Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Augusto da Silva e Souza Miranda

On the Use of Blockchain Structures in a Multiagent Based Software Engineering Method: A Healthcare Example

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
April 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Augusto da Silva e Souza Miranda

On the Use of Blockchain Structures in a Multiagent Based Software Engineering Method: A Healthcare Example

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Carlos José Pereira de Lucena
Advisor
Departamento de Informática – PUC-Rio

Prof. Hugo Fuks
Departamento de Informática – PUC-Rio

Prof. Marx Leles Viana
Pesquisador Autônomo

Prof. Donald Don Cowan
University of Waterloo

Rio de Janeiro, April 30th, 2019

Pedro Augusto da Silva e Souza Miranda

The author graduated in Information Systems from the Pontifícia Universidade Católica do Rio de Janeiro (Rio de Janeiro, Brazil) in 2017.Works at Laboratório de Engenharia de Software researching software engineering inovations for the healthcare domain.

Bibliographical data

Miranda, Pedro Augusto da Silva e Souza

On the use of blockchain structures in a multiagent based software engineering method : a healthcare example / Pedro Augusto da Silva e Souza Miranda ; advisor: Carlos José Pereira de Lucena. – 2019.
60 f. : il. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2019.
Inclui bibliografia

1. Informática – Teses. 2. Agentes de software. 3. Blockchain. 4. Cuidados de saúde. 5. HL7 FHIR. 6. Engenharia de software. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my beloved wife Bruna Miranda and to my parents Antonio Carlos Miranda e Luiza Cristina da Silva e Souza Miranda.

Acknowledgment

To my teatcher and advisor Carlos José Pereira de Lucena
for all his guidance since my graduation.

To my beloved wife Bruna Miranda which help
me get by all difficult times these last 2 years.

To my co-adivisor Marx Leles Viana
for all the guidance and friendship that helped me achieve this goal.

To Andrew Diniz da Costa for the guidance and friendship all these years.

To my co-workers from LES, André
Lucena, Antonio Benchimol, Jefry Satre that always made possible for
me to work and study at the same time.

To my friend Alberto
Yamamoto which friendship was very important all these years by giving me
moral support.

To CNPq and PUC-Rio for the support, guidance and infra-
strucutre that without i would not be able to finish my research.

This study was financed in part by the Coordenação de Aperfeiçoamento
de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Thank you very much !

Abstract

Miranda, Pedro Augusto; Lucena, José Carlos. **On the Use of Blockchain Structures in a Multiagent Based Software Engineering Method: A Healthcare Example**. Rio de Janeiro,2019. 48p. Dissertação de Mestrado - Departamento de informática, Pontifícia Universidade Católica do Rio de Janeiro.

This paper presents an implementation approach for a private blockchain that is managed by software agents for healthcare data storage. Blockchain technology is changing the way we store private information. Now, it is possible to share private information while not revealing its owner's identity. This paper presents a solution, which enables users to store medical data by using blockchains along with software agents. Further research work has shown that healthcare data, as sensitive information, should be stored in private blockchains through the use of appropriate methods. Software engineering techniques have been used to achieve the proposed solution. The result is a private blockchain that is suitable for manipulating HL7 data, ensuring anonymity and privacy. This can be achieved through software agents that act as regulatory controls for the private blockchains.

Keywords

blockchain; software agents; healthcare; HL7; FHIR; Software engineering;

Resumo

Miranda, Pedro Augusto; Lucena, José Carlos. **Um Método de Engenharia de Software para o Uso de Estruturas Blockchain em Sistemas Multiagentes: Um Exemplo no Domíno da Saúde**. Rio de Janeiro,2019. 48p. Dissertação de Mestrado - Departamento de informática, Pontifícia Universidade Católica do Rio de Janeiro.

Este trabalho apresenta uma proposta de implementação de uma *blockchain* privada gerenciada por agentes de software para armazenar dados de sáude. A tecnologia *blockchain* está mudando a maneira com que dados privados são armazenados. Agora é possível compartilhar informações sem revelar quem é o dono da informação. Este trabalho apresenta um sistema que permite que usuários armazenagem dados de saúde em uma *blockchain* gerenciada por agentes de software. Pesquisas revelaram que dados de saúde devem ser armazenado em *blockchains* privadas, pois são privadas e não devem ser acessíveis para qualquer pessoa como em *blockchains* públicas. Técnicas de engenharia de software foram utilizadas para desenvolver a arquitetura proposta. O resultado é uma blockchain privada capaz de manipular dados no padrão HL7 e armazena-los de forma anônima e privada. Agentes de software foram utilizados para realizar todo o processo de recebimento, validação e inserção de dados no *blockchain*.

Palavras-chave

blockchain; agentes de software; cuidados de saúde; HL7; FHIR; Engenharia de Software;

Table of Contents

Abbreviation List

1. HL7 – Health Level Seven.
2. FHIR – Fast Healthcare Interoperability Resources.
3. POW – Proof of Work.
4. POS – Proof of Stake.
5. WA – Worker Agent.
6. TDA – Transaction Distribution Agent.
7. DAA – Data Auditioning Agent.
8. ICA – Integrity Control Agent.
9. PA – Peer Agent.
10. SPADE – Smart Python Agent Development Environment.
11. REST – Representational State Transfer.
12. API – Application Programming Interface.
13. HIPAA – Health Insurance Portability and Accountability Act.

Image List

1
Introduction

Blockchain Technologies [1][2][3] is responsible for a new trend in software engineering solutions. Blockchain is a public or private distributed ledger that allows for transactions to be stored by a user in a private and anonymous manner. To achieve such objective encryption techniques such as RSA standard [4] are used. Users private key is used to generate public key and signatures that are used to identify the owner of a transaction stored in a block. This type of signature is known as the user address. This address is used to identify the user in the network so that transactions can be directed to him if this is the case. For an example if BOB wants to send a transaction do BILL then BOB would give BILL his address so that he can use that address to create a transaction to BOB. Transactions are stored in a structure called Block. Once a block is validated it is stored in the blockchain. The validity of the information stored in the blockchain is guaranteed by a process called Proof of Work [5].

The technology called blockchain was originally created to support the cryptocurrency knows as Bitcoin [6]. Bitcoin used the blockchain structure to store the history if transactions performed using the bitcoin currency. The reason that blockchain was imperative for the success of the bitcoin is due to the fact that each block inserted in the blockchain have a unique hash signature generated using meta information from the block. This hash is obtained by using the resultant Merkle Tree Root [7] of the transactions stored in the block, a timestamp of the moment the block was created, the previous block hash and a nonce. The nonce is a number that when inserted in the cryptographic calculations to generate the block hash guarantees that a certain characteristic of the resulting hash is obtained. In bitcoin case that nonce is to guarantee that a block unique hash starts with a pre-determined amount of 0s. The amount of 0s is to increase the difficulty of finding such a hash since the nonce is not given to the miner. The miner is the entity responsible to verify the validity of the transactions that are going to be inserted in the block, create the block structure and finding the nonce that when encrypted with the block meta information results in a hash with N 0s in the front. Section 2 of this document explains in detail how this process works.

Blockchain can be applied in different domains such as finances [8], law [9], governance [10] and healthcare [11]. In the spectrum of this work we focused in the healthcare application of the blockchain technologies. The reason for the selected area of application is because one of the main challenges of the healthcare domain is the storage and sharing of patient information. The Health Insurance Portability and Accountability (HIPAA) [12] enforcement makes imperative to any business that handle patient information to be extremely careful with private information being leaked or stolen. HIPAA presents guidelines that have to be followed by companies that handle patient information. The main concern is the breach of privacy information from patients since legal actions can be taken against any actors involved in the information transaction history. With HIPAA guidelines in mind it would be very hard and time consuming if each entity developed its own protocol for data standardization in order to ensure send and receive data security. Thus, the Health Level Seven - Fast Healthcare Interoperability Resources framework (HL7 FHIR) [13] comes to mind. FHIR offers implementations of the HL7 standard through a framework that allows for developers to complies with HL7.

The standard HL7 is important in the healthcare domain is because it was created to provide interoperability between healthcare entities. HL7 defines types of messages for each process that exists in the healthcare domain and within each message the standard defines properties that should be present in order to provide the necessary information for the receiver. FHIR provides a workflow [14] that describes how FHIR instances and

definitions relate between themselves so that different systems can request e receive information using the same protocol. It is organized in modules that define different functional areas of the specification [15]. Section 2.3 of this document showcase these structures.

The usage of HL7 FHIR allows for healthcare data to be stored in a blockchain structure. This is because for a transaction to be stored in a block it must follow a standard to be stored. If transactions are in HL7 FHIR format they can be automated to be inserted into a block. But if a block containing HL7 information from a patient is stored in a public blockchain HIPAA compliance is immediately violated since anyone would be able to see private information from patients. A private blockchain is a more secure and controlled way of storing HL7 information from patients in a blockchain structure since only registered users have access to the blockchain. Users from a private blockchain have limited access so that a user can only see his own information stored in the blockchain. This makes more sense for healthcare information stored in the blockchain but still if traditional POW is used to validate the transaction, we would be giving access of private information to miners which also violates HIPAA principles. A good solution would be to remove third party's participation in the POW process. In order to achieve that the usage of software agents can be a valid solution. Software Agents [16][17][18] are self-contained software robots with capability of changing behaviors and exchanging information between themselves in an orderly fashion. It's possible to use agents that can receive transactions in HL7 and validate those transactions before inserting them into the blockchain. More than that, since all information is in the HL7 standard, agents can be used to anonymize the information before it is stored in the blockchain by looking into the ontology of the HL7 message. When a message type that have patient's private information is detected by an agent he can anonymize that message before its added to a block. This process further protects the privacy of the patient healthcare data. Other characteristic of using agents in a blockchain management is that if the validation of a block is made by agents in an orderly fashion no cryptocurrency is needed unless the domain of the application is to be a new cryptocurrency such as Ethereum [19]. In healthcare data storage there is no need for involving cryptocurrencies since the intent is to safely store and retrieve patient's information. This is called preemptive consensus in the scope of the proposed solution. Agents decide in an orderly fashion each block is going to be inserted before another starts to be validated. This type of consensus also prevents forks between copies of the blockchain because agents will not be competing to "*mine*" blocks.

In this author's opinion using public blockchain does not offers the needed security since third parties would be have copies of patient's information which offers risk in privacy breaching. Another problem is within the legal realm since different countries have different laws involving healthcare data storage making it difficult to send patient's information for a country such as United States with HIPAA enforcement to another country where that kind of laws are not enforced. Moreover, patients would need to authorize their information before it can be sent to different countries whereas otherwise if it is kept in the same country in different datacenters consent is more simplified.

1.1
Problem and Proposed Solution

As mentioned in the first section the main problem faced by healthcare institutions is how protected stored patient's data. HIPAA heavily enforces that breach of privacy is a very serious offense and the keepers of patient's information should take all measurements to avoid such risk. This document proposes an implementation of a multiagent based system for storing HL7 FHIR information in a blockchain structure. The MAS will not only store HL7 data in blocks structures but also will anonymize any information contained in such data by using knowledge of the HL7 standard. All management of the blockchain is going to be performed by software agents meaning that:

- An Agent is going to be responsible for receiving transactions
- An Agent is going to be responsible for verifying the validity of the transactions
- An Agent is going to be responsible for anonymization of any private information from HL7 data present in a transaction.
- An Agent is going to be responsible to maintain a "*God*" copy of the blockchain. The reason for this will be explained in detail in section 4 of this document.
- An Agent is going to receive confirmations from peers' agents that included the block in their blockchain copies
- An Agent is going to be responsible for retrieving information from the blockchain when solicitated.

The consensus between the agents in order to decide which block is going to be inserted into the blockchain is called preemptive consensus as mentioned in the introduction. This consensus is used because not only the proposed solution is a private blockchain but also because there is no need for agents to compete between themselves to "*mine*" a block. The "*mining*" is substituted by an auditioning agent that verifies the validity and privacy of the HL7 data. Section 4 of this document will detail the proposed implementation and agent roles.

1.2
Expected Contributions

The proposed solution is an attempt to unify the concepts between MAS, Blockchain and HL7 FHIR to create a system capable of receiving and retrieving patient's data in a safe and anonymous blockchain. The expected contributions of this are:

- Demonstration of how software agents can be applied in the healthcare domain to help with relevant issues.
- Demonstration of how software agents can be paired with blockchain concepts helping to create a new approach for blockchain implementation.
- Demonstration of how can HL7 standard help to keep patient's data private and anonymous when stored in a private blockchain.
- Showcase how can a society of software agents can manage a private blockchain avoid problems such as forks through the use of a preemptive consensus.

1.3 Organization

This document is organized as follows. Chapter 2 contemplates the background concepts needed to understand the proposed solution implementation. Chapter 3 presents related works studied to validate the relevance of the proposed solution as well to help finding similar application that helped with the modeling and implementation of the proposed solution. Chapter 4 details the implementation of the proposed solution for further understanding of how all concepts studied were applied.Chapter 5 presents the formalization of the proposed solution using statescharts. Chapter 6 presents a case study used for the proposed solution and how patients send and retrieve their healthcare data stored in the blockchain. Finally, Chapter 7 presents the conclusion that also mentions limitations and future work needed to enhance the solution.

2
Background

This section presents the background concepts used in the research and implementation of the proposed solution. Section 2.1 explains the software agent paradigm used in the conception of MAS systems. Section 2.2 presents the concepts of blockchain technologies such as blocks, block mining and consensus algorithm. In Section 2.3 will present HL7 FHIR standard, it's main concepts and relevance to the healthcare domain.

2.1
Software Agents

This section will present the main concepts that describes the software agent paradigm. In section 2.1.1 an introduction to the paradigm will be presented. Section 2.1.2 presents the main concepts of software agents such as behaviors and messaging. Finally, in section 2.1.3 the framework chosen for the implementation of the proposed solution is described.

2.1.1
Introduction

Software agents are self-contained computer programs capable of interacting with the environment that they are inserted. Software agents is a paradigm that allows for modeling complex system using more intuitive description due to the nature of its properties and behaviors. Wooldridge and Jennings [20] define agents as "... self-contained program capable of controlling its own decision making and acting, based on its perception of its environment, in pursuit of one or more objectives". To formalize this definition is necessary to define a set of characteristics that every agent must have. Below these characteristics are defined:

- **Autonomy**: An agent is capable of having its own state, and making its own decisions based on its experiences and interactions with its environment and other agents.
- **Reactivity**: An agent is present in an environment and should be capable of reacting to stimulations that can occur on that environment. These stimulations can be through sensors, interaction with users, internet messages or a set of any of those possibilities combined.
- **Pro-activeness**: An agent has a goal-oriented behavior. This means that an agent does not only act in response of a stimulation of its environment, but it takes initiative to achieve its goals and react to stimulation in a manner that helps to achieve such objective.
- **Social ability**: An agent has the capability of interacting with other agents and other entities. The Agent Communication Language (ACL)[21] was designed to help theses communications allowing for more structured messages with properties that help an agent to achieve its goals.

Next section will define how these characteristics are used in defining the main concepts used by software engineers when creating software agent frameworks that allows for the development of MAS in different domains.

2.1.2
Main Concepts

This section is going to present the main concepts used in most implementations of software agent frameworks. The main concepts for instantiating an agent are Agent Class, Behavior Class and Agent Service Class.

Agent Class is normally an abstract class that defines the basic functions and properties that the most basic agent of a framework must have to be accepted as a functional agent. This class is normally extended to allows the creation of more specialized agents that are better suited to solve problems in a specific domain. A basic agent class have properties that let the agent defines a set of behaviors, a set of the agent's services and other properties that are necessary for distributed communication such as an agent name and domain host.

