SANTA CATARINA STATE UNIVERSITY - UDESC
COLLEGE OF TECHNOLOGICAL SCIENCE - CCT
GRADUATE PROGRAM IN APPLIED COMPUTING - PPGCAP

JOÃO HENRIQUE FAES BATTISTI

RESISTANCE ANALYSIS OF VIRTUAL MACHINE BASED ON ETHEREUM CONSORTIUM BLOCKCHAIN NETWORKS AGAINST INTERNAL DOS ATTACKS

JOINVILLE

2021

JOÃO HENRIQUE FAES BATTISTI

RESISTANCE ANALYSIS OF VIRTUAL MACHINE BASED ON ETHEREUM CONSORTIUM BLOCKCHAIN NETWORKS AGAINST INTERNAL DOS ATTACKS

Master thesis presented to the Graduate Program in Applied Computing of the College of Technological Science from the Santa Catarina State University, as a partial requisite for receiving the Master's degree in Applied Computing.

Supervisor: Dr. Charles Christian Miers

JOINVILLE

2021

Battisti, João Henrique Faes

Resistance analysis of virtual machine based on Ethereum consortium blockchain networks against internal DoS attacks/ João Henrique Faes Battisti. – 2021 83 p.

Supervisor: Dr. Charles Christian Miers

Thesis (Master's degree) – Santa Catarina State University, College of Technological Science, Graduate Program in Applied Computing, Joinville, 2021.

1. Cloud Computing. 2. Blockchain. I. Miers, Charles Christian. II. Santa Catarina State University, College of Technological Science, Graduate Program in Applied Computing. III. Title.

João Henrique Faes Battisti

Resistance analysis of virtual machine based on Ethereum consortium blockchain networks against internal DoS attacks

Master thesis presented to the Graduate Program in Applied Computing of the College of Technological Science from the Santa Catarina State University, as a partial requisite for receiving the **Master's degree in Applied Computing**.

Joinville, 17^{th} December 2021

I dedicate this work to my family, friends, colleagues and teachers who accompanied me and gave me strength in this magnificent trajectory.

ACKNOWLEDGMENTS

Initially, I dedicate this work to my parents (Adriana and Darcio), to my brother Jonas Gabriel and to my grandparents, who are no longer here, for all the support, dedication and motivation they had with me for all these years, since graduation. until the master's degree. I am very grateful to them for everything they represent to me and how much they taught me to be strong and persistent during all these years.

I want to give a special thanks to Professor Charles Christian Miers, for all the teaching, friendship and patience he had with me throughout this process. Mainly for having believed in me at times, when maybe I didn't even believe in myself. Furthermore, I would like to thank the other professors at Graduate Program in Applied Computing (PPGCAP) and collaborators at Laboratory of Parallel and Distributed Processing (LabP2D), who directly or indirectly participated in this process of my evolution as an academic.

I thank Salaar and Siddhartha, who are my faithful companions at all times and who accompany me wherever I go. I would like to thank my friends and colleagues who throughout this journey helped me to obtain knowledge, the various exchanges of ideas we had, the coffees, mate and beers that were consumed. Finally, I thank Buk, Neruda, Vinicius and many other poets and philosophers for their tireless moments of reflection.

“Education is a progressive discovery of our own ignorance...”

Voltaire

ABSTRACT

Solutions based on blockchain technology are growing their support from researchers and developers of their systems. Institutions from several areas (e.g., Finance, automotive, aerospace, food, security, technology, etc.) apply blockchain technology in their systems. A usual way to include new blockchain nodes is through virtualization technology virtual machines (VMs). This work aims to analyze the security/performance of blockchain networks based on the Ethereum platform, using its private model, against internal Denial of Service (DoS) attacks, since the amount and speed of insertion of blocks and transactions may be directly linked to the *flavor* of the instance. Thus, three scenarios were defined to carry out the experiments in order to analyze the behavior of these VMs of blockchain nodes applying the Ethereum solution using the following consensus mechanisms: RAFT Istanbul Byzantine Fault Tolerance (iBFT), and Proof-of-Authority (PoA). Regarding the results, it showed DoS attacks are efficient in a private or consortium blockchain network, as they compromise the proper functioning of the application when the flavor of the VM of blockchain nodes does not take into account security aspects. However, it is essential to point out the results of each scenario with their consensus mechanisms were different, some of them being more promising for specific applications and number of transactions. Other important aspects to be validated from the experiments is related to the efficiency of the consensus algorithms, mainly compared to the mechanisms applied in public blockchains that need high computational power.

Keywords: cloud computing, security, blockchain.

RESUMO

As soluções que utilizam a tecnologia blockchain ganham cada vez mais suporte de pesquisadores e desenvolvedores de seus sistemas. Cada vez mais instituições, de diferentes áreas: Financeira, automotiva, aeroespacial, alimentos, segurança, tecnologia, etc, aplicam a tecnologia blockchain em seus sistemas, e uma maneira usual de incluir novos nós de blockchain é através da utilização da tecnologia de virtualização de VMs. Este trabalho tem como objetivo realizar uma análise de segurança e desempenho de redes blockchain baseada na plataforma Ethereum, com seu modelo privado, perante à ataques DoS internos, uma vez que a quantidade e velocidade da inserção de blocos e transações estão diretamente ligadas as características de *flavor* da instância. Deste modo, é desenvolvido três cenários para realização dos experimentos afim de analisar o comportamento destas VMs de nós de blockchain aplicando a solução Ethereum com os mecanismos de consenso RAFT, iBFT e PoA perante a exploração do ataque. Em relação aos resultados, foi revelado ataques DoS são eficientes em uma rede blockchain privada ou consorciada, pois comprometem o bom funcionamento da aplicação. Contudo, é importante pontuar que os resultados de cada cenário com seus mecanismos de consenso foram diferentes, alguns destes sendo mais promissores para determinadas aplicações e quantidade de transações. Outros pontos importantes à serem validados a partir dos experimentos é em relação à eficiência dos algoritmos de consenso, principalmente comparados aos mecanismos aplicados em blockchains públicos que necessitam de um alto poder computacional.

Palavras-chaves: Computação em nuvem, Segurança, Blockchain.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
BFT	Byzantine Fault Tolerance
CFT	Crash Fault Tolerance
CPU	Central Processing Unit
CSA	Cloud Security Alliance
DAO	Data Access Object
DoS	Denial of Service
DDoS	Distributed Denial-of-Service
DLT	Distributed Ledger
ENISA	European Union Agency for CyberSecutiry
FAPESC	Foundation for Support to Research and Innovation of the State of Santa Catarina
IaaS	Infrastructure-as-a-Service
iBFT	Istanbul Byzantine Fault Tolerance
IoT	Internet of Things
IDC	International Data Corporation
LabP2D	Laboratory of Parallel and Distributed Processing
MIT	Massachusetts Institute of Technology
VM	virtual machine
NIST	National Institute of Standards and Technology
P2P	Peer-to-Peer
PaaS	Platform-as-a-Service
pBFT	Practical Byzantine Fault Tolerance
PoA	Proof-of-Authority
PoS	Proof-of-Stake
PoW	Proof-of-Work
PPGCAP	Graduate Program in Applied Computing
SaaS	Software-as-a-Service
SO	operating system
TA	Timestamp Authority
TTP	Trusted Third Party
UDESC	Santa Catarina State University

CONTENTS

1 INTRODUCTION

The use of public or private computing clouds has become predominant for hosting systems and services (COSTELLO; RIMOL, 2020). Organizations, universities, companies and users use cloud computing in an intrinsic way that has made the technology pervasive and ubiquitous (JANSEN; GRANCE, 2011). Among the service models offered by computing clouds, the Infrastructure-as-a-Service (IaaS) model has stood out as a model that offers fundamental computing, storage and networking resources on demand. The consumer does not manage or control the infrastructure, but has control over these fundamental resources through virtualization technology, especially virtual machines (VMs).

