TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

‘Blockchain smart contract system for secure health data sharing’

Georgios Vasilakis

Thesis Committee:

Associate Professor Sotirios Ioannidis

Professor Apostolos Dollas

Professor Eftychios Koutroulis

A Thesis

To be Submitted to the

School of Electrical and Computer Engineering (ECE)

Technical University of Crete

Chania, Crete

ACKNOWLEDGEMENTS

Dr. Sotirios Ioannidis, my supervisor, was instrumental in making this thesis a reality. He was the one who gave me the opportunity to complete my studies, and I will be eternally grateful for his help and support. Furthermore, words cannot express how grateful I am to Mr. Leonidas Kallipolitis for the many hours he spent assisting my work and generously sharing his knowledge, expertise, and guidance. Moreover, Stylianos Lavdas and Stavros Zarpas deserve special recognition for their technical assistance.

I'd be remiss not to mention my family, particularly my parents and sister. Throughout this exhausting process, their faith has kept my spirits and motivation high. Additionally, I'd like to express my gratitude to Eleni Neonaki for her emotional support. Special thanks to Emma Katsani, Nektaria Mariolou, as well as, Fotios Giariskanis as my closest life companions on this journey, as well as for their unwavering trust, patience, and guidance.

ABSTRACT

The main objective of this study is to review, assess and critically evaluate state-of-the-art research conducted around blockchain technology and its adoption in healthcare sector. Furthermore, a blockchain application was designed and implemented to orchestrate, store, and then exchange health data transactions among all parties involved.

To achieve its main objective, the study has been developed in three directions: (1) a broad background study of the blockchain thematic field has been conducted, with emphasis on the theoretical foundation and applications of blockchain in health sector, (2) a comprehensive review of blockchain techniques and related software platforms and tools was assessed, with a focus on the Ethereum platform, which appears to be well-suited for rapid prototype implementations and is thus used extensively by the research community, and (3) a proof-of-concept implementation, based on a private blockchain network, was created to demonstrate the applicability and use of blockchain technology in storing, moving, and securely accessing medical records and health data.

A number of important features inherited by blockchain technology, such as decentralization, immutability and traceability have made it particularly appealing in today's technological environment and sparked significant research efforts for their potential use in the health sector. The effective use of blockchain technology in the health domain may provide significant benefits to patients, doctors, and the medical community, as well as provide health data flow in a simple and secure manner. Even as blockchain networks and applications continue to grow in popularity, significant issues and aspects must be studied and addressed, with scalability and applicability in various domains being two of the most prominent.

The originality and value of the current study are justified because it (1) provides a synopsis and structured presentation of current blockchain qualitative and quantitative research, as well as the related technology evolution, with a particular focus on use cases applicable to the health domain and (2) proves that blockchain-based applications could significantly increase the value of sharing and exchanging sensitive medical records because they can be safely stored and managed, improving accessibility for patients and healthcare professionals and (3) demonstrates how private blockchain networks could be implemented, integrated, and deployed, using a set of software tools and scalable infrastructure, using a fast developed prototype.

Keywords: Blockchain, Proof of Authority (PoA), Healthcare, Ethereum.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AWS	Amazon Web Services
CAs	Contract Accounts
CoA	Control Authority
DPoS	Delegated Proof of Stake
EMV	Ethereum Virtual Machine
EOAs	Externally Owned Accounts
PoAc	Proof of Activity
PoA	Proof of Authority
PoC	Proof of Capacity
PoET	Proof of Elapsed Time
PoI	Proof of Importance
PoL	Proof of Location
PoSp	Proof of Space
PoS	Proof of Stake
PoST	Proof of Stake Time
PoW	Proof of Work
PBFT	Practical Byzantine Fault Tolerance
VM	Virtual Machine

CHAPTER 1: INTRODUCTION

1.1 Problem statement and importance

The healthcare industry is currently undergoing rapid growth as a result of significant recent advances in disease prevention and curation based on patients' genetic identities and modern treatment techniques. At the same time, concerns about system security on the Internet have sparked a debate about the security and privacy of online transactions and has, thus, paved the way for the adoption of new demanding technologies, such as blockchain. Blockchain can be defined as a decentralized immutable ledger, which records data in a structured and secure way. Although blockchain adoption in the health sector is still in its infancy, blockchain applications and technology can effectively address and manage major health domain challenges such as interoperability, data security, and access control. (Mettler [42], Linn and Koo [36], Zhang and Lin [63], Frost and Sullivan [17], Hasselgren et al. [23]).

Due to the sensitivity of medical data, particularly when accessed at the individual level, data security and controlled access to medical records are critical in healthcare. Currently, many large healthcare institutions around the world are digitizing their medical records. Nonetheless, widespread electronic health data sharing among medical institutions, hospitals, doctors, and the medical community as a whole is fraught with technical, operational, and privacy-related reasons and concerns. It is thus of great importance for researchers and practitioners to convince the general public, individuals and policy makers that access to medical information is both safe and necessary for societal benefit. (Linn and Koo [36], Roehrs et al [55], Hasselgren et al. [23]).

Leveraging blockchain technology to ensure secure, well-integrated and controlled data access, will undoubtedly assist health sector in expanding the use of electronic medical records for the benefit of all parties involved, including patients, doctors and the medical community. As a result, the current study proposes the development of a blockchain-based system, responsible for recording and exchanging health data. Blockchain technology provides the capabilities needed to address interoperability and data security challenges and is thus the proposed technology that will allow individuals, healthcare entities and medical researchers to securely record and share electronic health data (Peterson et al. [51], Dagher et al. [14], Patel [50] and Hasselgren et al. [23]).

Its functionality is distinct from that of a traditional database, and it offers distinct benefits to health organizations. Unlike databases, which can be changed, managed, updated, and controlled at any time by a single user, blockchain transactions are final and irreversible. This is due to the network's design, which only allows data insertion. Furthermore, blockchain creates a secure environment by utilizing advanced cryptographic technology and a shared decentralized network. Another advantage of blockchain databases is their high fault tolerance, which is achieved through built-in redundancy. Because each node processes each transaction, no node is required for the database's overall performance. Similarly, several communication lines may fail as nodes communicate with one another in a dense peer-to-peer network before everything comes to a halt. Failure-prone nodes can always catch up on transactions thanks to the blockchain. (Multichain [75]).

Finally, taking into consideration the sensitivity of medical records and health data, it is necessary to include a public entity that will act as the control body of every transaction and will ensure compliance with local and global health regulations (Dagher et al. [14], Hasselgren et al. [23]). Because network nodes in the chain cannot act completely independently, a private blockchain is required. Private blockchain, also known as permissioned blockchain, has a single operator or entity that controls who can access the network and whether or not they can view or create data on the blockchain.

1.2 Aims and objectives

The primary goal of this research is to review, assess, and critically evaluate cutting-edge research on blockchain technology and its application in the healthcare sector. Furthermore, the design and implementation of a blockchain application, capable of orchestrating health data transactions and managing storage and sharing between all relevant involved parties, has been proposed. To achieve the main objective, the following goals have been set:

1) A broad background study of the blockchain thematic field will be conducted, with emphasis on the theoretical foundation and applications of blockchain in health sector. For this purpose, a large number of relative research papers will be studied and evidence for blockchain foundations and its evolution will be provided.

2) An extensive review of blockchain techniques and related software platforms and tools will take place, with emphasis on Ethereum platform, which looks most appropriate for fast prototype implementations and is thus extensively used by the research community. Fast prototyping is of significant importance, since it provides the means to prove that selected technology works and is adequate for the selected use cases.

3) The implementation of a proof-of-concept system that will demonstrate how the blockchain technology can be used to store, transfer, and securely access medical records and health data. The core platform will be Ethereum, on top of which will be built-in features like contract management and an appropriate front end for user interaction.

1.3 Contribution

In the last decade, blockchain technology has proven its applicability especially in implementation and management of cryptocurrencies, such as Bitcoin. Blockchain's applicability and practical use in the health sector has taken some initial steps, but it has yet to be widely adopted. Main contribution of the current research can be summarized as follows:

1) Provides a synopsis and structured presentation of current blockchain qualitative and quantitative research, as well as related technology evolution, with a particular emphasis on use cases applicable to the health domain.

2) Using a rapidly developed prototype, demonstrates how private blockchain networks can be implemented, integrated, and deployed using a set of software tools and scalable, cloud-based infrastructure.

3) Proves that blockchain-based applications may add significant additional value to sharing and exchanging medical records and sensitive health data and could thus provide efficient information management solutions to the health community.

1.4 Main terminology

Blockchain is usually defined in the literature as an "*immutable ledger, logging data entries in a decentralized manner*". It is a type of distributed ledger that is shared among computers on a network, capable to record transactions and track assets; immutability is achieved by a peer-to-peer network of nodes and is not supported by any centralized entity. Assets can be either tangible, e.g., cash, or intangible such as patents and copyrights. Blockchains can be public (permissionless), consortium (public permissioned), or private. The method by which data is accepted and stored in a blockchain network's distributed ledger is known as the consensus mechanism. A consensus mechanism is governed by a consensus protocol, which decides how data entered into the ledger is validated. There are several proposed consensus protocols in the relevant literature, three of which were pioneers and remain among the most popular, namely PoW (Proof of Work), PoS (Proof of Stake), and PBFT (Practical Byzantine Fault Tolerance) (Hasselgren et al. [23]).

Smart contracts are digital agreements between two or more blockchain participants that use predefined functions to store information, process inputs, and record outputs. Smart contracts make it possible to automate and digitize the agreement process and transactions among involved parties. From a technical standpoint, smart contracts are self-executing agreements that are automatically reinforced with pre-agreed terms and provisions that are incorporated into the blockchain platform's source code. State of Arizona legal agreements Legislation, De Beers Group diamond source platform, Robomed Network and Fizzy under AXA are some of the most common real-world examples of smart contract implementations. (Arizona [71], De Beers Group. [72], Robomed Network [73], Medium.com [74]).

Ethereum platform is commonly used to build blockchain applications, since it provides a user-friendly language that can run on virtual machines, known as Ethereum Virtual Machines (EVMs). The Ethereum network consists of nodes (or computers) connected in a decentralized, peer-to-peer architecture. The Ethereum platform is built around the concept of "accounts," which can be either Externally Owned Accounts (EOAs) or Contract Accounts (CAs). EOAs are naturally owned and controlled by a third party and are accessible via private keys. They commonly contain Ether balances and are capable to send transactions to other parties or trigger CAs.

1.5 Thesis structure

The structure of this work is the following:

- Chapter 2 contains a comprehensive background study on the thesis subject. Background and blockchain fundamentals, as well as key of blockchain features, are presented in Sections 2.2 and 2.3 respectively, while Section 2.4 outlines blockchain types and Section 2.5 explains blockchain mechanisms of consensus. Section 2.6 provides a detailed presentation of smart contracts, while Section 2.7 discusses the vertical applicability of blockchain in the health industry. Finally, in Section 2.8, the main blockchain challenges and areas for improvement are presented, and the Chapter concludes with an overall discussion and outcome in Section 2.9.
- Chapter 3 includes the practical, "*hands-on*" work of this thesis and presents as a case study a blockchain-based prototype implementation in the health domain. Section 3.2 discusses the fundamentals of the Blockchain Ethereum platform and environment while the prototype design is presented in detail in Section 3.3. More specifically, this Section includes main design principles, functional and non-functional features, functional requirements in detail, user types and privileges, overall prototype architecture, prototype deployment and PoA with AWS, smart contracts, and front end, other prototype support components and final prototype's integration and interoperability. Section 3.4 visually presents and discusses the prototype's implementation results based on two hypothetical scenarios, namely the CoA creating a new campaign and a doctor requesting to participate in a campaign and receiving approval from the CoA.
- Chapter 4 provides a summary of the study conducted, along with the conclusions derived in Section 4.1, while Section 4.2 outlines thesis limitations and potential future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

During the last decade, blockchain technology has disrupted a significant number of data driven areas, spanning many different sectors such as banking, insurance, retail, and others, while significant efforts have been made to establish applicability of blockchain technology in the health sector.

When Bitcoin first appeared in 2008, blockchain technology became widely known. A number of important features inherited by this technology, such as anonymization, decentralization, and transparency made it particularly appealing and sparked significant research efforts in related, but also in different domains. (Nakamoto, 2008; Mettler, 2016). Bitcoin is currently the king of cryptocurrencies, with nearly half a billion transactions in 2019, 0.7 billion in 2021 and 0.8 billion in 2022. (blockchain.com [4]).

According to a recent IBM study, approximately 70% of healthcare leaders believe that blockchain technology will have a significant impact in the health sector, leading to improvements in the execution and management of clinical testing, essentially supporting regulatory compliance and securely enabling the sharing of medical records between interested parties, by leveraging its decentralized framework (IBM, 2016). Furthermore, the blockchain technology market value in healthcare, is expected to be around $500 million by 2022 (Frost & Sullivan, 2019).

Although blockchain technology has cultivated promises of widespread applicability and smart applications in health care, current results show little evidence of this because many of these cases are based on unrealistic ideas that are difficult to implement. Current literature also proves that only a small number of real-world applications have been developed, tested, deployed and are actually used on the field (Mettler, 2016; Hasselgren et al., 2020).

2.2 Background and blockchain fundamentals

A blockchain, in theory, should be viewed as "*a distributed append-only timestamped data structure*" (Casino et al. [9]). In practice, Blockchain enables interoperability between different parties without the need for a central entity to coordinate. Data is stored in a structured way, bundled in blocks that are growing continuously, while each block is connected to the previous and next one with the aid of cryptographic algorithms to ensure security (Raikwar et al. [54]). Each block contains a unique identifier, also known as digital fingerprint or hash and includes groups of timestamped transactions. Blockchains are created by connecting blocks in chronological order. In practice, changing any of the blocks in the middle of the chain is impossible because doing so requires changing all subsequent blocks after the modified one at the same time. This feature ensures that the data on a chain of blocks remains immutable. From its inception, Blockchain technology allowed data blocks to be accessible by community members in both read and write mode; the capability of decentralized data management, was one of the main reasons that made this technology attractive in many different domains and applications (Hasselgren et al. [23]).

Blockchain is appropriate for delivering fast and effectively business information to network members because it provides direct access to traceable and immutable data via a secure, decentralized ledger. After a transaction has been accepted and stored in the immutable ledger, no network member has the ability to change or update it. A blockchain network could thus be used to track, store and access important information and its evolution chain in critical business areas such as payments, production goods, patients' health data, etc. Another very important Blockchain feature that has sparked a lot of interest is the "*smart contracts*" feature. This allows for the creation and execution of contracts without involvement of any central body or authority (Buterin, 2014). Exploitation of the blockchain technology is based on the development of decentralized applications, known as Dapps. Dapps are built on top of existing blockchain platforms like Ethereum (decentralized platform), taking advantage of specialized frameworks such as Hyperledger (Cachin [8], Hasselgren et al. [23]).

The use of blockchain technology in the healthcare industry could be extremely beneficial to the medical community. The reason for this is that blockchain could make health data exchange more efficient and transparent while also ensuring data security. For example, patient's medical records, which typically exist in multiple locations (hospitals, medical care providers, insurance companies) with no integration, interoperability and consolidated view, could benefit from the use of a blockchain network, which would enable involved entities to combine information of interest. Furthermore, it ensures high system availability and virtually eliminates the need for any specialized disaster recovery solution because immutability is an intrinsic feature of the blockchain network.

2.3 Main features of blockchain

A number of researchers have already studied blockchain. Many studies, among others, focus on how a blockchain network works and what its core characteristics are. If one were to attempt to summarize the main features of Blockchain, the following list would be exhaustive (Hasselgren et al. [23]; Zheng et al. [67]; Sultan et al. [58]; Chen et al. [11]):

- Decentralization/ Consensus driven validation. No central entity controls the data stored, accessed, validated, and maintained in the blockchain. On the contrary, transactions are carried out in a distributed manner between network members, with the help of specific consensus mechanisms that provide rules for block validation.
- Immutability/ Persistency. Because of the nature of the decentralized ledger, data entered into the blockchain remains immutable, or persistent, because it is not practical to alter or delete it after creation. Transaction blocks are linked using hash keys, while each network node runs a validation algorithm to ensure validity of those hash keys. Due to the large numbers of blockchain nodes that are synchronized in real time, it is impossible to change any transaction after its propagation and storage across network members. Immutability is one of the main reasons why blockchain networks are trusted.
- Anonymity/ Pseudonymity. Blockchain is making it possible to keep data anonymity and pseudonymity, ensuring data access and identification through hash keys and specific validation algorithms.
- Traceability/ Auditability. Data in blockchain networks is auditable and traceable because all transactions are timestamped in chronological order and each block is linked to its neighbors using a hashing algorithm. Moreover, transactions within each block are structured in the form of a Merkle tree, with each transaction represented by a leaf that can be linked to the block root. (Merkle [41]).

In Figure 1, the main features of blockchain networks are presented.

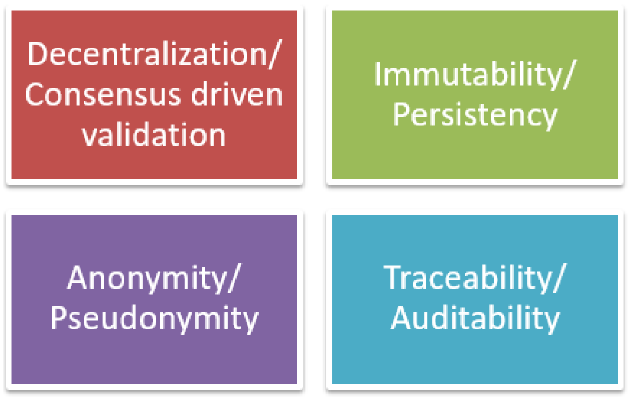
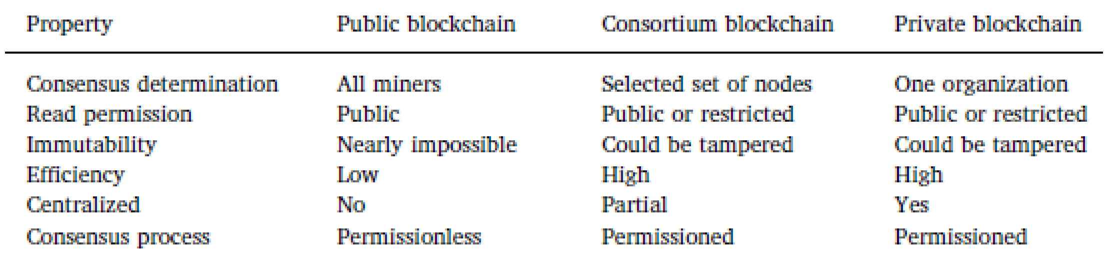

Decentralization/ Consensus driven validation
Immutability/ Persistency
Anonymity/ Pseudonymity
Traceability/ Auditability

Figure 1: Main features of Blockchain networks.

2.4 Blockchain types

The blockchain community is constantly debating the various types and categories of blockchain networks. Even though currently there is not any broadly and widely accepted blockchain categorization, there are significant studies available in the literature that have attempted to group blockchain networks based on specific features. In most cases, main criterion for the categorization is who has read and write access to the data. In Table 2.1, a synopsis of the blockchain types is presented as has been proposed by Zheng et al. [67] and Hasselgren et al. [23]. As can be easily noticed, there are three main blockchain types; public blockchains which are permissionless, consortium blockchains which are public permissioned and private blockchains.

Table 2.1: Synopsis of blockchain types.



Property	Public blockchain	Consortium blockchain	Private blockchain
Consensus determination	All miners	Selected set of nodes	One organization
Read permission	Public	Public or restricted	Public or restricted
Immutability	Nearly impossible	Could be tampered	Could be tampered
Efficiency	Low	High	High
Centralized	No	Partial	Yes
Consensus process	Permissionless	Permissioned	Permissioned

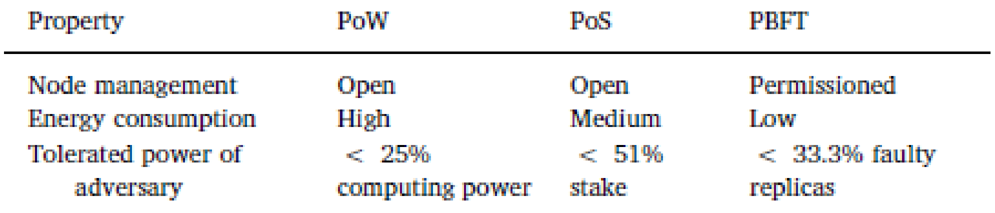
Source: (Hasselgren et al.* [23]*)

In more detail, in a public blockchain, anyone can view the data, while the system is open to anyone who wants to contribute consensus and updates to the core blockchain software. Cryptocurrencies are examples of public blockchains; for example, both Bitcoin (Nakamoto [43]) and Ethereum (Buterin [7]) are examples of public permissionless blockchains. Consortium blockchains are distinguished by their partial centralization, with only a small number of groups of entities having both view access and permission to participate in the consensus protocol. Finally, a private blockchain is distinguished by a distributed network that can frequently become centralized. A private blockchain is usually monitored and managed by a central body, and only a limited number of specific nodes have access to it (Zheng et al. [67]).

2.5 Blockchain mechanisms of concensus

The manner in which data is accepted and stored in the distributed ledger of a blockchain network is of significant importance and is critical to maintain integrity of the blockchain ecosystem; it is usually known as consensus mechanism. A consensus mechanism is governed by a consensus protocol, which decides how data entered into the ledger is validated. Consensus protocols are necessary in decentralized systems to impose the norms necessary to ensure consensus, even in networks where nobody trusts anyone. In other words, consensus protocols are mandatory in any distributed system to ensure system integrity and appropriate functionality, as well as to avoid the presence of specific nodes that will act as single points of failure. There are a number of proposed consensus protocols in the relevant literature, three of which were pioneers and remain among the most popular, namely PoW (Proof of Work), PoS (Proof of Stake), and PBFT (Practical Byzantine Fault Tolerance), as shown in Table 2.2. The main features and functionality of those protocols are explained in detail below (Hasselgren et al. [23]).

Table 2.2: Blockchain Consensus Mechanisms.