Behavior Class is normally an abstract class also. That main difference is that most frameworks provides a set of extensions ready to be used by the developers using the framework. Mostly commonly provided behavior classes are OneShotBehavior Class and CyclicBehavior Class. OneShotBehavior Class is a type of behavior that an agent executes only once and stops. It's used for simple tasks that don't need to be repeated over time. CyclicBehavior Class is a type of behavior that is used to implement an agent's task that should run continuously until a pre-determined condition is encountered by the agent. Agent Service Class is a class instantiated by an agent that allow him to publish services into the network so that other agents can find these services and message agents with the services needed for the agent to achieve its goals. A software agent framework usually has an agent responsible for the yellow pages listing from all the agents in the network [22].

2.1.3
Smart Python Agent Development Environment (SPADE)

SPADE Agents [23] is the chosen agent-based framework for the development of the proposed solution. SPADE Agents is implemented using the python language [24] which is a great feature because it works in any operational system. SPADE presents a model for the connection, messaging and executing of its agents. It uses an Extensible Messaging and Presence Protocol (XMPP) [25] server to handle the connections and messages between its agents. Each agent have a Jabber ID (JID) and a password that allows the connections to the XMPP server in the format: myagent@myprovider.com . Each agent once connected to the host holds an open and persistent stream of communications with the platform. This stream is opened once the agent is started by the framework with the correct parameters.

The message dispatcher is a component that each SPADE agent has. This component is responsible for organizing the received messages into the correct behavior and also foe sending message for other agents when needed.

In SPADE a behavior is a task that an agent executes using patterns. SPADE provides behaviors such as cyclic and One Shot behaviors (see section 2.1.2) and some others such as Periodic and Finite State Machine behaviors. Each agent can have as many behaviors as it wants. When a message is sent to the agent its message dispatcher redirects the message to the correct behavior queue. In SPADE a behavior has a message template attached to it. It is this template that the message dispatcher uses to match a received message with the correct behavior. This flow allows for a behavior to only receive messages that are intended by using messages templates making communication much more efficient because there is no need analyze the ontology of a message before delegating it to a specific behavior.

2.2
Bitcoin Blockchain Technology – Public Ledger

This section will describe the blockchain technology and its main concepts and implementations. Section 2.2.1 will mention the history of the technology and its first appearance on the internet. Section 2.2.2 will describe Bitcoin main concepts. Section 2.2.4 will present the Ethereum implementation of the blockchain technology and how it introduced the smart contract concept into the blockchain technology. Finally, section 2.2.5 will discuss the future of blockchain in the software engineering field.

2.2.1
History

In 2008 a paper called “Bitcoin: A Peer-To-Peer Electronic Cash System ” by a man called Satoshi Nakamoto [26]. The electronic cash system intended to revolutionize the way payments are performed in modern days by transferring money directly between parties without the need of trusted parties to ensure safeguard of the transaction such as banks and governments. Bitcoin was the first implementation of such concept and used the term “cryptocurrency” to reference its currency also called bitcoin. The name Satoshi Nakamura is not a real name because the original creator wanted to be anonymous by unknown reasons. In [26] the authors definition for a blockchain is: “ …a distributed database of records, or a public ledger of all transactions or digital events…” . Originally intended for money transactions each block of the blockchain contains transactions between peers of the network. The blockchain uses a distributed consensus between the peers of the network called network nodes. These nodes agree to one block that is “verified” by a process called Proof-of-Work. Other characteristics of the blockchain is that it achieves trust from its transactions without breaching the privacy between parties involved. Anyone can go through all the transaction since the beginning of the blockchain and verify each one without knowing the real owner of the transactions. In January of 2009 the first block called the Genesis block was created with 50 bitcoins in its transactions. Soon after the first transaction between peers was created in block number 170 of the chain from Satoshi to Finney. These transactions are public for anyone to see at: https://www.blockchain.com/explorer .

2.2.2
Main Concepts

This section will explain how the main concepts of the blockchain technology works. All the necessary information for further understanding the proposed solution are defined in the next sections. Section 2.2.2.1 will explain how the structure of transactions works. Sections 2.2.2.2 will explain how the proof of work ensures that valid transactions are propagated to the blockchain. Section 2.2.2.3 explains how the consensus of the network is achieved and how problems such as forks are resolved. Finally, section 2.2.2.4 will present how a bitcoin block is constructed and its main properties. Figure 01 [27] shows the main flow of the blockchain life cycle.

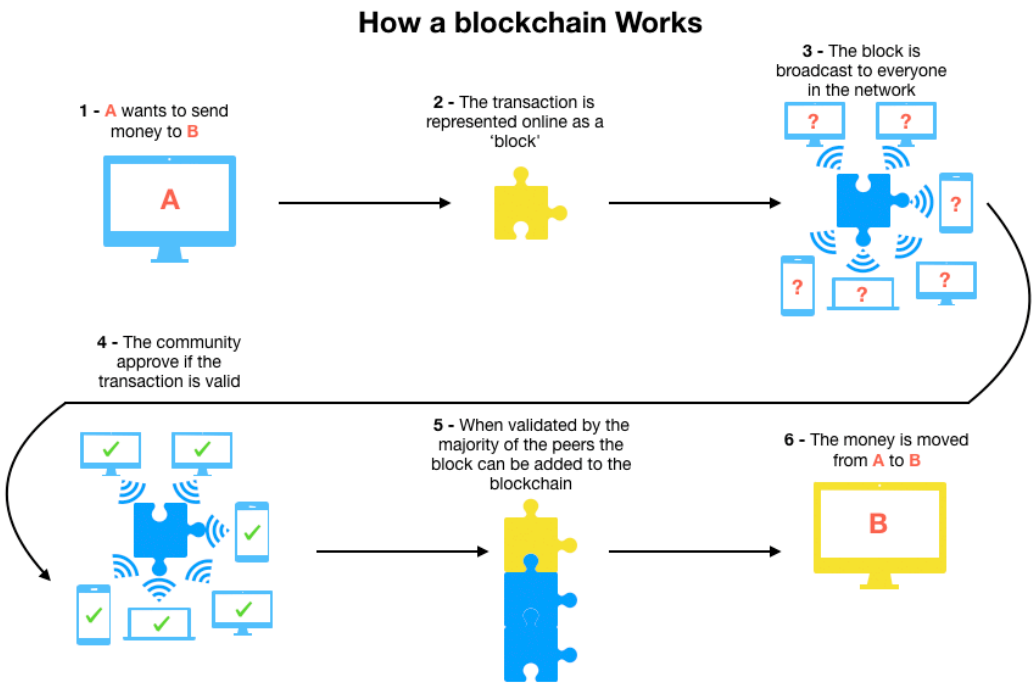


Figure 01 – The main flow of a blockchain network.

2.2.2.1 Transactions

Transactions in Bitcoin's blockchain is the representation of a transfer of a number of bitcoins that is broadcasted into the network. Then a transaction, or many transactions, are collected into blocks. Transaction structure has an output and an input, the input of a transaction is the output of a previous transaction and collects these outputs into a new transaction output. The owner of the previous output provides a signature and a public key that guarantee that the transaction is his own and valid. Bitcoins uses RSA private and public keys standard to provide proof of ownership without breaching the sender or receiver of transactions privacy. For the signature Bitcoin uses Elliptic Curve Digital Signing Algorithm (ECDSA) [29] to verify transactions. Figure 02 [28] show the structure of a basic transaction.

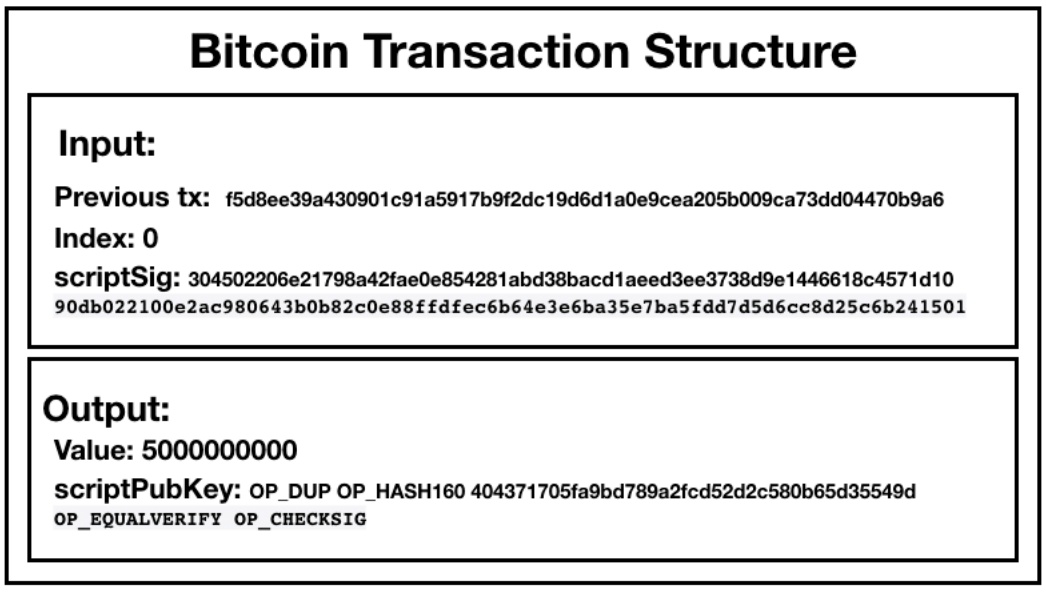


Figure 02 – An example of a bitcoin transaction.

In Figure 02 the input section of the example transaction has three fields:

- **Previous tx**: It means previous transaction hash. A hash given to the previous transaction that it is used to locate transactions in the Bitcoin blockchain.
- **Index**: It represents the index of the referenced output transaction. This is because a transaction between two peers can have multiple inputs and outputs and the protocol need to know which transaction it is being referenced by the new transaction.
- **scriptSig**: It's the proof that the peer A that is sending the bitcoins to peer B is the actual owner of that bitcoins.

The output section has two fields:

- **value**: Represents the number of bitcoins that are going to be transferred. The amount present in the inputs must be exact or the sender must provide another output to himself with the "change" of the transaction. Otherwise the remainder will become part of the fee to the miners of the network.
- **scriptPubKey**: It represents a script that must be executed by the receiver of the transaction in order to unlock the bitcoins received. This script can only be solved by the receiver that holds the private key that generated the address "404371705fa9bd789a2fcd52d2c580b65d35549d". If the return of the script is true, then the transaction is valid for the receiver to spend at his own will.

There are more details in the execution and encryption details that are present in the sending and receiving transactions with the Bitcoin network. These details can be found at [30].

2.2.2.2 Proof of Work Consensus

The consensus algorithm used in Bitcoin is called Proof of Work because it represents that a relevant amount of processing power must be executed by an actor in order to complete a new block and broadcast it to the network. In bitcoin one of the actors is elected a leader that decides the contents of the next block. This leader is the one responsible for broadcasting the new and valid block to the network.

In Bitcoin in order for these actors to generate a new block they have to solve a difficult mathematical problem called a hash puzzle. In order to resolve this puzzle actors must find a hash that is smaller or equals then the current target of the network. The target is a value defined to control the difficult of the hash puzzle that controls the flow of how many blocks are inserted per hour into the blockchain. This hash puzzle also prevents against Denial of Service (DOS)[31] attacks and limits the number of forks generated in the chain.

An actor that creates a new block a solves the hash puzzle is called a "miner", this is because the algorithm used to solve the hash puzzle is based on brute force alone. The name Proof of Work is due to this effort that its needed to solve the hash puzzle. The solution found needs to comply with the target defined by network and it is called a "Nonce"[32]. The nonce is a value that when hashed with the other properties of a block generates the block hash and it represents the solution of the hash puzzle. This ensures that it is a hard puzzle to solve but easy to verify by the other nodes in the network because all the other nodes need to do to verify is to hash the block with the nonce given by the miner to see the valid block hash. This process makes possible for the network to decide very easily if a new block is accepted as the next block to be inserted into the blockchain or not.

The guarantee that a new mined block is valid and immutable comes from the chain

heritage used in the block mining algorithm. When solving a hash puzzle one of the parameters used is the most recent block hash present in the blockchain hash. This ensures that is any information of any previous block is changed the hash generated is also going to change, making possible for the nodes in the network to verify such change and reject this difference since the majority of the network already agreed on the hash previously changed. Although is possible for someone to try and re-hash a previously hashed blocks to try and manipulate the chain. The hash puzzle algorithm would need to be solved again for each block successor of the block being violated. Figure 03 [33] show an example of how such attempt is very improbable due to the proof-of-work algorithm.

Bitcoin uses the hashcash [34] function which requires traditionally a service string, a nonce mentioned before and a counter. In bitcoin the service string is replaced by the block header encoded data (see next section), the block header contains the hash of the previous block inserted into the blockchain which ensure the cryptographic heritage of the chain. The miner must generate a Merkle tree root with all the transactions present in the block being mined with the left most leaf node being reserved by the nonce called ”extraNonce”. The counter wraps the “extraNonce” and it must be incremented at each attempt to solve the hash puzzle and to avoid repeating work. When a miner is mining a new block it repeatedly hashes the block header while incrementing the counter and the “extraNonce“ fields of the block header. After each Merkle tree generation the miner attempts a SHA256(SHA256(block header)) [35] encryption function to try and satisfy the target defined by the network. If the resulting hash is not enough the algorithm increases the counter and “extraNonce” and starts over until the target is satisfied. Details in the Bitcoin’s implementation of how the target is defined is at [36] but it’s not necessary for further understanding of the proposed solution. The next section will demonstrate details of a blockchain block structure and describe its properties.

In [37] the algorithm for achieving the distributed consensus is demonstrated in Figure 03.

Algorithm 2 The additional field and functions used by the Bitcoin consensus at p_i

19: $m = 5$, the number of blocks to be appended after the block containing
20: tx, for tx to be committed in Bitcoin

21: get-main-branch()$_i$: ▷ *select the longest branch*
22: $b \leftarrow$ genesis-block(B_i) ▷ *start from the blockchain root*
23: **while** $b.children \neq \emptyset$ **do** ▷ *prune shortest branches*
24: $block \leftarrow \text{argmax}_{c \in b.children}\{\text{depth}(c)\}$ ▷ *root of deepest subtree*
25: $B \leftarrow B \cup \{block\}$ ▷ *update vertices of main branch*
26: $P \leftarrow P \cup \{\langle block, b\rangle\}$ ▷ *update edges of main branch*
27: $b \leftarrow block$ ▷ *move to next block*
28: **return** $\langle B, P\rangle$ ▷ *returning the Bitcoin main branch*

29: depth(b)$_i$: ▷ *depth of tree rooted in b*
30: **if** $b.children = \emptyset$ **then return** 1 ▷ *stop at leaves*
31: **else return** $1 + \max_{c \in b.children}$ depth(c) ▷ *recurse at children*

Figure 03 – Bitcoin consensus algorithm.Source: [36]

In the above algorithm each node from the network continually keep running these processes to keep their node in sync with the rest of the network. One main feature of this algorithm is that it keeps constantly selecting the main branch of the network, meaning the longest chain. This is because since de consensus is based on the propagation of a new block trough any of the miners at any given point in time can generate temporary forks in the network. To resolve such problem the nodes of the blockchain always search for the longest chain in the network as the valid chain. Figure 04 [38] demonstrate such process.

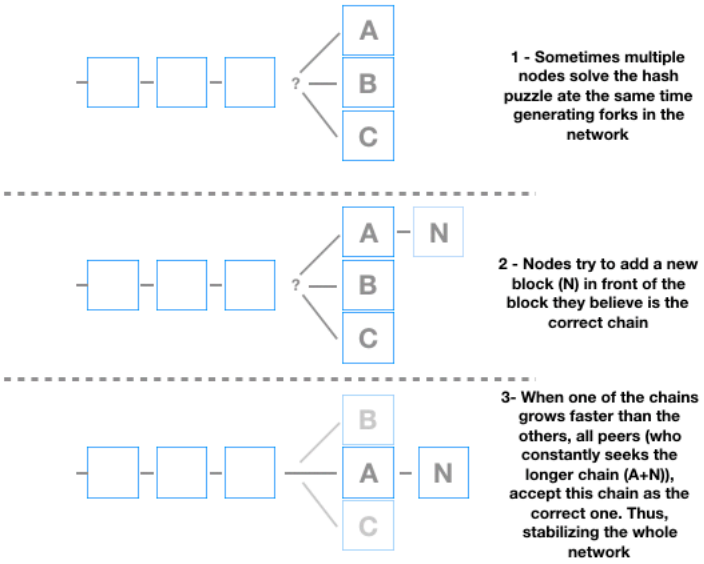


Figure 04 – How bitcoin resolves forks.