Applications based on VMs are quite diverse, making it possible to host traditional applications, such as a *web* server, and complex distributed applications, such as Peer-to-Peers (P2Ps). In this context, blockchain technology is formed by P2P networks, cryptography, algorithms and a consensus mechanism (LIN; LIAO, 2017). Decentralization requires functioning guarantees for the blockchain network to operate correctly, making the consensus mechanisms one of the critical areas of the blockchain solution. The consensus mechanisms determine the organization of the parts involved, the way they interact and what roles each involved part must agree. In this way, the consensus mechanisms are designed for different contexts and each application needs to identify the most efficient consensus mechanism to meet its requirements, in order to use resources effectively. Thus, it is natural that there are several different consensus mechanisms and that each one has its particularities. The needs of a blockchain are usually identified according to the number of parts involved, types and amounts of transactions, trust in the parts involved, guarantees to be provided, etc. Therefore, it is clear that blockchains used for a general public, with significant amounts of users and transactions, have consensus mechanisms that are very different from blockchains that are employed internally by a company, or even a restricted set. Furthermore, the way in which blockchains are planned and deployed can differ considerably. The focus of the present research is on one or more companies that employ blockchains in a reserved way, for their own use, in what is called a private or consortium blockchain model. Among the consensus mechanisms, for private and consortium blockchain models, RAFT, Istanbul Byzantine Fault Tolerance (iBFT) and Proof-of-Authority (PoA) stand out.

The consensus algorithm RAFT, is an algorithm Crash Fault Tolerance (CFT) which are algorithms resistant to loss of components but not tolerant to Byzantine faults. This algorithm is equivalent to Paxos (LAMPORT, 1998) in fault tolerance and performance. However, unlike Paxos it is decomposed into relatively independent subproblems and cleanly handles all the main pieces. Regarding how the algorithm works, some characteristics are relevant (ONGARO; OUSTERHOUT, 2014a): (i) Strong Leadership: The elected leader "centralises" all transaction *log* and sends them to the other peers; (ii) Election of leader: RAFT uses random timers for the election of the new leader, not allowing them to be repeated or to remain for a long time, preventing users from taking advantage of this process; and (iii) Changing validators: Allows new validators to be accepted or exited in a simple and uncomplicated manner, allowing the application to continue operating normally during these configuration changes. As for possible applications with RAFT, this is a consensus algorithm that is being disseminated, it is applied at scale as a solution to problems in distributed systems, in NoSQL systems, in communication transmission systems, implementation in several languages, etc.

As for the consensus algorithm iBFT, it is a consensus algorithm Byzantine Fault Tolerance (BFT) resistant to the loss of components and to malicious users, and it has similarities with Practical Byzantine Fault Tolerance (pBFT) and PoA , the latter mainly in relation to the system for electing validators. iBFT uses a *pool* of previously known validating nodes operating on a blockchain network to determine if a proposed block is added to the chain. In this consensus mechanism, there is no leader, all nodes can receive client *logs* and perform *broadcast* among themselves to replicate the entire machine state. However, only one node is selected for insertion of the new block, being changed in *voting times*. The iBFT consensus mechanism is a relatively new mechanism, but one that has expressive expectations by the *Enterprise Ethereum Alliance*, of penetration in corporate environments, being used by Ethereum and Hyperledger Besu.

The PoA engine is a consensus algorithm based on BFT, resistant to malicious users and the loss of network components. This consensus algorithm does not need high computational power or cryptocurrency *stakes*, but the reputation of the network validator. In general, in this consensus algorithm, the participating nodes are arbitrarily pre-selected and have all their real identification exposed, which is why they put their identity reputation at stake. In the process of validating a block, a few steps are necessary: confirmation of the real identity, tests with certain levels of difficulty to ensure commitment, and a standardized system for validators. Regarding applications, there are several platforms that use this consensus algorithm, such as Ethereum, Hyperledger, Polkadot, etc.

The use of blockchains in applications involves the configuration and deployment of nodes that allow the inclusion of new entries in the blockchain, as well as providing the process of convergence for a consensus between the parts involved. Institutions, which adopt the private or consortium blockchain models, have been employing the creation of their nodes from VMs in their private/hybrid computing clouds. Typically, developers of blockchain solutions already make available on their *websites* images of VMs optimized for ease of use. However, nothing prevents an organization from creating its own VM or container with blockchain nodes to be used by their applications. The present work focuses on the use of VMs as they are still widely used, leaving experiments with containers for future work. In this context, the security of these cloud environments is one of the most complex challenges for distributed environments, especially for services hosted in IaaS environments, in which internal users of the organization have access to participating nodes, or networks that VMs are part of. In this scenario, from blockchain nodes in VMs susceptible to unauthorized access, it is observed that an internal user (malicious or not) can generate an attack against the institution intentionally or accidentally. Regardless of the motivational nature of the attack, an attack surface for various security incidents emerges. This work addresses the possibility of intentional/accidental internal attacks by several requests that can subvert the system and generate a blockchain Denial of Service (DoS) situation. In this context, questions arise related to the environment in which the blockchain is inserted and also about its resistance, as well as the stability of the blockchain services and their transactions during the occurrence of a DoS attack.

The use of VMs implies the definition of a *flavor*[1] on which each blockchain node will run. An VM intended for a blockchain application is typically not resource intensive, so configurations with few vCPUs and memory are typically recommended (ie, 2 vCPUS, 4Gb RAM, 4Gb storage and 1Gb/s NIC) (Ethereum Foundation, 2021; Linux Foundation, 2018). A lean *flavor* configuration avoids wasting resources and even the transaction demand of most applications. On the other hand, a lean configuration can become a problem if an unusually high number of transactions occurs and memory, processor, or network resources reach 100% or thresholds that hamper consensus engine operations.

Regarding the development of private and consortium model applications, many platforms are using it (e.g., Ethereum, Multichain, Hyperledger, Polkadot, R3 Corda, EOSIO). About the applications, these are applied in these exemplified platforms and in many others, being applied in different sectors, e.g., Financial (REINMUELLER, 2018; SINGAPORE. . . , 2020); health (Blockpharma, 2021); government (E-Estonia, 2021); and supply chains (IBM, 2020; VECHAIN, 2021a).

Regarding the scope presented in this work, in relation to that presented in the qualification, there were some changes during the process, mainly in relation to the choice of the blockchain platform and the consensus mechanisms. Previously, the

[1] A *flavor* typically determines the amount of resources that each VM will have available, eg, amount of vCPUs, memory , storage, network throughput, etc.

Hyperledger Fabric platform had been chosen and the scenario with the RAFT consensus mechanism realized. However, during the implementation of other consensus mechanisms, this tends to be quite complex, even with attempts to use Application Programming Interface (API) to facilitate the implementation, not showing satisfaction and doubts in its application. Hyperledger is a modularized platform, which allows the application of other consensus mechanisms, however, for each different mechanism applied, a new *branch* was created in which the tool is applied to different solutions, changing the *core* of the same, directly influencing the search. With these validations, the Hyperledger Fabric was changed to Ethereum, with the RAFT, iBFT and PoA consensus mechanisms.

With the presentation of this problem using virtualization technology, the present work aims to perform a security and performance analysis of blockchain networks based on the Ethereum platform, with its private model, during the execution of DoS internal attacks. To achieve this objective, specific objectives are defined:

- Analyze the number of transactions per minute that an instance can perform without harming the service;
- Analyze the integrity and immutability of transactions; and
- Establish the relationship between the instance's *flavor* and the intensity of DoS attacks on the monitored computing resources.

The work is organized as follows. The Chapter 2 that seeks to address the fundamental concepts of cloud computing, virtualization and blockchain technology. In Chapter 3 the functional and non-functional requirements, related works, the proposal with the general and specific objectives and the presentation of the test environment are defined. Finally, Chapter 4 presents the results in a descriptive way and also analyzes them.

2 FUNDAMENTAL CONCEPTS

Applications with VMs are quite diversified, allowing from hosting traditional applications to distributed and more complex ones. The services provided by the blockchain are made up of P2P networks, cryptography, algorithms and a consensus mechanism. However, decentralization requires functional guarantees for these blockchain networks to operate correctly, which makes consensus mechanisms one of the critical areas of a blockchain solution.