Property	PoW	PoS	PBFT
Node management	Open	Open	Permissioned
Energy consumption	High	Medium	Low
Tolerated power of adversary	< 25% computing power	< 51% stake	< 33.3% faulty replicas

Source: (Hasselgren et al. [23])

1) Proof of Work (PoW). Because of its early use in Bitcoin, where decentralization and cryptography were combined under the umbrella of a blockchain network, this protocol has grown in popularity. Being the pioneer, PoW was preferred by other major blockchain networks that followed Bitcoin. PoW, in particular, facilitates the mining process, in which "miners" compete against one another by using as much computational power as possible to discover a part of the data block (known as hash) with a value less than a predetermined base value. The miner who uses the most efficient computational techniques and has the most available computational power will usually compute the hash value faster and receive an award, such as a portion of a bitcoin. The large amount of energy required and consumed during the mining process is a significant weakness of this protocol, especially when used on large networks. Furthermore, recent studies, including the 2020 Ethereum Classic hack, (D.Howarth [24]), have provided some evidence that PoW does not meet the highest security standards and may be vulnerable. Even though peer-to-peer scalability is high, its performance is considered low because transaction rates are generally low. It is applicable to public networks, with Litecoin being a popular example. (Nakamoto [43]; O' Dwyer and Malone [48]; Howarth [24]).

2) Proof of Stake (PoS). The node selected to approve the data entered into the blockchain is determined by its current stake, which in the case of cryptocurrencies is the balance of the currency. Even though this protocol does not suffer from the drawback of high-energy consumption, it does have a fairness issue because it favors nodes with the highest stake values while also being vulnerable. To address the fairness issue, a number of alternative PoS versions have been proposed, in which the stake value is not the only criterion considered, but it is combined with others, such as considering a level of randomness. Its performance is considered high, and it is applicable to both public and private networks, with Tezos, NXT and Ethereum serving as notable examples.

3) Practical Byzantine Fault Tolerance (PBFT). This protocol is applicable only under a permissioned environment, since to be feasible each node must be known by all other nodes. The PBFT protocol is implemented in three stages: the pre-prepared phase, the prepared phase, and the commit phase. To advance from one phase to the next, each node must receive

two thirds of the votes cast by the other nodes. One of the most popular examples of ledgers using the PBFT protocol, is Hyperledger Fabric (Castro and Liskov [10]; Hyperledger [25]).

4) Recent research has shifted toward increased security, revealing Proof of Authority (PoA) as an alternative consensus mechanism with significant advantages over both PoW and PoS. In practice, PoA is a variant of PoS in which nodes are working to enhance their reputation and identity rather than identifying tokens. PoA, in particular, employs an enhanced security and sustainability approach by utilizing validators that define reputation, while remaining highly scalable due to the use of a small number of validators. It was introduced in 2015 by G. Wood, one of the founders of Ethereum platform, and has since grown to become one of the most prominent consensus protocols. Main industries where PoA is applied are insurance and banking. It is a high throughput protocol, and its consensus is relatively centralized, making it suitable for public, private and consortium networks. Because of the way it works, particularly the identity requirement, PoA use in very large public blockchains is practically impossible, as the number of validating nodes is enormous in such cases. Obviously, its usage is optimal on blockchains with small number of nodes, and especially in enterprise environments. Real life examples of PoA include Kovan, Rinkeby and Giveth. (Zheng et al. [67]).

Other consensus algorithms that have become quite popular over the years are the following (Cachin [8]; Zheng et al. [67]):

- Delegated Proof of Stake (DPoS). This is an extension of PoS in which network nodes delegate the creation of new data blocks to a small number of validators in order to increase speed and fairness. It is a high-throughput protocol that can be used on both public and private networks. EOS and BitShares are two networks that use DPoS.

- Proof of Activity (PoAc). This is a hybrid of PoW, PoS and PoA that aims to provide a balance between miners and other members. It is a low throughput protocol, suitable for public networks. Decred is an example of network using PoAc.

- Proof of Location (PoL). This protocol uses beacons to identify nodes in synchronized state and stamps their participation with a stamp. It is an average throughput protocol, suitable for public networks. FOAM and Platin are two networks that makes use of PoL.

- Proof of Importance (PoI). This protocol is similar to PoS, but it possesses a number of additional properties that effect in ranking. It is a high throughput protocol, suitable for public networks. NEM is an example of network that makes use of PoI.
- Proof of Elapsed Time (PoET). PoET is a PoW-like in which blocks are created in equal intervals. It is an average throughput protocol, suitable for private networks, utilizing much less energy than PoW. Intel is an example of network that makes use PoET.
- Proof of Stake Time (PoST). While retaining the efficiencies of PoS, this protocol also provides enhanced security and decentralization by introducing a mechanism that increases the probability of stake with time. It is a high-throughput protocol that is appropriate for public networks. VeriCoin Blockchain Explorer is a PoST network example.
- Proof of Capacity (PoC)/ Proof of Space (PoSp). This protocol works similarly to PoW, but is different in the amount of work undertaken to verify blocks, and in the timing when this verification takes place; verification process is called "*plotting*". It is a high throughput protocol, suitable for public networks. Burstcoin is an example of a PoC network.

2.6 Blockchain smart contracts

Smart contracts are digital agreements between two or more blockchain participants that use predefined functions to store information, process inputs, and record outputs. Smart contracts enable the automation and digitization of the agreement process and transactions between involved parties. As previously stated, "*smart contracts*" are a very appealing feature of some blockchain platforms such as Ethereum, because they enable for the creation and execution of contracts without involvement of any central body, authority, or third party. From a technical standpoint, smart contracts are self-executing agreements that are automatically reinforced with pre-agreed terms and provisions that are incorporated into the blockchain platform's source code. Practically, smart contracts are supported by executable programming code that runs in the blockchain network nodes to ensure validity of an agreement between two blockchain parties without the intervention of any third party (Buterin [7]; Hasselgren et al. [23]).

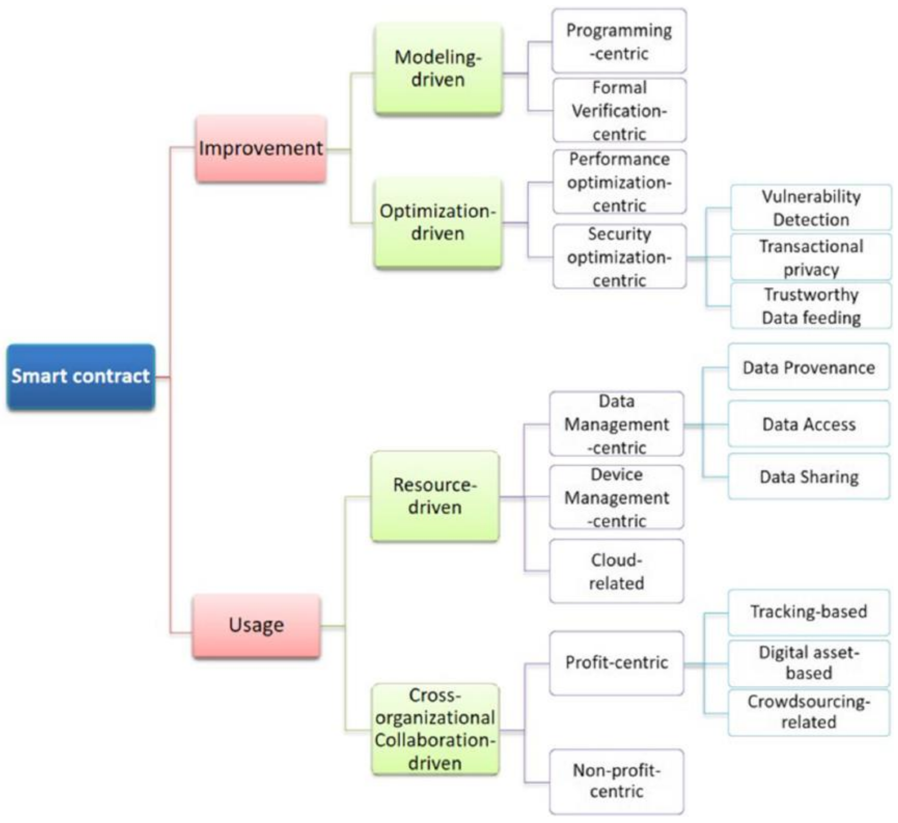
From a programming point of view, smart contracts could be seen as classes that contain functions, function modifiers, state variables, and other programming structures required for enablement of proper control and event execution, according to defined smart contract terms. Furthermore, constructor and destructor functions could be used to support smart contract structures. Constructors could be used to facilitate contract creation. For example, a member of the blockchain could initiate a new contract by invoking a constructor function in a transaction; in this case this member acquires smart contract ownership. The contract's owner is typically the only blockchain member who can destroy it by invoking a self-destruct function. In addition to the previously described basic functionality, smart contracts may be able to call other smart contracts, and a portion of the code usually supports state definition and management. States can be constant (the ones that can never be changed) as well as writable (the ones that can be updated). States can be read and updated by smart contract functions; functions can either be read only functions, not requiring computational effort to run, or write functions where computational effort to run, or write functions, in which computational effort is needed to run (Buterin [7]; Khan et al. [30]).

Smart contracts are stored in every network node to ensure that contract modification is not possible within the blockchain environment. Obviously, automating contract execution reduces the possibility of human error and significantly reduces potential disputes between parties involved. On the other hand, because smart contracts are part of a large cyber community, they could be targets of cyber-attacks, and there are numerous examples of malicious activity, primarily in cryptocurrency platforms. For example, in an attack on the Ether currency platform in 2016, attackers hacked its smart contract to steal nearly two million of Ether by exploiting a specific vulnerability, specifically the reentrancy issue (Wired.com [61]). Legal and privacy issues, as well as performance issues during execution, are among the other challenges that smart contracts face (Singh et al. [57]; Khan et al. [30]).

As previously stated, smart contracts necessitate the creation, deployment, and use of an underlying blockchain platform. Ethereum, Hyperledger Fabric, Bitcoin, and NXT are some of the most popular platforms for smart contracts. These platforms include features and capabilities to aid in the creation and management of smart contracts, such as security features, execution enablement capabilities, and specific coding or scripting languages for their development. More specifically, the smart contracts capabilities provided by these platforms are summarized as follows (Khan et al. [30]):

- Ethereum was the pioneer blockchain platform to support the creation and management of smart contracts. EMV (Ethereum Virtual Machine), a runtime environment that executes the same code in every blockchain node of the network, is the fundamental structural element of Ethereum on which smart contract capabilities are based. Specific code required for smart contract execution is sent to the blockchain network via a transaction and is then executed in all modes using its local EVM. Solidity is the scripting language used to implement and manage smart contracts. Ethereum is the most popular blockchain platform in terms of smart contracts, and it is used to support the implementation and execution of Dapps (Decentralized Applications) in a variety of business and functional domains (Buterin [7]).
- IBM proposed Hyperledger Fabric, a modular enterprise platform based on an open-source distributed ledger that supports smart contracts among other things. Members of Hyperledger Fabric are businesses that are constructing their own networks and can benefit from a wide range of industry-specific use cases by utilizing “plug and play” modules. In terms of smart contracts, Hyperledger Fabric supports a variety of programming languages, including Javascript and Java. If a smart contract transaction is generated, it is only propagated to and executed by a subset of peers known as endorsing peers. Endorsing peers only, execute received transactions by running the required chaincode, which runs in an isolated environment for security purposes (Androulaki et al. [1]).
- Bitcoin is the most well-known public blockchain platform, best known for its use in the processing of cryptocurrency transactions. Bitcoin’s smart contract capabilities are limited. Although a bytecode scripting programming language is used, as previously stated, foundational changes to the mining functions are required to support effective smart contracts capabilities (Lewis [32]).
- NXT is an open-source platform that uses the PoS (Proof of Stake) consensus protocol exclusively. One of its key features is that smart contracts can only exist in the platform if they are built on existing contract templates, which means that smart contract personalization is not possible (NXT [47]).

Many researchers have proposed groupings of smart contracts. A representative example is the categorization proposed by Khan et al [30] in their relative study, attempted a categorization of blockchain smart contracts based on a variety of different criteria as depicted in Figure 2 below.



Source: (Khan et al. [30])

Figure 2: Taxonomy of blockchain smart contracts.

2.7 Blockchain in Health

The healthcare industry is currently experiencing an explosion of data generation and dissemination among interested parties. Simultaneously, long-standing issues such as data leakage and unauthorized data sharing are worsening, having a significant impact on public trust in healthcare institutions as well as the operational and reputational risks they face. The sensitivity of patient data when stored and accessed on an individual level complicates matters even further. All the above factors trigger the need for an alternative data management and data exchange approach in the healthcare sector. Blockchain technology is a great candidate that might cope with the aforementioned issues of healthcare sector, because of its nature and inherited data driven characteristics (Coiera [13]; Hasselgren et al. [23]). Blockchain is ideal for security applications because it can keep an incorruptible, decentralized, and transparent log of all patient data. Furthermore, blockchain is both private and transparent, concealing any individual's identity with complex and secure protocols capable of safeguarding the sensitivity of medical data.

Multidisciplinary teams of healthcare professionals, with a variety of skills and competencies, along with required information and technology, are necessary to achieve effective hospitalization and patient treatment. For the effective, end to end, patient treatment, multiple operations are needed that require important data as an input and produce outcomes that are critical for the proper patient treatment. Particularly, required data falls into one of the following three main areas. (Hasselgren et al. [23]):

1) health domain problem solving
2) clinical condition assessment and treatment
3) decision on care, based on previous knowledge.

Accessing (reading or writing) information in any healthcare system ought to be conducted in accordance with strict security policies, especially when it comes to patient medical records or data that is critical to the proper operation of hospitals or any healthcare unit. At the same time, effective collaboration between the healthcare institutions and the academic community necessitates the provision of granular, anonymized patient data, while universities and research institutes return expertise and skills to the healthcare sector. In the field of clinical trials, healthcare institutions help by providing experiment data, while the academic community provides research results, new more effective treatment methods, specialized in-depth knowledge, and new technology and tools. The aforementioned collaboration requires secure exchange of test data and research results but is also based on obtaining and maintaining the necessary consents that should be provided by patients and involved parties regarding personal and sensitive information (Linn and Koo [36]; Hasselgren et al. [23]).

According to the preceding discussion, the effective and secure exchange and maintenance of private and sensitive data between involved parties in the health care domain boils down to four (4) critical factors: access control, provenance, data integrity, and interoperability. In more detail, the following hold true for each of those factors:

1) Access control is accomplished through the assumption of trust between the entities that store the data and the data owner. In practice, the entities involved are usually trusted servers that can define and enforce secure access control policies. Many research studies in recent literature have focused on improving access control. Various approaches, such as the use of public and private keys (Patel [50]), the use of queries to identify and retrieve specific data (Peterson et al., 2016), and the use of Hyperledger membership services, have been proposed (Liang et al. [34]).

2) Interoperability, or in other words the ability of different applications or systems to operate together in an orchestrated way, is critical for effectively exchanging data between interested parties, individuals or groups of people. Various approaches to achieve effective interoperability, have been discussed in the literature. In some cases, URL references are used (Peterson et al. [51]; Zhang et al. [63]), whereas in others, a translator is used to convert data from one format to another that is more appropriate (Roehrs et al. [55]).

3) Data provenance is very important for a patient's medical record. Data provenance ensures the full historical trace of patients' data and can thus ensure information transparency and auditability. In the context of data, it refers to the mechanism that tracks and versions every change made to the original data so that authorized viewers can be confident in its legitimacy. Blockchain technology has the potential to be extremely useful because it ensures that data can never be deleted, modified or tampered once it is committed to the ledger. Blockchain has been positioned to improve data provenance in a variety of ways, including the use of trusted IoT for smart contract execution (Bocek et al. [5]) and the use of a specialized system to track clinical trial data (Nugent et al. [46]; Angeletti et al. [2]).

4) Data integrity is a necessary condition to ensure that consolidated health information is complete and meets expected data quality criteria. Despite the fact that data integrity is naturally supported in blockchains, due to the property of immutability, efforts to further improve it can be found in the recent literature. Some researchers, for example, have approached data integrity by storing and using medical data in hashed forms or using hash pointers (Dagher et al. [14]; Wang and Song [59]; Li et al. [33]), whereas others use smart contracts and blockchain integration with IoT devices (Nugent et al. [46]; Angeletti et al. [2]).

Consortium blockchain is the most popular type in healthcare, while private and public blockchains are less common. This is a natural occurrence due to the high sensitivity of health data. Ethereum is one of the most widely used platforms in healthcare, where it is used to maintain health records, conduct clinical trials, monitor patients, improve safety, display information, and enhance transparency. Furthermore, it preserves hospital financial statements while reducing data transformation time and cost, and it offers interoperability with smart contracts and applications that have the potential to transform the healthcare industry. It should be noted, however, that a significant number of implementations are built on Hyperledger Fabric. The majority of research studies conducted so far, have been based on Ethereum platforms controlled by institutions and focusing on efficient management of personal health records or on exploitation of M-Health capabilities (Agbo et al. [69]). Main reasons for the wide applicability of Ethereum and Hyperledger platforms are their advantage to effectively implement and manage smart contracts providing efficiencies in interactions with third parties, as well as because of their wide market penetration and big number of developers with skills and expertise working on those platforms. Regarding consensus mechanisms, in most of the cases PoW, PoS and PBTF have been proposed in the health care domain (Hasselgren et al. [23]).

Regarding the specific content that health care blockchain implementations deal with, the following are the most popular areas in the literature:

- <u>Health Records Management</u>: Blockchains are fundamental in terms of implementing required functionality to share data from medical health records between clinical teams, such as doctors, researchers and other field professionals. Team and community care management, as well as controlled access across various health systems, are common use cases. A number of researchers have proposed the implementation of complementary health records, with additional data that amend already included information in institution specific health records, in order to improve care continuity and empower patients (Zhang et al. [63]; Liang et al. [34]; Zhang and Lin [63]; Liang et al. [34]; Wilkowska and Ziefle, [60]).
- <u>M-health Capabilities Exploitation</u>. M-health refers to the ability of patients to collect, store and control their own data from mobile devices with the aid of sensors. Patients can then share this data for their own benefit, such as by taking advantage of virtual care methods like remote patient monitoring and telecare. Although M-health is a new research area lacking

common principles, standards and regulations, it should be definitely further studied given the rapid increase of M-health application users and the high value of the data collected (Griggs et al. [21]; Ichikawa et al. [27]; Liang et al. [34]; Zhao et al. [66]; Zhou et al. [68]).

- Biomedical Research and Clinical Trials Use Cases. There are also cases where blockchain is used to support research in biomedical science and clinical trials. Medical research organizations and institutions, in particular, are inextricably linked with the health care industry because they use data for their research while also providing useful research outcomes to support health care improvements (Rahman et al. [53]; Zhou et al. [68]; Mamoshina et al. [38]; Angeletti et al. [2]; Nugent et al. [46]).
-

2.8 Main blockchain challenges and areas of improvement

Although blockchain technology is very promising and has the potential to penetrate into many different industries and applications, claiming that it is suitable for any area or use case would be misleading. While blockchain can be beneficial and add significant value in some cases, its nature and limitations make it unsuitable for use in others. Main challenges and limitations that blockchain technology faces can be summarized as follows (Casino et al. [9]):

1) Latency and scalability issues: The speed of transaction execution in blockchain networks is generally slow, much slower than traditional transactional systems such as Visa or Mastercard. This is due, among other things, to the high processing costs, but also to the time required by applicable consensus algorithms. Even in private blockchain networks, which are much faster by definition, there is still room for improvement. To be widely adopted, blockchain technology must address the issue of latency and provide effective solutions to increase current transaction rates. Simultaneously, optimizing and effectively scaling resources such as computational power and storage will necessitate additional efforts from the blockchain research community. Researchers have already tried to reduce storage requirements while improving performance, such as by deleting old transactions (Bruce [6]), or splitting blocks into two parts (key block part and micro-block part) [Eyal et al. [16]]. To improve performance, the Ethereum platform introduced the logic of "*sharding*", or in other

words the partitioning of blockchain networks into "*shards*" each one of which stores part of the overall state and transaction history. It should not be overlooked that by applying techniques aimed at reducing latency and improving performance, side effects in other important aspects such as scalability, security, and the level of blockchain decentralization may occur. Before applying, these side effects must be considered in order to balance any effort to improve latency.

2) Sustainability challenges: Overuse of resources such as computational power has traditionally been one of the most significant challenges that blockchain networks particularly public ones, face. Overconsumption of computational power results in waste of energy and consequently, waste of natural resources, which poses a growing threat to climate sustainability today. In 2017, a study conducted by Digiconomist (Digiconomist [15]) provided evidence that mining of Bitcoin consumes electricity quantities equivalent to the consumption of 159 countries, and today the situation is even worse. The complexity of consensus algorithms is largely to blame for energy consumption, as many of them require massive computational power. Renewable energy is a potential solution to the sustainability problem, but despite recent research efforts in this area, there is still a long way to go (Potenza [52]; Gogerty and Zitoli [18]).

3) Limited adoption and interoperability challenges: As previously stated, blockchain technology is rapidly gaining traction; however, the current level of adoption is merely at the beginning of its journey, as only a few industries and applications are currently adopting blockchain technology. Apart from the fact that it is a new technology, the main reasons are its extensive resource requirements (e.g., data storage, computational power, etc.), as well as the fact that the heterogeneity of blockchain networks and solutions creates rapidly increasing challenges in standardizing interoperability. Existing cryptocurrency APIs, for example, are frequently difficult to use. To address this issue, many businesses and organizations are attempting to transition to standardized and configurable integration solutions that take advantage of more interoperable architectures. For example, efforts are currently underway in the healthcare sector, as well as in education and citizenship blockchain solutions, where significant work for standardized interoperable services is underway (Cheng et al. [12]; Governatori et al., [20]; Kosba et al. [31]).

4) Data confidentiality, privacy and security issues: Another challenging area for blockchain networks to address, mainly due to their public ledger nature, is data confidentiality, privacy and security. Blockchain vulnerabilities, particularly those affecting cryptocurrency networks, have repeatedly proven to be a big challenge; the DAO and Parity wallet attacks, for example, resulted in losses of 47 million dollars and 280 million dollars, respectively. (Siegel [56]; Parity Technologies [49]). Because of the required traceability, transactional privacy is jeopardized; techniques such as pseudonymization and anonymization are used to ensure appropriate functionality and security (Kosba et al. [31]; Goldfeder et al. [19]). Diverse implementations are gradually developed to cope with existing challenges, ranging from generic (Dannen [15]) to most efficient such as the ones that rely on smart contract checkers (Mavridou and Laszka [39], Nikolic et al. [45], Kalra et al. [29]).

2.9 Discussion and outcome

Obviously, blockchain solutions are not appropriate for all data management use cases. A number of factors should be considered when determining suitability, such as whether and how long data must be stored, how many users read or store the data, the importance and criticality of performance, and so on. Lo et al. [37] used a domain-specific evaluation framework to conduct such an assessment, taking into account factors such as identity management and data properties. Similarly, in (Wust and Gervais [62]), the blockchain type and properties are taken into account.