This scenario demonstrated in Figure 08 (further down) happens because when a miner solves a new block it starts to broadcast the block to its immediately nearby nodes, and only when the neighbor's nodes check the validity of the received block, they start to broadcast the block to their neighbor's nodes. When this is happening, another miner could resolve a block ate the same time and start the broadcast himself, thus result in forks in the chain. That's, again, the reason why nodes are always searching for the main branch. In [39] the way that the network resolves these left-over blocks called orphan blocks is explained. In summary what happens is that the transactions present in the orphan block is returned to the transaction pool to be later added to a block that is going to be mined.

The transaction pool [40] is a cache of received transactions sent by user's wallet that is kept being collected by miners that are trying to create and mine blocks in order to receive their rewards for their work.

In order to the Proof-of-Work algorithm work the miners need to keep hashing to solve new blocks constantly. But due to the increase in difficult of the target over time more and more power is needed to solve a new block. This creates a necessity to encourage the miners to give their hashing power to the network to the means of rewards in bitcoins. Each miner that solve successfully a block that is accepted by the network is rewarded by receiving a reward in bitcoins to keep stimulating them to keep hashing. This is also very important because incentivize miners to be fair players in the managing of the blockchain since if a majority of miner tries to disturb the blockchain eventually would means their own loss.

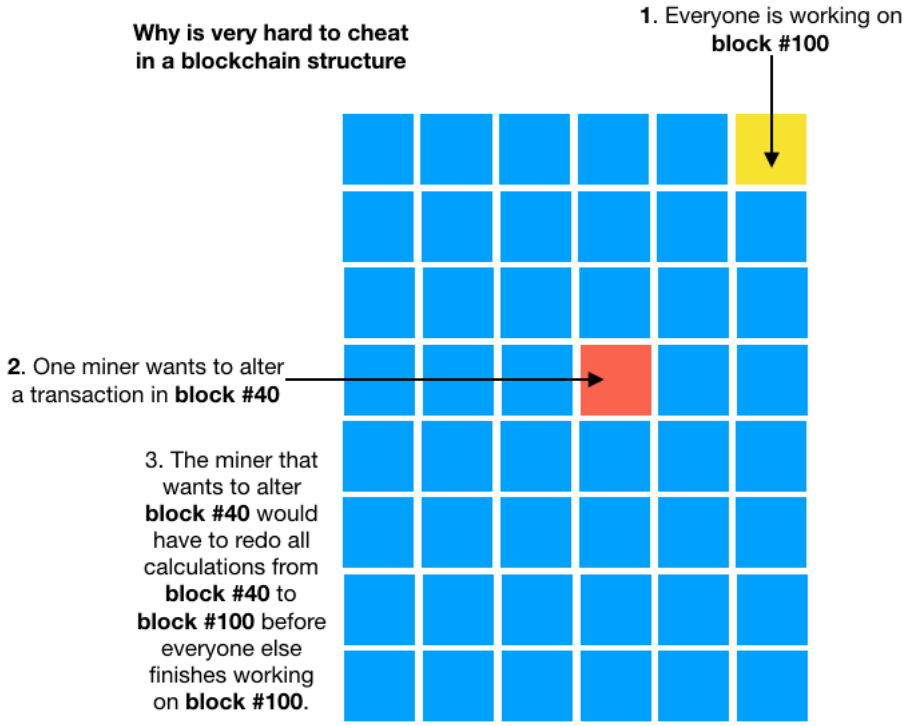


Figure 05 – Demonstration of how attempt to change previous blocks is very improbable to be achieved. Source: https://steemit.com/bitcoin/@darkflame/why-you-cant-cheat-at-bitcoin

2.2.2.3 Block Structure

Bitcoins transactions are permanently stored in structures called blocks. In [41] they are described as pages of a record book or records on a public ledger. They are organized into a linear sequence with an incremental timestamp known as blockchain. Miners group transactions into new blocks and mine such block in a race to see who can solve a block first. The winner if the conditions are valid receives a reward. Since each block have cryptographic heritage from the most recent bock added to the blockchain the successor will also carry that heritage. As older the block becomes its harder and harder for some bad intentioned miner to change its values as shown in Figure 05. This is the reason that blockchain ensures that its transactions are irreversible. Figure 06 bellow presents the structure of a Bitcoin block and its properties.

Field	Description	Size
Magic Number	value always 0xD9B4BEF9	4bytes
Blocksize	number os bytes up to the end of block	4 bytes
Blockheader	meta information from block	80 bytes
Transaction Counter	positive integer	1-9 bytes
Transactions	non-empty list of transactions	<transaction counter>

Figure 06 – Bitcoin block structure.

The fields present in a blockchain block are:

- **Magic no**: It is an identifier used by the blockchain node to determine with block structure is expected. This is because variations of the blockchain project can define different block structures and change this field. In case of the Bitcoin is always the value presented in Figure 04.
- **Blocksize**: Represents the numbers of bytes of the block.
- **Blockheader**: It is the structure used to create the block hash, maintain the hash heritage and to keep the nonce of the block. It also keeps the Merkle tree root of the transactions of the block.
- **Transaction counter**: Indicates how many transactions are present in the transactions field of the block.
- **Transactions**: List of the transactions in the block. They are no encrypted and are visible to anyone willing to read them.

As mentioned above the block header is the structure used to generate the block hash. Figure 07 presents such structure.

Field	Purpose	Size
Version	Block version	4 Bytes
hashPrevBlock	number os bytes up to the end of block	32 Bytes
hashMerkleRoot	meta information from block	32 Bytes
Timestamp	positive integer	4 bytes
Bits	non-empty list of transactions	4 bytes
Nonce	The nonce used to generate this block, to allow variations of the header and compute different hashes	4 bytes

Figure 07 – Blockchain block header structure.

The fields present in the block header are:

- **Version**: Used to keep the version of the block being encrypted. Changes overtime change the version of the block and is used by miners to know with process to use when mining a block.
- **HashPrevBlock**: Represent the hash of the previous block in which this block was the successor. It is imperative the presence of such field otherwise the cryptographic heritage would be lost and destroy the chain integrity.
- **HashMerkleRoot**: It's the hash generated using the transactions plus the nonce and counter.
- **Time**: The timestamp of the block that is also part of keeping the hash of each block unique.
- **Bits**: It the target used by the miner to know with difficult he should use when mining a block.
- **Nonce**: It's the value found by a miner when a block is solved. This is used by other nodes to quickly verify if a block is valid or not.

Figure 08 shows how blocks are chained together by their hash heritage.

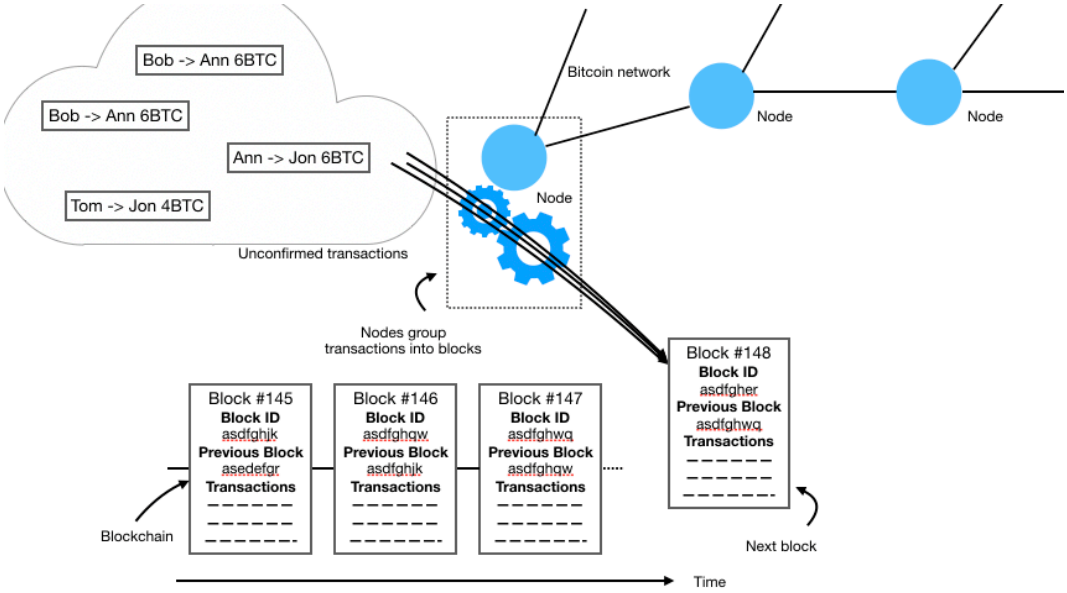


Figure 08 – How the blockchain clock relationship works.

As shown in Figure 08 the nodes group transactions into nodes and form them into a valid block by solving the hash puzzle. Once that is done the it is added to the nodes blockchain and propagates to its neighbors' nodes, keeping the cryptographic heritage.

2.2.3
Ethereum

Ethereum is a public blockchain such as Bitcoin. The main difference between them is that Ethereum introduce the concept of smart contracts aside from money transactions. Ethereum currency is called ether and is generated very similar to bitcoins meaning that Ethereum transactions are stored into block and mined to validate that block. Ethereum messages can be simply transactions of ether or inputs to a smart contract. Smart contracts are implemented in the Solidity [42] language which is Turing complete. These contracts run inside a middleware called Ethereum Virtual Machine (EVM). Each new contract is deployed inside the EVM and becomes available to be used to intermediate messages between parties.

Another deviation of Ethereum in comparison with Bitcoin is that the restricted public access to data can be public or private. Figure 09 [43] show a comparison between Bitcoin and Ethereum comparing both implementations main features.

Bitcoin X Ethereum	Bitcoin	Ethereum
Permission restriction	Permissionless	Permissionless
Restricted public access to data	Public	Public or Private
Consensus	Proof-of-Work	Proof-of-Work
Scalability	High node-scalability, Low performance scalability	High node scalability, Low performance scalability
Centralized regulation (governance*)	Low, decentralized decision making by community/miners	Medium, core developer group, but EIP process
Native currency	Yes, bitcoin high value	Yes, ether
Scripting	Limited possibility, stack-based scripting	High possibility, Turing-complete virtual machine, high-level language support (Solidity)

Figure 09 – Comparison between Ethereum and Bitcoin.

The table above shows that Ethereum although shares lots of similarity with Bitcoin such as the consensus, permissions and anonymity, its scripting deviation allows for much more possibilities for its implementation besides money transactions. One such feature is the Decentralized Applications (DAPS). DAPS are web-based applications that instead of using traditional APIs to run their backend, they use smart contracts and blockchain. This means that their business logic is run based on smart contract and their persistence of data happens on the blockchain. More details can be found at [44].

2.2.4 Private Blockchain

Private blockchain [45] are blockchains in which the public ledger notion is not true. In Private blockchains only authorized and registered users can access information on the chain. The main advantages of a private blockchain are:

- **Enterprise controlled**: This means that any users that wants to access any given information on the blockchain must be registered and have the necessary permissions. This heavily diminishes the chances of anonymity breach and chain manipulation such as 51% attacks.
- **Faster transactions**: Since the node distribution is controlled, meaning that not any given person can become a new node in the network, transactions are much simpler and faster than a public blockchain. Although a public blockchain have valid reasons to use a POW Consensus as was discussed before.
- **Better scalability**: The enterprise responsible for the managing of the blockchain can add new nodes as its needed in a controlled and orderly manner.
- **Flexible consensus**: Since it's a closed and private network the consensus algorithm becomes much more flexible to be implemented. This is because there is no need to implement a really robust protocol since only authorized and compliant actors are participant of the network.

The adoption of private blockchain are really based on the necessity of the domain in which they are going to be applied. For money transactions and currency, a public blockchain is more interesting since you want everyone using its currency to grow the number of nodes and the value of the currency. In other domains such legal contracts, democratic elections and healthcare, the case study of the proposed solution, a private blockchain is much more intuitive since healthcare data cannot be public for anyone to see.

Another point that is making the adoption of private blockchain for certain domains, such as healthcare, is the fact that sending patient information to different countries is not possible due to laws and regulations to protect the breaching of information. The proposed solution of this document adopted a private blockchain in part for these reasons and also for the flexibilization of consensus algorithm together with the user-controlled access to information.

2.2.5 Future

Blockchain is a technology that is being heavily used by software engineers to propose solution for no more than 10 years. Although the concept of a cryptographic secured chain of blocks existed before [46], Satoshi Nakamura was the first to implement a functional and successful blockchain solution. Since then multiple implementations of blockchains have being presented for not only cryptocurrencies but for many other domains [47]. Different concepts and application being smart contracts, Distributed Applications over blockchain, Distributed Business Governance over blockchain, Deed records over blockchain, Legislations over blockchain, stocks purchase and transfer over blockchain and Democratic voting system over blockchain are some of the solutions that spinoff from Satoshi's Bitcoin success.

Much discussion is still needed to see how all these different domains will deal with the concept over time as the technology matures and becomes more robust.

2.3 Health Level Seven - Fast Healthcare Interoperability Resources (FHIR)

2.3.1 Introduction

FHIR [48] is a standard framework created by HL7. The main reason for the creation of FHIR was to implement the HL7 standard into a RESTful architecture since older versions V2 and V3 weren't implemented in such manner. FHIR segmented its solution into modules that are called "Resources". These resources can be implemented into systems giving them the capability of interoperability with other systems as long they are also HL7 compliance.

The main reason for the compliance to such standard is to promote the interoperability between healthcare systems without the need for continuous integration between multiple systems. With the usage of FHIR web based architectural framework, applications can trade information without the need of ad-hoc implementations for each new system that appears to trade information. As long as a system complies with FHIR it can benefit of interoperability with any other FHIR compliance system.

FHIR can be applied in many contexts of healthcare such as mobile apps, cloud communications, HER-based data sharing, server communication between healthcare institutions and more. To actually attend to all these fields necessities FHIR defines a framework that allows the extension of FHIR resources and uses pre-defined profiles that describes the usage of such extensions.

Figure 10 [49] shows and example of a FHIR resource that represents a patient.


```
<Patient xmlns="http://hl7.org/fhir">
  <id value="glossy"/>
  <meta>
    <lastUpdated value="2014-11-13T11:41:00+11:00"/>
  </meta>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456. Male, 24-Sept 1932</p>
    </div>
  </text>
  <extension url="http://example.org/StructureDefinition/trials">
    <valueCode value="renal"/>
  </extension>
  <identifier>
    <use value="usual"/>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
      </coding>
    </type>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <active value="true"/>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender value="male"/>
  <birthDate value="1932-09-24"/>
  <careProvider>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </careProvider>
</Patient>
```

Figure 10 – FHIR patient resource. Source: https://www.hl7.org/fhir/summary.html

As shown in Figure 10, each resource have a identifier that defines to other system what type of resource is being presented, a human readable summary to enhance the transparency of the resource, an extension with a URL to the definition of such extension along with the value of such extension and the standard data of the patient which includes properties such as name, gender and more.

FHIR modules represents different areas of the specification to help developers choose which area of the standard they want to comply. Figure 11 [50] shows such modules and their areas.