Companies and institutions have adopted as a practice for blockchain solutions using private and consortium models the creation of their blockchain nodes using VMs within public or private clouds. However, the security of blockchain services hosted on a computational cloud IaaS deserves attention regarding the amount of transactions that the instances can perform and also due to issues of access to blockchain nodes by the employees of this organization. In this context, attacks like DoS can be caused accidentally or maliciously.

From the questions presented, there is a need to understand about the issues of blockchain networks applied to VMs. This chapter covers the fundamentals of computing clouds mainly related to virtualization and also blockchain, presenting its main vulnerabilities and models.

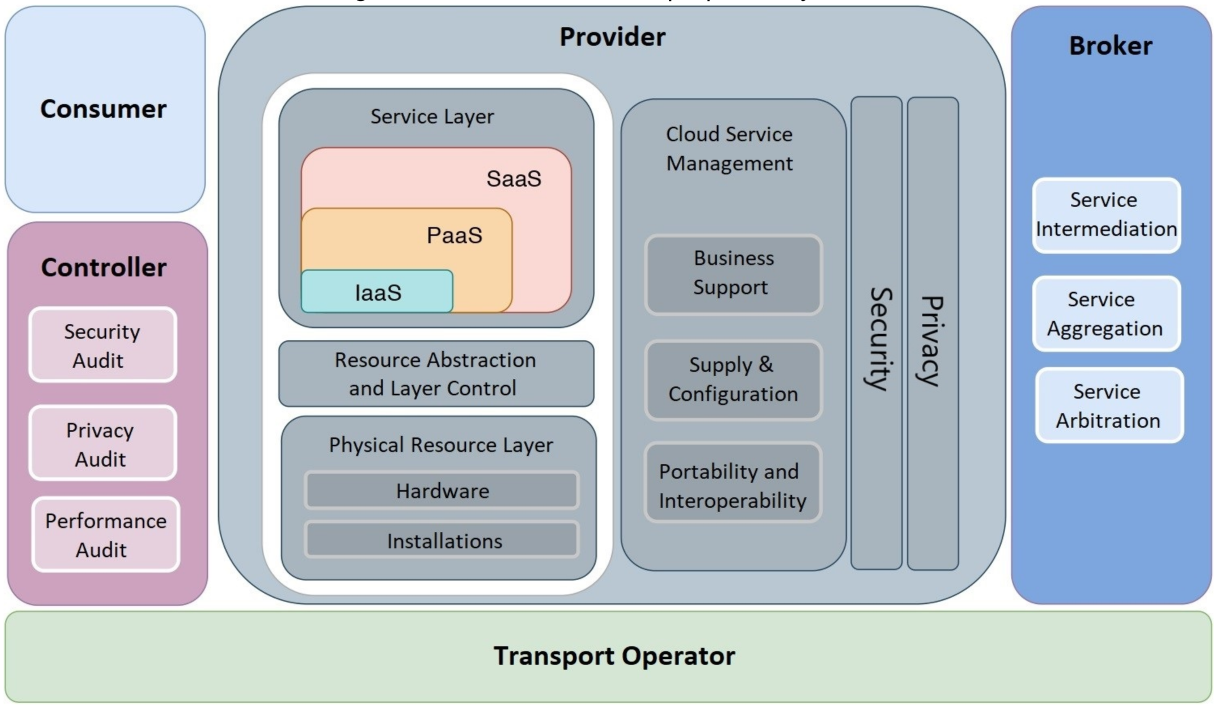
2.1 CLOUD COMPUTING

Adoption of cloud computing has become prevalent in the world market (COSTELLO; RIMOL, 2020). Companies, institutions and users have been using cloud computing in a way that the technology has become pervasive and ubiquitous (JANSEN; GRANCE, 2011; MELL; GRANCE, 2011).

According to National Institute of Standards and Technology (NIST) cloud computing is a model for enabling ubiquitous, convenient and on-demand access, which by network allows access to a set of computing resources (*eg*, networks, servers, storage , applications and services) delivered quickly and released with minimal management effort or service provider interaction (MELL; GRANCE, 2011; JANSEN; GRANCE, 2011). NIST defines three service models that describe the different categories of cloud service: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and IaaS. SaaS is a service model that is basically a complete application managed and hosted by the server. The PaaS model provides the consumer with a platform for installing applications developed or created that require a cloud infrastructure. The IaaS model allows the consumer to have access to lower-level resources, these are fundamental computing resources, such as storage, communication and processing. In IaaS the consumer does not manage or control the cloud infrastructure, but has controls over operating systems, storage and applications deployed through VM.

Computing clouds are also classified by their deployment model, with the public and private models being the most common. Private clouds operate under their own infrastructure, where their resources are provisioned for the use of a single organization made up of multiple users with different permissions. Unlike private clouds, public clouds have their infrastructure provisioned for open use, which is managed by an entity (*e.g.*, company, government organization, academic organization). Figure 1 illustrates the reference model for cloud computing, according to NIST.

Figure 1 – Reference model proposed by NIST.



Source: (LIU et al., 2011)

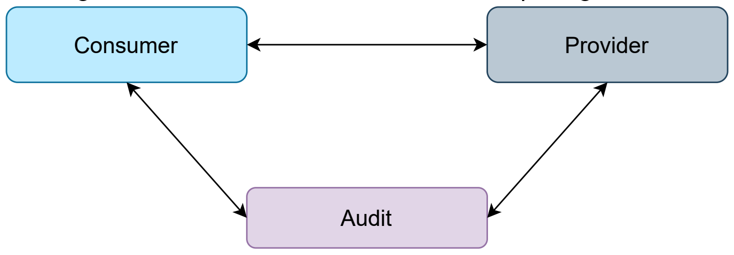
The main actors involved in the cloud computing model (Figure 1) are (LIU et al., 2011):

- **Consumer**: A person or organization that makes use of the cloud provider's services;
- **Supplier/Provider**: Entity responsible for making the cloud computing service available to interested parties;
- **Auditor**: An entity responsible for conducting independent assessments of the cloud services, information system operations, performance and security of the cloud implementation.

- **Broker**: Entity that controls the use, performance and distribution of cloud services; and
- **Transport Operator**: acts as an intermediary between the provider and consumer, in order to enable connectivity and transport of services between them.

As shown in Figure 1 the NIST cloud computing reference architecture defines five main actors, where each of these actors represents an entity that participates in a process, transaction and/or performs a task in the cloud computing. Based on the analysis of Figure 1, this work focuses on the issues of performance analysis and security of a VM, linked to auditing. In Figure 2 it is possible to observe the relationship of the actors related to this work.

Figure 2 – Main actors of the cloud computing model.



Source: Author.

Figure 2 illustrates the relationship between the auditor, consumer and provider. The auditor is responsible for carrying out assessments of the operations and security of the cloud computing service, these examinations may be performed independently of the cloud service controls. The purpose of this actor is to ensure that the content has not been modified and that legal and data archiving requirements have been maintained.

The issue of computing clouds security is one of the most complex challenges for distributed environments. Cloud Security Alliance (CSA) and European Union Agency for CyberSecutiry (ENISA) list the risks, recommendations and benefits for cloud computing (MOGULL et al., 2017; SOMOROVSKY et al., 2011). Highlighting key concerns such as: software, infrastructure, storage and network security.

Among the main concerns of the agencies, the one that stands out most in terms of concern is network security, in Figure 1 in light green, as it deals with the communication issues of computing clouds, in which there are concerns about internal and external attacks, whether on the physical or virtual network (WU et al., 2010). One of the best known attacks that seek to exploit network vulnerabilities is DoS, whose main objective is to make the resources of a system/application unavailable to its users.

As a premise of this model of cloud computing, several VM can be created through virtualization. The application of VM is quite diverse, they can host traditional applications such as web servers and also more complex distributed mechanisms such as services based on P2P. All these possibilities are subject to these vulnerabilities, mainly related to internal issues. Therefore, regardless of the model, scenario and solution used, the audit of these services becomes essential to guarantee the quality of their offer and cloud management.

2.2 RESOURCE VIRTUALIZATION IN COMPUTING CLOUDS

Virtualization is one of the main technologies used in cloud computing. This technology developed to help IT organizations optimize the performance of their applications in a simple and cost-effective way.