Casino et al. [9] presented and tested an evaluation framework that compares the suitability of blockchain versus traditional databases, which by nature are not immutable and operate by assigning different roles per (known) user, while administrators are capable to change information structure and content. The main distinctions are summarized in four areas:

1) blockchain networks do not rely on trusted third parties, while databases do,

2) blockchains are preferable when traceability is needed and security and privacy are strict requirements and deviations cannot be accepted, whereas databases are more vulnerable due to their centralized nature,

3) blockchains are more cost effective in terms of maintenance, while databases require hosting and have thus more stringent maintenance requirements, and
4) blockchains, as opposed to databases, are better suited to achieving consensus among multiple users.

Even though blockchain networks and applications are gradually expanded proving their value, still significant issues and aspects must be further studied and addressed, with scalability and applicability in various domains being two of the most prominent. Although an individual examination of the features of blockchain networks will reveal that the majority of them are not the result of new research but have been well known for a long time, it is their combination that distinguishes blockchain technology. However, as previously stated, blockchain is not a technology for every application. Traditional databases are far more appropriate in many cases, or specific vertical solutions with industry-specific added value are preferable (Casino et al. [9]).

Blockchain has permeated many different research areas and industry domains due to its remarkable features of decentralization, anonymity, traceability, and persistence. For example, blockchain technologies have been applied to content delivery networks, data management applications, and smart grids, while the health industry also has a strong potential to use blockchain characteristics in the direction of eliminating third parties where possible and providing significant benefits resulting from efficient healthcare data sharing. Because healthcare data is not commonly managed in a pure distributed manner in the current information sharing landscape, research efforts targeting to a more distributed process are expected to provide efficiencies in improving performance and security issues and making massive healthcare data flows feasible through different channels, enabling for example more efficient internet, or mobile healthcare applications (Ni et al. [44]).

Furthermore, healthcare domain is “*problem driven*” and “*data intensive*”. Daily operations include clinical assessments and decision making, resolution of health problems and issues, enrichment and processing of knowledge stores, and treatments requiring combination of medical skills and technology or collaboration between different teams and experts. The type and criticality of those operations implies that capabilities such as medical information management and patient data security, become very critical for the successful execution of its mission. Healthcare activities involve among others efficient consent management and sharing of patient data to interested parties, while health organizations, institutions and companies need to be very careful to securely transmit, manage and share patients' personal sensitive data. That is why data access control, data security, integrity, traceability, and interoperability must be secure and effective (Hasselgren et al. [23]).

CHAPTER 3: CASE STUDY – BLOCKCHAIN-BASED PROTOTYPE IMPLEMENTATION FOR SECURE DATA HEALTH SHARING

3.1 Introduction

In this chapter a proof-of-concept blockchain implementation for recording data transactions between healthcare stakeholders, is proposed. Its overall architectural approach, along with its technical design and practical considerations are presented, while a pilot, prototype implementation that has been developed, is presented and documented. The pilot work product developed, named Chain-Health, is nothing more than a web application that uses a blockchain smart contract system for secure health data sharing. Its goal is to assist related health companies, doctors, patients and the medical community in securely storing and exchanging health transactions, while being monitored by a Control Authority (CoA) entity. Aside from Chain Health, an auxiliary application called Health Petition has been developed to allow interested parties who use Chain Health to transfer files.

3.2 Blockchain Ethereum Platform and Environment Fundamentals

Ethereum platform is commonly used to build blockchain applications, since it provides a user-friendly language that can run on a virtual machine; virtual machines are simulating actual machines by using software that can understand a set of instructions and execute them in some logical order, just like a real computer does. The virtual machines used by Ethereum, are known as Ethereum Virtual Machines (EVMs). When a smart contract or other application is built using Ethereum's programming language, the EVM compiler takes that code and translates it into lower-level machine code that the EVM can understand, process, and execute. The Ethereum network, as shown in Figure 3, is made up of nodes (or computers) that are linked in a decentralized, peer-to-peer architecture. An EVM runs in each node, processing the same set of instructions to ensure that consensus is reached on each transaction.

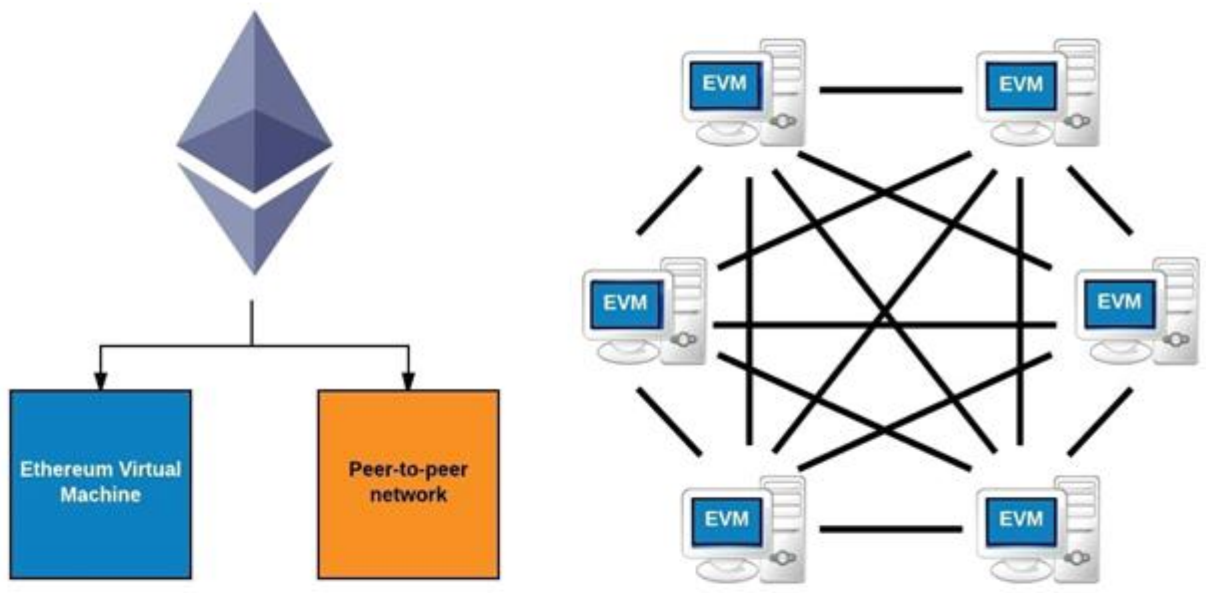


Figure 3: Ethereum Virtual Machines (EVMs) architecture.

An Ethereum account is a legal entity with an ether (ETH) balance that can send Ethereum transactions. Accounts can be controlled by the user or deployed as smart contracts. Ethereum has two types of accounts:

- **Externally-owned account (EOA)** managed by anyone who has access to the private keys and
- **Contract account**, a smart contract that has been deployed to the network and is controlled by code

Anyone who wants to participate in the Ethereum ecosystem, needs to own an EOA along with the keys needed to operate the account. Figure 4 presents schematically the two account types.

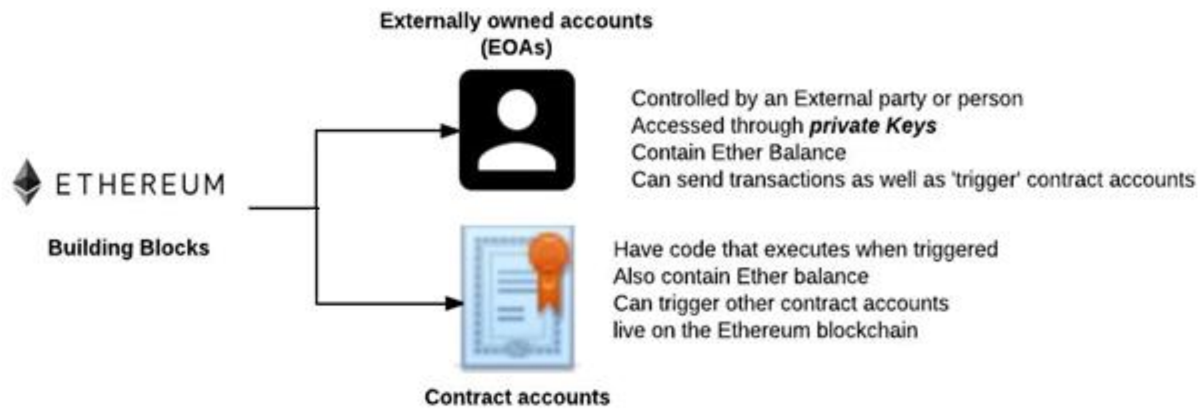

Externally owned accounts (EOAs)
Controlled by an External party or person
Accessed through private Keys
Contain Ether Balance
Can send transactions as well as 'trigger' contract accounts
ETHEREUM
Building Blocks
Have code that executes when triggered
Also contain Ether balance
Can trigger other contract accounts
live on the Ethereum blockchain
Contract accounts

Figure 4: Ethereum Account Types.

Developers who intend to build applications or interact with smart contracts on Ethereum, need Ether to pay for the required computational resources within EVM. The smallest denomination of Ether is called Wei and one Ether has 10^{18} Wei. A CA contains code to execute a particular function for which it was designed; this code 'lives' in the Ethereum blockchain platform. Once a CA is triggered, its code is executed by all EVMs on every node of the network. CAs enable Smart Contracts, which are contract accounts that facilitate transactions in a transparent way. The steps to create a Smart Contract, can be summarized as follows:

1) A contract account is created by defining rules and the actions that will be taken based on those rules. Rules provide logic of the "If this is true, then do that" variety.
2) The contract is then coded in an Ethereum high level language such as Solidity and deployed on the Ethereum blockchain platform.
3) The contract receives a public key address upon deployment, which allows it to be identified and its code to be executed. A smart contract cannot be changed after it has been deployed, not even by the EOA that created it.

A simple example of a smart contract could be an escrow account between two parties, without any third party involved. The two parties can agree on a set of criteria for a predetermined value of fund disbursement, and then create and code a smart contract and deploy it on Ethereum. As soon as the criteria for disbursing the funds are met, the contract can disburse out the funds to the appropriate party.

The Ethereum environment is inherently idle until a user initiates an action by sending a transaction, typically from an EOA. In practice, a transaction is a validated message sent by an EOA to another user account. A message of this type typically includes the following characteristics:

- The address of the recipient of the message.
- A signature of the sender that proves the owner of the account.
- A "*value*" field, including the amount of Ether to transfer from the sender to the recipient.
- An optional data field, which could be an arbitrary message, or function call to a contract, or even a code script that creates a contract.

In addition to the attributes mentioned above, transactions should include the maximum number of computational steps, along with the computational resources required. The transaction flow is depicted schematically in Figure 5 below.

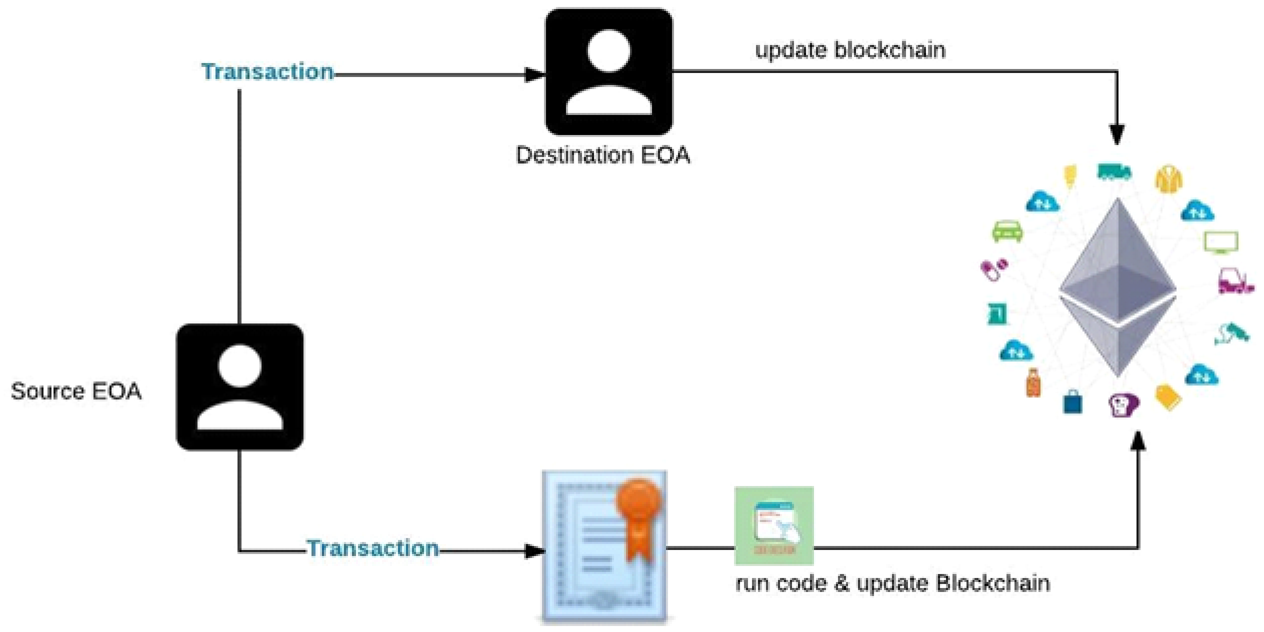


Figure 5: Transaction flow in Ethereum.

3.3 Proposed prototype design

3.3.1 Main design principles

Health-Chain, the current study's work product, is a web platform that uses a blockchain smart contract system for secure health data sharing. Its mission is to assist related health organizations, doctors, patients, and the medical community in securely storing and exchanging health transactions while being overseen by a Control Authority (CoA) entity. Chain-Health is the platform's main program, and it is in charge of the main communication between the user interface and the blockchain network. The following are the main design principles upon which the proposed Chain-Health System is based:

- There are two main user types, (1) the health entities which are health care institutions, doctors, or patients, and (2) the CoA which plays the role of the entity that controls and guarantees network transactions.
- In Health-Chain, each user has their own address, which is linked to specific roles and privileges. CoA, Doctor, Hospital, and Patient are the roles that are supported in this implementation.
- Any user can create a new account or login to one of his or her existing accounts to gain access to and use the services offered.
- In this proof of concept, the campaign concept is about creating requests, but it is also about transferring files for the needs of ONE patient. When the CoA creates a campaign, it generates a copy of the smart contract that will be used to carry out the system functions described above. So, in the end, every campaign is about a patient. The CoA oversees the Health-Petition subsystem, which is in charge of associating each campaign in the system with the patient it represents. Each campaign includes requests, and each approved request includes a file transfer between the parties involved.
- CoA has the authority to conduct campaigns. Users can participate in campaigns based on their roles and privileges.
- The exchanges that take place between the parties involved in each campaign are referred to as requests. Each data transaction is counted as one request by the platform.
- Users can submit requests. CoA has the authority to accept or reject a request, and every request results in a transaction. In Ethereum's immutable layer, all transaction data is stored in the form of blocks.

The core objective of the proposed implementation is to support health entities communicate effectively and transparently in a secure environment supervised by the aforementioned CoA. From the standpoint of functionality, the proposed implementation is based on a web application that allows users to:

- submit an application for national health Campaigns
- create requests for specific health data categories
- monitor the results of their transaction

The main principles on which functionality of the proposed application has been based are the following:

- Efficiency: From the perspective of the end user, the proposed application should provide speed and efficiency in access and functionality.
- User-friendliness: The proposed application should be easily accessible and user friendly to all users that need to access and work on it.
- Reliability: The proposed application should have proven and dependable functionality, as well as solid performance and stability.
- Security: The proposed application must be secure not only in terms of who has access to it, but also in terms of data management and communication between end users. Only administrators should be able to validate transactions.
- Maintainability: The proposed application should be simple to maintain, with as little disruption to its functionality and user access as possible.

Software technologies used to build Chain-Health, are a synthesis of React, Solidity, Javascript and Ethereum. All transactions are stored and validated by a PoA consensus mechanism.

In addition to the Chain-Health implementation, an assisting application has been developed to facilitate data transfer between the relevant parties. Because actual data cannot be transferred via this blockchain network, all interested parties must sign up for the Health-Petition System, which acts as a web copycat of the Chain-Health system since all relevant requests and campaigns can be found within it. All authorized users in Health-Petition can monitor the entire hash tree of the Chain-Health System, learn about each campaign, and submit and receive the files required to complete a request. From the standpoint of functionality, the proposed implementation is based on a web application that allows users to:

- associate patients with actual campaign addresses,
- keep track of the entire hash tree of requests for each campaign,
- send and receive health-related files through the Google Firebase Firestore Database.

Software technologies used to build Health-Petition, are a synthesis of Angular (Html-CSS & Typescript) and Google Firebase Cloud Firestore Database.

3.3.2 Main functional and non-functional Features

The proposed application provides a user-friendly front end, combined with the required security measures. Unregistered users do not have access to enter the application or view any transaction. At the same time, although registered users have access to the application, they can view detailed information of their own transactions but can only see very specific information of other users' transactions. The main pillars of functional requirements for the proposed prototype application are the following:

1) Registration of users: The application should provide the capability for new users to register online, using the appropriate web interface.
2) Automatic database update: When a new user registers, the application should automatically update the underlying database, with no further intervention from the administrator.
3) Execution of transactions: End users should be able to execute transactions and track their progress using blockchain structures such as blocks and hash algorithms.

In terms of non-functional requirements, the proposed application should comply with the following:

- Performance requirements: The system must have fast response, acceptable performance and scalability capabilities; application response should be fast, i.e., in the order of a few seconds for execution time for simple functions.

- Security and integrity requirements: The proposed application should meet stringent security requirements and ensure the integrity of data stored and transmitted between users. Only registered users with verified usernames and keys have access to and can execute transactions.
- Resilience requirements: The proposed application should be resilient, producing minimum errors from which recovery is possible within acceptable time frames.

The underlying swimlane diagram, along with proposed information structure of the proposed application is depicted in Figure 6.

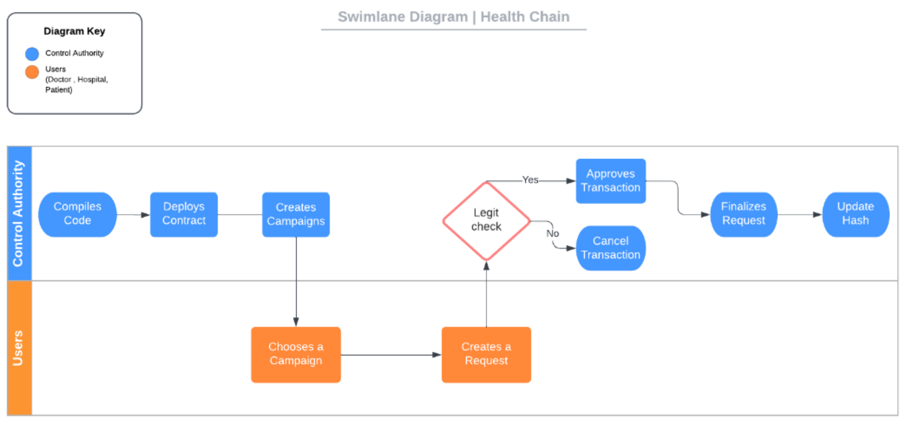


Figure 6: Activity flow of Chain-Health.

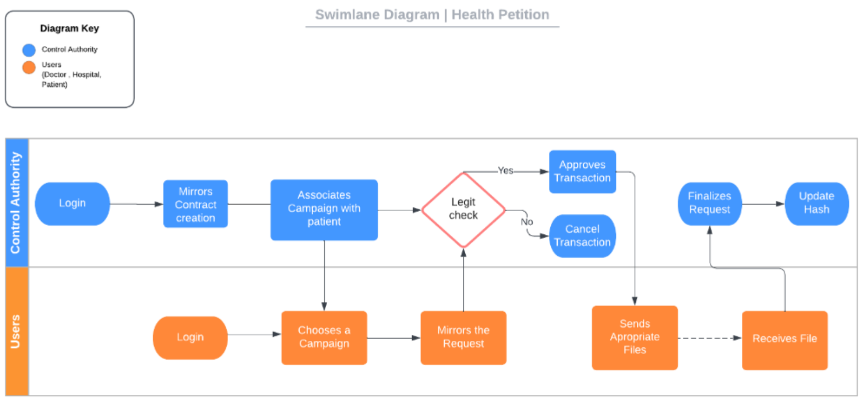


Figure 7: Activity flow of Health-Petition.

Capabilities of users, depending on their specific role, are also depicted in Figure 8.

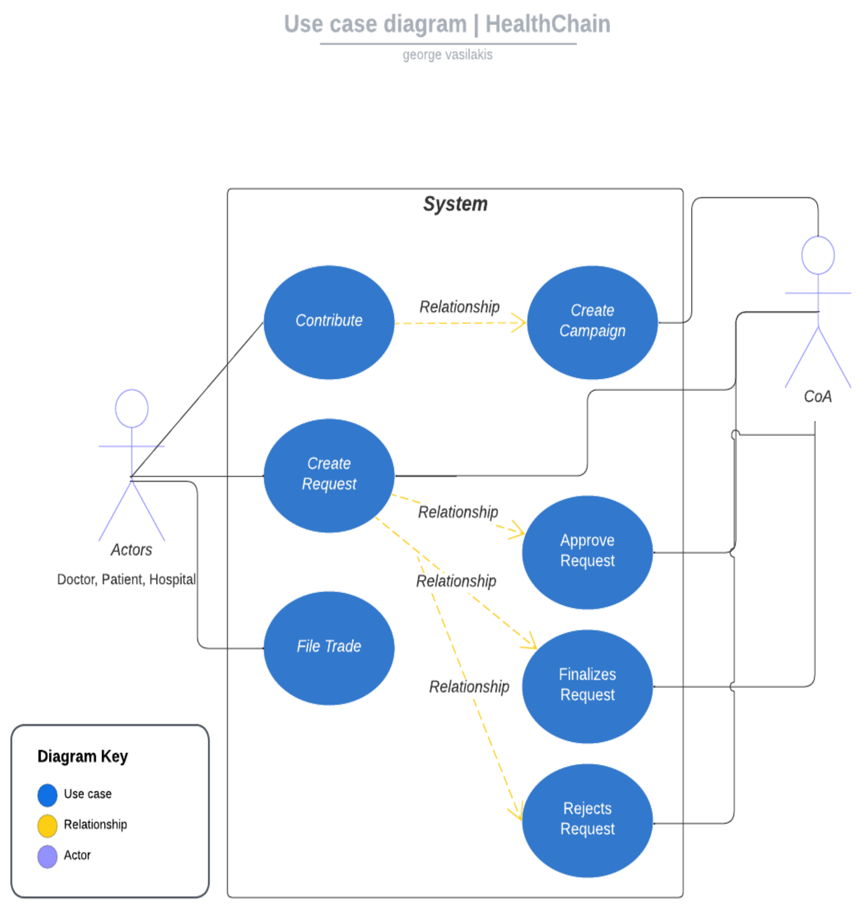
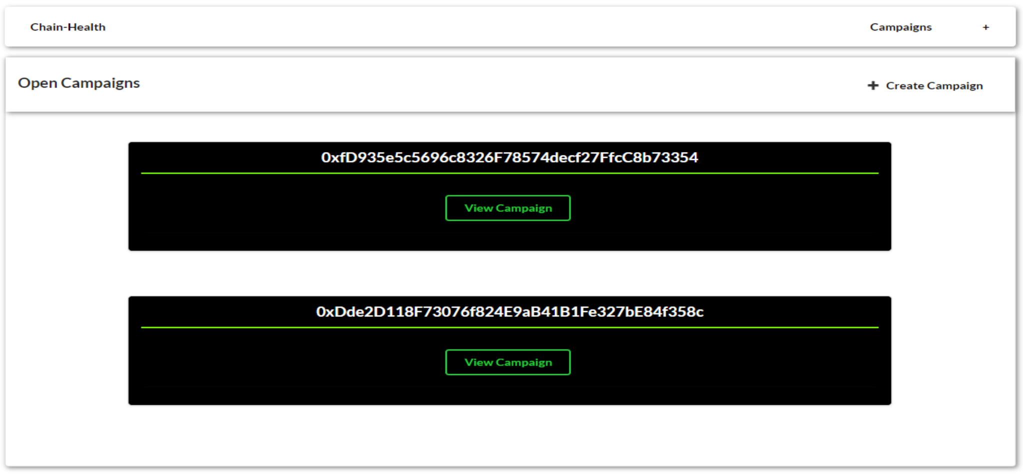


Figure 8: Use Case Diagram in Health-Chain.