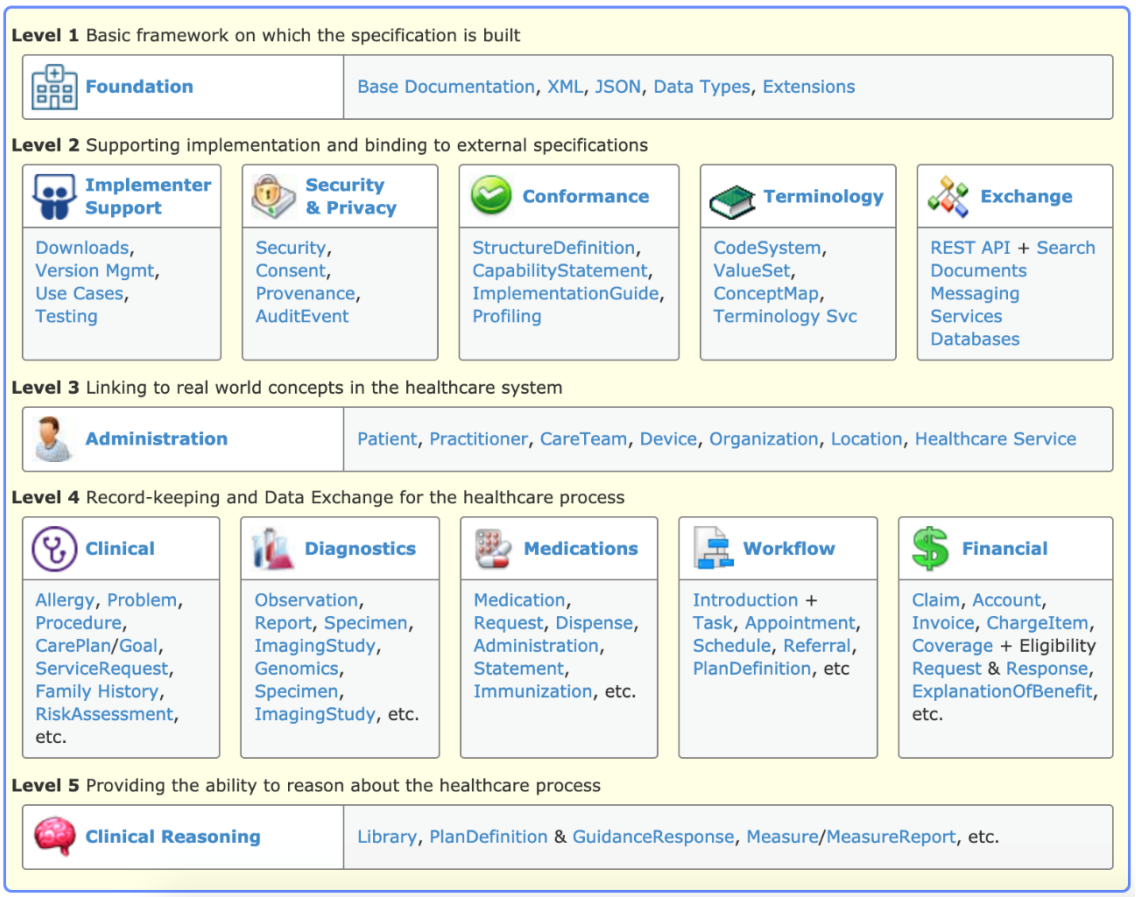


Figure 11 – FHIR modules. Source: https://www.hl7.org/fhir/modules.html

Each FHIR modules contains:

- **Scope and Index**: Description of the content covered by the module with index.
- **Use cases**: Used to exemplify the usages of the module and to help developers.
- **Security and privacy information**: Information about what each module security and privacy should be contemplated by the developer.
- **Roadmap**: Defines the progress of the module and what is still in development.

Any given application or systems that wants to be FHIR compliance must implement all modules. But since FHIR was designed to be flexible if a system only needs part of the modules to solve their needs it is also possible to be done. The specification for developers to become compliance is very large and complex and it is not in the scope of the proposed solution to discuss the inner concepts used to implement a FHIR application. The FHIR foundation provides guides, examples and implementations to help developers. Also, there are a variety of solutions in the market that already implemented the standard being a framework or a complete system.

2.3.2 Observation Resource

For the scope of the proposed solution implementation FHIR Observation resource were used to be stored in the blockchain. This is for the first version of the implementation and is sufficient to demonstrate the usability of the proposed architecture.

A FHIR observation resource is one of the most important elements of the framework since it is used for a variety of applications. Some uses of the observation resource are:

- Vital signs such as body weight, vital sings and respiratory measurements.
- Laboratory data.
- Imaging results.
- Clinical findings, for example, abdominal tenderness.
- Device measurements such as EKG data.
- Clinical assessment tools.
- Personal characteristics such as eye color.
- Social history such as tobacco use.
- Core characteristics such as pregnancy status, or a death assertion.

Normally the observation resource is a series of name-value pairs along with meta information of the characteristics of the conditions and context where that observations was created. The XML snippet below shows the structure of an Observation resource.

Observation Message XML Snippet:

```
<Observation xmlns="http://hl7.org/fhir">
 <!-- from Resource: id, meta, implicitRules, and language -->
 <!-- from DomainResource: text, contained, extension, and modifierExtension -->
 <identifier><!-- 0..* Identifier Business Identifier for observation --></identifier>
 <basedOn><!-- 0..* Reference(CarePlan|DeviceRequest|ImmunizationRecommendation|
   MedicationRequest|NutritionOrder|ServiceRequest) Fulfills plan, proposal or order --
></basedOn>
 <partOf><!-- 0..* Reference(MedicationAdministration|MedicationDispense|
   MedicationStatement|Procedure|Immunization|ImagingStudy) Part of referenced event -
-></partOf>
 <status value="[code]"/><!-- 1..1 registered | preliminary | final | amended + -->
 <category><!-- 0..* CodeableConcept Classification of  type of observation --
></category>
 <code><!-- 1..1 CodeableConcept Type of observation (code / type) --></code>
 <subject><!-- 0..1 Reference(Patient|Group|Device|Location) Who and/or what the
observation is about --></subject>
 <focus><!-- 0..* Reference(Any) What the observation is about, when it is not about the
subject of record --></focus>
 <encounter><!-- 0..1 Reference(Encounter) Healthcare event during which this
observation is made --></encounter>
 <effective[x]><!-- 0..1 dateTime|Period|Timing|instant Clinically relevant time/time-
period for observation --></effective[x]>
 <issued value="[instant]"/><!-- 0..1 Date/Time this version was made available -->
 <performer><!-- 0..* Reference(Practitioner|PractitionerRole|Organization|
   CareTeam|Patient|RelatedPerson) Who is responsible for the observation --
></performer>
```

<value[x]><!-- 0..1 Quantity|CodeableConcept|string|boolean|integer|Range|Ratio|
SampledData|time|dateTime|Period Actual result --></value[x]>

<dataAbsentReason><!-- 0..1 CodeableConcept Why the result is missing --
></dataAbsentReason>

<interpretation><!-- 0..* CodeableConcept High, low, normal, etc. --></interpretation>

<note><!-- 0..* Annotation Comments about the observation --></note>

<bodySite><!-- 0..1 CodeableConcept Observed body part --></bodySite>

<method><!-- 0..1 CodeableConcept How it was done --></method>

<specimen><!-- 0..1 Reference(Specimen) Specimen used for this observation --
></specimen>

<device><!-- 0..1 Reference(Device|DeviceMetric) (Measurement) Device --></device>

<referenceRange> <!-- 0..* Provides guide for interpretation -->

<low><!-- 0..1 Quantity(SimpleQuantity) Low Range, if relevant --></low>

<high><!-- 0..1 Quantity(SimpleQuantity) High Range, if relevant --></high>

<type><!-- 0..1 CodeableConcept Reference range qualifier --></type>

<appliesTo><!-- 0..* CodeableConcept Reference range population --></appliesTo>

<age><!-- 0..1 Range Applicable age range, if relevant --></age>

<text value="[string]"/><!-- 0..1 Text based reference range in an observation -->

</referenceRange>

<hasMember><!-- 0..* Reference(Observation|QuestionnaireResponse|
MolecularSequence) Related resource that belongs to the Observation group --
></hasMember>

<derivedFrom><!-- 0..* Reference(DocumentReference|ImagingStudy|Media|
QuestionnaireResponse|Observation|MolecularSequence) Related measurements the
observation is made from --></derivedFrom>

<component> <!-- 0..* Component results -->

<code><!-- 1..1 CodeableConcept Type of component observation (code / type) --
></code>

<value[x]><!-- 0..1 Quantity|CodeableConcept|string|boolean|integer|Range|
Ratio|SampledData|time|dateTime|Period Actual component result --></value[x]>

<dataAbsentReason><!-- 0..1 CodeableConcept Why the component result is
missing --></dataAbsentReason>

<interpretation><!-- 0..* CodeableConcept High, low, normal, etc. --></interpretation>

<referenceRange><!-- 0..* Content as for Observation.referenceRange Provides guide
for interpretation of component result --></referenceRange>

</component>

</Observation>

In the context of the proposed solution intent to only send anonymized FHIR data to the blockchain. The fields that must be anonymized are the fields subject and performer.

These fields can possibly be used for linking with a patient resource. The patient resource contains private information such as addresses and billing information. Because of this it must be anonymized to prevent propagation of private data to the blockchain.

The example above demonstrates how the use of FHIR standard can be applied to all modules and resource to guarantee that private information can be anonymized automatically by software agents before being sent to the blockchain.

2.3.3 Future

The HL7 FHIR framework is still a work in progress. Most of its module are ready to be used in production but some functionalities and limitations are still being developed. The developer should always consult the roadmap section of each resource to be aware of its limitations.

The future for the HL7 FHIR is still uncertain outside of the United States of America since the implementation by institutions takes time and its costly. Furthermore, if only some institutions comply to the standard, the motivation and reasoning for using FHIR becomes much less relevant because the interoperability is not going to be used enough to justify the effort and money invested. That's the reason why such standard must become part of the regulatory system for healthcare institutions, thus making interoperability possible. Remembering that the interoperability has one main reason: Enhancing patients health and treatment.

3
Related Works

This section will present the related works studied to help with the research and development of the proposed solution. Many implementations were studied and the more relevant to the topic in hand will be explained and compared with the proposed solution. Section 3.1 mention a literature review [51] that compares MAS and blockchain technology that was very important for the conception of the proposed architecture since it was necessary to understand if MAS and blockchain could be applied successfully in the healthcare domain. Section 3.2 presents a work [52] that lists and explain possible use cases for using blockchain in the healthcare domain. Finally, section 3.3 presents a work [53] that presents a prototype for "MedRec" a case study for applying electronic health records and medical research data in blockchain. Other works were studied in the research [54] [55] [56] [57] [58] [59] [60] and provides valid information, but the main contributors for the conception of the proposed solution are presented below.

3.1
Multiagent Systems and Blockchain: Results from a Systematic Literature Review

This work presents a systematic literature review discussing how using blockchain in MAS and stablishing a research road-map, as well identifying challenges in these fields. The main contributions of this paper are:

- Better understanding the motivation and relevance of the existing works that combine blockchain technology and MAS.
- An analysis that justifies using blockchain technology to address the requirements of MAS.
- Presents challenges to apply in practice blockchain technologies in MAS frameworks and provides directions for future research.

In its discussion section it shows that most works studied proposed the usage of MAS to solve issues in blockchain technology such as scalability, processing power and smart-contracts non-repudiation. But none of them provides in-depth analysis or results indicating that they are in early stages of development and need maturation. Another relevant point discussed by this work is the critique of using private blockchain to manage anonymized information sharing by using a blockchain to keep track of all events of data sharing. The author makes a case that by the law in its country any anonymized data management it is not considered property of the original creator. Meaning that using a very costly to manage blockchain to ensure historical control of the creator data's sharing events is not necessary or relevant. It may be true that maintaining a blockchain only to keep track of permissions of data sharing is not that worth based on the cost of processing but, maintaining the actual data that it is going to be shared in a private blockchain to ensure immutability and anonymization and control the distribution in an orderly manner can be useful in this author opinion and experience.

The paper follows up with a listing of the open challenges of applying blockchain technology towards MAS applications:

- The creation of a legal base for blockchain technology.
- Verifying the correctness of the chain code and smart contracts.
- Preserving the distributed nature of blockchain, especially in public blockchains since mining pools can be disruptive and break the actual distribution.

- Developing solutions to ensure privacy and anonymity.
- Dealing with the adoption of blockchain technology by the community.
- Managing membership in private blockchain scenarios.
- Dealing with scalability issues of blockchain.
- Ensuring that all properties such as digital signatures, hashing algorithm and chain correctness are robust.

Lastly, the literature review presents a mapping between MAS requirements and the main properties of blockchain technology. Figure 12 below presents the mapping.

		BTC Properties				
		Immutability	Complete History	Distributed Consensus	Cryptography primitives (eg., hash, digit sign)	Smart Contract
MAS Requirements	Trust	X	X	X	X	X
	Reputation	X	X	X		
	Data integrity	X	X	X	X	
	Traciability	X	X	X		X
	Transparency	X	X	X	X	X
	Anonymity				X*	
	Privacy	X			X*	
	Authenticity	X	X	X	X	

Figure 12 – Mapping between blockchain technology and MAS.

The above mapping is used to show that reputation, transparency and traceability are essential in competitive behaviors between agents, whereas trust and accountability are important for collaborative behavior. The mapping also shows that not all features of MAS could gain benefits from adopting blockchain into their architecture without some extra mechanism. They are marked with "*" in Figure 12 being the properties of anonymity and privacy. The point made is that additional mechanism has to employed to ensure privacy and anonymity such as cryptographic primitives used off blockchain, secret sharing scheme [61] and other examples.

The main contribution of this paper for the proposed solution is that although blockchain can benefit MAS with its main features the necessity of external controls to ensure anonymity and privacy are needed. These arguments were essential when to choose a private blockchain architecture in which the control of access can be applied by service layers outside of the blockchain layer providing more control over how an information is anonymized and kept private before being forward to the blockchain.

3.2
Blockchain Technology Use Cases in Healthcare

This work presents a study that focus on the applicability of blockchain technology in healthcare by:

- Identifying potential blockchain use cases in healthcare.
- Providing a case study that implements blockchain technology.
- Evaluating design considerations when applying this technology in healthcare.

In its analysis of challenges this paper also brings the difficult in keeping secure links that connects all independent healthcare systems involved in the healthcare environment. It mentions HL7 FHIR as a solution to interoperability but makes a case of the difficult in

adapting all systems to the standard. Figure 13 present in the paper show the challenge in integrating all these systems.

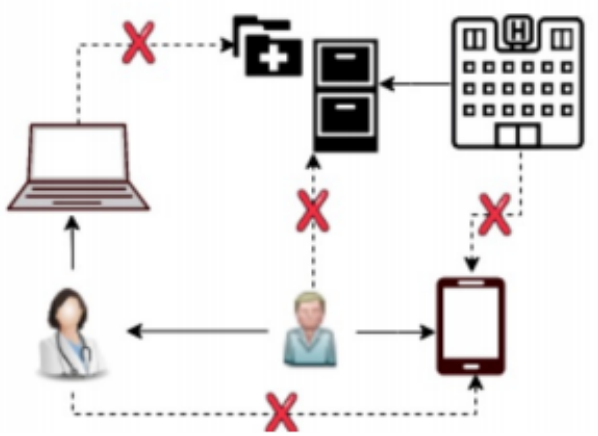


Figure 13 – Shows the difficult challenge to provide interoperability between all parties involved in healthcare environment.

It follows by presenting a table with seven possible applications of blockchain in healthcare based on the challenges the author discussed that involved interoperability and patient centered care. The applications of blockchain in healthcare presented were:

1. Prescription Tracking to Detect Opioid Overdose and Over-Prescription.
2. Data Sharing to Incorporate Telemedicine with Traditional Care.
3. Sharing Cancer Data with Providers Using Patient-Authorized Access.
4. Cancer Registry Sharing to Aggregate Observation Cancer Cases.
5. Patient Digital Identity Management for Better Patient Record Matching.
6. Personal Health Record for Accessing and Controlling Complete Health History.
7. Health Insurance Claim Adjudication Automation to Surface Error and Fraud.