The growth in the use of virtualization revolves around VMs, mainly because it is a technology that is present in the most modern cloud computing infrastructures (*eg,* Amazon Elastic Compute Cloud – EC2, Google App Engine and OpenStack). The use of virtualization in computing clouds has gained acceptance for providing isolation, performance and portability. It also presents its share of challenges that cause some difficulties in ensuring some security aspects.

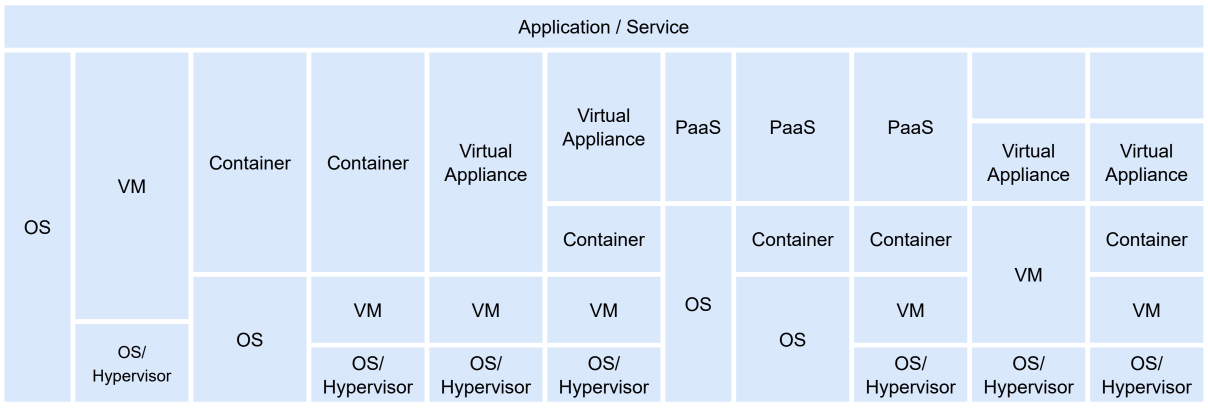
The application of virtualization in cloud computing has different types of approaches, such as:

- Operating system-based virtualization: This is enabled by an operating system *host* that supports various operating systems (SOs) *guest* isolated and virtualized on a single physical server with characteristics that are all in the same *kernel* with exclusive control over the infrastructure (SABAHI, 2012).
- Application-based virtualization: It is a virtualization that is hosted on top of the hosting operating system. This virtualization application emulates each VM that contains its own guest OS and related applications (SABAHI, 2012).
- Hypervisor-based virtualization: The hypervisor is available at machine boot time. Physical resources are managed by the hypervisor, virtualized and made available to VMs. In this model you can have two categories of hypervisor: *Bare-metal* (Type 1), which offers kernel-level isolation and *hosted* (Type 2), which provides isolation between processes (BUI, 2015; KULKARNI, 2016). In the case of Type-1 it guarantees a higher level of isolation and security and that Type-2 is totally dependent on the host OS.
- Container-based virtualization: This virtualization focuses on isolation between processes, for that, it relies on SO core resources and does not seek to abstract the host hardware. A container enables virtualization by implementing task management, scheduling and resource multiplexing (PANIZZON et al., 2019).

Comparing this approach, one can see that hypervisor-based virtualization has security benefits. What makes it attractive, because exploiting vulnerabilities of host-s/guests is an arduous and complex task, for the reason that providers of IaaS and PaaS have developed virtual resource managers for this virtualization technique, with that provide a high level of isolation (MIERS et al., 2014).

The approaches presented are technologies that complement each other, thus favoring the implementation of virtualization with different needs. Figure 3 presents different approaches to providing computing resources to an application, from without virtualization to with several layers of virtualization.

Figure 3 – Combinations to provide computational resources to an application.



Source: (PANIZZON et al., 2019)

From Figure 3 there are several combinations of VMs, containers and applications, in which the choice of the ideal combination depends on the purpose of the proposed solution. The objectives can vary according to the need, *e.g.*, applications with minimal structures is recommended the traditional approach of SO - containers, for performance reasons. In this work, the combination Hypervisor - VM - Container was used, according to the objective of the work.

Table 1 presents typical instance *flavors* configurations for different use cases. These instance *flavors* consist of various combinations of Central Processing Unit (CPU), memory, storage and network capacity, which aim to offer flexibility in choosing the right composition for your application.

Table 1 – Combinations between flavors configurations

Model	vCPUs	Memory (GB)	Network bandwidth (Gbps)
Medium	1	2	until 10
Large	2	4	until 10
x.Large	4	8	until 10
2x.Large	8	16	until 10
4x.Large	16	32	until 10

Source: (AMAZON, 2020)

The combination presented in Table 1 is applied for general use of virtualization, but there are other types of instances focused on certain computing tasks, such as optimizations for processing, memory and storage. Each of these instance types are optimized to meet different use cases and deliver the best possible user experience.

There are several use cases for virtualization applications, such as servers, distributed analytics, databases, web applications, microservices, containers, among others. One of these applications that has seen growth in the adoption of cloud applications is blockchain technology.

2.3 BLOCKCHAIN

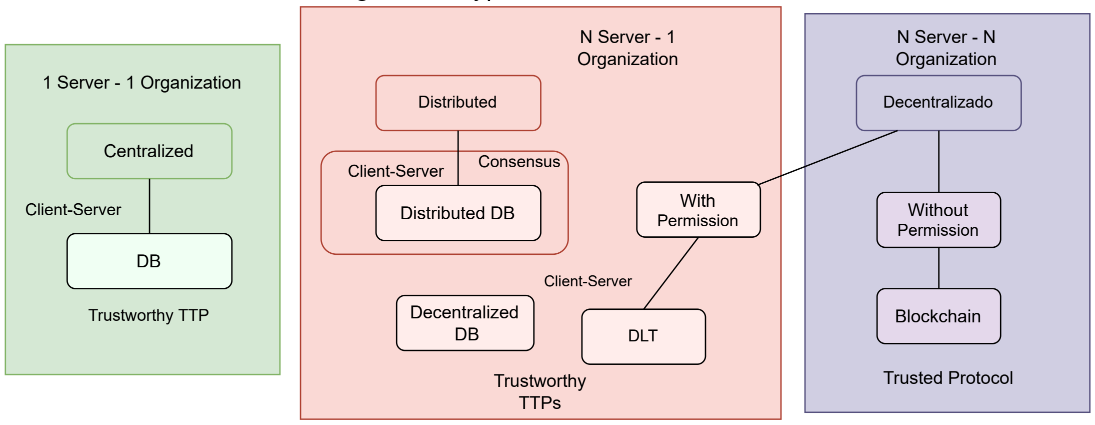
Blockchain technology had its first case of widespread success when applied to the financial area and, in general, associated with this domain, different areas of application and use of the technology have emerged with the growth of attention and popularity around blockchain (RODRIGUES et al., 2019). In fact, the technologies that enable blockchain implementation have been the subject of a growing number of scientific research, generating significant interest by various sectors of industry, government and researchers due to their characteristics of transparency, reliability and security (LIN; LIAO, 2017). Since the first popular blockchain implementation, Bitcoin, different blockchain systems have emerged with proposals outside the financial system, highlights include Ethereum (BUTERIN, 2015a), Hyperledger (Linux Foundation, 2018) and Multichain (GREENSPAN, 2015).

As for the implementation of a blockchain network, there are variations according to the purposes of the application. The first question is regarding the model of the blockchain network, which there are some variations. Some models characterize the blockchain based on its permission model, a determining question to know who can participate and maintain the network.

The NIST (YAGA et al., 2018) template defines with only two templates the permissioned and permissionless / private or public. Basically, in a network where any user can participate and publish a new block, this network is considered non-permissioned, on the other hand if only specific users can publish, this is a permissioned network. The blockchain model defined by Massachusetts Institute of Technology (MIT) adds that blockchain is only when the network is completely decentralized and public, other types of networks are called Distributed Ledger (DLT), illustrated in Figure 4.

In Figure 4 it is possible to observe the differences between the blockchain and DLT. A comparison between the traditional model, in Trusted Third Party (TTP), distributed and decentralized database encryption is illustrated. In this comparison it is denoted that the large scope of distributed and decentralized applications are not

Figure 4 – Types of data architecture.