3.3.3 Functional requirements in detail

As previously stated, the proposed Chain-Health proof of concept implementation is a smart contract system-based blockchain-based application for secure health data sharing. Its goal is to assist related health organizations and companies, doctors, patients, and the medical community in securely storing and exchanging health transactions under the supervision of a Control Authority (CoA) entity. Users can access this Web3 application using their personal credentials and based on their role, they will be given specific responsibilities. For example, only the CoA can approve a request and not a doctor. To sign up for and gain access to the application, a user must provide a specific unique identifier, which in our case is a unique address created manually by Metamask. A doctor named Georgios Vasilakis, for example, has his own unique address 0x3C0789F57e1f40B95D105000EBN7CM2cET27bafc38c0.

As system administrator, the CoA is in charge of configuring, managing, and maintaining the application. Furthermore, CoA is in charge of creating Campaigns for registered users. Campaigns are contracts created by CoA (for which users can apply) by completing the appropriate Campaign Form, as shown in Figure 3.6. For example, CoA could create a campaign called "Insurance Information," and registered users could apply for it by filling out the relevant campaign form and providing the necessary data and documentation. Figure 9 shows an example of a Campaign request.



0xfD935e5c5696c8326F78574decf27FfcC8b73354

View Campaign

0xDde2D118F73076f824E9aB41B1Fe327bE84f358c

View Campaign

Figure 9: Campaign set up screen of Chain-Health.

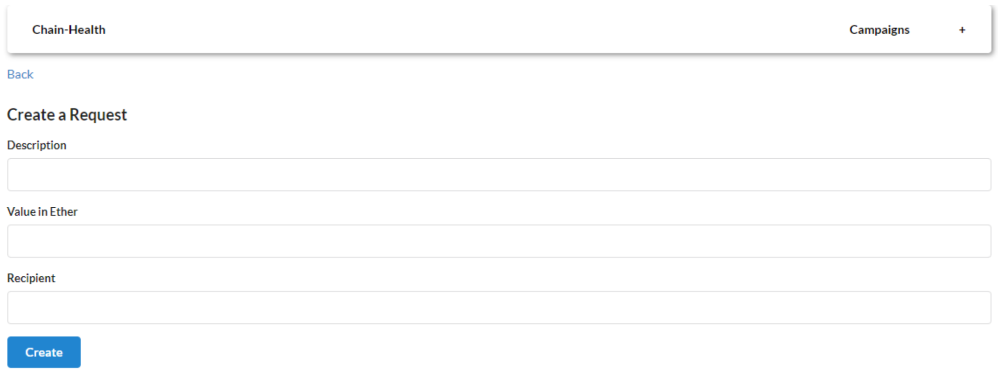


Figure 10: Campaign request screen of Chain-Health.

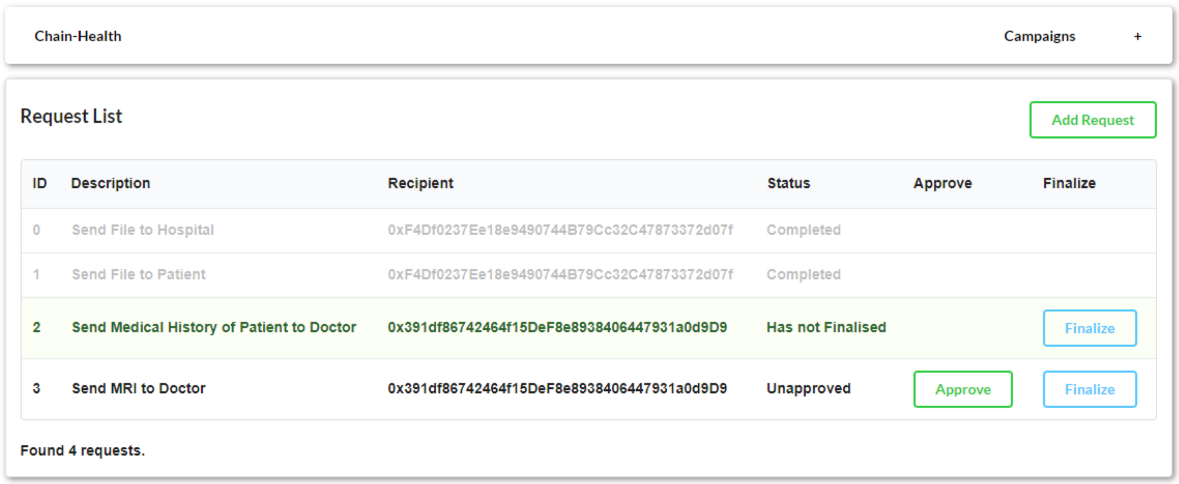
The following are the main characteristics of a CoA-generated campaign:

- A Campaign can only be created by the CoA.
- Every Campaign has a distinct address. This is necessary because it must be uniquely identified and mapped into the underlying hash tree.
- To create a campaign, the CoA must create a specific template for users who want to create campaign requests.
- Users create campaign requests by filling out the relevant request form.
- Every Campaign request has also its own distinct address.
- The CoA has the authority to approve or reject a Campaign request based on information and documentation provided by the requestor.
- Requests are saved in the application's hash tree upon approval or rejection. They are also saved as transactions in the private Ethereum PoA blockchain.

The Ethereum PoA blockchain was built using Geth, the official Ethereum client written in Google's "Go" programming language. Geth is in charge of making local copies of the Ethereum network's current state. The application also includes a "hash tree" that contains a list of requests that can be viewed by all network users because request addresses are appropriately encrypted for security and to make those requests untraceable. The following are the attributes of the requests as they are stored in the hash tree:

- address of the campaign as part of which the request has been created
- address of the specific request transaction
- CoA request evaluation result (approved or rejected).

Figure 11 presents an indicative request list screenshot.



Request List

ID	Description	Recipient	Status	Approve	Finalize
0	Send File to Hospital	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f	Completed		
1	Send File to Patient	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f	Completed		
2	Send Medical History of Patient to Doctor	0x391df86742464f15DeF8e8938406447931a0d9D9	Has not Finalised		Finalize
3	Send MRI to Doctor	0x391df86742464f15DeF8e8938406447931a0d9D9	Unapproved	Approve	Finalize

Found 4 requests.

Figure 11: Campaign request list screen of Chain-Health.

3.3.4 Non-Functional requirements in detail

As previously stated, the system must adhere to certain specifications in order to meet its non-functional requirements. To summarize, the three main pillars of these non-functional requirements that the system must meet in order to be considered stable and efficient are performance, security and integrity, and resilience.

1) Performance:

- Speed: These system components must be fast enough to compete with traditional applications that use a database and a web interface.
  - i. Smart Contract Compiling and Deploying: The total time required to compile and deploy the contract is **35.28** seconds, which is reasonable and efficient.
  - ii. Blockchain Launching: The total time to launch the Blockchain is **4.45** seconds.
  - iii. Campaign Creation (Block Time): The total elapsed time to create a new Campaign is **6.82** seconds.
  - iv. Request Creation (Block Time): The total elapsed time to create a new Request or “mine a new block” is **9.59** seconds.
  - v. Data Transfer and Link Acquisition: The total time to transfer a PDF and get a Download Link is **3.41** seconds. For a PNG file is **2.24** seconds

- Scalability: Because the application is built in web-based environments, it is quite extensible and already includes functions that can be extended in the future, such as contribution mechanisms, as well as the ease of creating extended smart contracts in the future without significantly changing the code. Concurrently, the contract factory-Campaign allows the user to generate copies of the contract without having to go through the compilation and deployment processes again.

2) Security and Integrity: The application page is currently inaccessible to non-authorized users. To add the network via Metamask, each user must obtain the necessary network information from the administrator, such as the IP address and chain ID. To use the utility, one must have an authorized username and password, both of which can only be created by the administrator. Transfer files and authentication credentials are also kept safe in the Google Firebase vault. Finally, even if the blockchain ledger is traced, the individual campaign's address cannot be linked to the patient.
3) Resilience: Because the current system is heavily reliant on the blockchain network, its efficiency is proportional to the network's ability to flow continuously. However, if the network fails, the ledger data remains in the system rather than disappearing. Simultaneously, the system allows the user to modify the smart contracts he has created to meet the needs of the users while also using them as a backup for flexibility and resiliency.

3.3.5 User types and privileges

As previously stated, the proposed implementation is linked to four (4) types of users: (1) CoA administrators, (2) Health Companies or Organizations, (3) Doctors, and (4) Patients. More specifically, the functional features for the aforementioned user roles are as follows:

1) CoA administrator: The Chain-Health application is built around the CoA. CoA is responsible for creating health campaigns for interested parties, offering health transactions to the user community, and finally monitoring the results of user requests. The CoA has the authority to accept or reject a Campaign request based on the information and documentation provided by the involved health entities.

2) Health Companies/ Organizations: The proposed blockchain application is used by health companies and organizations. They can apply for a company-specific campaign, resulting in the creation of a Campaign request tailored to this user role.

3) Doctors: Doctors can also be users of the proposed application. They can only apply to a doctor-specific campaign and they also need to provide specific documentation required by CoA to evaluate their requests.

4) Individuals: Individuals who work as doctors or in other medical-related fields or patients may also use the proposed application. They can only apply to a single campaign, and they must also provide the specific documentation required by CoA to evaluate their requests.

3.3.6 Overall prototype architecture

In this section the overall prototype architecture along with the proposed application interfaces are described. To support user access and proposed functional and non-functional requirements, the following user interfaces have been implemented:

1) Campaign List Interface: Every user of this interface has access to the full list of Health-Chain campaigns. The address of each campaign and a button for more information are provided in the fields on the relevant screen that supports this interface.

2) New Campaign Interface: The CoA uses this interface to create new Campaigns. When a Campaign is created, a unique Metamask key (address) is assigned to it, and a minimum Contribution is set for users to contribute.

3) Campaign Details Interface: On this interface, users can view all of the campaign's information, including the administrator's metamask address, the required minimum contribution in wei, the number of requests for each campaign, and the campaign's current balance. A form is also provided for users to contribute to the specific campaign.

4) Add Request Interface: This is the interface where users can create new requests based on predefined Campaign templates. The user types available in this interface should be the same as those available in the Campaign interface. This interface will include fields and selections for the user's Metamask key and the required request forms.

5) Requests List Interface: When users request to be notified of the outcome of their transactions, this interface will be invoked. The fields accessible via this interface will be the request ID, the description, the recipient's Metamask address, and the transaction result (approved or rejected).

Regarding software components, structural elements of the proposed prototype are the following:

1) the development software stack based on Javascript and React and enabled by Solidity
2) a cloud Amazon Web server Services server
3) the assistance software stack based on Angular
4) a Firestore Database on Google Firebase
5) a PoA consensus mechanism based on a Geth blockchain network

Figure 12 summarizes used software components and application interfaces.

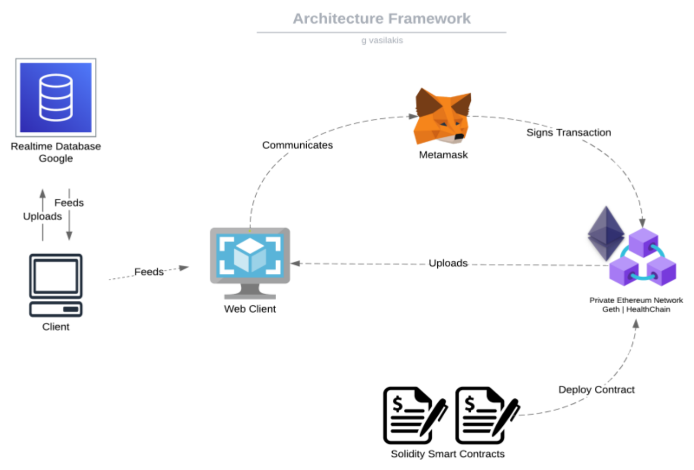


Figure 12: Logical architecture, software components and interfaces of Health-Chain.

3.3.7 Prototype Deployment and PoA with AWS

From an infrastructure standpoint, cloud Amazon Web Services (AWS) was chosen as part of the proof-of-concept implementation, while Ethereum Geth was used for PoA implementation and deployment. The exact steps taken for both deployments are described in this section.

3.3.7.1 Reserve required Infrastructure in AWS

The first step in the proof-of-concept implementation has been the provision of the necessary infrastructure and middleware on which the Ethereum PoA will be deployed. The AWS cloud platform was used for this purpose because it provides speed and flexibility in resource provision and configuration. The following are the steps taken as part of the prototype implementation for infrastructure provision and configuration:

1) Create an Ubuntu Virtual Machine (VM) instance in AWS with the below configuration settings:

    a) a x86 Operating System

    b) a t2. micro type instance with its standard settings

    c) 16GB storage

2) Define and open the ports of the below table with the indicated settings

Type	**Protocol**	**Port**	**Source**
ssh	tcp	22	anywhere
http	tcp	8545	anywhere
custom tcp rule	tcp	30303	anywhere
custom tcp rule	tcp	30301	anywhere
custom ICMP rule-IPv4	echo reply	n/a	anywhere
https	tcp	443	anywhere

3) Create, download and save locally a new "*key pair*" (.pem file) for future use.

4) Create an elastic IP address and associate it with the instance created to avoid changing IP every time the instance is being restarted.

3.3.7.2 Proof of Authority (PoA) Blockchain Network using Geth

The environment is ready to proceed with the deployment of the Ethereum PoA blockchain network now that the necessary infrastructure is in place and properly configured. The Go Ethereum (Geth) client is used for this purpose. Geth is the standalone client for Ethereum that is required to create and run a node on any Ethereum network. By "running a node," network users can execute transactions and interact with one another. Technically, this is accomplished through the use of smart contracts, which are programs that define the functionality of the Ethereum blockchain. To participate in the chain, each node or Ethereum blockchain network runs a Geth client. Geth's most important code modules and functionalities are as follows:

- Core module: The core code module of Geth is responsible for the management of blocks, transactions, etc., but also contains the Ethereum Virtual Machine (EVM), which is responsible for executing the smart contract code.
- Eth module: This part of the code is responsible to keep any local blockchain network synced with the overall Ethereum blockchain network.
- Ethdb: This module is responsible for accessing and managing the database where the Ethereum blockchain is stored, a key value database called LevelDB.
- Consensus module: In this part of Geth code the Ethereum consensus mining algorithm *ethhash* is implemented.
- Miner module: This part of the Geth code is responsible for the mining process in Ethereum blockchain. It uses the *ethhash* mining algorithm that defines the consensus mechanism.
- Rpc module: This module takes care of definition and management of all Geth interfaces, such as the web socket API and the json rpc API, thus providing the capability for a fast and efficient interaction with other Geth clients.

- Params module: In this part of the code, configuration of the various Geth parameters, such as the block numbers, is included.
- P2p: This is a low-level networking library responsible for peer-to-peer exchange of data, between network nodes.
- Tests folder: This part of the Geth code includes integration tests needed to ensure that performed implementations are compatible with the specifications of Ethereum protocol.
- Accounts module. The main functionality of this module is management of Ethereum wallets.
- Cmd: This is the command line interface of Geth, which is used in order to interact with the Geth client with Geth commands.

The steps required to develop the proposed prototype of the Ethereum PoA Blockchain network, are the following:

1) Connect to the Ubuntu VM created and configured as explained in the previous section. The "*key pair*" .pem file (in our case aws_key_pair.pem) defined and stored during VM creation will act as password for this connection. The sub-steps needed for this connection are:
   a) Open a new command line terminal (such as the Git Bash) on the location where the .pem file has been stored, to define and manage the connection.
   b) On the new terminal, type chmod 400 aws_key_pair.pem.
   c) To log in the VM, type ***ssh -i "aws_key_pair.pem" ubuntu@ec2-3-9-185-202.eu-west-2.compute.amazonaws.com*** (the command with the proper attributes has been derived from the "*Connect to your Instance*" interface after selecting "*Connect to instance*" in AWS).
2) Install Geth in the VM just connected, by typing the following commands:
   a) ***sudo -s*** to obtain root privileges
   b) sudo add-apt-repository -y ppa:ethereum/ethereum
   c) sudo apt-get update
   d) ***sudo apt-get install ethereum***

   e) ***geth*** to make sure the gate is installed on the VM and Ctrl+C to stop it if everything was installed and runs properly.

3) Verify that ports opened are working properly. This could be done by opening new terminal and typing ***ping 3.9.185.202***; if ping receives response, the connection is ok, if it times out, port definition and configuration must be reviewed and updated appropriately.

4) Create a folder (boot_health) and required subfolders (node1, bootnode) responsible for the deployed network, by typing the following commands:

   a) ***sudo -s*** command

   b) ***cd /opt/*** (enter opt folder)

   c) ***mkdir boot_health***

   d) cd boot_health

   e) ***mkdir node1*** (a directory acting as a "*miner*")

   f) ***mkdir bootnode*** (a directory acting as the first node of the blockchain)

5) For node1 create an Ethereum node account, by typing the following:

   a) cd node1

   b) ***geth --datadir "./data" account new*** (create a new account using geth and store it in the folder named data). The new account is locked with a password. Provide and repeat a password for your Ethereum account. Also, a public address of the key is provided, which is 0xc63B93AB994BF05F37e96e1270Ddb03c1E8a9740. This is the address of your node1.

   c) ***ls-la*** to see account details in the folder. Files in this folder should remain unchanged.

6) Build genesis configurations by running puppeth command in health_chain directory, and providing requested information:

   Please specify name: type **boot_health** *(no spaces, hyphen or capital)*

   What you would like to do: select ***2 (= Configure new genesis)***

   What you would like to do: select ***1 (= Create genesis from scratch)***

Consensus engine: select ***2 (= clique – proof-of-authority)***

How many secs? type ***5 (= Any number of your choice actually)***

Which accounts are allowed to seal? type
0x56E1767E67F2335104eC08D69885E84Eb6f4FEBE

Which accounts should be pre-funded? type
0x56E1767E67F2335104eC08D69885E84Eb6f4FEBE

pre-compile addresses: type ***yes***

*Specify the network ID: type 7****979***

7) Export genesis configurations file by running again puppeth command in node1 directory, and providing requested information:

   Please specify name: type **boot_health**

   What you would like to do: select ***2 (= Manage existing genesis)***

   What you would like to do: select ***2 (= Export genesis configurations)***

   type "enter" to export files in current directory

As a result boot_health_harmony.json and boot_health.json files are generated in folder node1.

8) Initialize genesis file on node1 using geth by typing the below command. Proper initialization is confirmed by a message "*Successfully wrote genesis file*".

   geth --datadir ./data init ./boot_health.json

   Proper initialization is confirmed by a message "*Successfully wrote genesis state*".

9) Create and initialize a boot node (acting as the first node of the blockchain), under bootnode directory, by typing the following:

   a) ***cd bootnode*** (enter in the boot node directory)

   b) ***bootnode -genkey boot.key*** (generate boot node's key)

   c) ***bootnode -nodekey boot.key*** (start the network). This provides the enode address which is required for other nodes to connect to boot node. The enode address produced is:

enode://172f4c19c675fd39fdf4f1ec8cc36530e1762ff57c885f94f7d207e3c3ade79d6e9 14d268478b0153eedaac707f33b22f92208097bca16941548a8c333a01bb@127.0.0.1:0?discport=30301

Prepare to connect node1 to bootnode.

a) Create a passwort.txt file under the node1 directory and store the selected password (123456 in our case).