The chosen application that was further studied was the sixth on the list because it was similar to the proposed solution.

The current practice adopted by most healthcare institutions is to use a provider centric electronic health record to keep patient's data. But recently personal health records applications are empowering patients and helping them to keep their own records and control of access by others since they are the actual owners of such information. The paper argues that with blockchain if connect securely with existing health systems can manage patient's information without having the patient having to request manually or digitally their information from each healthcare institution that they visit. It also promotes the idea of using smart contracts to improve the control of origin and sharing of their health data.

The proposed solution of this document is in alignment with this case study since what is intended is to store patient's health data in a secure and anonymous way allowing patients to request their data at will.

The paper also presents a case study using Ethereum DApps for electronic health records of patients called DASH. Figure 14 show the overview structure of DASH.

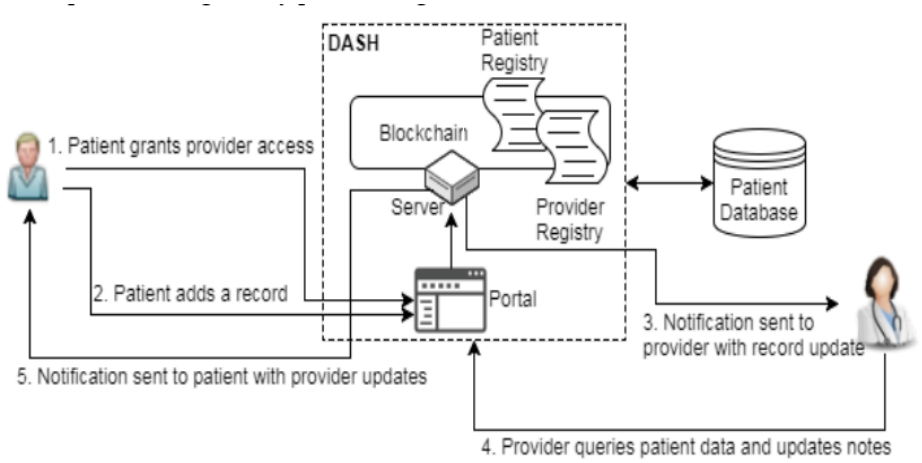


Figure 14 – Overview of DASH DApp.

The DASH DApp is meant for allowing not only patient health records to be stored in the Ethereum blockchain, but also to allows providers to access such information through the usage of smart contracts. This is an approach that deviates from the proposed solution because it involves providers being able to access patient's information and update notes from it. It is very important to notice that this kind of application directs revolves around Ethereum blockchain to keep registry from patient and provider actions, the health record in itself is stored in a database. The proposed solution of this work is to stored final state HL7 FHIR data from patients, not to allow manipulation by third parties.

3.3
A Case Study for Blockchain in Healthcare: "MedRec" Prototype for Electronic Health Records and Medical Research Data

MedRec is a prototype for electronic health records and medical research data built on top of Ethereum's blockchain. MedRec uses smart contracts to intermediate the interaction between patients and providers. Patients register themselves through the use of public keys to ensure privacy. It has three main contracts:

- **Registrar Contract**: Used to register patients and providers to enable interaction between them.
- **Patient-Provider Relationship Contract** (PPR): It used between two nodes in the system when one node stores and manages medical records for the other. This contract creates a hash of the patient data stored in the provider database through and storing the SQL query used to retrieve its information on the contract. This ensures that the patient have control of any changes of his records and also allows his to retrieve his records in an automated manner without having to issue requests to the provider.
- **Summary Contract**: This contract is responsible to locate the record history of the participants of the system. It holds a list of references to all participants patient-provider contracts registered in the system. Patients summary contract contain all references to all care providers they engaged to and providers would have references to the patients they served.

The MedRec works based on network nodes being deployed on providers and for patients they have to install they nodes on their PC. Figure 15 shows the workflow of the system.

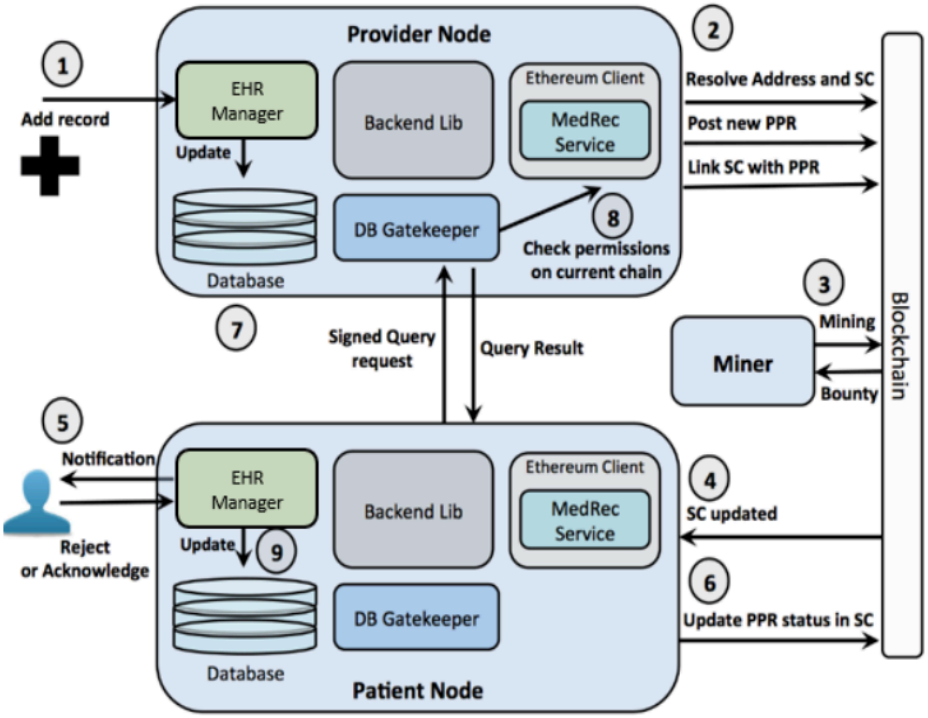


Figure 15 – The workflow of MedRec.

MedRec is a proof-of-concept implementation of how an electronic health record can be implemented based on a blockchain architecture. The proposed solution of his work studied MedRec to better understand how can records be tracked on the blockchain in an automated manner. The main deviation of the proposed solution from MedRec implementation is that it is not built on top of the Ethereum. Ethereum is very a powerful platform but it creates a dependency that cannot be reverted for the continuous working of MedRec. If Ethereum comes to a halt by any reason the system also stops working and patients lose access to their records. This is the reason that the proposed solution does not use any mainstream blockchain platform.

4
Architecture and Workflow

This section will explain how the architecture of the proposed solution was structured. Section 4.1 will explain how the architecture is defined and explain its layers. In Section 4.2 will explain each agent role in the management of the MAS Blockchain layer. In Section 4.3 will explain the consensus that is used by the agents to agree into a block. In Section 4.4 explain why no cryptocurrency is needed for the proposed architecture. In Section 4.5 the performance achieved so far with the first implementation is discussed. In Section 4.6.discusses the security involved to maintain the patients records secure and anonymous Finally, in Section 4.7 the extension of HL7 FHIR compliance is explained.

4.1
Architecture

Figure 16 shows the architecture of the proposed solution and its workflow. The next sections will explain how each layer works and each agent roles.

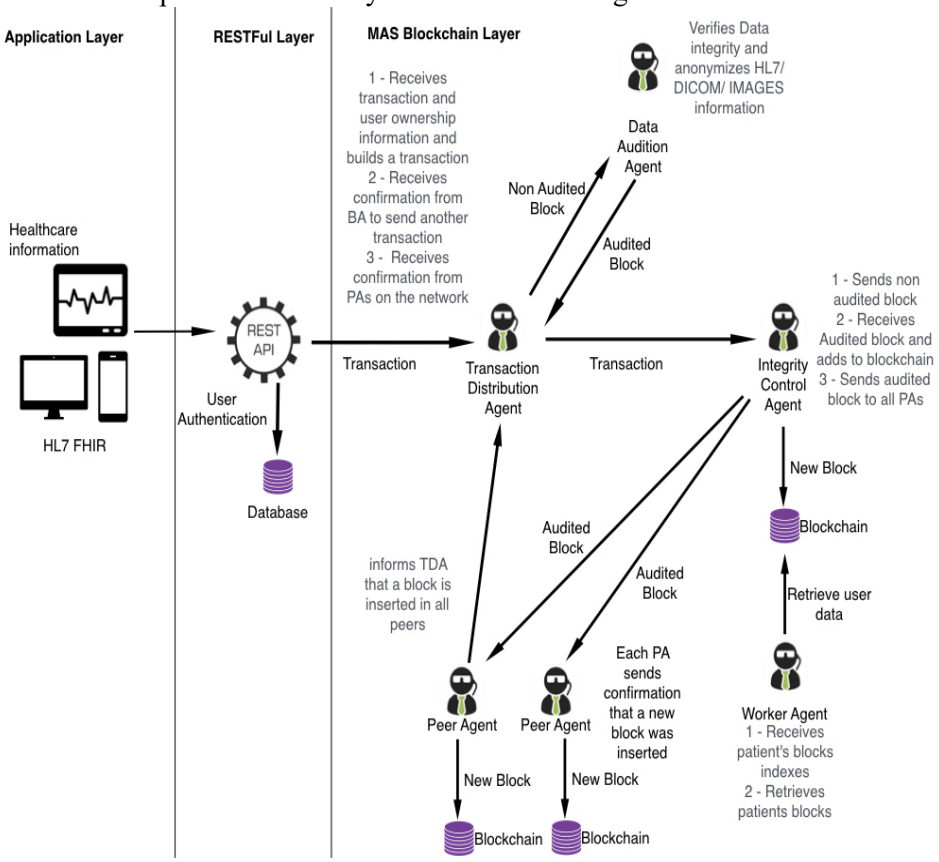


Figure 16 – Proposed solution architecture.

4.1.1
Application Layer

The Application Layer is responsible to collect information from the user. It can be received by any device as long as that information is HL7 compliant. Authenticated users send their information through a RESTful service. For example, a health application can collect data from users (patients) monitoring devices and send that information to the blockchain using the RESTful services of the solution.

4.1.2
RESTful Layer

The RESTful Layer offers services needed to control who and how healthcare information from a patient is sent and stored in the blockchain. The user authentication guarantees that only registered users can send information to the blockchain. This layer also manages the user private/public keys and addresses in order to create a valid transaction to be stored in the private blockchain. Another responsibility of the RESTful layer is to keep the transaction pool ready for the MAS blockchain Layer requests. The RESTful layer also keeps patient's block indexes to ensure that when information from a patient must be retrieved from the blockchain there is no wasteful querying through useless information. This is achieved by organizing the blockchain files in ranges of 128 blocks per file. For example, one patient has two blocks in the blockchain, one with an index of 90000 and other with index of 100000. If we did not know these indexes and only had the block hash or the user address, we would have to run through the entire blockchain files until such blocks were found. But by keeping users' block indexes and knowing that the blockchain files are kept in 128 block range files the search has a maximum length of 128 blocks per block to be retrieved. This is achieved by naming the blockchain files using simple integers indexes such as:

0.dat; 1.dat ... n.dat.

To find such files, an algorithm is used, where "a" is the dividend that is given by the index of the block we want retrieve from the blockchain, "b" is the divisor that is always 128, "q" is the quotient, "r" the remainder and N the file name where the block index we want is located.

If a < 128 then N = 1
If a % b > 0 then N = q + 1
If a % b == 0 then N = q

Example of the above algorithm: a = 100000, b = 128 then a > 128; a % b = 32 and 32 > 0 so the block requested is on file 782, which is the integer part of the quotient of the division plus 1.

4.1.3
Agent Blockchain Layer

The MAS Blockchain layer is responsible for managing the blockchain. Each software agent has a role that keeps the blockchain balanced and the transactions valid. They will be described in more detail in Section 4.2. The block structure generated by the agents is presented in Figure 17. Each block holds transactions from a patient until a maximum of 1 Megabyte. This limitation is to prevent network latency when broadcasting blocks to the network's agents. One characteristic of the proposed solution that differs from other implementations is the fact that each block has a transaction of only one patient. The reason for this is to facilitate retrieving transactions from one patient without having to sift through hundreds of other patient transactions. In the "data" field, a Merkle tree root would normally be found, but for illustrative purposes the transaction was not converted to show a type of HL7message that can be added to a block.

```
{'index': '0',
'hash': '03a92477604cc9a4cf8781f30288d02d10045dd0be0ec5501d282e4f4f424150',
'timestamp': '2018-08-20 13:27:45.453951',
'address': '1Pun3vFqHHxNP544qm2Wc75Vp9T7Rb7nZq',
'previous_hash': 'first',
'data': {
    'status': 'final',
    'category': {
        'text': 'MTidalVolInsp',
        'coding': [{
            'code': 'vital-signs',
            'system': 'http://hl7.org/fhir/observation-category',
            'display': 'Vital Signs'}]},
    'code': {
        'coding': [{
            'code': '29274-8',
            'system': 'http://loinc.org',
            'display': 'Vital signs measurements'}]},
    'extension': [{
        'valueDateTime': '2018-07-02T16:35:19.986-03:00'}],
    'resourceType': 'Observation',
    'valueQuantity': {
        'code': 'ml',
        'value': 24,
        'unit': 'ml',
        'system': 'http://unitsofmeasure.org'},
    'meta': {'versionId': 'iX5'},
    'bodySite': {'text': 'Lungs'},
    'device': {
        'id': '00450047-004F-0044-0045-496E7465726D656400',
        'display': '00450047-004F-0044-0045-496E7465726D656400'},
    'effectiveDateTime': '2018-07-02T16:35:19.986-03:00',
    'identifier': [{
        'system': 'http://hl7.org/fhir/sid/us-ssn',
        'value': 'urn:uuid:00450047-004F-0044-0045-496E7465726D656400'}]}
    )
}
```


Figure 17 – Block Structure generated by the proposed solution.

The block structure of the proposed solution is a variant of the traditional bitcoin block structure. The reason for the changes is the fact that many properties used for controlling block versions and that make mining difficult are not necessary, since no cryptocurrency is generated through block mining. The block version is used for controlling the version of the consensus rules used by the network nodes. Non-upgraded nodes cannot validate new blocks preventing forks from being created.
The first field in Figure 17 is the index of the block on the blockchain. Next is the generated hash of the block, which follows the SHA256(SHA256(Block properties)) used by the Bitcoin's implementation. The timestamp represents the moment the block was created, and it is used to guarantee that an older block cannot be inserted in front of a newer block, and also is used for maintaining the cryptography heritage from one block to the next. The address field holds the bitcoin address of the owner of the information stored in the "data" field; it should be noted that the bitcoin address standard is used because it is very safe and makes a collision almost impossible. There are 2^160 possible addresses. Bitcoin addresses are generated using the user's public key, thus making it possible to prove this transaction's ownership.

4.2
Agent Roles

4.2.1
Integrity Control Agent

The Integrity Control Agent (ICA) is responsible for maintaining and inserting new blocks into the private blockchain. When the ICA receives a new transaction from the Transaction Distributing Agent (TDA) the agent creates a new block using the information that comes with the it and sends it to the Data Audition Agent (DAA). The DAA returns either a valid block, which means that no privacy information is stored in the data field of the block, or receives an invalid block indicating that either a privacy

violation was identified or that the block data is inconsistent. Once the ICA receives a new valid block, it inserts the new block into the blockchain using hash information from the most recent block of the blockchain. The last action this agent performs is to broadcast the new block to the network peer agents so that the blockchain copies remains synchronized.