Source: Author.

considered blockchain, but DLT or decentralized database.

For this work, the model defined by Buterin (2015b), Lin e Liao (2017), Joshi, Han e Wang (2018) is considered. This model classifies blockchain into three main models, differentiated by their access, which are public, consortium and private blockchain.

- **Public Blockchain**: is a model with a fully decentralized and open accounting platform so that any user can publish blocks, read these blocks/transactions and validate transactions. Due to the open feature of the public model allowing any user to participate, some of these users may be malicious, that is, they try to publish blocks in a way that subverts the system. In order to avoid these types of attacks and damages, these blockchain networks usually use a multi-party system of agreement, known as consensus mechanisms (Subsection 2.3.2), which requires the user beyond the necessary computational power. , also spend or provision resources in an attempt to publish new blocks. In this way, preventing or hindering the system from being easily subverted. Some examples of consensus mechanisms that are applied in the public blockchain model are Proof-of-Work (PoW), Proof-of-Stake (PoS) among others (Subsection 2.3.2). These mechanisms make this model safe and with low malicious behavior, providing rewards to users, known as miners, who insert new blocks in the chain and validate transactions.

- **Consortium Blockchain**: The consortium model is a combination of public and private blockchain characteristics, being composed of two or more institutions and perceived as partially decentralized. This blockchain network has pre-defined institutions and their participants, the latter need authorization to publish new blocks on the network or participate in the validation process. In general, this model is a hybrid between the low trust that exists in the public model and the single-entity model of the private, allowing the implementation of restrictions regarding the access to read, issue and validate transactions, pre-defined and configured in the code. network start.

- **Private Blockchain**: The private model is the most restrictive type of blockchain, involving only one institution as the blockchain network manager. This model has a pre-defined maintaining institution, as well as its participating users, who need authorization to participate in the process of inserting new blocks, making this model the only one that can be centralized. In this model and in the consortium, only allowed/authorized users maintain the blockchain network, which makes management more rigorous and restrictive of the authority to access data on the network. Providing a higher level of efficiency in verifying and validating transactions, making it safer, more efficient and faster.

From this definition, it is possible to observe the main differences between the public, consortium and private blockchain models. The adhesion, in number of users, to the public blockchain is considerable due to the fact that all accesses are released and that there is no need for trust on either side. Private and consortium blockchain networks have many similar characteristics that make it possible to implement restrictions that aim to make the network more secure, less accessible and controllable.

Another interesting feature in these models is that both maintain the main traceability of digital assets, the distributed, resilient and redundant storage system of the public blockchain models. One of the main differentials of these networks compared to the public model is their consensus mechanism, as they generally do not require resource provisioning and previously imposed hardware limitations. Some examples of the consensus mechanisms applicable to these models are PoA, pBFT among others presented in Subsection 2.3.2.

In relation to possible cyber attacks, mainly involving malicious users, these models have as maintainers of the blockchain network users who have a minimum level of trust with each other, who need authorization to participate in the blockchain network and who can be revoked in the occurrence or detection of bad behavior. This minimal relationship of trust between the consortium and private models allows the blockchain to use other consensus mechanisms, which make the network more efficient, faster, safer and with less need for computational power.

On these issues, current research shows a strong growth of investment in solutions using blockchain technology. Research carried out by International Data Corporation (IDC) points to an annual growth of 48% of investments in blockchain until 2024, reporting that the COVID-19 pandemic highlighted the need for greater investments in resilience and transparency (NEEDHAM, 2021). The main sources of investment are from the industrial and government sectors, with a focus on private and consortium models. These institutions have various application issues, such as financial services, real estate, manufacturing, construction, transportation, supply chain, telecommunication and *etc.* Used mainly by organizations that need to more tightly control and protect their system through blockchain. What makes auditing and supervision entities constant and/or periodic occurrences for the proper functioning of the entire system.

Extending the blockchain implementation issues, from the blockchain network model decision, there are two main underlying elements that cover this entire implementation process (MIERS et al., 2019):

- **An integrity-checked data structure:** As its name implies, the blockchain uses a data structure based on a chain of blocks, where each block (a set of any records, accompanied by metadata) carries the value of *hash* from the previous block. By digitally signing the records stored in the individual blocks, the blockchain allows the detection of local changes and, indirectly, also allows verifying the integrity of the previous blocks (whose *hashes* would be altered in case of improper modifications).
- **A distributed consensus mechanism:** the blockchain uses techniques to ensure ledger consistency, i.e. that all users will eventually have the same view of the order in which the blocks were entered into the chain. Thus, it is guaranteed that the distributed characteristic of the database does not lead to inconsistent decisions on the part of users. In particular, this mechanism is important to prevent the same currency from being spent multiple times: as long as all users agree on the first transfer transaction of that currency, only that first transaction will be considered valid. By performing this procedure, the blockchain can be seen as a distributed Time Stamp Authority (Timestamp Authority (TA)), which defines the temporal relationship between blocks inserted in the system. It is important to note, however, that this temporal relationship is: (1) only relative, i.e., it does not seek to determine the exact times when a transaction took place; and (2) it does not necessarily correspond to real time instants, so that a transaction occurring at time t may appear on the blockchain after a transaction occurring later, at time $t+\epsilon$. Therefore, the blockchain creates a kind of "*alternative timeline*", which does not necessarily correspond to the actual temporal order of transactions.

Blockchain technology may seem complex, however it can be simplified by examining these two elements individually. In a high-level context, blockchain uses well-known mechanisms in computer science and cryptographic primitives (cryptographic *hash* functions, digital signatures, asymmetric key cryptography) integrated with record keeping concepts. The following subsections seek to discuss each of these elements in more depth.

2.3.1 Blockchain Structure

Users of a blockchain network submit candidate transactions to the blockchain network through applications. This software is responsible for communicating and sending these transactions to one or more nodes that belong to the blockchain network. These transactions are propagated to other nodes on the network, but this does not guarantee that they will be published and validated. These transactions are dependent on validation, through consensus mechanisms (Subsection 2.3.2), and must wait in a queue to be added to a block on the blockchain network.

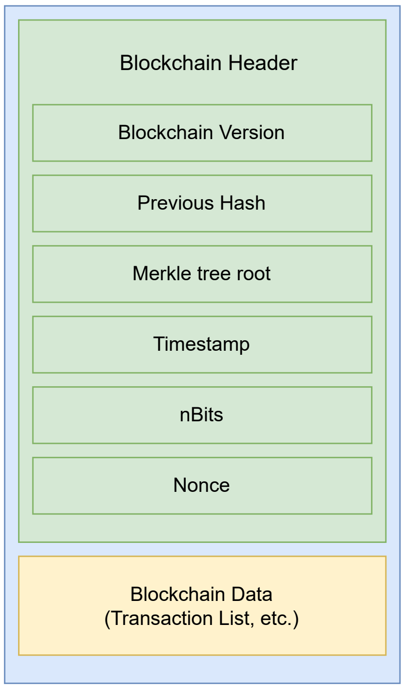
The data structure underlying blockchain schemas consists of a sequence of blocks, illustrated in Figure 5. In the sequence presented, a complete list of records is stored, which creates a set of data available for consultation by any user of the system (NAKAMOTO, 2008). According to the needs of the scenario, the rules for storing the records are carried out, which is up to the blockchain to only store these records, ensuring the integrity of the blocks and their order in the system.

Basically, a block contains (1) block header and (2) block data, illustrated in Figure 5. The block header stores the block metadata, while the block data contains a list of validated and authenticated transactions that were sent to the blockchain network (YAGA et al., 2018). Validity and authenticity are guaranteed by verifying that the transaction is correctly formatted and that the digital asset providers in each transaction have cryptographically signed the transaction, as illustrated in Figure 6, making it possible to verify that the asset providers of a transaction had access to the private key that makes it possible to sign the digital assets that are available. Thus, the other nodes are responsible for verifying the validity and authenticity of all transactions in a published block and have not accepted certain blocks that contain non-validated transactions.