10) Launch created network by executing the below commands:

a) In bootnode directory start boot node by typing the command ***bootnode -nodekey "./boot.key" -verbosity 7 -addr "127.0.0.1:30301"***

b) In node1 directory run the following (one command) to start mining blocks:

sudo ./launch.sh,

Where the complete command is

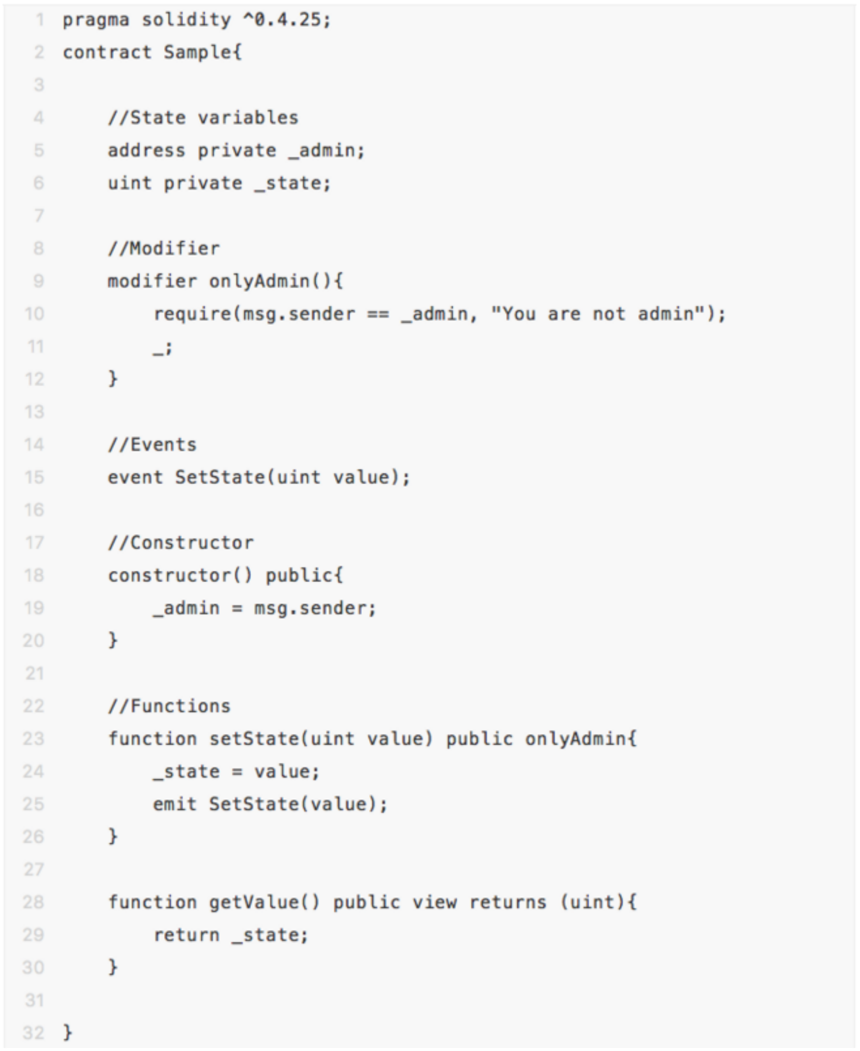
geth --networkid 7979 --datadir ./data --port 30303 --ipcdisable --syncmode full --http --allow-insecure-unlock --hrrp.corsdomain "" --http.port 8545 --http.addr "172.31.12.115" --unlock 0x56E1767E67F2335104eC08D69885E84Eb6f4FEBE --password password.txt --mine --http.api personal,admin,db,eth,net,miner,ssh,txpool,debug,clique --ws --ws.addr 0.0.0.0 --ws.port 8546 --ws.origins "*" --ws.api personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique --maxpeers 25 --miner.gasprice 0 --miner.etherbase*

3.3.8 Prototype smart contracts

3.3.8.1 The Solidity programming language

As previously stated, the proposed prototype includes fully functional smart contracts. Solidity is an object-oriented programming language that was developed to aid in the implementation of Ethereum's blockchain smart contracts. Solidity's syntax is curly-bracket or curly-brace, which means that statement blocks are defined and separated by curly brackets, i.e., and. Solidity's creators were inspired by C++, JavaScript, and Python, while its primary use is intended to be the implementation of smart contracts within Ethereum Virtual Machines (EVMs). Solidity's main characteristics are that it is statically typed, supports inheritance, and employs libraries. Voting, crowdfunding, auctions, and wallets are the most popular blockchain smart contracts that can be built with Solidity.

Solidity is a Turing-complete language that is widely used for blockchain platform development, with features such as function calls, constructors, modifiers, overloading, event management, inheritance, and others. Based on the sample Solidity program structure shown in Figure 13, the following sections present and describe the basic solidity features required to develop smart contracts.



```
pragma solidity ^0.4.25;
contract Sample{

    //State variables
    address private _admin;
    uint private _state;

    //Modifier
    modifier onlyAdmin(){
        require(msg.sender == _admin, "You are not admin");
        _;
    }

    //Events
    event SetState(uint value);

    //Constructor
    constructor() public{
        _admin = msg.sender;
    }

    //Functions
    function setState(uint value) public onlyAdmin{
        _state = value;
        emit SetState(value);
    }

    function getValue() public view returns (uint){
        return _state;
    }

}
```

Figure 13: Sample Solidity program structure.

The main sections of this program could be described as follows.

1. **State variables** (_admin, _state) are variables that are permanently stored and can be modified by functions. Typically, the business information of a smart contract is included in this type of variable. They can be altered by user-defined functions that can be included in any transaction. The syntax of those variables is: <*type*><*access modifier* – (*optional*)><*variable name*>.
2. **Constructors** are structures that are used for smart contract deployment and initialization. Constructors are used in practice to pass significant contract data to state variables. It should be noted that Solidity constructors are unique in that they do not support overloading, unlike other languages such as Java.
3. **Events** (SetState) are structures similar to logs, used to record the occurrence of events in a blockchain network. Their names do not have to be related with the function names from which they are called, even though it is recommended to do so. Clause "*emit*" is being used to distinguish events from functions.
4. **Modifiers** (onlyadmin) are critical components of a smart contract. Their primary function is to modify existing functions before they are called by extending their natural behavior with extra functionality such as cleansing, checking and other similar activities.
5. **Functions** (setState, getValue) are commonly used to read from, and write to state variables. Changes to those variables can be incorporated into transactions, and their values can be stored in the blockchain's immutable ledger. Other function modifiers, such as the "view", or the "pure" modifier might also be applied. Function definition attributes might be input and output parameters, and modifiers, while it very common that they return single or multiple values.

The most common Solidity data types used for Solidity variables are the following:

1. <u>Integer</u>: Variables of this type can be signed of various lengths (int, int8, int16, …, int256), or unsigned of various lengths (uint, uint8, uint16, …, uint256).
2. <u>Fixed length bytes</u>: Variables of this type can be of a predetermined fixed length in bytes from one (1) to thirty-two (32) bytes (bytes1, bytes2, …, bytes32).

3. Variable length bytes: In addition to the fixed length bytes data type, Solidity provides the variable length data type, which in practice is used for array variables of non-fixed length.
4. String: This type is used for variable length string variables and is compatible with the variable length bytes data type.
5. Address: Variables of this type are used to store account or node addresses generated for example by private keys.
6. Mapping: This data type is used for variables that store the hash value of a key.
7. Array: The array data type is used to encode variables that might contain variable length arrays. Array data type can be used for either state or local variables.
8. Struct: Solidity, like many other programming languages, includes the structure data type, which can be used to create specific structure objects. Structures can be used to represent both state and local variables.

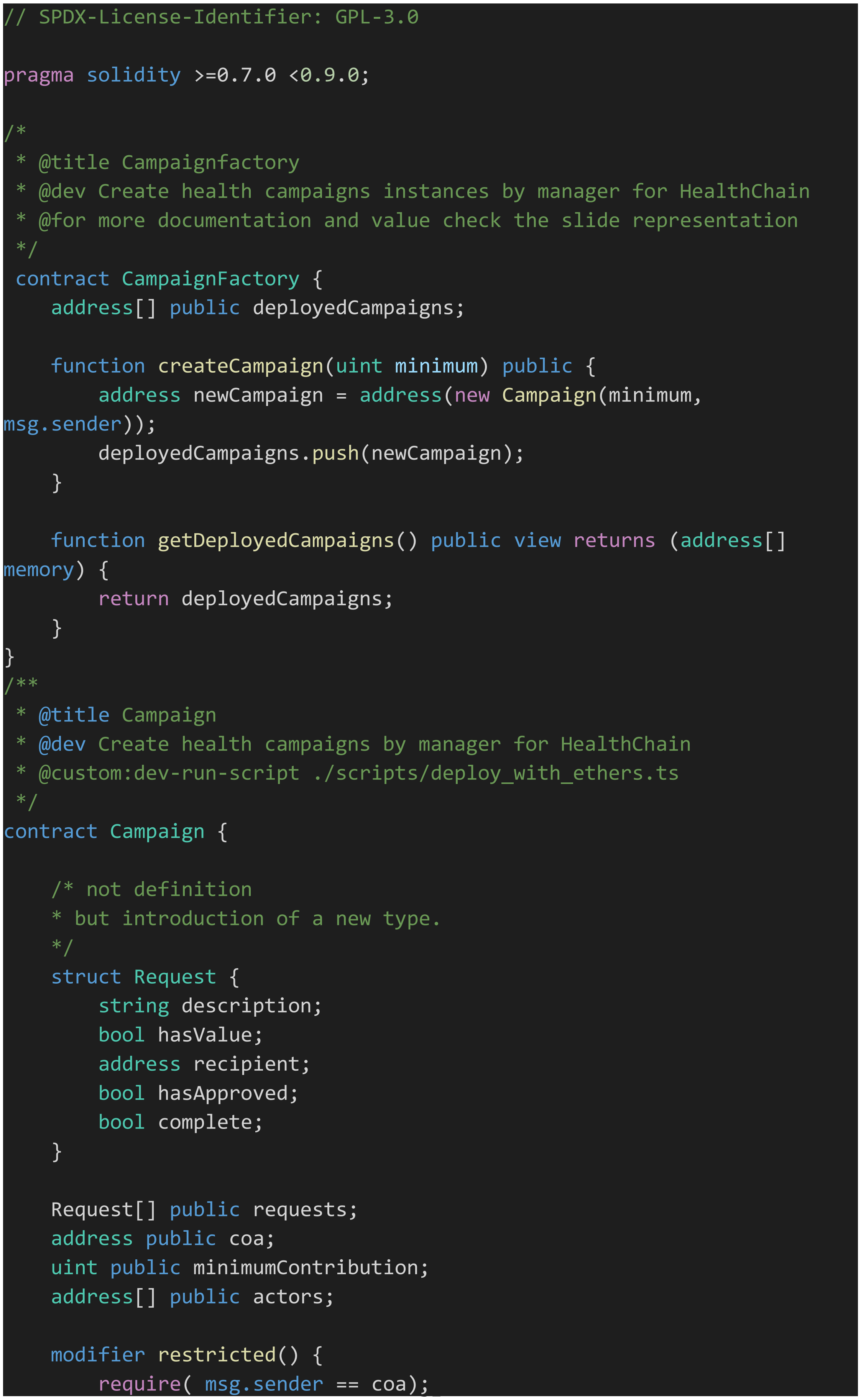
Solidity also supports global variables (or global methods). Their primary function is to provide basic information about a current block or transaction, such as block time, contract caller, and so on. The following are the most common global variables:

- **msg.sender**: Represents a direct call to a smart contract.
- **tx.origin**: Returns the initiator of a transaction, i.e., the starting point of a call chain.
- **msg.calldata**: Returns detailed call information, such as function identification, parameters, etc.
- **msg.sig**: Contains the first four (4) bytes of msg.calldata, needed to identify the function.
- **block.number**: Returns the current block number.
- **now**: Returns a timestamp with the current data and time.

There are several options for editing, compiling, debugging, and running a Solidity program, including the fisco bcos console, the fisco web-based IDE, and the Remix Online IDE, which is the most popular and will be used as part of the proposed prototype implementation. Remix Online IDE includes communication development tools and functionality such as a scripting editor, compilation, debugging, and deployment, as well as transaction logging.

3.3.8.2 Proposed smart contract implementation

The Solidity implementation used to support the smart contract of the proposed prototype is presented in the following sections. With the smart contract in place, the CoA can create and submit new campaigns, while health professionals can submit requests to participate, triggering CoA approval or rejection.



```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/*
 * @title Campaignfactory
 * @dev Create health campaigns instances by manager for HealthChain
 * @for more documentation and value check the slide representation
 */
 contract CampaignFactory {
    address[] public deployedCampaigns;

    function createCampaign(uint minimum) public {
        address newCampaign = address(new Campaign(minimum,
msg.sender));
        deployedCampaigns.push(newCampaign);
    }

    function getDeployedCampaigns() public view returns (address[]
memory) {
        return deployedCampaigns;
    }
}
/**
 * @title Campaign
 * @dev Create health campaigns by manager for HealthChain
 * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
 */
contract Campaign {

    /* not definition
    * but introduction of a new type.
    */
    struct Request {
        string description;
        bool hasValue;
        address recipient;
        bool hasApproved;
        bool complete;
    }

    Request[] public requests;
    address public coa;
    uint public minimumContribution;
    address[] public actors;

    modifier restricted() {
        require( msg.sender == coa);
```

3.3.9 Prototype front end

End users of the proposed blockchain network, interact with each other by using specific user interface capabilities developed as part of the proposed prototype. The front end has been developed using React JS for creating web-based interfaces with all user types. Creation and execution of campaigns, submission of requests along with their response (approval or rejection), and view of campaign transactions are supported by the proposed front-end implementation. Indicatively, the React JS code for creating and interacting with a new campaign is presented below.

```
class CampaignNew extends Component {
    state = {
        minimumContribution: '',
        errorMessage: '',
        loading: false
    };

    onSubmit = async (event) => {
        event.preventDefault();
        this.setState({ loading: true, errorMessage: '' });
        try {
            const accounts = await web3.eth.getAccounts();
            await factory.methods
                .createCampaign(this.state.minimumContribution)
                .send({
                    from: accounts[0]
                });
            Router.pushRoute('/');
        } catch(err) {
            this.setState({ errorMessage: err.message })
        }
        this.setState({ loading: false });
    };

    render() {
        return(
            <Layout>
                <h3> Create Campaign </h3>
                <Form onSubmit={this.onSubmit}
error={!!this.state.errorMessage}>
                    <Form.Field>
                        <label>Minimum Contribution</label>
                        <Input
                        label="wei"
                        labelPosition='right'
                        value={this.state.minimumContribution}
                        onChange={ event => {
                            this.setState({ minimumContribution:
event.target.value })
                        }}
                        />
                    </Form.Field>
                    <Message error header="Oops!"
content={this.state.errorMessage} />
                    <Button color='green' loading={this.state.loading}>
                         Create
                    </Button>
                </Form>
```

Moreover, the JS code for compiling a new Campaign Contract is presented below.

```
const path = require('path');
const fs = require('fs-extra');
const solc = require('solc');

const buildPath = path.resolve(__dirname, 'build');

fs.removeSync(buildPath);

const campaignPath = path.resolve(__dirname, 'contracts', 'Cam-
paign.sol');
const source = fs.readFileSync(campaignPath, 'utf8');


var input = {
    language: 'Solidity',
    sources: {
      'Campaign.sol': {
        content: source
      }
    },
    settings: {
      outputSelection: {
        '*': {
          '*': ['*']
        }
      }
    }
  };

  const output = JSON.parse(solc.compile(JSON.stringify(input))).con-
tracts[
    "Campaign.sol"
  ];

  fs.ensureDirSync(buildPath);

  solc.loadRemoteVersion('v0.8.11+commit.d7f03943', function(err,
solcSnapshot) {
    if (err) {
      reject(false);
    } else {
      for(let contract in output) {
          fs.outputJSONSync(
              path.resolve(buildPath, contract.replace(":", "") +
".json"),
              output[contract]
          );
      }
```

Finally, the JS code for deploying a new Compiled **campaign factory** is presented below.

```
const HDWalletProvider = require("@truffle/hdwallet-provider");
const Web3 = require("web3");

const mnemonicPhrase = "***** ***** ***** ***** ***** ***** ***** *****
*****  *****  *****  ***** ";
const rinkeby_endpoint =
"https://rinkeby.infura.io/v3/40fe6433ded94800b0139cbc7abeeb9c";
const healthchain_endpoint = "http://3.9.185.202:8545";

const provider = new HDWalletProvider(mnemonicPhrase,
healthchain_endpoint);
const web3 = new Web3(provider);

const compiledFactory = require("./build/CampaignFactory.json");
const abi = compiledFactory.abi
const bytecode = compiledFactory.evm.bytecode.object

const deploy = async () => {
  const accounts = await web3.eth.getAccounts();
  console.log("Attempting to deploy from account", accounts[0]);
  const result = await new web3.eth.Contract(abi)
    .deploy({
      data: '0x' + bytecode,
    })
    .send({
      gas:"1400000",
      from: accounts[1],
    })
    .catch(err => console.log(err))

  console.log("Contract deployed to", result.options.address);
  provider.engine.stop();
};

deploy();
```

3.3.10 Other prototype support components

3.3.10.1 Configuration and usage of Metamask

Metamask, which can natively support "wallet" functionality, is one of the tools used to support the current prototype implementation. MetaMask provides access to transaction requests through a simple and easy-to-use front end, while also acting as an Ethereum cryptocurrency wallet. Metamask works as a browser plugin for Chrome, Firefox, and other browsers One of its main advantages is that it can connect to existing Ethereum networks as well as new custom private Ethereum nodes. Using the Metamask wallet, it can function as a vault, storing, sending, or receiving actual or "test" Ethers with their respective local security keys. Before beginning to use Metamask, one must first create an account; during account creation, a "secret seed" is provided, which should be remembered in order to gain future access to Ethers. A password should also be created in order to gain access to the MetaMask.

Metamask includes pre-configured integration with the main Ethereum network (Main net), which is used to support miners who "dig" for Ethers. As shown in Figure 14, Metamask is also pre-integrated with a number of test Ethereum networks (Ropsten, Kovan, Rinkeby, and Goerli) created and used by developer communities. The mainnet is the official Ethereum production network, where actual transactions are carried out. Ropsten is the official Ethereum test network, where transactions and smart contract deployments are tested before being deployed to production. The PoW (Proof of Work) consensus mechanism is used, but the transactions are only for testing purposes. Rinkebay is a PoA (Proof of Authority) consensus-based Ethereum test. It is important to note that available Ethers in a Metamask account are linked to the network from which they were obtained.

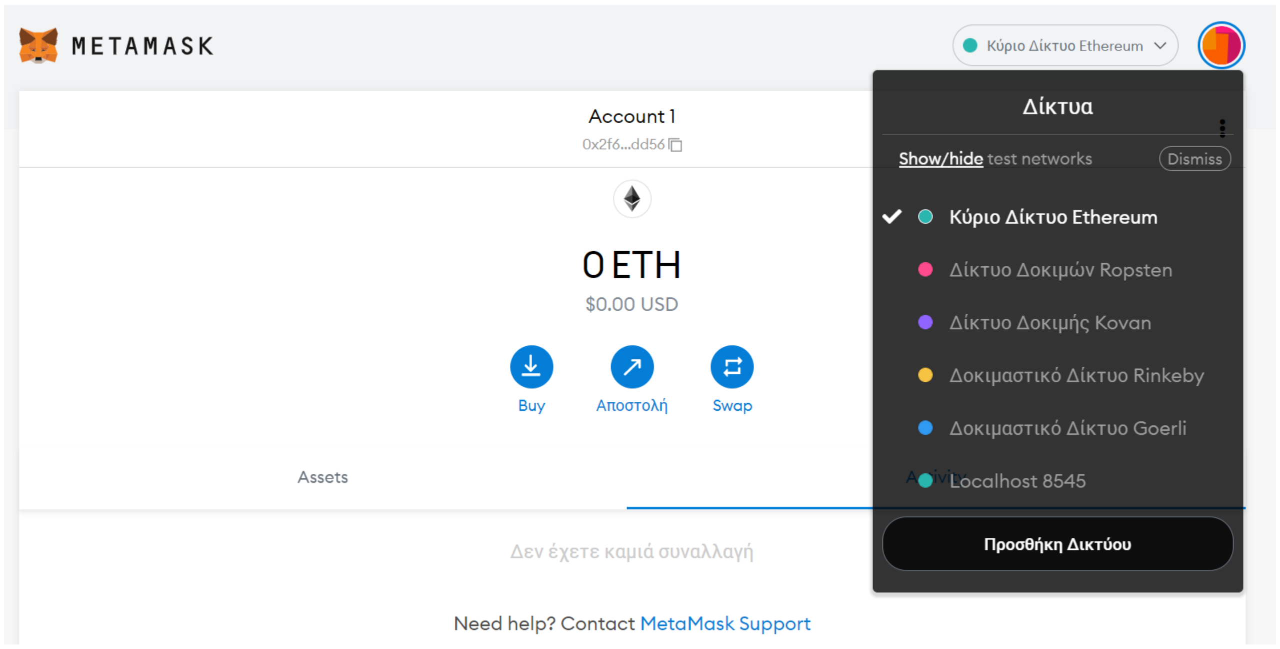
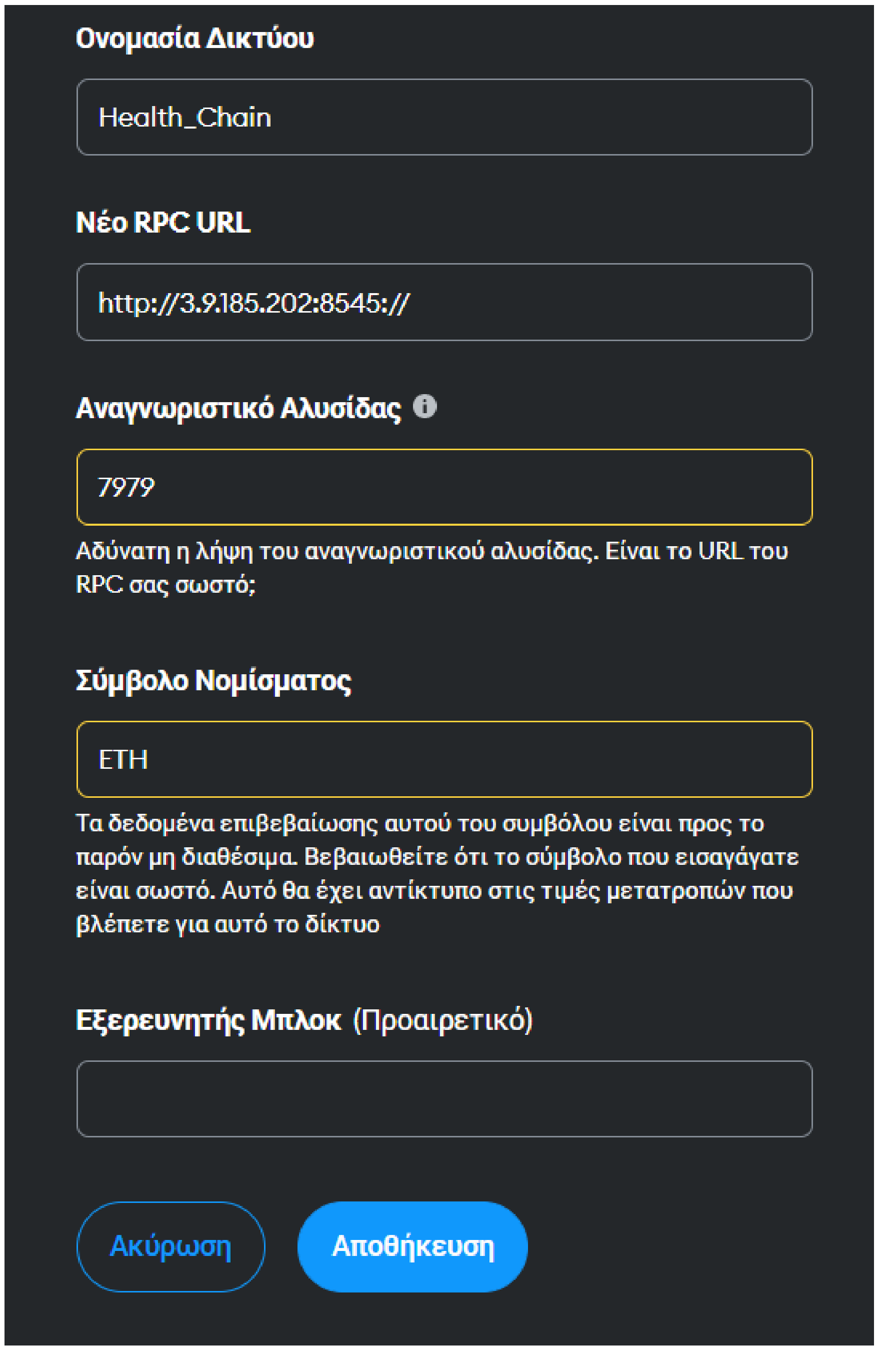


Figure 14: Available Ethereum Blockchain Networks in Metamask.