4.2.2
Data Audition Agent

The Data Audition agent is responsible for checking if personal information is present in the new block's data field. If personal data is present in the block, the agent automatically classifies it as invalid and the ICA is informed. Another function of this agent is to verify if all the block's fields are correct, such as, for example, that a timestamp should be in RFC3339 format. If any field is not correct, the block is invalidated. The agent also creates a Merkle tree root using the transaction information.

4.2.3
Transaction Distribution Agent

The Transaction Distributing Agent (TDA) is responsible for receiving health data from a user of the platform. Once it receives the healthcare data, the agent retrieves the user's blockchain address and public key from the private database. The user must have been previously registered on the platform for that purpose. Once the TDA has the following information, (i) User blockchain address, (ii) User public key, and (iii) User healthcare information in HL7 JSON format, it creates the new block and sends it through the SPADE framework to the ICA. TDA is also responsible for only accepting a new transaction once all the peers of the network have finished inserting the block into their copies of the blockchain. TDA receives confirmations from the PAs on the network and when all have finished, he gets the next transaction in the transaction pool.

4.2.4
Peer Agent

The Peer Agent (PA) role is to keep one copy of the blockchain synchronized with the rest of the network on each new block inserted. Once a new block is received, the agent adds it to the copy of the blockchain and notifies the TDA agent that the block has been inserted.

4.2.5
Worker Agent

The role of the Worker Agent (WA) is to retrieve information from the blockchain. The agent receives an array of block indexes from a patient and retrieves this information from the blockchain. To achieve this goal, it uses the algorithm mentioned in section A above. Once all information is retrieved, the agent returns it to the front- end using the RESTful architecture layer.

4.3
Preemptive Consensus

The agent in this solution performs what the author of this paper calls a "preemptive" consensus, meaning that agents agree with a block before the agent is inserted into the blockchain layer. There are three steps to the agent consensus of this solution. A new

transaction received by the TDA is valid, so the transaction is sent to the ICA. The ICA creates the new block and sends it to the DAA. The DAA verifies the block's information and, if valid, returns a valid block to the ICA. Once the ICA receives the new valid block, it inserts it into this agent's local private blockchain and propagates it to the peer agents in the network that also have a copy of the private blockchain. Each peer agent notifies the TDA once its new block is added to its current blockchain. Once all of the peer agent notifications are received by the TDA, the next transaction can be accepted into the private blockchain.

4.4 Why no Cryptocurrency

It is important to clarify why the prosed solution do not make usage of cryptocurrency. All management and control of the blockchain is performed by software agents without the need for human interaction. This makes having a cryptocurrency a needless overhead in complexity. There is no mining because agents perform the validation of data and the blockchain is private since sending personal records from patients to third parties in a public blockchain structures can be very risky to maintain privacy and anonymity of the patients. Another reason for no usage of cryptocurrency is because the purpose of the proposed solution is to store and retrieve patients HL7 FHIR records in a clean manner benefiting from the properties of a blockchain data structure without bring huge complexities from publics blockchain intended for financial transactions and also without requiring huge amounts of processing power to keep it running.

4.5 Performance

The performance of the proposed solution is satisfactory at the moment since it is in early staging of testing. There is still more work to be done on a production environment. We achieved a controlled insertion of HL7 data from a patient every 50 seconds in an environment with seven peer agents distributed in different machines. Thanks to HL7 HFIR the verification of an observation resource is achieved in a very efficient process by the DAA. The more time-consuming task is the propagation of the blocks to the PAs and their confirmation to the TDA.

Another point of relevance is that since software agent's consensus is pre-emptive and organized there is no forks happening in the blockchain. Making much more efficient the process to maintain the blockchain balanced.

4.6 Security

Security breach of patient's information is a very important issue since breach of privacy is not only illegal, but it also may represent danger to the patient. To avoid breach of patient privacy not only public key encryption is used on the stored transactions but the management of the patients private and public keys are managed in the RESTful layer to avoid human mistakes such as the patient losing his keys. Also, the private architecture uses OAuth authentication between the RESTful layer and the MAS blockchain Layer to ensure that only authorized users can send and receive transactions. There are no third parties involved in verifying the transactions, only software agents, making safer than having strangers checking the validity of such transactions.

4.7 Extending Compliance

As of now, the proposed architecture is compliance only with HL7 FHIR observation resource (since is one of the most relevant). But for extending the compliance for the other resources in HL7 FHIR what needs to be done is to add new behaviors to the Data Auditioning agent. One of the main advantages of the private blockchain choice is that updating the system is possible without losing any of its records because the transaction pool is maintained in the RESTful layer. This allows for updating the MAS Blockchain layer without risking loss of patient's records.

5
Formalization

This section presents the states that exist in order to validate and maintain the private blockchain with the use of agents. Statecharts is used to formalize the states of the system. Statechart extends conventional state- transition diagrams, allowing for compact and expressive diagrams capable of explaining complex behaviors in small diagrams. Statecharts is compositional and modular, which allows for organized diagrams with hierarchy, concurrency and economical description language. Statecharts makes it possible to view the description of the solution in different levels of detail, therefore making the understanding of a complex architecture more comprehensible. Another relevant characteristic of statecharts diagrams is the inter-components communications, which is used in this paper to explain the messaging that occurs between states and agents. Hence, describing all states of the solution becomes much simpler.

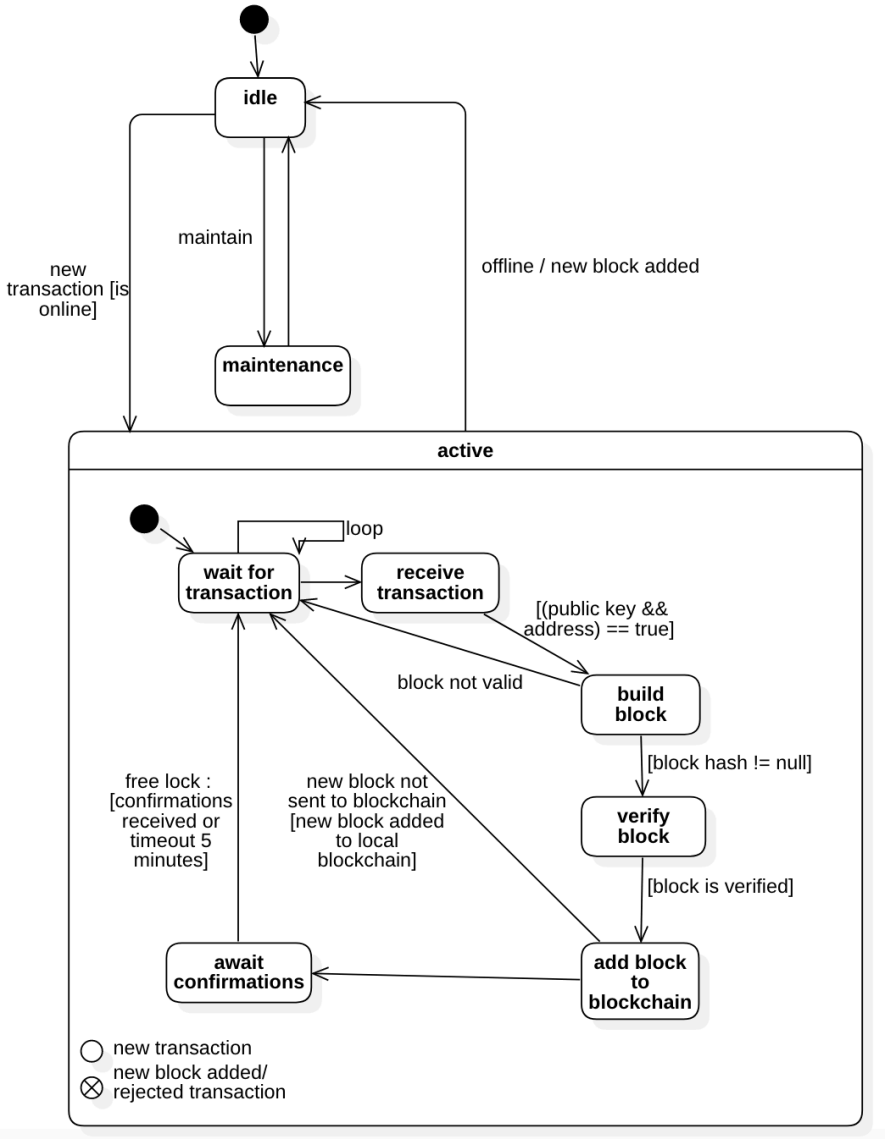
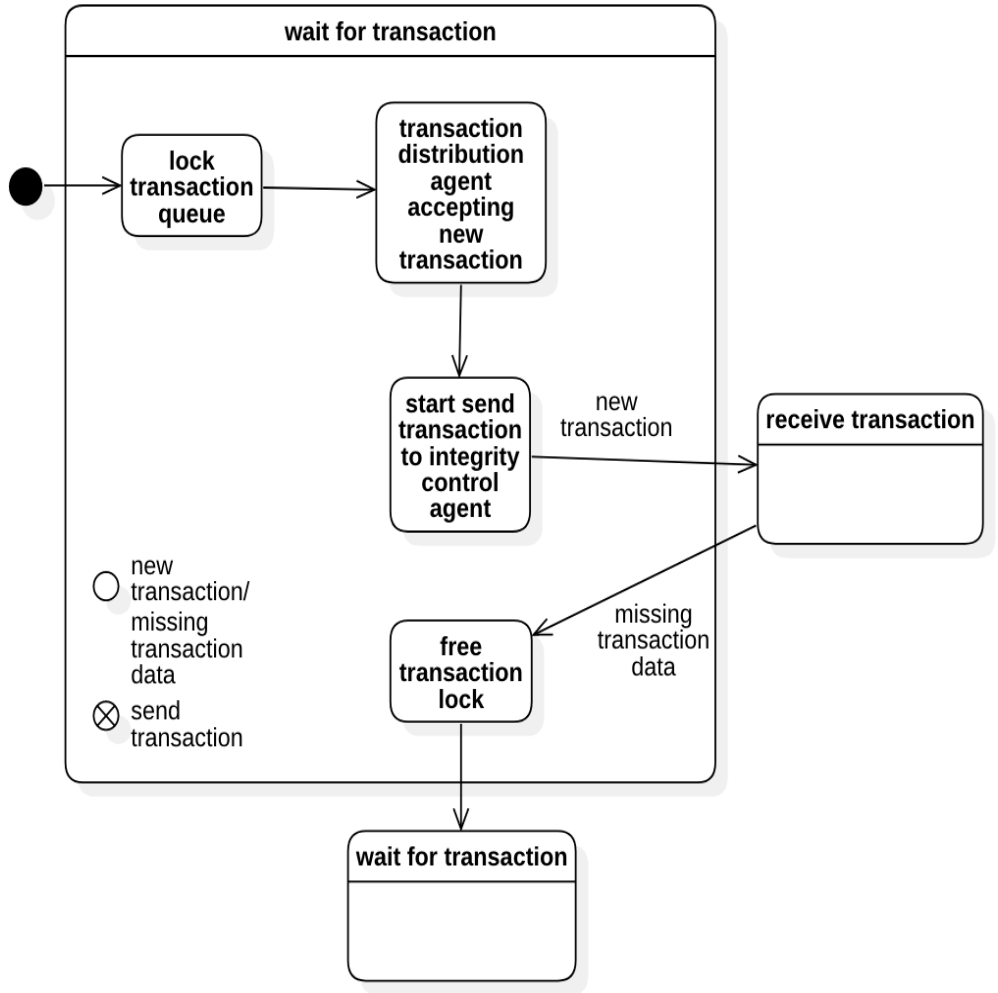


Figure 18 – Main program statechart.

The diagram above shows the states of an active instance of the private blockchain. The instance comes out of the idle state when the first transaction is received. The first state is the “wait for transaction” state. This state is executed in a loop until a new transaction is received and it is explained in 5.1. After the “receive transaction” state, the system enters the “receive state transaction” state. This state is responsible for verifying that the

transaction data is correct, see 5.2. The “build block” state is also presented in 5.3. This state is responsible for creating an unverified block with the transaction data. Once the block is built it is sent for verification. The “verify block” state is responsible for checking the type of transaction and to guarantee anonymization and privacy of the contained transaction information. Once the block is verified the next state is the “add block to blockchain” state. This state is responsible for inserting the new verified block into the blockchain and it is explained in 5.4. The “await confirmation” state is responsible for receiving new block insertion confirmation from at least two-thirds of the peer agents on the network. If that condition is met, the program goes back to the “wait for transaction” state. In 5.5 the “await confirmation“ state is presented, followed by the “synchronize peers” state explanation.

5.1
Wait for Transaction



FigFigure 19 - Wait for transaction statechart.

This state initially locks out any new transactions from entering the system until the current one is resolved. As it is possible to observe when a state is dictated by an agent, it is reflected in the state name. This is to make it clear that this state is dependent upon the agent’s behavior. The TDA accepts a new transaction and changes its state to the "receive transaction" state. If any transaction information is not correct, the transaction is discarded. Afterwards, the state frees the transaction lock and enters the "wait for transaction" state.

5.2
Receive Transaction and Build Block

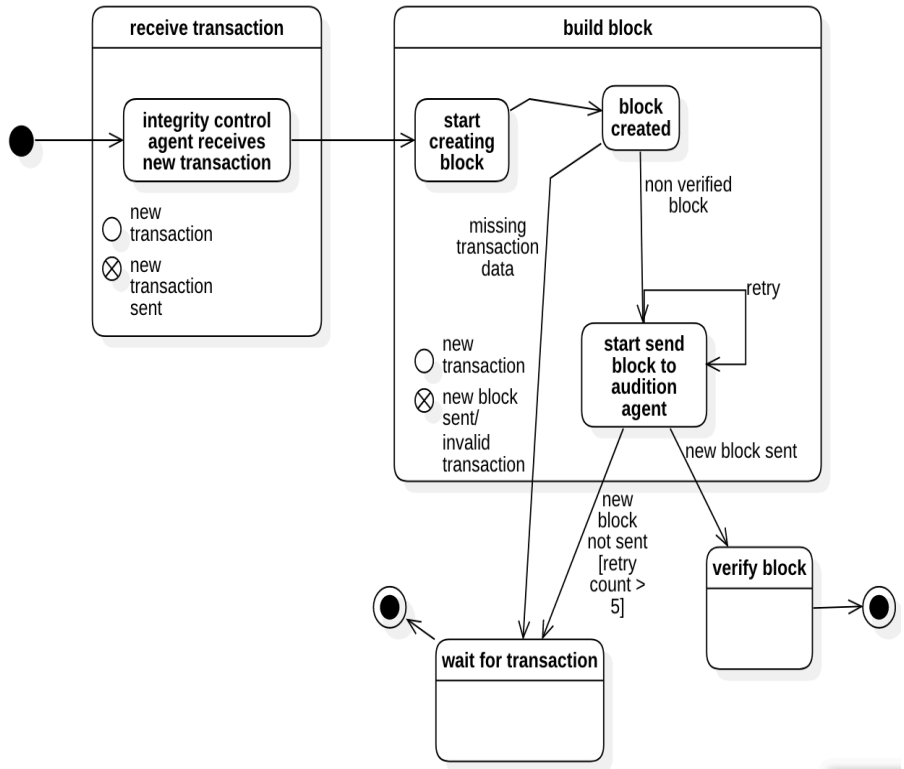


Figure 20 – Receive transaction and Build block statechart.

These states interact as follows: The “receive transaction” state has the ICA receiving new transaction data. Afterwards, it transitions to the “build block” state. In this state, the ICA creates the new unverified block and sends it to the DAA to verify this block’s information. If the new block is not sent to the DAA, the block is discarded, and the program enters the “wait for transaction” state.