Figure 5 illustrates the composition of a generic blockchain block. Generally speaking, the following fields are present in blocks in addition to transactions chosen by miners (PLURASIGHT, 2017):

- Blockchain header:
  - Blockchain version: Block structure version.
  - *Hash* Previous: The value of the *hash* of the previous block, calculated using a secure *hash* function (in the case of Bitcoin, SHA-256 (NIST, 2015)).
  - *Merkle Tree Root*: The value of *hash* corresponding to the root of a Merkle tree (Merkle, 1980) constructed from all transactions included in this block.
  - *Timestamp*: Block creation date and time.

Figure 5 – Generic block structure.



Source: Author.

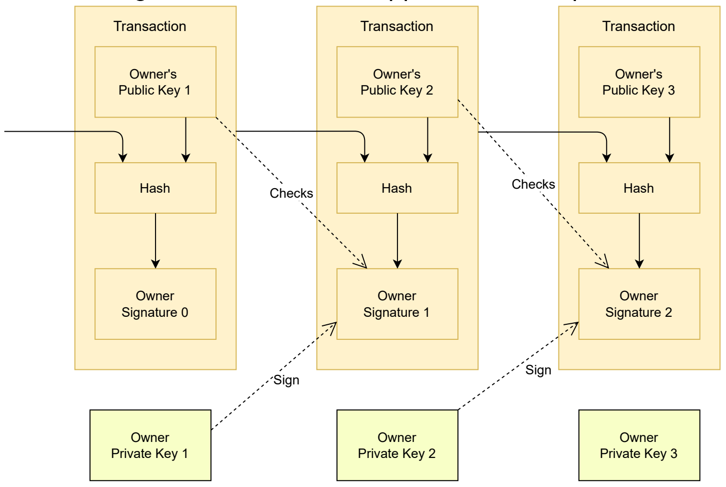
  - nBits: Value that represents the computational difficulty for mining this block. This field is directly related to the consensus mechanism used by Bitcoin, known as PoW, as discussed further in Subsection 2.3.2.
  - *Nonce*: Arbitrary value added to the block to give variability to the *hash* value of the block. In blockchains that use mining, this is the number that is manipulated by the publishing node to resolve the hash.

- Blockchain data:
  - List of transactions and accounting events included in the block.
  - Other data may be stored.

When a new blockchain network is created, a block known as *Genesis* is created. The *Genesis* is a different block from the others created, as it is the first in the chain, it does not have the previous *hash*. Another issue is that this block may contain different information from the blockchain, according to its implementation (e.g., configuration data) (MIERS et al., 2019).

From this structure based on *hash* chaining, blockchain technology guarantees the integrity and their relative order of the blocks. Allied to the digital signature of the stored records, the authenticity of the data can also be verified in a distributed way. In particular, when this data takes the form of asset transactions between users, the current owner of any asset can be identified simply by analyzing the chain of valid transactions carried out. Figure 6 aims to illustrate how different transactions change

Figure 6 – Blockchain application example.



Source: Author.

the owner of the transaction object (e.g., a cryptocurrency): when signing a transaction, the asset owner indicates his consent to transmit it to the new owner; the last user identified as the asset's receiver is then its current owner.

2.3.2 Consensus Mechanism

One of the key aspects of blockchain is determining the user, known as a node, responsible for publishing the next block in the chain. It is important to note that it is a process carried out asynchronously and without central coordination, in which each of these nodes, at a given moment, may have a different view of the blockchain chain. These issues are resolved by implementing many possible consensus mechanisms. The consensus mechanism is essential for the development of a consistent system, in which all participating nodes agree on the order of blocks and their certain transactions/information. Basically, the consensus mechanism makes it possible to build a highly resilient environment to breaches, in which transactions are verified by all participants, whether they are trusted or not (SWAN, 2015).

The main ideas behind the technology emerged in the late 1980s and early 1990s. In 1989, Lamport developed the protocol *Paxos*, submitting the paper *Part-Time Parliament* (LAMPORT, 1998). This article presented a consensus model, in order to democratically guarantee that it was possible to reach an agreement on a certain result in a computer network where the network or computers are not trusted. In any scenario that involves these issues, the consensus has the function of applying and/or verifying a set of rules established from the Genesis block and applied through the participants who act in an organized way. In this way, the specification of a consensus scheme involves, in addition to organizing the nodes and defining their operating rules, the very infrastructure on which the blockchain operates (MIERS et al., 2019).

Tasca e Tessone (2019) define the main subcomponents that form the consensus mechanisms are:

- Consensus Network Topology: This component is related to the level of decentralization in the validation process and factors such as applied reward mechanisms. The topologies applied in the blockchain are similar to other P2P networks, being these decentralized, partially decentralized and centralized.
- Agreement for Consensus and Conflict Resolution: It is the definition of a set of rules in which records are updated independently and timelessly by the nodes of a distributed system. This question is important to understand how a distributed system is able to deal with Byzantine faults, which present themselves in different ways to different observers, due to the absence of a global view of the network, thus creating inconsistency in the system. Depending on the scenario, this type of failure can be handled through synchronous or asynchronous communications, combined with a voting mechanism to resolve conflicts. When consensus is established on the network, it can be considered (TASCA; TESSONE, 2019): deterministic, in which the information stored in a blockchain after consensus cannot be changed later; or non-deterministic, in which when consensus is established, at a given moment it does not prevent the state of the blockchain from being changed by a later consensus.
- Communication and messaging: Blockchain is also a decentralized and redundant storage system. Redundancy makes it difficult to steal and/or hijack the information stored on them. As this information travels through the network and, in general, there is no central authority, each node must transmit the information it has to other participating nodes, which are at the same level of the system. There are two types of local message exchange, in which the message exchange takes place first between neighboring nodes, through a local validation process, and then propagates through the network until global consensus is reached. The second type of message exchange is global, which is the most common type in blockchain implementations, where communication takes place on a list of selected nodes, which are participants in the network.
- Algorithm for inserting blocks: A blockchain has a special type of distributed and fault-tolerant system, due to the replication of its states by all nodes. This issue allows the blockchain to continue in perfect operation even if a node ceases to be part of the network, whether for intentional or accidental reasons, having the ability to reestablish itself and adapt to maintain the same degree of reliability and validity of the information stored in the your records. In order for this consistency to exist between the data stored from different nodes, it is necessary to define

rules for updating this data in a distributed manner. If you do not have these well-defined rules, there would hardly be a convergence between the different versions of blockchain spread across the network. Since the development of blockchain technology, highlighting the consensus mechanisms PoW initially applied to Bitcoin, it has been accompanied by the development of different mechanisms and regulation of the insertion of blocks in the blockchain that help to maintain the consistency of the information contained in the blockchain. ledger. For this work, which has guidelines focused on private and consortium blockchain models, the following mechanisms stand out:

- RAFT: This is a *Crash Fault Tolerance* algorithm and is equivalent to Lamport's Paxos, but designed in a way that any user can apply and understand it. This Algorithm is decomposed into relatively independent sub-problems and clearly addresses all the main approaches to systems (ONGARO; OUSTERHOUT, 2014b). This consensus mechanism is applied in private and consortium blockchain models, being an algorithm of easy application, of low computational cost and that perfectly solves the needs of users.
- pBFT: This model is a replication algorithm, originally created to allow any system to tolerate Byzantine failures. Basically, the nodes organize themselves and operate through rounds, so that in each round a primary node is selected according to certain rules. The selected node is responsible for inserting the next block in the chain, this process is separated by three phases: pre-prepared, prepared and committed. In each of these phases, the nodes perform a vote for approval, requiring 2/3 of the votes to be validated. In this mechanism, there is no need for a level of trust between the parties, nor their real identity, it is only necessary to identify the address and the total number of participants, to carry out the vote. Another interesting feature is that this mechanism does not require large computational resources, such as PoW, rather only the consultation between nodes to obtain consensus. The model is suitable for use in private/consortium networks (MIERS et al., 2019; CASTRO; LISKOV, 1999).
- PoA: The PoA model is based on the partial trust of the publishing nodes, this link is due to the knowledge of real world identities. In this model, participants are not solicitors to solve arbitrarily difficult math problems, but use a set of pre-configured authorities to collaborate without trust. The idea behind this consensus mechanism is that participating users stake their identity/reputation for the publication of new blocks, as these can lose reputation by acting in a questionable way and also being banned. The PoA mechanism is only used in blockchain with private and consortium models, as it requires

a minimum of trust between peers. In order to prevent the impersonation of these special nodes by others, the pre-defined authorities receive valid digital certificates for signing the new blocks, in this way each block header that a client sees can be compared/verified with a defined list of trusted signatories. (TASCA; TESSONE, 2019).