Developers can easily create their own integrations on top of the pre-integrated networks by using the RPC protocol or exploiting connectivity with the localhost. The localhost integration capability was chosen and configured for the current implementation. As shown in Figure 15, the localhost's IP address is 127.0.0.1 and the local port is 8545. The following attributes must be defined as part of the overall configuration of the localhost test network:

- The name of the network under construction
- The RPC URL and port
- The ChainID
- The supported currency symbol (optional attribute)
- The URL of the block explorer (optional attribute)


Ονομασία Δικτύου

Health_Chain

Νέο RPC URL

http://3.9.185.202:8545://

Αναγνωριστικό Αλυσίδας

7979

Αδύνατη η λήψη του αναγνωριστικού αλυσίδας. Είναι το URL του RPC σας σωστό;

Σύμβολο Νομίσματος

ETH

Τα δεδομένα επιβεβαίωσης αυτού του συμβόλου είναι προς το παρόν μη διαθέσιμα. Βεβαιωθείτε ότι το σύμβολο που εισαγάγατε είναι σωστό. Αυτό θα έχει αντίκτυπο στις τιμές μετατροπών που βλέπετε για αυτό το δίκτυο

Εξερευνητής Μπλοκ (Προαιρετικό)

Figure 15: Using Localhost Test Network with Metamask.

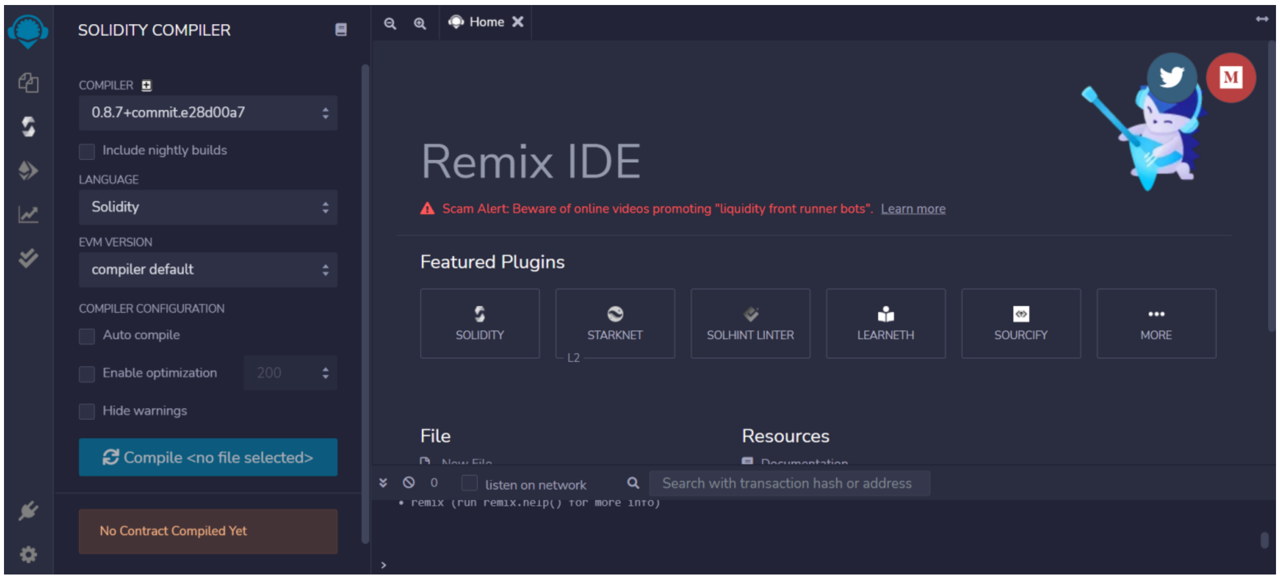
Developers can easily create their own integrations on top of the pre-integrated networks by using the RPC protocol or exploiting connectivity with the localhost. The localhost integration capability was chosen and configured for the current implementation. As shown in Figure 3.3.10, the network's elastic IP address is 3.9.185.202 and the local port is 8545. The following attributes must be defined as part of the overall configuration of the localhost test network:

ssh -i aws_key_pair.pem -N -v ubuntu@ec2-3-9-185-202.eu-west-2.compute.amazonaws.

Network traffic is directed to the appropriate AWS port 22, already configured to accept SSH connection requests from any IP address. Since a continuous connection without interrupts (tunnel) is required, the "-N" attribute is used to ensure that no remote command is executed. Additionally, to provide visible logging to the SSH window, the "-v" attribute is used. Connection is established, using the AWS elastic IP address, already configured as explained in Section 3.3.6. With the above script, any traffic received by the local machine on port 8545, which is the port listening RPC requests, is transparently and securely redirected to the AWS blockchain node (port 22) through the established tunnel.

3.3.10.2 Creation and deployment of smart contracts

To create and test a smart contract, as already stated, Solidity scripting language has been used. In terms of tooling, this is done by using Remix IDE, a tool set easily accessible directly through the URL https://remix.ethereum.org. Remix is a development IDE supported by Ethereum organization with a Solidity compiler (as a plug in) and contract execution capabilities incorporated. Figure 16 below shows the main introductory screenshot of Remix IDE.



COMPILER 0.8.7+commit.e28d00a7

LANGUAGE Solidity

EVM VERSION compiler default

Enable optimization 200

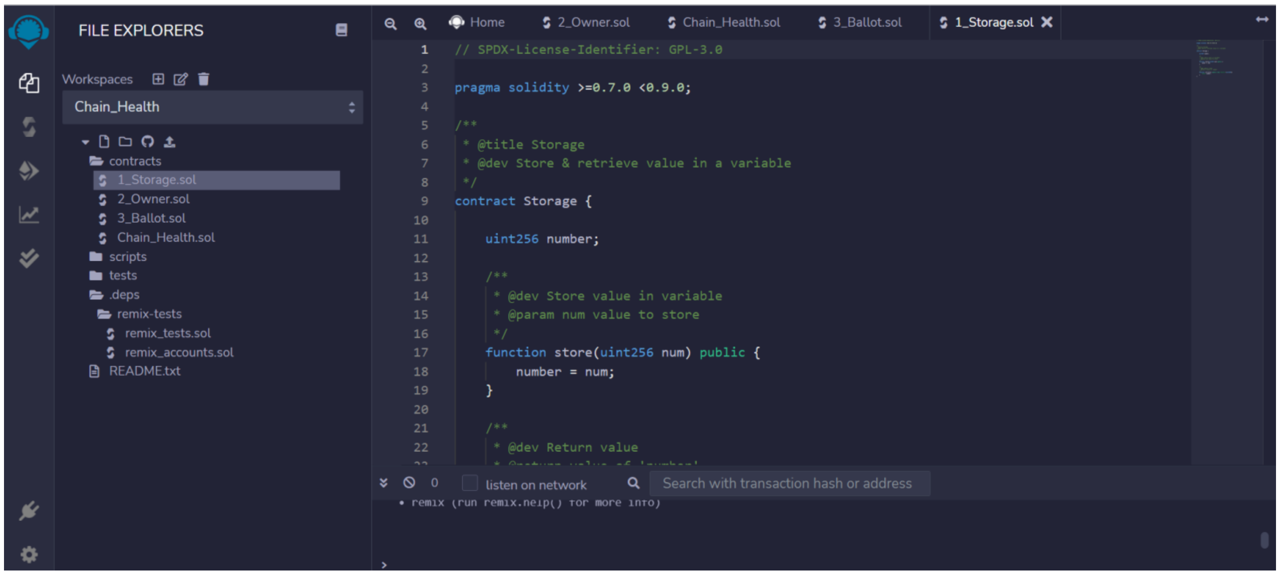
No Contract Compiled Yet

Scam Alert: Beware of online videos promoting "liquidity front runner bots". Learn more

```
• remix (run remix.help() for more info)
```

Figure 16: Remix IDE tool used for the development of Health_Chain prototype.

Remix IDE, for example, includes an editor for creating the Solidity code required for the proposed prototype implementation, as shown in Figure 17 below. This is the primary reason it was chosen. Chain_Health.sol is the name of the smart contract created to support the Chain Health prototype.



```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */
contract Storage {

    uint256 number;

    /**
     * @dev Store value in variable
     * @param num value to store
     */
    function store(uint256 num) public {
        number = num;
    }

    /**
     * @dev Return value
```

```
• remix (run remix.help() for more info)
>
```

Figure 17: Solidity coding editor in Remix IDE tool used for the development of smart contracts in Chain_Health prototype.

3.3.11 Prototype integration and interoperability

This section describes the orchestration and interoperability of the various tools and solution components used in the proposed prototype. Obviously, the implemented prototype is complex because a variety of different solutions, programming languages, and tools are used and must be integrated to make the prototype functional. The use of cloud and local infrastructure elements adds to the complexity, and interoperability between them has proven to be difficult at times. The following components have been integrated to make the overall prototype functional and interoperable:

1. <u>AWS infrastructure:</u> From an infrastructure standpoint, cloud Amazon Web Services (AWS) has been chosen as part of the prototype implementation. The provision of the necessary infrastructure and middleware on which the Ethereum PoA will be deployed is critical to

ensuring the proposed implementation's scalability and expandability. The AWS cloud platform allows for rapid resource provisioning and configuration.

2. Ethereum network deployment and Geth client: With the required infrastructure in place and properly configured, deployment of the Ethereum PoA blockchain network has been made feasible. For this purpose, Go Ethereum (Geth) client has been used. Geth is Ethereum's standalone client required to create and run a node on any Ethereum network to enable participation in the chain. By "*running a node*", the network users can execute transactions and interact with each other. This is technically achieved with the aid of smart contracts, which are programs that define the functionality of the Ethereum blockchain, as will be explained later in this section.

3. Metamask browser extension: This is used in conjunction with a Chrome browser to improve end-user interaction and experience. MetaMask provides access to transaction requests through a simple and easy-to-use front end, while also acting as an Ethereum cryptocurrency wallet. One of its main advantages is that it can connect to existing Ethereum networks as well as new custom private Ethereum nodes. Using the Metamask wallet, it can function as a vault, storing, sending, or receiving actual or "test" Ethers with their respective local security keys. Before beginning to use Metamask, one must first create an account; during account creation, a "secret seed" is provided, which should be remembered in order to gain future access to Ethers. A password should also be created in order to gain access to the MetaMask. Metamask includes pre-configured integration with the main Ethereum network (Main net), which is used to support miners who "dig" for Ethers. It is also pre-integrated with a number of test Ethereum networks created and used by developer communities (Ropsten, Kovan, Rinkeby, Goerli). The mainnet is the official Ethereum production network, where actual transactions are carried out. Ropsten is the official Ethereum test network, where transactions and smart contract deployments are tested before being deployed to production. The PoW (Proof of Work) consensus mechanism is used, but the transactions are only for testing purposes. Rinkebay is a PoA (Proof of Authority) consensus-based Ethereum test. It is important to note that available Ethers in a Metamask account are linked to the network from which they were obtained. Developers can easily create their own integrations on top of the pre-integrated networks by using the RPC protocol or exploiting connectivity with the localhost.

4. Git Bash command shell and "ssh tunnel": This tool was used to make network communication and interoperability between a local PC and AWS infrastructure easier. Connectivity is achieved by using "ssh" with the necessary attributes, such as the key that allows access to AWS servers and VMs, the "elastic" IP address that has been set up in AWS to ensure a reserved, constant IP address, and other flags and switches. The tricky part about using Metamask for the use case developed as part of this work is that a network transfer mechanism from the localhost of the local PC as configured in Metamask to the private blockchain network set up in AWS Virtual Machine (Chain health) must be implemented in order to use the AWS virtual server. This is accomplished by establishing a "ssh tunnel" between the local machine and Chain health. The network traffic is routed through the "ssh tunnel" to the appropriate AWS port 22, which is already configured to accept SSH connection requests from any IP address. Using this method, any traffic received by the local machine on port 8545 (the port listening for RPC requests) is transparently and securely redirected to the AWS blockchain node (port 22) via the established tunnel.

5. Solidity programming language for smart contract implementation: The proposed prototype includes smart contracts that have been properly implemented. Solidity is an object-oriented programming language that was developed to aid in the implementation of Ethereum's blockchain smart contracts. Solidity's syntax is curly-bracket or curly-brace, which means that statement blocks are defined and separated by curly brackets, i.e., and. Solidity's creators were inspired by C++, JavaScript, and Python, while its primary use is intended to be the implementation of smart contracts within Ethereum Virtual Machines (EVMs). Solidity's main characteristics are that it is statically typed, supports inheritance, and employs libraries. Voting, crowdfunding, auctions, and wallets are the most popular blockchain smart contracts that can be built with Solidity.

6. Remix Online IDE: Remix Online IDE was used to create and test a smart contract written in the Solidity programming language. This is a tool set that can be accessed directly via the URL https://remix.ethereum.org. Remix is an Ethereum-supported development IDE that includes a Solidity compiler (as a plug-in) and contract execution capabilities. Remix Online IDE includes common development capabilities and functionality such as a scripting editor, compilation, debugging, and deployment, as well as transaction logging.

7. React JS for front-end integration: End users of the proposed blockchain network, interact with each other by using specific user interface capabilities developed as part of the proposed prototype. The front end has been developed using React JS for creating web-based interfaces with all user types. Creation and execution of campaigns, submission of requests along with their response (approval or rejection), and view of campaign transactions are supported by the proposed front-end implementation.

3.4 Prototype implementation results and discussion

In this section, an end-to-end walk-through of the proposed Ethereum blockchain prototype is attempted, aiming to provide a holistic, explanatory view of the overall functionality implemented and provided. More specifically, a number of selected scenarios that describe representative flows of activities in Health-Chain is presented, including specific user roles and activities, campaigns that are created and executed, and requests that are submitted and approved or rejected.

3.4.1 Scenario 1: New Campaign creation from the CoA

In this scenario the CoA creates a new campaign, which is a prerequisite for any other scenario. Campaign creation is initiated by clicking to the button "Create Campaign" or to the sign "+" on the upper right corner of the central campaign page as shown in Figure 18.

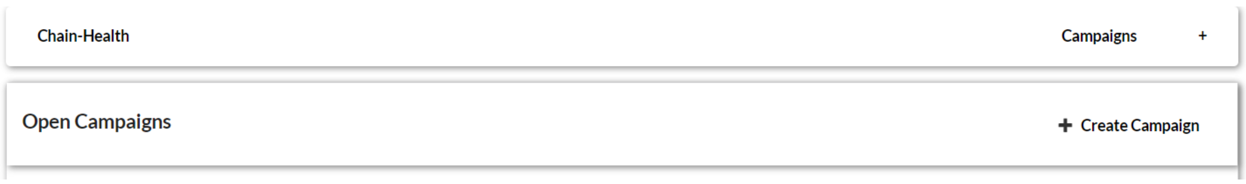
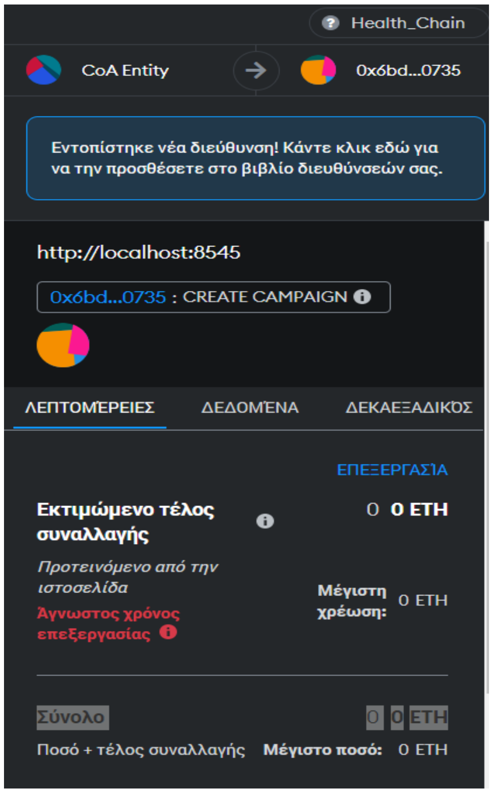


Figure 18: New campaign creation

After clicking to the "Create Campaign" button, the Metamask interface takes care to propagate the transaction to the blockchain and provides the option to the CoA to confirm or cancel the specific action as depicted in Figure 19.


Health_Chain

CoA Entity → 0x6bd...0735

Εντοπίστηκε νέα διεύθυνση! Κάντε κλικ εδώ για να την προσθέσετε στο βιβλίο διευθύνσεών σας.

http://localhost:8545

0x6bd...0735 : CREATE CAMPAIGN

ΛΕΠΤΟΜΈΡΕΙΕΣ ΔΕΔΟΜΈΝΑ ΔΕΚΑΕΞΑΔΙΚΌΣ

ΕΠΕΞΕΡΓΑΣΙΑ

Εκτιμώμενο τέλος συναλλαγής 0 **0 ETH**

Προτεινόμενο από την ιστοσελίδα

Άγνωστος χρόνος επεξεργασίας

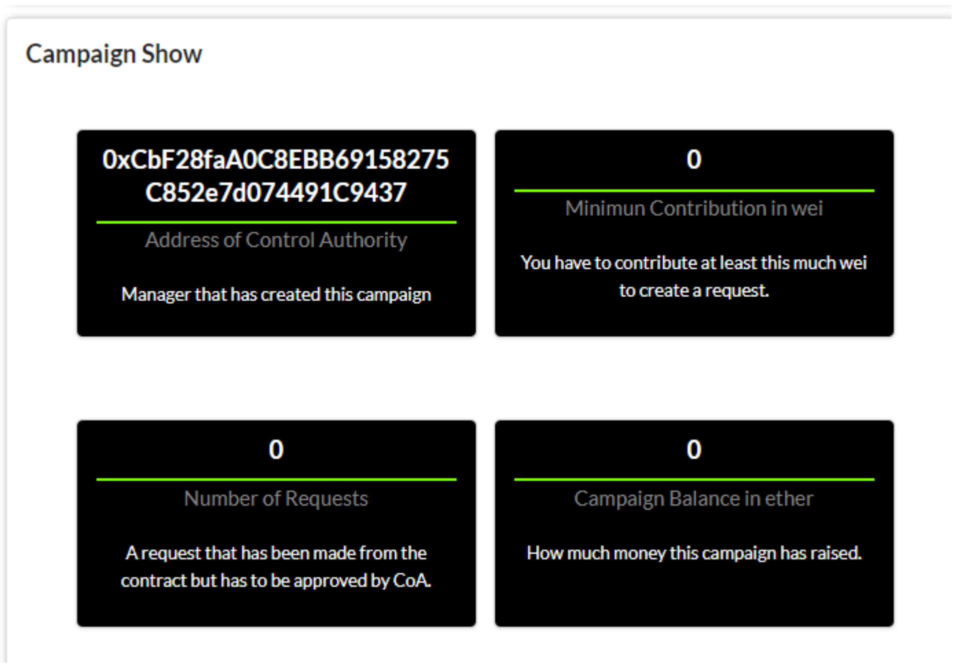
Μέγιστη χρέωση: 0 ETH

Σύνολο 0 0 ETH

Ποσό + τέλος συναλλαγής **Μέγιστο ποσό:** 0 ETH

Figure 19: Metamask interface for campaign creation

Upon confirmation of the CoA for the creation of the new campaign, the basic information of this campaign appears to the user interface as on can see in Figure 20.


Campaign Show

0xCbF28faA0C8EBB69158275C852e7d074491C9437
Address of Control Authority
Manager that has created this campaign

0
Minimun Contribution in wei
You have to contribute at least this much wei to create a request.

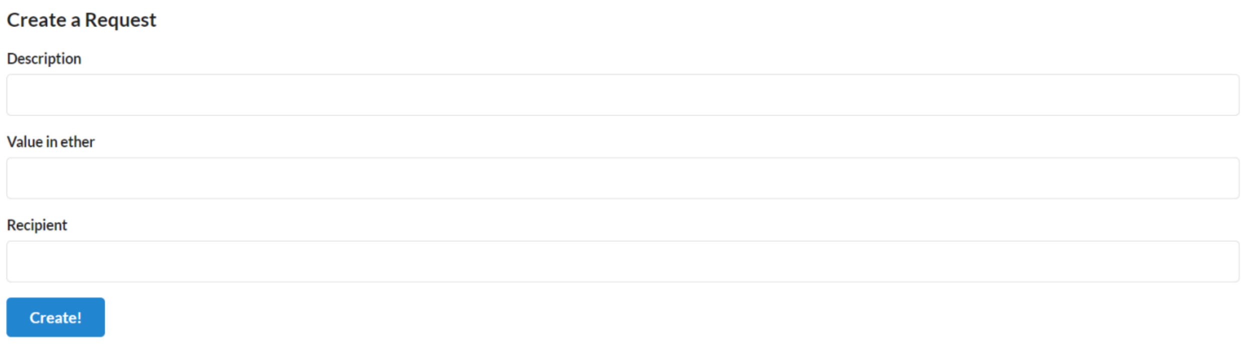
0
Number of Requests
A request that has been made from the contract but has to be approved by CoA.

0
Campaign Balance in ether
How much money this campaign has raised.

Figure 20: Basic campaign information

3.4.2 Scenario 2: Doctor request to participate to a campaign and approval from the CoA

An entity such as a doctor can participate in open campaigns by creating relevant requests. As shown in Figure 21, any request should include specific attributes such as a description of the request, its cost, and the recipient.