5.3
Verify Block

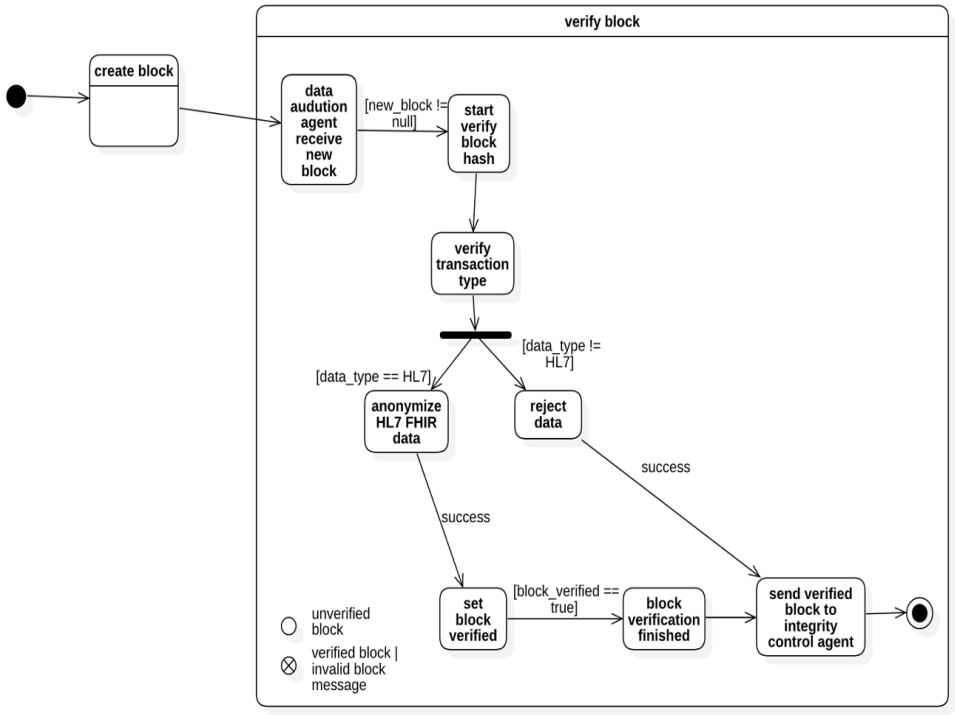


Figure 21 – Verify block statechart.

The state "verify block" is initiated by the DAA. If the new block is valid, the verification process is initiated. The first step is to verify the type of transaction; that is, if it is HL7 data or not. If it is HL7 data, the DAA uses the knowledge of the standard and anonymizes the HL7 information. This allows for safe storage in the blockchain. If it is not HL7 data, the DAA discards that transaction. Once all that it is concluded, the DAA sets the block as verified and broadcasts the verified block to the ICA.

5.4
Add Block to Blockchain

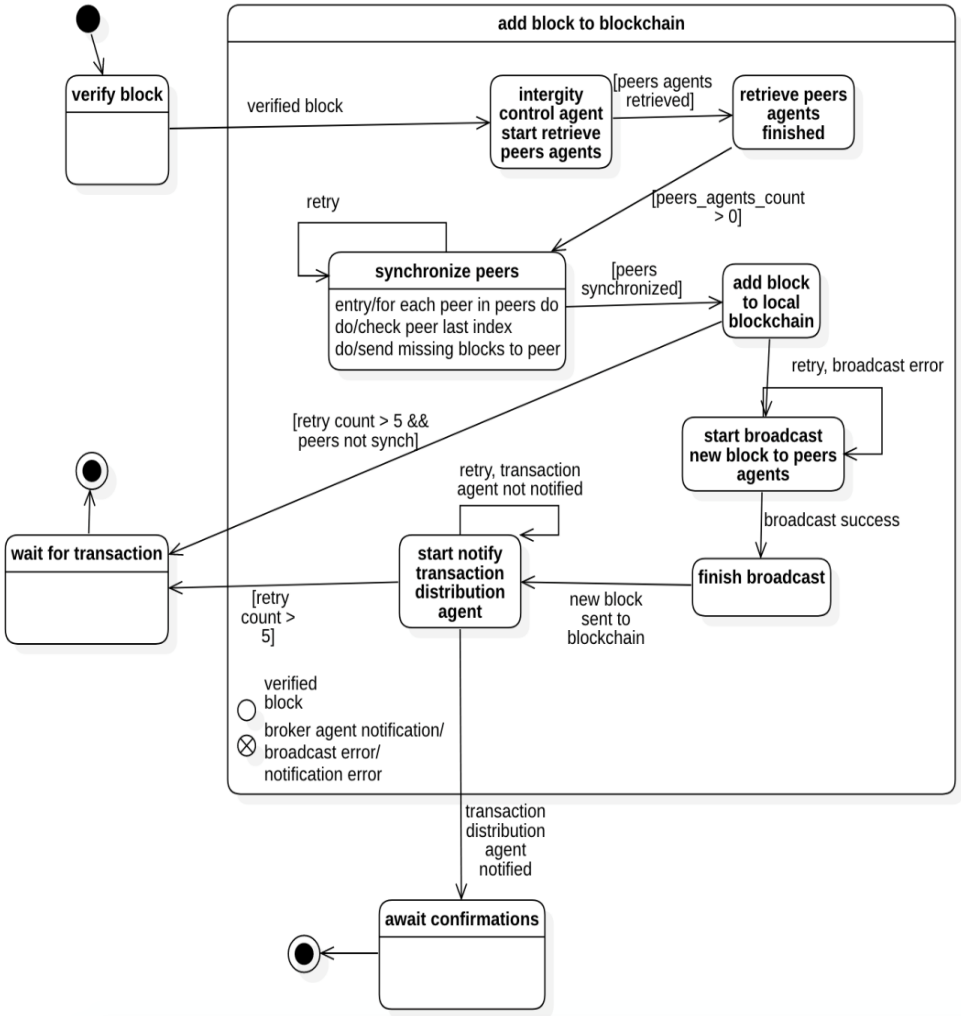


Figure 22 – Add block to blockchain statechart.

This state starts with the ICA retrieving all PAs on the network. If the peer count is greater than 0, the program enters the “synchronize peers” state, explained in 5.6. If the synchronization is successful, the ICA adds the block to the local blockchain and changes the state to start broadcasting the new block to the PAs on the network. Once the broadcast is concluded, the ICA notifies the TDA and the program enters the “await confirmations” state.

5.5
Await Confirmations

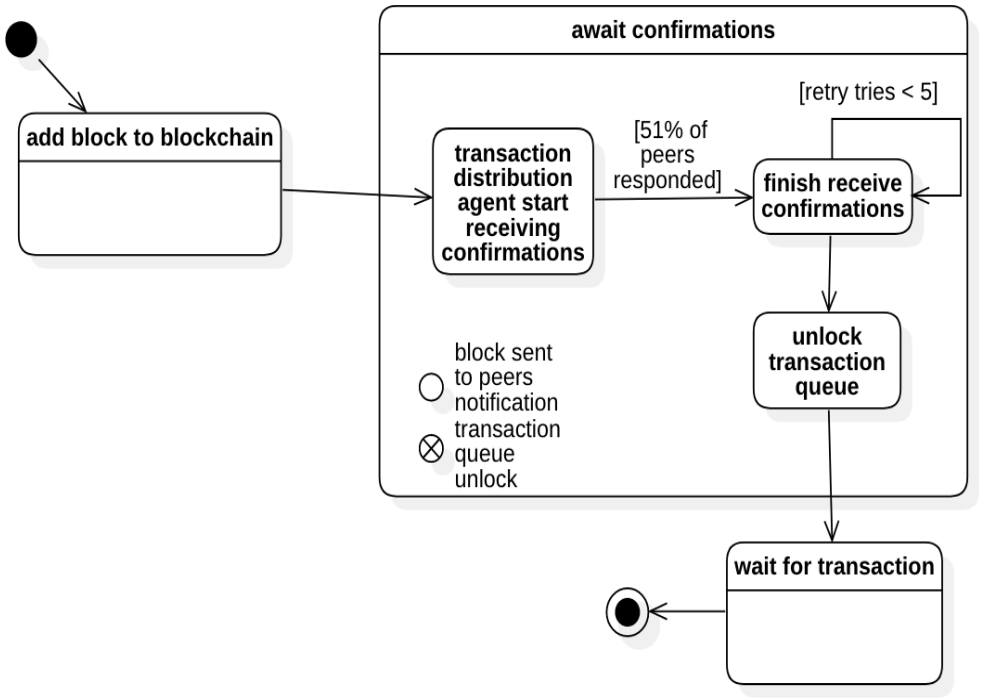


Figure 23 – Await confirmations statechart.

The TDA starts to receive confirmation from the peer agents present on the network. If all of the PAs confirm to the TDA, then it unlocks the transaction queue. The program then returns to the "wait for transaction" state.

5.6
Synchronize Peers

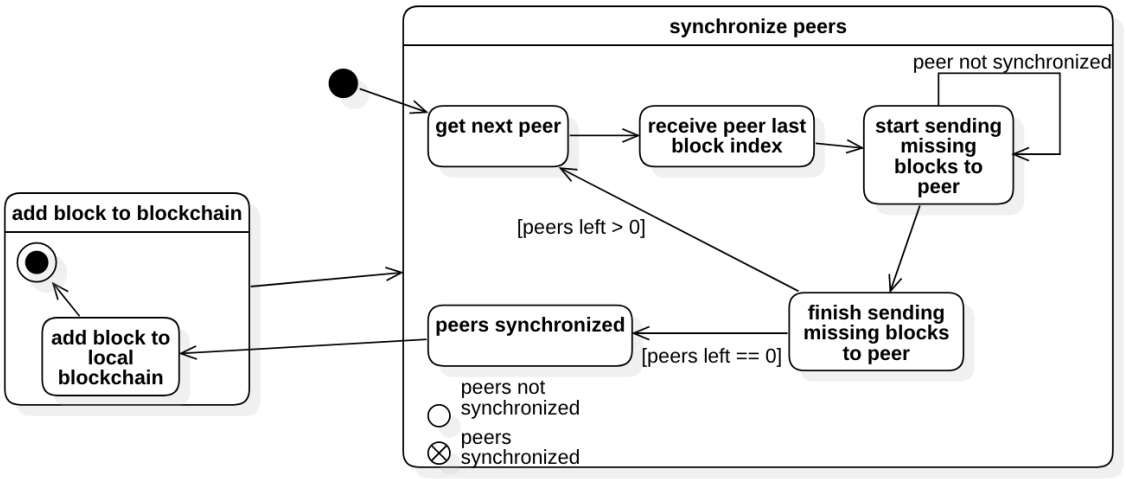


Figure 24 – Synchronize peers statechart.

This state is a sub-process of the ICA that runs for each PA on the network. This process checks the last index from every PA present on the network. If any of the PAs present is behind from the main chain the ICA sends all missing blocks to the corresponding PA. This step is repeated until all PAs are synched with the ICA. With this routine keeping the blockchain balanced is guaranteed. Once all the PAs are synched, the program resumes to the "add block to the blockchain" behavior.

6
Case Study

This section is going to present the case study that is was used to test the initial implementations of the proposed solution. In Section 6.1 the DOCPAD system is presented and its purposed explained. In Section 6.2 it is explained how the DOCPAD architecture works to retrieve patient's records from healthcare institutions. Section 6.3 presents how DOCPAD architecture was extended to be integrated with the blockchain of the proposed solution. In Section 6.4 it is shown how the DOCPAD system queries for information about a patient's record. Finally, in Section 6.5 the challenge of dealing with non HL7 FHIR information is discussed.

6.1
DOCPAD System

DOCPAD is a platform that allows for patients, doctors and clinics to manage healthcare records. DOCPAD provides services so that clinics can send patients information to their cloud and follows to distributes these exams trough mobile and web applications.

Patients can:

- Manage their exams and share with doctors.
- Upload exams that they already have by send pictures or files.
- Access clinic and doctor information.
- Create custom folders to organize their health records

Doctors can:

- Receive exams which they are referring physician of.
- Receive exams shared by patients.
- Manage their patient exams.
- Access patients and clinics information.

Clinics can:

- Manage their production status.
- Schedule appointments for patients.
- Visualize delivered exams.

DOCPAD system was used to make tests of the proposed solution initial implementation so that the system could also work with patients HL7 records. The next section will explain how patients HL7 records are received and integrated to the MAS blockchain and DOCPAD's system.

6.2
DOCPAD Architecture

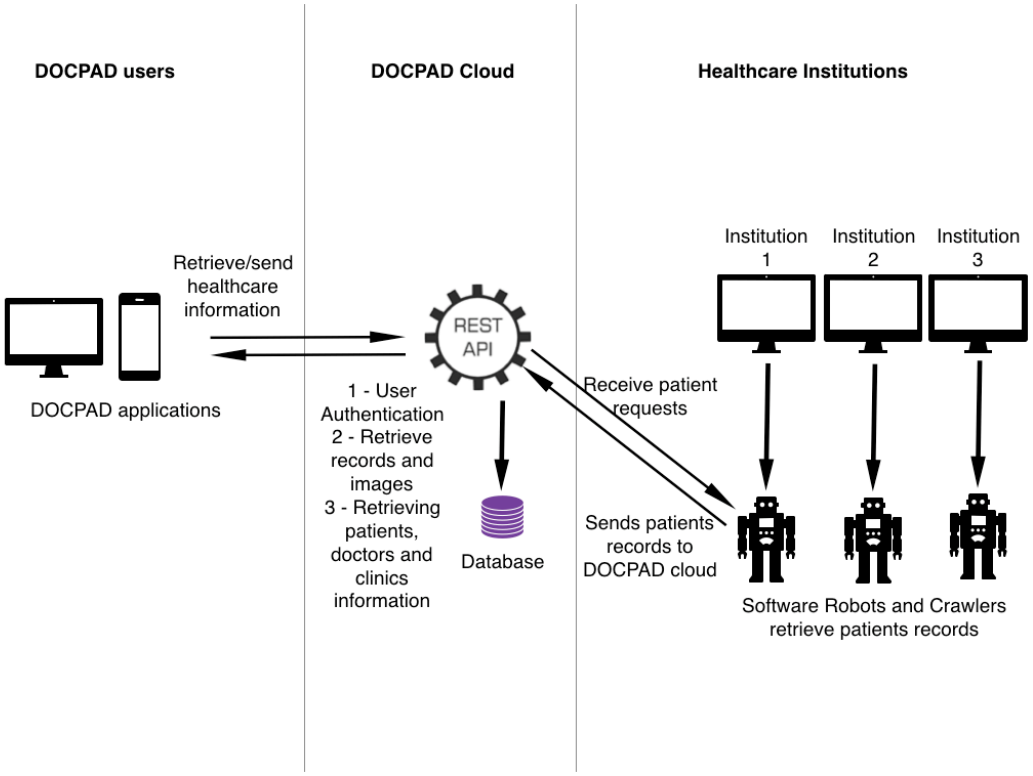


Figure 25 – DOCPAD Architecture

The Figure above summarizes DOCPAD's architecture. DOCPAD install software robots in healthcare institutions that are responsible to retrieve patients records and sends them to DOCPAD's cloud. The robots are also responsible to identify key information from the records such as patient social security number, date of birth and name so that the RESTful services present on the cloud can link the records with the correct patients. Once the link is complete patients can access their exams by mobile or web applications.