– iBFT: iBFT is a Byzantine fault and component fault tolerant consensus algorithm. Regarding the implementation of this algorithm, it is based on PoA for the voting process and list of validators, in which they use the epoch process, allowing the same node to validate for a certain time. Regarding the insertion of the next block in the chain, this process is based on the pBFT algorithm, going through four phases: pre-prepared, prepared, committed and change of rounds. For the validation process of this algorithm, at least 66% of the participating nodes must agree.(MONIZ, 2020).

Table 2 presents a comparison between the RAFT consensus mechanisms, iBFT, pBFT and PoA. This comparison includes the mechanism PoW and PoS to perform a broad comparison between the main mechanisms mentioned.

Table 2 – Consensus mechanism comparison.

Mechanisms	Scalability	Node management	Energy saving	Comments
RAFT	High	Permission	Yes	Doesn’t resolve Byzantine faults.
iBFT	Medium	Permission	Yes	Confidence in the pre-selection
pBFT	Medium	Permission	Yes	Confidence in the pre-selection
PoA	Medium	Permission	Partial	Required authorities
PoS	Under discussion	Open	Partial	Fork "not cost", measurable assets needed
PoW	controversial	Open	No	High energy cost, Need to ensure high level of security

Source: Author.

With the presentation of some block insertion algorithms and their characteristics illustrated in Table 2, it is possible to observe their differences and that each of these mechanisms is a project to obtain better performance in certain blockchain network models. . Blockchain networks with private and consortium models, for example, allow the existence of some level of security between the publishing nodes, in which case there is no need to use a resource-intensive consensus mechanism (resources, computing time, investment, *etc.*) to determine which of the participating nodes is responsible for adding the next block to the chain. A pertinent observation is that as the level of security between nodes grew, the need for resources such as trust-generating measures decreased, as observed in the mechanisms PoA.

2.3.3 Virtualized blockchain applications

One of the main ways of developing new applications occurs through computing clouds, mainly through the IaaS model, more specifically using its application resources, in private or public clouds (LIU et al., 2020). Not different from this development medium is blockchain technology, which is traditionally applied through containers, virtualizations or a combination of both, possibilities illustrated in Figure 3. Among the technologies mentioned, virtualization stands out for the facilitation of creating an instance and also for its positive security aspects. An example is public computing clouds such as AWS, Microsoft Azure, IBM Cloud Platform, Oracle Cloud Service, which have made investments and incentives to use blockchain technology through VM, which can be built from scratch or with pre-built platforms. -defined (eg, Hyperledger, Ethereum and Multichain).

Private and syndicated blockchains are being applied for a variety of purposes, *eg*, supply chain management, financial services, insurance claims, food security, cybersecurity, baggage management, copyright and *protection royalties*, Data Access Object (DAO), energy market, retail, Internet of Things (IoT), government services, among others, some of these examples are presented in Table 3. The Table 3 is divided into the categories of solution, platform, mechanism and *flavor* issues. As for the mechanisms and characteristics of *flavor*, these are information that are generally not public, for security reasons, for this reason the *flavor* configuration is used as indicated by the platform and the same question for the applied consensus mechanisms.

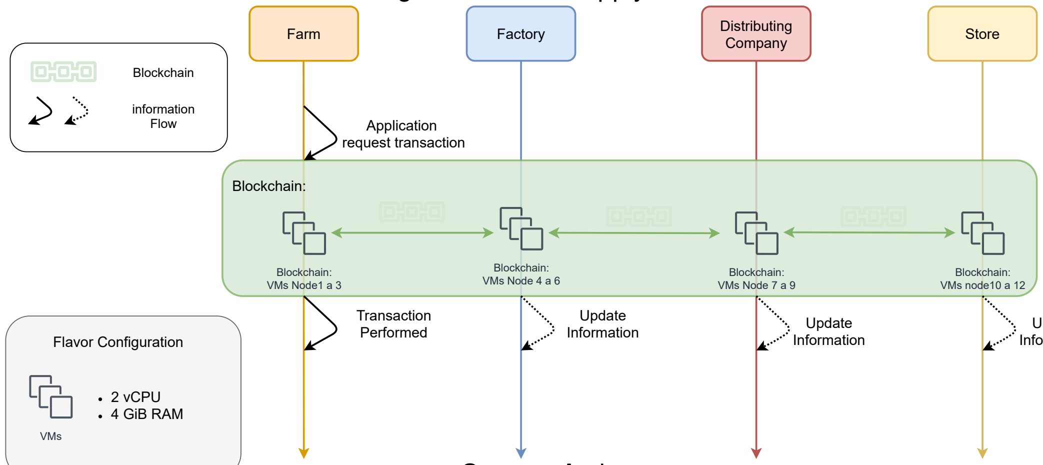
Table 3 – Blockchain Solutions.

Solution	Platform	Mechanism	Reference
Supply Chain	Corda CrystalChain Ethereum Hyperledger VeChain X-CEED	PoA PBFT PoS PoW Tendermint RAFT	(ESSBAUER; SCHMIDT, 2020) (IBM, 2020) (IBM, 2019) (VECHAIN, 2021a) (VECHAIN, 2021b) (Renault Group, 2021) (SHAHID et al., 2020)
Healthcare	BurstIQ Corda CrystalChain MedicalChain VeChain	PoA PBFT RAFT	(Blockpharma, 2021) (DALEY, 2021)
Digital Identification	Corda Ethereum Hyperledger VeChain	PoA PBFT PoS RAFT	(E-Estonia, 2021) (R3, 2021b)
Governmental	EW Chain KSI Blockchain	PoA	(E-Estonia, 2021) (Energy Web, 2021)
Travels	Ethereym Hyperledger Winding Tree	PoA PBFT PoS RAFT	(REINMUELLER, 2018) (AZURE, 2019) (IZMAYLOV et al., 2021)
Financial Services	Bitcoin Ethereum Hyperledger Tendermint	PoA PBFT PoS RAFT	(SINGAPORE. . . , 2020) (R3, 2021a)

Source: Author.

From Table 3 it is possible to observe some existing solutions using private and consortium blockchain. Some outstanding features are the issues of the most evolved platforms to support the technology, highlighting Ethereum with its smart contracts and decentralized applications and Hyperledger and its modules for distributed applications. One of the main differences of these models is the need for less computational power, or a *flavor* with low performance, since the mechanisms applied require many resources. As an example of a flow that occurs in the validation process of a new transaction or block in a private/joint blockchain chain, Figure 7 illustrates, in a simplified way, how this whole process occurs.

Figure 7 – Flow Supply Chain.



Source: Author.

From Figure 7 it is possible to observe the process of a blockchain network of a Supply Chain. It is possible to observe that each participating entity has its VMs to carry out the process of validating transactions and inserting new blocks. The process starts with the request or sending of information from an entity, through a *broadcast* this information is sent to all participating nodes, in order to validate this information. When this information is validated, a *commit* occurs and all processes in the chain receive an update of the new data that has been inserted into the blockchain network.