Create a Request

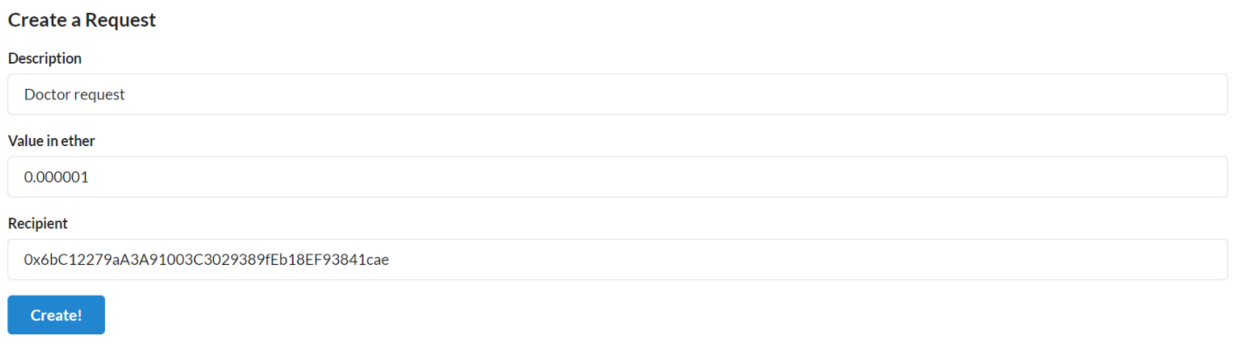
Description

Value in ether

Recipient

Figure 21: New request creation

An example of a request from a doctor for a specific campaign with its cost in ether, is presented in Figure 22.


Create a Request

Description

Doctor request

Value in ether

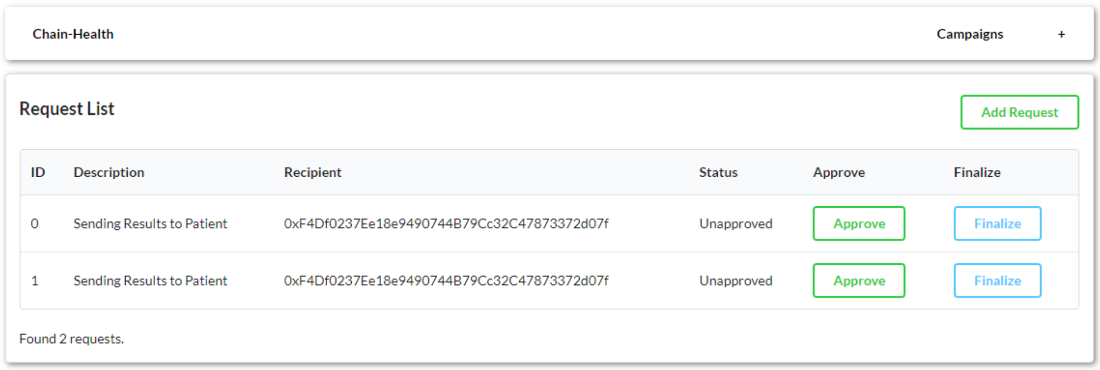
0.000001

Recipient

0x6bC12279aA3A91003C3029389fEb18EF93841cae

Figure 22: Example of a doctor request

Any request is approved or rejected by the campaign owner, i.e., the CoA, after it is submitted. The result of this transaction is accessible to the requestor via the "Request List" screen, as shown in Figure 23. The requestor can also create and submit new requests from this interface by clicking on the "Add Request" button.



Request List

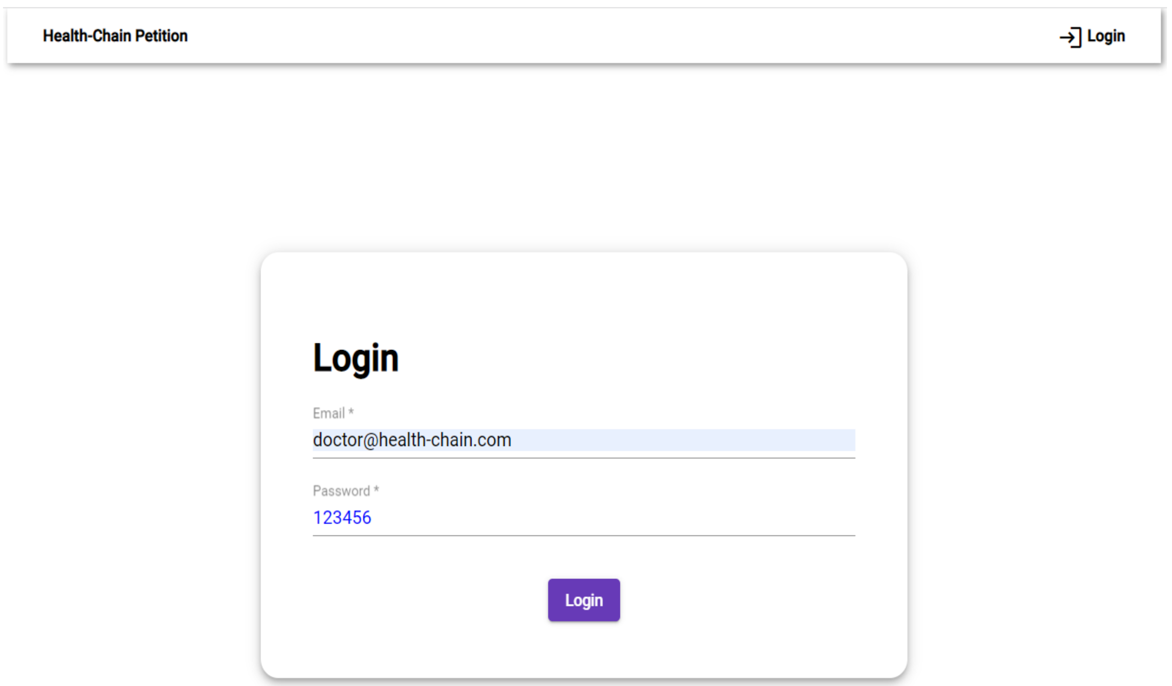
ID	Description	Recipient	Status	Approve	Finalize
0	Sending Results to Patient	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f	Unapproved	Approve	Finalize
1	Sending Results to Patient	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f	Unapproved	Approve	Finalize

Found 2 requests.

Figure 23: An example of a Request list

3.4.3 Scenario 3. Doctor uploads the results of the blood test to Health-Petition, and the patient receives the data transfer.

To complete the transaction, an entity, such as a doctor, must gain access to the Health Petition platform. First and foremost, he or she must log into the system using the credentials provided by the CoA, as shown in Figure 24.

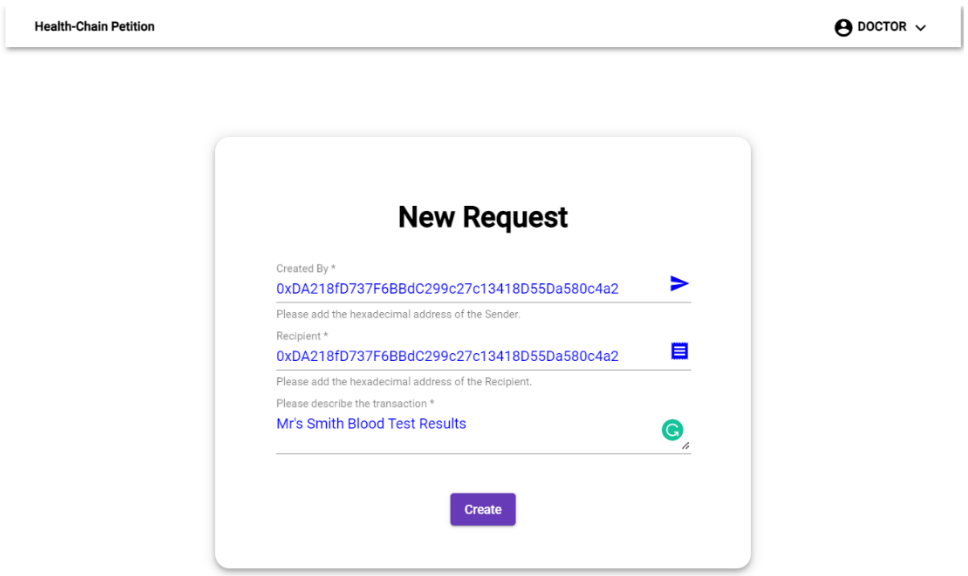


Email *
doctor@health-chain.com

Password *
123456

Figure 24: Log in to Health Petition as a Doctor

The next step will be to locate the campaign from the Campaigns list and to create an exact copy of the request that he/she previously created on Chain Health. As shown in Figure 25, any request should include attributes such as address of creator, its description, and the recipient.



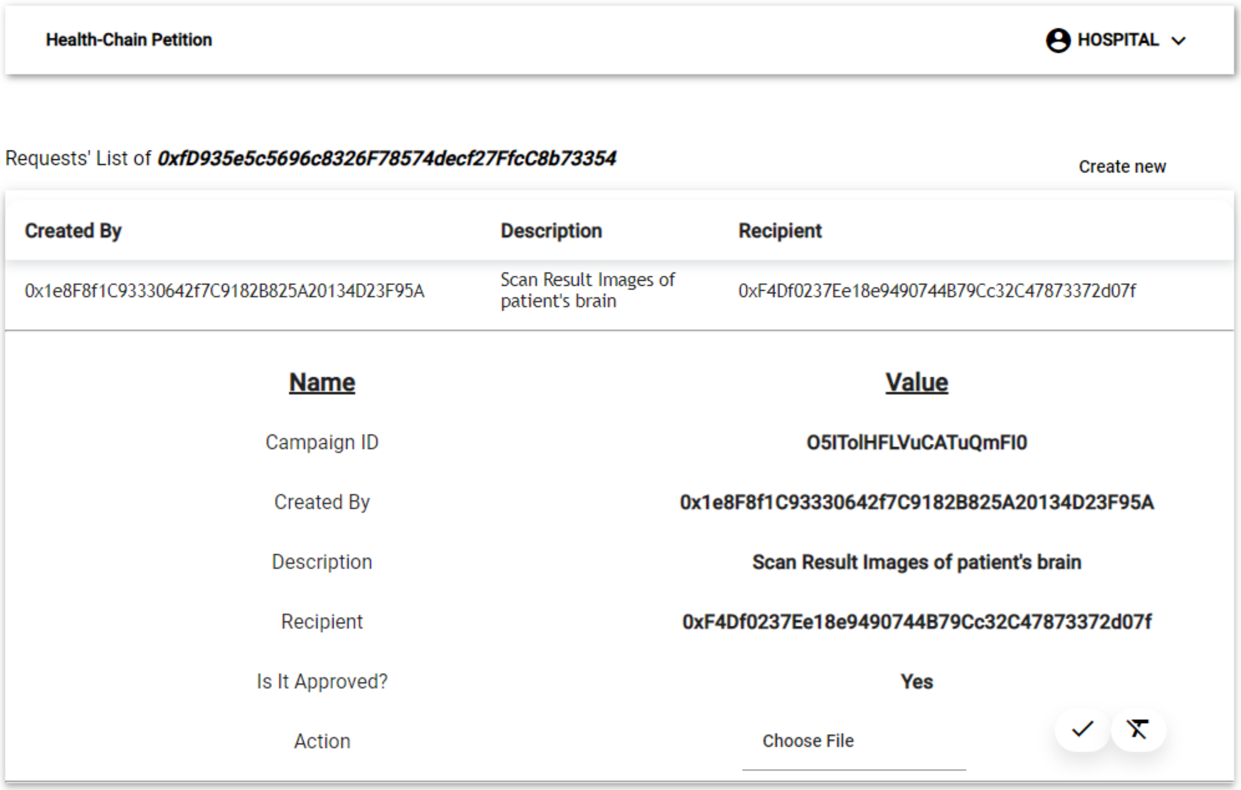
Created By *
0xDA218fD737F6BBdC299c27c13418D55Da580c4a2
Please add the hexadecimal address of the Sender.

Recipient *
0xDA218fD737F6BBdC299c27c13418D55Da580c4a2
Please add the hexadecimal address of the Recipient.

Please describe the transaction *
Mr's Smith Blood Test Results

Figure 25: Create Request Interface

The CoA must approve the request in the Health Petition platform after it is created, and then a file Uploader is presented for the Doctor to upload the file, as shown in figure 26.



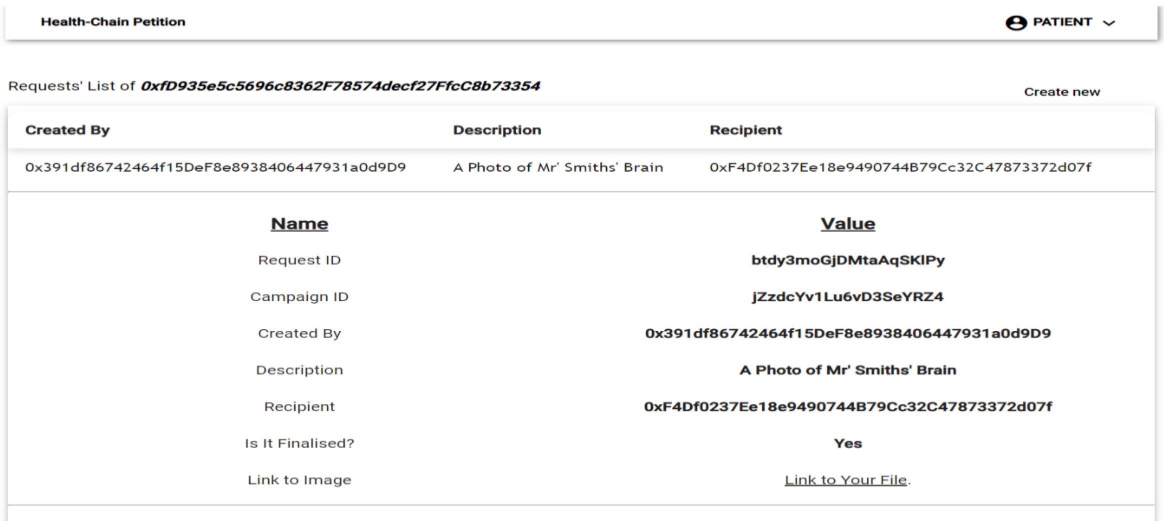
Requests' List of ***0xfD935e5c5696c8326F78574decf27FfcC8b73354***

Created By	Description	Recipient
0x1e8F8f1C93330642f7C9182B825A20134D23F95A	Scan Result Images of patient's brain	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f

Name	Value
Campaign ID	O5lTolHFLVuCATuQmFI0
Created By	0x1e8F8f1C93330642f7C9182B825A20134D23F95A
Description	Scan Result Images of patient's brain
Recipient	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f
Is It Approved?	Yes
Action	Choose File

Figure 26: Upload Image Interface

Finally, as shown in Figure 27, the requester can easily log in to the Platform and download the requested file via the link provided by the secure Firebase Firestore Database.



Requests' List of ***0xfD935e5c5696c8362F78574decf27FfcC8b73354***

Created By	Description	Recipient
0x391df86742464f15DeF8e8938406447931a0d9D9	A Photo of Mr' Smiths' Brain	0xF4Df0237Ee18e9490744B79Cc32C47873372d07f

Name	Value
Request ID	**btdy3moGjDMtaAqSKlPy**
Campaign ID	**jZzdcYv1Lu6vD3SeYRZ4**
Created By	**0x391df86742464f15DeF8e8938406447931a0d9D9**
Description	**A Photo of Mr' Smiths' Brain**
Recipient	**0xF4Df0237Ee18e9490744B79Cc32C47873372d07f**
Is It Finalised?	**Yes**
Link to Image	Link to Your File.

Figure 27: Download File Interface

3.4.4 Discussion

As previously stated, the current study's work product, Health-Chain, is a web application that uses a blockchain smart contract system for secure health data sharing. Its goal is to assist related health organizations, doctors, patients, and the medical community in securely storing and exchanging health transactions under the supervision of a Control Authority (CoA) entity. The main principles on which the proposed design of the Health-Chain System is based have been followed, as evidenced by the scenarios presented earlier in this Section. First, two types of users have been defined: health entities (health Organizations, doctors, or individuals), and the CoA, which serves as the entity that controls and guarantees network transactions. Second, the CoA has the authority to design and implement campaigns. Third, users can make requests, and the CoA has the authority to accept or reject those requests, with each request resulting in a transaction. In Ethereum's immutable layer, all transaction data is stored in the form of blocks.

The proposed implementation's primary goal, namely to assist health entities in communicating effectively and transparently in a secure non-traceable environment supervised by the aforementioned CoA, has been demonstrated. Functionally, the proposed implementation allows users to apply for health campaigns, create requests, and track the outcomes of their transactions. The main principles on which the proposed application's functionality is based have been met, i.e., the proposed application provides speed and efficiency in access and functionality from an end user perspective, is easily accessible and user friendly, and provides reliable functionality, solid performance, and stability. Based on the modern software stack and tooling used and integrated, security and maintainability principles have also been met.

In terms of non-functional requirements, the proposed application adhered to the principles and best practices established as goals. Because AWS cloud infrastructure was used, the implemented solution has good response times as well as acceptable performance and scalability capabilities. It also meets high security standards and ensures the integrity of data stored and transmitted between users. Equally important, based on the testing performed thus far, the proposed application appears resilient, producing minimal errors from which recovery is possible within acceptable time frames.

As system administrator, the CoA is in charge of configuring, managing, and maintaining the application. Furthermore, the CoA is in charge of creating campaigns by filling out the appropriate campaign forms. The main features required for the proper operation of a CoA-generated campaign have been implemented. Specifically, only the CoA can launch a campaign, and each campaign has its own unique address. The CoA must create a campaign template before users can create campaign requests by filling out the relevant request form. Finally, the CoA can approve or reject a Campaign request, and once approved or rejected, the request is saved in the application's hash tree.

CHAPTER 4: CONCLUSIONS

4.1 Summary and conclusions

This paper provides a comprehensive background research and critical analysis of the blockchain topic, as well as a case study of a specific pilot blockchain deployment in the healthcare industry. Blockchain is commonly defined as an immutable ledger that stores and manages data in a decentralized way. It is a new disruptive technology that is being used in a variety of data-driven industries. From another angle, blockchain is a public, distributed ledger capable of recording transactions and tracking assets; immutability is achieved through a peer-to-peer network of nodes and is not supported by any centralized entity; assets can be tangible or intangible. A number of important characteristics inherited with this technology, such as anonymity, decentralization, and transparency, have made it particularly appealing and sparked significant research efforts (Buterin [7]; Hasselgren et al. [23]).

In practice, blockchain enables interoperability between different parties without the need for a central entity to coordinate. To ensure security, data is stored in a structured manner, bundled in blocks that grow continuously, with each block connected to the previous and next one using cryptographic algorithms. Using the "smart contracts" feature, blockchain also allows for the creation and execution of contracts without the involvement of any central body or authority. Smart contracts are digital agreements between two or more blockchain participants that use predefined functions to store information, process inputs, and record outputs. Smart contracts enable the automation and digitization of the agreement process and transactions between parties involved (Buterin [7] Raikwar et al. [54]).

There are three types of blockchains: permissionless public blockchains, public permissioned consortium blockchains, and private blockchains. The manner in which data is accepted and stored in a blockchain network's distributed ledger is critical to the integrity of the blockchain ecosystem and is commonly referred to as the consensus mechanism or protocol. There are many consensus protocols in the relevant literature, three of which were pioneers and remain among the most popular, namely PoW (Proof of Work), PoS (Proof of Stake), and PBFT (Practical Byzantine Fault Tolerance). However, recent research has moved towards the direction of enhanced security and has revealed Proof of Authority (PoA) as an alternative consensus mechanism with significant advantages versus both PoW and PoS. PoA is a variant of PoS in which nodes are working to enhance their reputation and identity, instead of working to identify tokens (Zheng et al. [67]; Hasselgren et al. [23]; Sultan et al. [58]; Chen et al. [11]).

The healthcare industry is currently experiencing an explosion of data generation and dissemination among interested parties. Simultaneously, long-standing issues such as data leakage and unauthorized data sharing are worsening, having a significant impact on public trust in healthcare institutions as well as the operational and reputational risks they face. The sensitivity of patient data when stored and accessed on an individual level complicates matters even further. To achieve effective hospitalization and patient treatment, multidisciplinary teams of healthcare professionals with a variety of skills and competencies, as well as the necessary information and technology, are required. Multiple operations are required for effective, end-to-end patient treatment, requiring important data as input and producing outcomes that are critical for proper patient treatment. Because of its nature and inherited data-driven characteristics, blockchain technology is an excellent candidate for addressing the aforementioned healthcare sector issues. The effective use of blockchain technology may provide significant benefits to patients, doctors, and the medical community, as well as convert massive healthcare data flows into feasible and achievable goals (Ni et al. [44]; Coiera [13]; Hasselgren et al. [23]).

In healthcare, consortium blockchain is the most common, while private and public blockchains are less common. This is a natural occurrence, owing to the high sensitivity of health data. In terms of platform, Ethereum is by far the most popular in healthcare, while Hyperledger Fabric is used in a significant number of implementations. The majority of research studies conducted thus far have been based on Ethereum platforms controlled by institutions and have focused on the efficient management of personal health records or the utilization of M-Health capabilities. The main reasons for Ethereum and Hyperledger platforms' wide applicability are their ability to effectively implement and manage smart contracts, providing efficiencies in interactions with third parties, as well as their broad market penetration and large number of developers with skills and expertise working on those platforms. In terms of consensus mechanisms, PoW and PBTF have been proposed in the majority of cases in the health care domain (Hasselgren et al. [23]).

A proof-of-concept blockchain implementation for recording data transactions between healthcare stakeholders has been implemented in this study. Its overall architectural approach, along with its technical design and practical considerations are presented, while a pilot, prototype implementation has been presented and documented. Its core application, Chain-Health, was designed with the goal of assisting associated health organizations, doctors, patients, and the medical community in securely storing and exchanging health data by conducting transactions overseen by a Control Authority (CoA) entity. The relevant files are transferred through an auxiliary application called Health-Petition, which is responsible for acting as a web replica of the Chain-Health system because it contains all relevant requests and campaigns. The functionality of the proposed platform is built on efficiency, usability, reliability, security, and maintainability, with the primary goal of aiding health organizations in interacting effectively and transparently in a secure non-traceable environment regulated by the CoA. In terms of functionality, the suggested implementation enables users to apply for national health campaigns, create requests for specific health data categories, track the outcomes of their transactions, and eventually transfer the relevant files.