6.3
DOCPAD Architecture Integration with the Proposed Solution

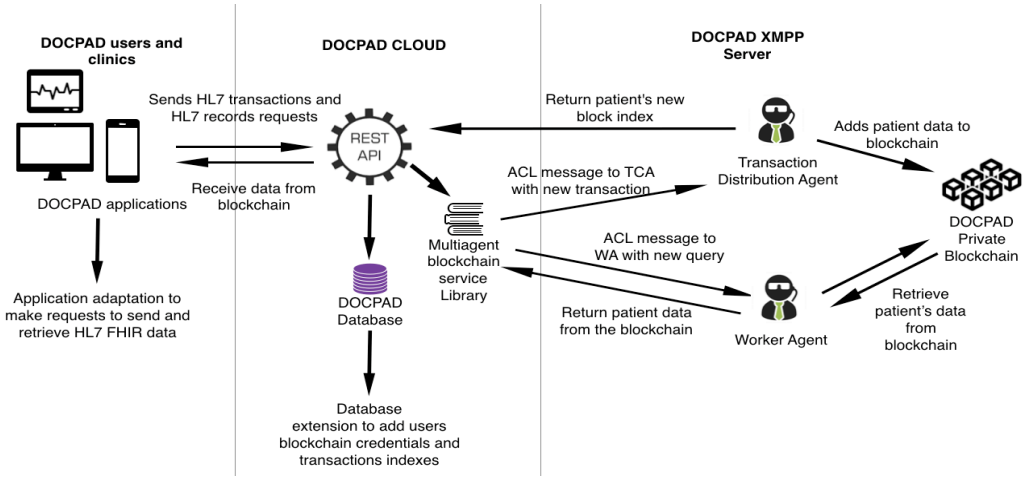


Figure 26 – DOCPAD integration with the proposed solution

In Figure 26 it is demonstrated how the integration of the proposed solution was done with DOCPAD system. It was necessary a few extensions in DOCPAD's architecture to be able to store patient's information in the private blockchain. First the RESTful services had to be extended to use a service library provided by the proposed solution. This library contains classes necessary to create users' public and private key as well as classes used to create the users address. Another function of this library is a set of classes used to transform HL7 FHIR data together with users' public key and address to generate a transaction that will be delivered to the TDA. Next the database had to be extended to store patient's public and private key as well his address used to identify him in the blockchain. Another database extension that had to be done is a set of tables that keeps patients' blocks indexes. Finally, the RESTful layer had to be extended to provides services capable of receiving HL7 FHIR transactions form monitoring devices. This is an ADHOC process and depends on how each device provides its communication capabilities such as an API or sockets communication for an example.

6.4
Querying HL7 FHIR Data Transaction

Each block from the blockchain is indexed and contains the patient address on its structure. Once a patient wants to receive information from the blockchain the RESTful layer retrieves from the DOCPAD's database his transaction index history and sends it to the WA. Once the WA receives the message with the list of transactions to be retrieved, he uses the algorithm mentioned before to find the users records on the blockchain and user a RESTful service to send the patient data to the front-end.

6.5
Non HL7 FHIR Data Challenge

The challenge of storing non HL7 FHIR healthcare data on a blockchain structure still open due to the non-guarantee of anonymization due to the lack of standardization of the different types of data. PDF, Word documents, DICOM images [63], JSON data and XML data have different layouts for the information in each healthcare institution. This is makes adapting the proposed solution for each integration very time consuming and costly to be implemented. Furthermore, since there is no standard such as HL7 FHIR imposed on these files, changes can happen at any time and break the premises of privacy and anonymization of the information sorted on the MAS blockchain layer. That's also the reason the first implementation only stores HL7 FHIR data on the blockchain.

7
Conclusion and Future Work

This section presents the final conclusions about the research and the proposed solution in section 7.1. In Section 7.2 its limitations are discussed. Finally, in Section 7.3 shows future work that need to be done so that the solution becomes more robust.

7.1
Main Contributions of the Solution

The main contributions of the prosed solution include promoting the usage of MAS on top of a blockchain structure, to show how agents can manage a blockchain structure to avoid the necessity of miners and a public ledger to ensure the lifespan of the blockchain community. A private blockchain environment is the suitable for this purpose since it allows control of access and the process executed by software agents in an orderly manner. Using software agents can make a private blockchain more simple, efficient and without the need for a cryptocurrency.

Another contribution how the HL7 FHIR and MAS can be used to ensure anonymity and privacy of patients records in an automated manner. The problem is that many systems to this date are not still HL7 compliance which makes difficult to ensure automated anonymization of healthcare data. When all systems are Hl7 compliance the usage of the proposed solution will be much more efficient interesting since any systems that want to store anonymized data from its patients will be able to do it without too much developing effort.

Demonstrating how software agents can have a pre-emptive consensus that is simple and efficient diminishing the amount of processing power needed in classic consensus algorithm such as POW. The main motivation for this contribution is to promote viable usage of blockchain structures without the need of huge amounts of processing power or the usage of third parties blockchains that keep information from multiple domains.

Finally, the last contribution is the presentation of an architecture that binds together Multiagent system with the HL7 FHIR standard and blockchain technology together to achieve a common objective.

7.2
Main Limitations of the Proposed Solution

So far, the main limitation of the proposed solution is the fact that it only works with HL7 FHIR observation resource messages. This can be enhanced by creating new behaviors for the DAA so that different resources from HL7 FHIR can be anonymized by the proposed solution as well. Another limitation is the amount of transactions per minute that the solution can handle at the moment due to the fact that observation messages from monitoring devices can have a ratio of 1 message per second or less, making an enormous amount of transactions that are not relevant to be stored. To surpass this limitation further studies of which message from monitoring devices should be stored on the blockchain. Only relevant changes on patient's conditions should be stored and not continuous observation data do not add value to the patient history.

7.3
Future Work

Future work to the proposed solution includes:

- Extending the DAA to become fully compliance with all HL7 FHIR resources.

- Creating a logic mechanism to only stored observation messages from devices that are relevant and represent changes in the patient's condition.
- Enhance the compressing of the data stored in the blockchain.
- Enhance the block querying algorithm to be more efficient.
- Accept more types of data such as DICOM and Word documents.
- Further integration with other system's such as DOCPAD

8
References

1 SWAN, MELANIE. **Blockchain: Blueprint for a new economy**. " O'Reilly Media, Inc.", 2015.
2 IANSITI, Marco; LAKHANI, Karim R. **The truth about blockchain**. Harvard Business Review, v. 95, n. 1, p. 118-127, 2017.
3 CROSBY, Michael et al. **Blockchain technology: Beyond bitcoin**. Applied Innovation, v. 2, n. 6-10, p. 71, 2016.
4 JONSSON, Jakob; KALISKI, Burt. P**ublic-key cryptography standards** (PKCS)# 1: RSA cryptography specifications version 2.1. 2003.
5 LIU, Debin; CAMP, L. Jean. **Proof of Work can Work**. In: WEIS. 2006.
6 NAKAMOTO, Satoshi et al. **Bitcoin: A peer-to-peer electronic cash system**. 2008.
7 BECKER, Georg. **Merkle signature schemes, merkle trees and their cryptanalysis**. Ruhr-University Bochum, Tech. Rep, 2008.
8 KUO, Tsung-Ting; KIM, Hyeon-Eui; OHNO-MACHADO, Lucila. **Blockchain distributed ledger technologies for biomedical and health care applications**. Journal of the American Medical Informatics Association, v. 24, n. 6, p. 1211-1220, 2017.
9 ANGRAAL, Suveen; KRUMHOLZ, Harlan M.; SCHULZ, Wade L. **Blockchain technology: applications in health care**. Circulation: Cardiovascular Quality and Outcomes, v. 10, n. 9, p. e003800, 2017.
10 PUTHAL, Deepak et al. **The blockchain as a decentralized security framework [future directions]**. IEEE Consumer Electronics Magazine, v. 7, n. 2, p. 18-21, 2018.
11 MERTZ, Leslie. (Block) chain reaction: **A blockchain revolution sweeps into health care, offering the possibility for a much-needed data solution**. IEEE pulse, v. 9, n. 3, p. 4-7, 2018.
12 ANNAS, George J. et al. **HIPAA regulations-a new era of medical-record privacy?**. New England Journal of Medicine, v. 348, n. 15, p. 1486-1490, 2003.
13 BENDER, Duane; SARTIPI, Kamran. HL7 FHIR: **An Agile and RESTful approach to healthcare information exchange**.
In: Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems. IEEE, 2013. p. 326-331.
14 **FHIR Workflow**. Access at:
<https://www.hl7.org/FHIR/workflow.html
>. Last access on 05 May 2019.
15 **FHIR Modules**. Access at:
<https://www.hl7.org/FHIR/modules.html>. Last access on 02 May 2019.
16 FRANKLIN, Stan; GRAESSER, Art. **Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents**. In: International Workshop on Agent Theories, Architectures, and Languages. Springer, Berlin, Heidelberg, 1996. p. 21-35.
17 NWANA, Hyacinth S. **Software agents: An overview**. The knowledge engineering review, v. 11, n. 3, p. 205-244, 1996.

18 BRESCIANI, Paolo et al. **Tropos: An agent-oriented software development methodology.** Autonomous Agents and Multi-Agent Systems, v. 8, n. 3, p. 203-236, 2004.
19 WOOD, Gavin et al. **Ethereum: A secure decentralized generalized transaction ledger**. Ethereum project yellow paper, v. 151, p. 1-32, 2014.
20 JENNINGS, Nick; WOOLDRIDGE, Michael. **Software agents**. IEE review, v. 42, n. 1, p. 17-20, 1996.
21 AMETLLER, Joan; ROBLES, Sergi; BORRELL, Joan. **Agent migration over FIPA ACL messages**. In: International Workshop on Mobile Agents for Telecommunication Applications. Springer, Berlin, Heidelberg, 2003. p. 210-219.
22 HUHNS, Michael N. **Agents as Web services**. IEEE Internet computing, v. 6, n. 4, p. 93-95, 2002.
23 GREGORI, Miguel Escrivá; CÁMARA, Javier Palanca; BADA, Gustavo Aranda. A **jabber-based multi-agent system platform**. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. ACM, 2006. p. 1282-1284.
24 SANNER, Michel F. et al. **Python: a programming language for software integration and development**. J Mol Graph Model, v. 17, n. 1, p. 57-61, 1999.
25 SAINT-ANDRE, Peter. **Extensible messaging and presence protocol (XMPP): Core**. 2011.
26 NAKAMOTO, Satoshi et al. **Bitcoin: A peer-to-peer electronic cash system**. 2008.
27 **Technology: Banks seek the key to blockchain**: Access at: <https://www.ft.com/content/eb1f8256-7b4b-11e5-a1fe-567b37f80b64#axzz41DtGYYEo>. Last access on 23 April 2019.
28 **Bitcoin transaction**. Access at: <https://en.bitcoin.it/wiki/Transaction >. Last Access on 20 April 2019.
29 JOHNSON, Don; MENEZES, Alfred; VANSTONE, Scott. **The elliptic curve digital signature algorithm (ECDSA)**. International journal of information security, v. 1, n. 1, p. 36-63, 2001.
30 JOHNSON, Don; MENEZES, Alfred; VANSTONE, Scott. **The elliptic curve digital signature algorithm (ECDSA)**. International journal of information security, v. 1, n. 1, p. 36-63, 2001.
31 MIRKOVIC, Jelena; REIHER, Peter. **A taxonomy of DDoS attack and DDoS defense mechanisms.** ACM SIGCOMM Computer Communication Review, v. 34, n. 2, p. 39-53, 2004.
32 **Bitcoin Nonce**. Access at: < https://en.bitcoin.it/wiki/Nonce > Last access on 20 April 2019.
33 **Bitcoin fundamentals**. Access at: <https://medium.com/loom-network/understanding-blockchain-fundamentals-part-2-proof-of-work-proof-of-stake-b6ae907c7edb>. Last access on 15 April 2019.
34 **Blockchain hashing**. Access at: <https://en.bitcoin.it/wiki/Block_hashing_algorithm>. Last access on 10 April 2019.

35 PILKINGTON, Marc. 11 **Blockchain technology: principles and applications**. Research handbook on digital transformations, v. 225, 2016.
36 Bitcoin target. Access at: <https://en.bitcoin.it/wiki/Target>. Last access on 10 April 2019.
37 GRAMOLI, Vincent. **From blockchain consensus back to byzantine consensus**. Future Generation Computer Systems, 2017.
38 **Blockchain overview**. Access at: <https://www.researchgate.net/publication/319984012_From_blockchain_consensus_back_to_Byzantine_consensus>. Last access on 05 April 2019.
39 **Orphan blocks**. Access at: <www.bitcoinwiki.com>. Last access on 05 April 2019.
40 **Bitcoin's protocol rules**. Access at: <https://en.bitcoin.it/wiki/Protocol_rules#Transactions>. Last access on 03 April 2019.
41 **Bitcoin block structure**. Access at: <https://en.bitcoin.it/wiki/Block>. Last access on 25 March 2019.
42 HILDENBRANDT, Everett et al. Kevm: **A complete semantics of the ethereum virtual machine**. 2017.
43 **Comparison between Bitcoin and Ethereum**. Access at: <https://medium.com/blockchainspace/3-comparison-of-bitcoin-ethereum-and-hyperledger-fabric-cd48810e590c>. Last access on 25 March 2019.
44 **State of DApps**. Access at: <https://www.stateofthedapps.com/>. Last access on 25 March 2019.
45 **The truth about blockchain**. Access at: <https://enterprisersproject.com/sites/default/files/the_truth_about_blockchain.pdf>. Last access on 25 March 2019.
46 HABER, Stuart A.; STORNETTA JR, Wakefield S. **Digital document time-stamping with catenate certificate**. U.S. Patent n. 5,136,646, 4 ago. 1992.
47 CROSBY, Michael et al. **Blockchain technology: Beyond bitcoin. Applied Innovation**, v. 2, n. 6-10, p. 71, 2016.
48 **FHIR Summary**. Access at: <https://www.hl7.org/fhir/summary.html>. Last access on 20 March 2019.
49 **FHIR Summary Image**. Access at: <https://www.hl7.org/fhir/summary.html>. Last Access on 20 March 2019.
50 **FHIR Modules**. Access at: <https://www.hl7.org/fhir/modules.html>. Last access on 20 March 2019.
51 CALVARESI, Davide et al. **Multi-agent systems and blockchain: Results from a systematic literature review**. In: International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer, Cham, 2018. p. 110-126.
52 ZHANG, Peng et al. **Blockchain technology use cases in healthcare**. In: Advances in Computers. Elsevier, 2018. p. 1-41.
53 EKBLAW, Ariel et al. **A Case Study for Blockchain in Healthcare:"MedRec" prototype for electronic health records and medical research data**. In: Proceedings of IEEE open & big data conference. 2016. p. 13.

54 NADKARNI, Prakash M.; MILLER, Randolph A. **Service-oriented architecture in medical software: promises and perils.** 2007.
55 ZHANG, Peng et al. **Applying software patterns to address interoperability in blockchain-based healthcare apps**. arXiv preprint arXiv:1706.03700, 2017.
56 UDDIN, Md Ashraf et al. **A Patient Agent to Manage Blockchains for Remote Patient Monitoring. Studies in health technology and informatics**, v. 254, p. 105-115, 2018.
57 KUO, Tsung-Ting; KIM, Hyeon-Eui; OHNO-MACHADO, Lucila. **Blockchain distributed ledger technologies for biomedical and health care applications**. Journal of the American Medical Informatics Association, v. 24, n. 6, p. 1211-1220, 2017.
58 GORANOVIĆ, Andrija et al. **Blockchain applications in microgrids an overview of current projects and concepts**. In: IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2017. p. 6153-6158.
59 MENGELKAMP, Esther et al. **A blockchain-based smart grid: towards sustainable local energy markets. Computer Science-Research and Development**, v. 33, n. 1-2, p. 207-214, 2018.
60 YUE, Xiao et al. **Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control**. Journal of medical systems, v. 40, n. 10, p. 218, 2016.
61 FELDMAN, Paul. **A practical scheme for non-interactive verifiable secret sharing**. In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987). IEEE, 1987. p. 427-438.
62 HAREL, David. **Statecharts: A visual formalism for complex systems. Science of computer programming**, v. 8, n. 3, p. 231-274, 1987.
63 MILDENBERGER, Peter; EICHELBERG, Marco; MARTIN, Eric. **Introduction to the DICOM standard. European radiology**, v. 12, n. 4, p. 920-927, 2002.