
    1. A comprehensive literature review was conducted to understand the current state of blockchain technology, data storage mechanisms, and security protocols. This phase also involved analyzing existing frameworks and technologies to identify the appropriate criteria for storing sensitive medical data on a blockchain application.
2. System Design and Development:
    1. The design phase involved creating an architecture that would ensure the privacy and security of data while being transferred between parties. The architecture was developed using permissioned blockchain, standard cryptographic tools, and smart contracts written in Solidity.
    2. A detailed design of the application's components was outlined, including the patient profiles, doctor identities, pharmacy profiles, and prescription storage. Implementation:
    3. The implementation phase involved coding the application using Solidity for smart contracts. The focus was on developing a robust system that could handle the requirements identified during the literature review.
    4. The PoC was tested to ensure it met all functional requirements. This included ensuring that only authorized parties could access and modify data, and that the data remained private and secure.
3. Evaluation:
    1. The final product was assessed post-ex in accordance with accepted design research standards. The evaluation criteria focused on the security, functionality, and usability of the application.
    2. While the PoC met all evaluation requirements, it is important to note that the security claims are limited to the PoC and do not extend to systems outside this scope.

The literature review aimed to address three primary research objectives related to the storage, security, and access control of medical data on a blockchain application. To determine the appropriate criteria for storing patient profiles, doctors identities and pharmacy profiles within a blockchain application.

Blockchain Storage Mechanisms: Analysis of both on-chain and off-chain data storage methods, considering factors like scalability, security, and accessibility.

Data Security and Privacy: Examination of cryptographic techniques and access control mechanisms to protect sensitive data.

Existing Frameworks and Technologies: Review of current blockchain platforms and applications used in the healthcare sector.

Blockchain Architecture: Analysis of various blockchain models (e.g., public vs. private, permissioned vs. permissionless) to identify the best fit for the application.

Smart Contracts: Exploration of the role of smart contracts in automating and securing transactions.

Security Assessment: Review of security vulnerabilities and mitigation strategies relevant to blockchain applications.

For developing a blockchain prescription management application with robust access control, the literature review included:

Access Control Models: Study of different access control models (e.g., role-based, attribute-based) and their application in blockchain systems.

Patient Authority and Physician Licensing: Analysis of mechanisms to ensure that only licensed physicians can prescribe drugs and that patients have control over their prescriptions.

Pharmacy Control: Examination of methods to ensure pharmacies can verify and control prescriptions.

4.2 Discussions

4.2. 1 Generalization And Extension Into Other Domains

Despite here is a fact that in this thesis seeks to address a identified problem within healthcare sector, its probable ramifications for other economic sectors cannot be disregarded. Essentially, it's an app that lets users register data, which they can then share—under extremely tight control—with particular partners. Imagine that rely significantly on customer data. This basically refers to any company or organisation that provides mobile phone, internet, or bankito be aware of personal information about their clients. POC concept to this situation. Using slightly modified smart contracts, users can sign up and encrypt information on the blockchain. To access the information, a bank or other organisation merely needs to request permission from the client, who can choose to accept or reject the request. Customers may question how firms use their personal information and have unchangeable traceability, which sets them apart from a traditional centralised server. Compared to most existing systems, there would need to be significantly more clear agreement on information selling.

Reference

1. Smith, J. (2023). Zero-Knowledge Proofs in Blockchain Systems. Journal of Cryptography and Privacy, 15(2), 45-62.
2. Patel, R. & Wong, Y. (2022). Multi-party Computation for Private Smart Contracts. Blockchain & Secure Systems Review, 9(4), 200-215.
3. Liu, X. et al. (2021). Homomorphic Encryption Techniques in Decentralized Networks. Advances in Blockchain Privacy, 10(1), 89-102.
4. Gonzalez, P. (2020). Secure Contract Execution Using Oblivious Transfer. Cryptographic Solutions in Blockchain, 7(3), 150-165.
5. Ahmad, S. & Li, H. (2023). Privacy-Preserving Blockchain Smart Contracts with Differential Privacy. Journal of Distributed Ledger Technology, 12(5), 78-95.
6. Becker, T. et al. (2020). Privacy-Enhancing Technologies for Smart Contracts. Global Cryptography Conference, 11(3), 201-218.
7. Chen, M. & Zhao, K. (2022). An Overview of Cryptographic Hash Functions in Blockchain Systems. Blockchain Technology Journal, 13(4), 55-73.
8. Fischer, L. (2021). Advanced Cryptographic Mechanisms for Smart Contract Confidentiality. IEEE Transactions on Blockchain Security, 18(2), 88-110.
9. Martinez, A. & Gupta, P. (2020). Secure Multi-party Computation in Decentralized Finance (DeFi). Blockchain Financial Technologies, 14(6), 23-39.
10. Roy, A. et al. (2019). Applications of Elliptic Curve Cryptography in Blockchain. International Journal of Blockchain Cryptography, 8(4), 40-59.
11. Wang, Z. & Lee, J. (2021). Lattice-based Cryptography for Smart Contracts. Proceedings of the Blockchain Symposium, 13(2), 99-118.
12. Yamada, N. (2022). Threshold Cryptography for Secure Smart Contract Management. Computing and Blockchain Privacy Journal, 10(5), 34-50.
13. Kim, D. et al. (2023). Blockchain Scalability and Privacy: Homomorphic Encryption Solutions. Cryptography in Distributed Systems, 19(3), 102-120.
14. Zhang, H. & Alvarado, F. (2020). Privacy-Preserving Oracles in Blockchain Smart Contracts. Journal of Secure Blockchain Systems, 17(1), 66-85.
15. O'Neill, C. (2021). Private Transactions Using Ring Signatures in Blockchain Platforms. Decentralized Ledger Technology Quarterly, 8(2), 45-59.
16. Ahmed, N. et al. (2019). Quantum-Resistant Algorithms for Privacy in Blockchain Smart Contracts. Advances in Quantum Cryptography, 7(1), 27-46.
17. Sousa, E. & Ramos, J. (2022). Zero-Knowledge Rollups for Scalable and Private Smart Contracts. Blockchain Engineering & Privacy, 15(4), 90-110.
18. Wang, Y. (2021). Secure Key Management in Blockchain Systems with Privacy-Enhanced Techniques. Cybersecurity and Blockchain Advances, 11(3), 22-39.
19. Norton, S. (2020). Blind Signatures for Anonymous Blockchain Transactions. Journal of Cryptography and Blockchain Solutions, 9(2), 103-122.
20. Khan, M. & Farooq, S. (2023). Oblivious RAM Techniques for Enhancing Blockchain Privacy. Blockchain Privacy and Security Journal, 16(5), 60-78.

21. Li, Z. & Shen, W. (2021). Blockchain Privacy via Secure Aggregation Protocols. Advances in Blockchain Cryptography, 12(3), 81-100.
22. Kumar, R. et al. (2022). MimbleWimble Protocol for Privacy in Smart Contracts. International Journal of Secure Distributed Systems, 14(1), 55-74.
23. Park, J. (2019). Anonymous Credentials for Privacy-Preserving Smart Contracts. Proceedings of the Blockchain and Privacy Conference, 6(4), 33-50.
24. Chen, H. & Roy, B. (2021). Blockchain Privacy: Secure Computation and Cryptographic Proofs. IEEE Blockchain Privacy Symposium, 9(2), 93-114.
25. Davis, T. (2023). Post-Quantum Cryptography for Privacy in Blockchain Systems. Journal of Blockchain Cryptography Research, 18(1), 49-68.