The software tools used by Chain-Health are a hybrid of React, Solidity and Javascript whereas Health Petition uses Angular 12. Because a variety of different solutions, programming languages, and tools are used and must be integrated to make the prototype functional, the implemented prototype is quite complex. The use of cloud and local infrastructure elements adds to the complexity, and interoperability between them has proven to be difficult at times. AWS infrastructure, Ethereum network deployment and Geth client, Metamask browser extension, and Solidity programming language for smart contract implementation are combined to make the overall prototype functional and interoperable. A PoA consensus mechanism stores and validates all transactions. Smart contracts are properly implemented in the proposed prototype. With the smart contract in place, the CoA can create and submit new campaigns, while health professionals can submit requests to participate, triggering CoA approval or rejection. End users of the proposed blockchain network communicate with one another through the use of specific user interface capabilities developed as part of the proposed prototype. The proposed front-end implementation supports campaign creation and execution, request submission and response (approval or rejection), viewing campaign transactions and file transfer. Metamask, which can natively support "wallet" functionality, is one of the tools used to support the current prototype implementation. MetaMask provides access to transaction requests through a simple and easy-to-use front end while also acting as an Ethereum cryptocurrency wallet. Metamask works as a browser plugin for Chrome and Firefox, and one of its main benefits is that it can connect to Ethereum networks, either existing or new custom private Ethereum nodes.

To summarize, the current study (1) provided a synopsis of current blockchain qualitative and quantitative research, as well as related technology evolution, with a particular emphasis on use cases applicable to the health domain, (2) demonstrated that blockchain-based applications add actual value to sharing and exchanging medical records and sensitive health data, and could thus provide efficient information management solutions to the health community, and (3) demonstrated how private blockchain networks could be implemented, integrated, and deployed using a set of software tools and scalable cloud-based infrastructure using a prototype.

4.2 Thesis limitations and future work

Although blockchain technology has cultivated promises of widespread applicability and smart applications in health care, current results show little evidence of this because many of them are based on unrealistic and difficult-to-implement ideas. Current literature also demonstrates that only a small number of real-world applications have been developed, tested, deployed, and are currently in use. Even as blockchain networks and applications continue to grow in popularity, significant issues and aspects must be studied and addressed, with scalability and applicability in various domains being two of the most prominent. Although, if one examines the features of blockchain networks individually, he or she will conclude that the majority of them are not the result of new research but have been well known for a long time, it is their combination that distinguishes blockchain technology. Blockchain, on the other hand, is not a technology for every application. Traditional databases are far more appropriate in many cases, or specific vertical solutions with industry-specific added value are preferable (Nakamoto [43]; Mettler [42]; Casino et al [9]; Hasselgren et al. [23]).

The most popular blockchain applications, particularly in the health care industry, have been built around electronic or personal health records, with Ethereum and Hyperledger being the most popular platforms. According to a recent literature review, there has been a surge in interest in the use of blockchain in health care in recent years. The primary goals of blockchain applications in health care have been to improve functionality in areas such as interoperability, access control, data integrity, and provenance. The high level of interest, among other things, has led health institutions and organizations to become more open and willing to share medical data for the benefit of patients, health professionals, and, ultimately, society. Clinical testing, automated diagnostic services, and population health management are currently under-researched areas of blockchain in healthcare (Linn and Koo [36]; Hasselgren et al. [23]).

The main challenges and limitations that the proposed prototype faces are listed below.

1. **Latency and scalability issues**. Although the proposed prototype's transaction execution speed is generally fast, as the volume of data grows, performance issues with the specific application will emerge, even in private blockchain networks, which are much faster by definition. At the same time, because the network's operation will be continuous, it will require massive computing power, capacity, and energy consumption. One solution could be to delete old transactions or to use the sharding technique described earlier. However, any effort to improve efficiency should not conflict with the benefits of blockchain technology, which include transaction security, decentralization and scalability.
2. **Stability issues**. Because blockchain technology is not yet widely used, and because of limitations in the implementation of existing technologies, there are stability issues in the proposed network's operation. Since it is nearly impossible to keep the current network running continuously, there have been cases of smart contracts being lost in the network and having to be recreated. Unfortunately, because it alters the immutability of data, this fact has created limitations in the management of a blockchain network.
3. **Complexity challenges**. The proposed smart contract only regulates the fundamental principles governing the operation of a healthcare blockchain application. A realistic proposal for such a system would have far more rules governing its proper functioning. Unfortunately, because the Solidity language is still in its early stages, there is insufficient literature and guidance to create code suitable for the smooth operation of such a system.

Further work on the aforementioned challenging areas appears promising and valuable in terms of potential future research on the proposed prototype.

1. **Contribution mechanism**. To address the future increase in current costs of an integrated blockchain application in the health sector, a mechanism should be developed that allows all parties involved to contribute financially. This is already happening in the majority of public blockchain networks. As a result, if the CoA wishes, a minimum contribution in Wei should be assumed. In this case, before a user can create a request, the specified amount must be in his Metamask e-wallet.
2. **Deployment**. Health-Chain's applications run in a localhost environment to avoid additional development costs for the needs of the proposed prototype. The platform must be lined up and uploaded to a web server in order to create an integrated system. As a result, any application user can type the relevant URL and create requests using only Metamask, provided it is configured to connect to the blockchain network.
3. **User interface optimization.** The Health-Chain system employs React and Angular technologies to create a user-friendly and simple-to-use environment. Nonetheless, the front-end techniques used are quite simple, relying on basic Bootstrap Framework and React Bootstrap Framework usage. The system should receive an end-to-end graphic treatment, establish stylistic patterns, and develop its own distinct aesthetic.
4. **Authorization module.** The current implementation employs two-way user authentication. First, in the Chain-Health system, users acting as actors can communicate with the network because it has already been configured via the Metamask. Random users who have not implemented the above settings do not have network access, and this serves as a form of authentication**.** At the same time, the basic Password-based authentication technique is currently used in the Health-Petition utility application so that a user can access all of its functionalities. As part of platform optimization, a unified authentication solution must be found, which will of course be based on the creation of a single and integrated application with a unique URL.

5. **Smart Contracts.** Simultaneously, more sophisticated smart contracts should be put in place to allow for appropriate system testing and experimentation. As the system's applications become more complex, different health campaigns will necessitate different smart contract solutions.
6. **Merging.** The proposed prototype's end-to-end operation is based on the concurrent operation of two programs, Chain-Health and Health-Petition. Without a doubt, a comprehensive healthcare blockchain system solution that combines the functionalities of these two applications must be implemented. To accomplish this, a way to store and transfer files over the blockchain network, possibly in base64 format, would need to be discovered. Simultaneously, an intelligent solution for correlating national health campaigns with patients or other cases should be developed, taking into account the blockchain's operational rules.
7. **mHealth implementation**. M-health, as previously stated, refers to patients' ability to collect, store, and control their own data from mobile devices using sensors. Given the rapid growth of M-health app users and the high value of the data collected, a mobile system that replicates the functionality of the proposed prototype must be developed.
8. **Integration into the digital Health Record concept.** This system must be associated with the concept of an Electronic Personal Health Record in the grand scheme of things. Blockchain technology has the potential to provide answers and solutions to timeless problems, as well as accelerate the adoption of electronic patient records. Because of their portability and potential access by unscrupulous users and unauthorized individuals, electronic medical and health records raise concerns about patient privacy. Immutability, Traceability, and Enhanced Security, which are some of the most important features of Blockchain Technology, may be key to finding these answers and assisting Health Entities paving the way for a new reality in which every patient on the planet has easy access to his or her medical history, resulting in faster and more effective health care.

REFERENCES

[1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman Y. Manevich, et al, “Hyperledger fabric: A distributed operating system for permissioned blockchains,” in EuroSys ’18: *Proceedings of the Thirteenth EuroSys Conference, vol. 30*, pp. 1-15,2018, doi: 10.1145/3190508.3190538.

[2] F. Angeletti, I. Chatzigiannakis and A. Vitaletti, “Privacy preserving data management in recruiting participants for digital clinical trials,” in *Proceedings of the First International Workshop on Human-Centered Sensing, Networking, and Systems*; Delft, Netherlands, ACM, pp. 7-12, doi: 10.1145/3144730.3144733.

[3] D. Appelbaum, A. Kogan, M. A. Vasarhelyi; “Big Data and Analytics in the Modern Audit Engagement: Research Needs” in *AUDITING: A Journal of Practice & Theory*, 1 November 2017; vol. 36 (4), pp 1–27, doi: 10.2308/ajpt-51684

[4] blockchain.com (2021), *Currency Statistics*. https://cutt.ly/tE9GQrP (accessed 3/10/2021)

[5] T. Bocek, B. Rodrigues, T. Strasser, and B. Stiller, “Blockchains Everywhere - A Use-Case of Blockchains in the Pharma Supply-Chain” in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM), doi:*10.23919/INM.2017.7987376

[6] J. Bruce, “*Purely P2P crypto-currency with finite mini-blockchain”, 2013* https://cutt.ly/lRveXi3 (accessed 3/10/2021)

[7] V. Buterin, “*A Next-generation Smart Contract and Decentralized Application Platform*” in Ethereum White Paper, 2014, https://cutt.ly/JE9GSbf (accessed 3/10/2021)

[8] C. Cachin, “*Architecture of the hyperledger blockchain fabric*” in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, IBM Research, 2016 https://cutt.ly/lE9GG3c (accessed 3/10/2021)

[9] F. Casino, T. Dasaklis and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues” in *Telematics and Informatics, 2019, vol.36,* pp 55-81, doi: 10.1016/j.tele.2018.11.006

[10] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance" in *Proceedings of the third symposium on Operating systems design and implementation*, 1999, pp: 173-186, https://cutt.ly/FE9G6xl

[11] G. Chen, B. Xu, M. Lu and et al, "Exploring blockchain technology and its potential applications for education" in *Smart Learning Environments, 2018, vol. 5*(1), doi: 10.1186/s40561-017-0050-x

[12] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller and D. Song, "Ekiden: a platform for confidentiality-preserving, trustworthy, andperformant smart contract execution" in *2019 IEEE European Symposium on Security and Privacy (EuroS&P) in 2018*, *vol.*1804.05141, doi: 10.1109/EuroSP.2019.00023

[13] E. Coiera, "*Guide to Health Informatics* (3rd ed.)" in London: CRC press, 2015, doi: doi.org/10.1201/b13617

[14] G. Dagher, J. Mohler, M. Milojkovic and P. Marella, "Ancile: privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology" in *Sustainable Cities and Society, 2018, vol. 39, pp.* 283-297, doi: 10.1016/j.scs.2018.02.014

[15] C. Dannen , "*Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*" in New York: Apress. Digiconomist *Bitcoin Energy Consumption Index*, 2017
https://cutt.ly/ORvtfhV (accessed 3/10/2021)

[16] I. Eyal, A. Gencer, E. Sirer and R. Van, "Bitcoin-NG: a scalable blockchain protocol" in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation, USENIX Association*, 2016, vol. 45-59. https://cutt.ly/4RvtJfv (accessed 3/10/2021)

[17] Frost and Sullivan in "*Global Blockchain Technology Market in the Healthcare Industry 2018-2022",* 2019 https://cutt.ly/lE9HTJB (accessed 3/10/2021)

[18] N. Gogerty and J. Zitoli, "*DeKo: currency proposal using a portfolio of electricity linked assets",* 2011, doi: 10.2139/ssrn.1802166

[19] S. Goldfeder, H. Kalodner, D. Reisman and A. Narayanan, "*When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencie.*https://cutt.ly/9RU1bZi

[20] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor amd X.Xu, " On legal contracts, imperative and declarative smart contracts, and blockchain systems" in *Artificial Intelligence and Law, 2018, vol:26, pp:*377-409, doi: 10.1007/s10506-018-9223-3

[21] K. Griggs, O. Ossipova, C. Kohlios, A. Baccarini, E. Howson and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring." in *Journal of Medical System, 2018, vol:42*(7), doi: 10.1007/s10916-018-0982-x

[23] A. Hasselgren, K. Kralevska, D. Gligoroski, S. Pedersen, A. Faxvaag, "Blockchain in healthcare and health sciences - A scoping review" in *International Journal of Medical Informatics, 2020, vol:134,* pp:104040, doi:10.1016/j.ijmedinf.2019.104040

[24] D. Howarth, "*Hackers Launch Third 51% Attack on Ethereum Classic This Month"* https://cutt.ly/aRvrKaP (accessed 3/10/2021)

[25] Hyperledger (n.d.). https://cutt.ly/pE9HPPM (accessed 3/10/2021)

[26] IBM, "*Health rallies for blockchains: Keeping patients at the center.", 2016* https://cutt.ly/qE9HF99 (accessed 3/10/2021)

[27] D. Ichikawa, M. Kashiyama and T Ueno, "Tamper-resistant Mobile Health Using Blockchain Technology" in *JMIR mHealth and uHealth, 2017, vol:5*(7):e111, doi:10.2196/mhealth.7938

[28] H. Issa, T. Sun, T and M.Vasarhelyi, "Research ideas for artificial intelligence in auditing: the formalization of audit and workforce supplementation" in *Journal of Emerging Technologies in Accounting,2017, vol:13*(2), pp:1-20, doi:10.2308/jeta-10511

[29] S. Kalra, S. Goel, M. Dhawan and S. Sharma, "Zeus: analyzing safety of smart contracts" in *Network and Distributed Systems Security (NDSS) Symposium, 2018,* pp:1-15, doi:10.14722/ndss.2018.23082

[30] S. Khan, F.Loukil, C. Ghedira-Guegan, E. Benkhelifa and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends" in *Peer-to-peer networking and applications, 2021, pp:*1-25, doi:10.1007/s12083-021-01127-0

[31] A. Kosba, A. Miller, E. Shi, Z. Wen and C. Papamanthou in “Hawk: the blockchain model of cryptography and privacy-preserving smart contracts” in *2016 IEEE Symposium on Security and Privacy*,2016, pp: 839-858, doi:10.1109/SP.2016.55

[32] A. Lewis, “*A gentle introduction to smart contracts”, 2016,* https://cutt.ly/BRvilNB (accessed 3/10/2021)

[33] H. Li, L. Zhu, M. Shen, F. Gao, X. Tao and S. Liu, “Blockchain-based data preservation system for medical data” in *Journal of Medical Systems, 2018, vol:42*(8), pp:141, doi:10.1007/s10916-018-0997-3

[34] X. Liang, S. Shetty, J. Zhao, D. Bowden, D. Li, J. Liu, S. Qing, D. Liu, C Mitchell and L. Chen, L, “Towards Decentralized Accountability and Self-Sovereignty in Healthcare Systems” in *Information and Communications Security, 2018,* pp:387-398, doi: 10.1007/978-3-319-89500-0_34

[35] X. Liang, J. Zhao, S. Shetty, J. Liu, D. Li, D, “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), 2018, doi:* 10.1109/PIMRC.2017.8292361

[36] L. Linn and M. Koo, “*Blockchain For Health Data and Its Potential Use in Health IT and Health Care Related Research”,* 2016 https://cutt.ly/1RviUp3 (accessed 3/10/2021)

[37] S. Lo, X. Xu, Y. Chiam, Q Lu, “Evaluating suitability of applying blockchain” in *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems (ICECCS),*2017, pp:158-161, doi:10.1109/ICECCS.2017.26

[38] P. Mamoshina, L. Ojomoko, Y. Yanovich, A. Ostrovski, A. Botezatu, P. Prikhodko, E. Izumchenko, A. Aliper, K. Romantsov, A. Zhebrak, I.O Ogu and A.Zhavoronkov, “Converging blockchain and next-generation artificial intelligence technologies to decentralize and accelerate biomedical research and healthcare” in *Oncotarget*, 2018, *vol.9*(5), pp:5665-5690, doi:10.18632/oncotarget.22345

[39] A. Mavridou and A.Laszka, “Tool demonstration: FSolidM for designing secure ethereum smart contracts” in *Cryptography and Security*, 2018 arXiv:1802.09949. https://cutt.ly/eRIb3E3

[40] T. McConaghy, "*How blockchains could transform artificial intelligence",* 2016, https://cutt.ly/QRInedU (accessed 3/10/2021)

[41] R. Merkle, "A Certified Digital Signature" in *Conference on the Theory and Application of Cryptology*, 1989, pp:218-238, doi:10.1007/0-387-34805-0_21

[42] M. Mettler, "Blockchain technology in healthcare: the revolution starts here" in *2016 IEEE 18th International Conference on e-health networking, applications and services (Healthcom)*, 2016, pp:1-3, doi:10.1109/HealthCom.2016.7749510

[43] S.Nakamoto, "*Bitcoin: a Peer-to-Peer Electronic Cash System",* 2008, https://cutt.ly/7E9JtNq

[44] W. Ni, X. Huang, J. Zhang, and R Yu, "HealChain: A Decentralized Data Management System for Mobile Healthcare Using Consortium Blockchain" in *2019 Chinese Control Conference (CCC)*, 2019, pp:6333-6338, doi:10.23919/ChiCC.2019.8865388

[45] L. Nikolic, A. Kolluri, L. Sergey, P. Saxena, P and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale" in *Cryptography and Security*, 2018, arXiv:1802.06038, https://cutt.ly/5RIncop

[46] T. Nugent, D. Upton and M. Cimpoesu, "Improving data transparency in clinical trials using blockchain smart contracts" in *F1000Research*, 2016, *vol:5*, pp:2541, doi: 10.12688/f1000research.9756.1

[47] NXT, "*Nxt Whitepaper",* 2016, https://cutt.ly/7RvoeKA (accessed 3/10/2021)

[48] K. O'Dwyer and D Malone, "Bitcoin Mining and Its Energy Footprint" in *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communities Technologies* (ISSC 2014/CIICT 2014), 2014. doi: 10.1049/cp.2014.0699

[49] Parity Technologies, "*Security Alert",* 2017, https://cutt.ly/wRImbdj (accessed 3/10/2021)

[50] V. Patel, "A framework for secure and decentralized sharing of medical imaging data via blockchain consensus" in *Health Informatics Journal*, 2018, *vol:25*(4), pp:1398-1411. doi: 10.1177/1460458218769699

[51] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "*A Blockchain-Based Approach to Health Information Exchange Networks" in* Mayo Clinic, 2016, https://cutt.ly/fE52dtg

[52] A. Potenza, "*Can renewable power offset bitcoin's massive energy demands?",* 2017, https://cutt.ly/iRvocBf (accessed 3/10/2021)

[53] M. Rahman, E. Hassanain, M. Rashid, M and S. Barnes, "Spatial blockchain-based secure mass screening framework for children with dyslexia", *IEEE Access,* 2018, vol:*6,* pp:61876-61885, doi:10.1109/ACCESS.2018.2875242

[54] M. Raikwar, D. Gligoroski and K. Kralevska, K, "SoK of used cryptography in blockchain", *IEEE Access,* 2019, vol. *7,* pp:148550-148575. doi: 10.1109/ACCESS.2019.2946983

[55] A. Roehrs, C. Da Costa and R. Da Rosa Righi, "OmniPHR: a distributed architecture model to integrate personal health records" in *Journal of Biomedical Informatics,* 2017*, vol:71, pp:*70-81, doi:10.1016/j.jbi.2017.05.012

[56] D. Siegel, "*Understanding the DAO attack",* 2016, https://cutt.ly/IRImXNY (accessed 3/10/2021)

[57] A. Singh, R. Parizi, Q. Zhang, K. Choo and A. Dehghantanha, "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities" in *Computers & Security,* 2020*, vol: 88, pp:*101654, doi:10.1016/j.cose.2019.101654

[58] K. Sultan, U. Ruhi and R. Lakhani, "Conceptualizing Blockchains: Characteristics & Applications" in *11th IADIS International Conference Information Systems, 2018,* https://cutt.ly/1RvpQDh (accessed 3/10/2021)

[59] H. Wang and Y. Song, "Secure Cloud-Based EHR System Using Attribute-Based Cryptosystem and Blockchain" in *Journal of Medical Systems,* 2018, vol:*42*(8), pp:152, doi:10.1007/s10916-018-0994-6

[60] W. Wilkowska and M. Ziefle, "Privacy and data security in E-health: requirements from the user's perspective" in *Health Informatics Journal,* 2012, *vol:18*(3), pp:191-201, doi: 10.1177/1460458212442933

[61] Wired.com, "*A 50 million hack just showed that the DAO was all too human",* 2016, https://cutt.ly/fRvpDPi (accessed 3/10/2021)

[62] K. Wust and A. Gervais, "Do you need a Blockchain?" in *IACR Cryptology ePrint Archive, 2017*, pp:375, https://cutt.ly/yRIQzWJ

[63] A. Zhang and X. Lin, "Towards secure and privacy-preserving data sharing in e-Health systems via consortium blockchain" in *Journal of Medical Systems*, 2018, vol:*42* (8), pp:140, doi: 10.1007/s10916-018-0995-5

[64] P. Zhang, J. White, D. Schmidt , G. Lenz and S. Rosenbloom, "FHIRChain: applying blockchain to securely and scalably share clinical data" in *Computational and Structural Biotechnology Journal*, 2018, vol: *16*, pp: 267-278, doi: 10.1016/j.csbj.2018.07.004

[65] J. Zhang, N. Xue and X. Huang, "A secure system for pervasive social network-based healthcare" in *IEEE Access*, 2016, *vol:4*, *pp:*9239-9250, doi: 10.1109/ACCESS.2016.2645904

[66] H. Zhao, Y. Zhang, Y, Peng, and R. Xu, "Lightweight Backup and Efficient Recovery Scheme for Health Blockchain Keys" in *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*, 2017, doi:10.1109/ISADS.2017.22

[67] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An overview of blockchain technology: architecture, consensus, and future trends" in 2017 IEEE International Congress, 2017, doi: 10.1109/BigDataCongress.2017.85

[68] L. Zhou, L. Wang and Y. Sun, "MIStore, A blockchain-based medical insurance storage system" in *Journal of Medical Systems*, 2018, *vol:42*(8), vol:149, doi: 10.1007/s10916-018-0996-4

[69] C. Agbo, Q. Mahmoud, J. Eklund, "Blockchain Technology in Healthcare: A Systematic Review" in *Healthcare (Basel)*, 2019, vol:7(2), pp:56, doi: 10.3390/healthcare7020056

[70] A. Gad, D. Mosa, L. Abualigah, A. Abohany, "Emerging Trends in Blockchain Technology and Applications: A Review and Outlook" in *the Journal of King Saud University - Computer and Information Sciences*, 2022, vol:34(9), pp:6719-6742, doi: 10.1016/j.jksuci.2022.03.007

[71] 53 Legislature 1st Regular, State of Arizona House Bill 2417, Passed on March 29 2017: legiscan.com/AZ/bill/HB2417/2017

[72] De Beers Group, *"De Beers group introduces world's first blockchain*-*backed diamond source platform at scale*" De Beers - Tracr

[73] Robomed Network (Sistems Robohoney), *"Robomed the Blockchain platform for the medical organizations*", 20/11/2018 Robomed the Blockchain platform for the medical organizations

[74] Medium.com, *"fizzy by AXA: Ethereum Smart Contract in details*", 24/05/2019 fizzy by AXA: Ethereum Smart Contract in details

[75] Multichain.com, *"Blockchains vs centralized databases*", 17/05/2016 Four key differences between blockchains and regular databases