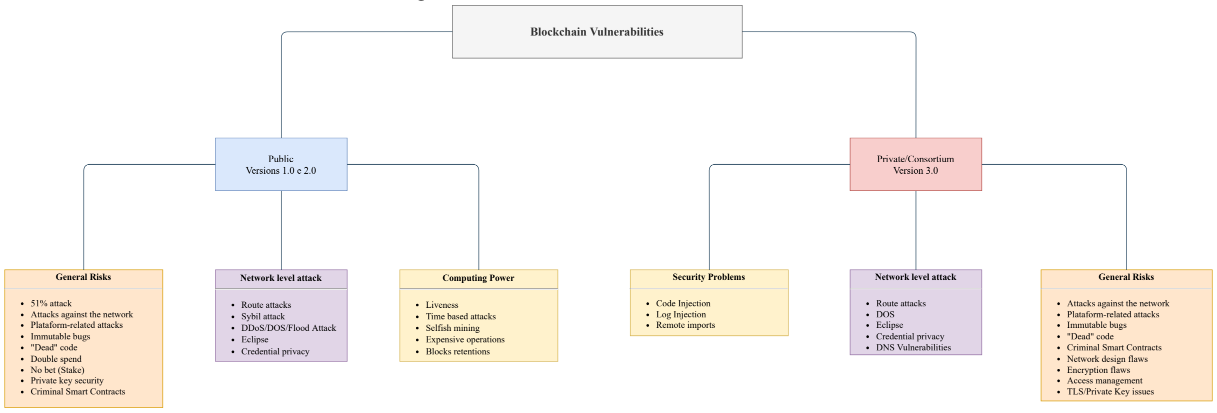
Another issue that is illustrated in Figure 7 is related to possible application vulnerabilities, especially when related to the private and consortium models. In Sections 2.2 and 2.3, virtualization and blockchain issues are addressed, represented through the VMs that exchange information. When different nodes exchange information and messages with each other, whether they are from the same institution or not, they use the network layer as communication, and in distributed computing one of the main challenges is related to this layer, which requires attention and mitigation as possible. vulnerabilities that the network layer may contain and also for platform and *etc.*

2.4 BLOCKCHAIN ATTACKS AND VULNERABILITIES

Blockchain networks are based on a data structure that stores all executed transactions, as a kind of "account book". Each block that is inserted into the network includes a *hash* that points to the previous block, thus creating a sequence that maintains a connection between the current block and the *Genesis*. The reversal of a block inserted in the network is methodologically complex, after a period that it is in the network.

Reversal issues, as well as any directly linked changes in the blockchain chain, are related to the chosen blockchain model and also the applied consensus mechanism. Making it necessary to make a relationship between the blockchain models with the main vulnerabilities known by the technology. Figure 8 presents a literature review that identifies the main vulnerabilities in relation to blockchain technology, relating them to their versions and the blockchain model (MIERS et al., 2019; HASANOVA et al., 2019).

Figure 8 – Blockchain Vulnerabilities.



Source: (HASANOVA et al., 2019; MIERS et al., 2019)

Figure 8 illustrates the main vulnerabilities of blockchain technology, based on its three models: public, private and consortium. It is possible to observe that the vulnerabilities presented are divided by categories.

Public model is divided into three categories and two versions of the blockchain:

- General Risks: This category presents the main vulnerabilities that are directly related to the applied blockchain platform, the rules of the consensus mechanism and the algorithm applied to smart contract.
- Network Level Attacks: The vulnerabilities presented in this category are related to network vulnerability issues, node or pool control, network inoperability and other related issues.

- Computing Power: In this category, vulnerabilities to issues related to increased computing power, obtaining benefits from malicious miners over honest ones, or simply reducing the reward benefit for certain pool.

Public blockchain models mainly use consensus mechanisms that use computational resources, opening precedents for these vulnerabilities to be exploited through attacks of the computational power category, as well as network-level attacks. One of the issues of public blockchain is immutability, which makes vulnerabilities such as bugs immutable and outdated code present and persistent mainly in long-time blockchain networks (HASANOVA et al., 2019).

The private/consortium model, unlike the public model, has a smaller scope of attacks, which are divided into two categories:

- General Risks: Vulnerabilities in this category have some issues related to the public model, such as blockchain platform issues, access management and monitoring and mainly related to network design and security and encryption guarantees.
- Network-level attacks: In private and consortium models, vulnerabilities related to network issues have a smaller scope of possibilities, but are more offensive in terms of network interoperability, making them necessary to be more careful.

From a comparison of vulnerabilities between blockchain models, the scope of vulnerabilities for exploitation is quantitatively greater in a public model. This fact is motivated by the fact that the network does not have any layer of trust between the parties, basically, public models are dependent on a higher computational performance, guaranteed by the consensus mechanism, which allows the participation and consultation of any user. Private and consortium models have, necessarily, a minimum level of trust between the parties, a characteristic used through the control of the participating nodes. However, these models are weak to vulnerabilities related to the network level, motivated by attacks that directly influence the operations performed by the node or by the blockchain network, affecting its performance.

Recently, blockchain network security issues have gained popularity in the fields of network security research (TAYLOR et al., 2019; HASANOVA et al., 2019). It is noticeable that despite the growth of research in security and application development, the main studies are related to issues of scalability, security in application functionalities, availability and sustainability.

Regarding the exploitation of vulnerabilities in blockchain networks and platforms, it is observed that with the rise of technology investments, it has become attractive to carry out cyber attacks that seek to influence the services of a blockchain network. Among the various attacks that are applied and presented in Figure 8, DoS/DDoS attacks stand out when the objective is the subversion or unavailability of the network or application (MIERS et al., 2019; HASANOVA et al., 2019). Basically, the main objective of a denial of service attack is the inaccessibility of computing resources by legitimate users. The main categories of attack methods aim to decrease bandwidth, characterized by flooding and amplification attacks and resource exhaustion through exploitation of the communication protocol (HEINRICH; OBELHEIRO, 2019).

The relationship of DoS or Distributed Denial-of-Service (DDoS) attacks with a blockchain network is interesting, because when it comes to a public model, due to the fully distributed/replicated nature of the blockchain, they are naturally resilient to denial-of-service attacks. service. In the case of private/consortium blockchain networks, which are partially decentralized or centralized, but which use P2P, DDoS attacks are inapplicable, as there are a limited number of participants, not allowing this to be carried out. . However, common DoS attacks are more successful in these models, this is due to the limited number of users participating in the network, which allows a malicious user to exploit vulnerabilities in a more accessible network.

In a real environment, as illustrated in Figure 7, it is possible to observe the VMs with its *flavor* which has 2 vCPU, 4 GiB RAM and 20 GB HD as characteristics. When an DoS attack is carried out, it aims to reduce bandwidth or exhaust resources, which makes this *flavor* configuration susceptible to being attacked and subverted. Thus, it is necessary to understand and mitigate the influence of these attacks on the flavor to guarantee the operationalization of the technology.

2.5 PROBLEM DEFINITION

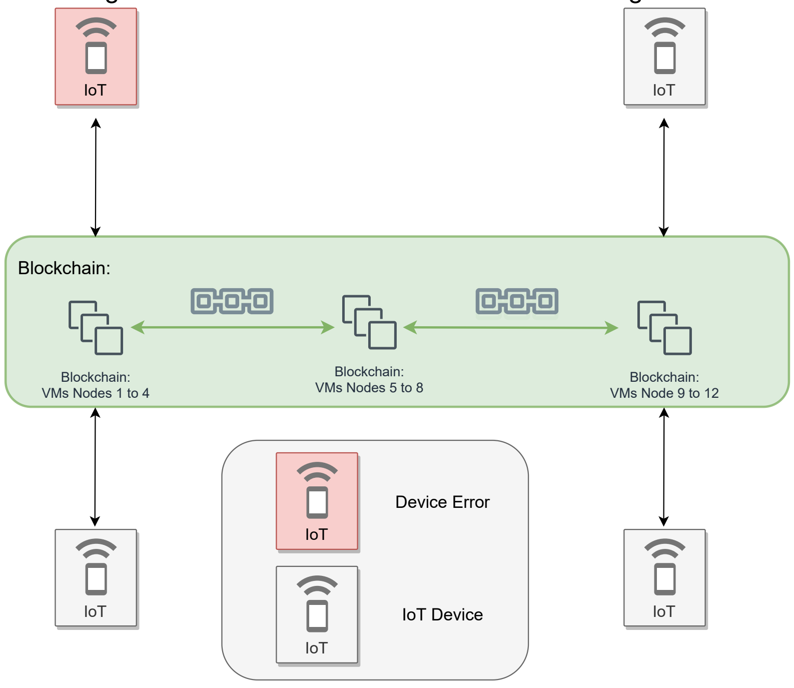
The popularity and growth in investment of blockchain technology is in evidence (IDC, 2019; NEEDHAM, 2021; TAYLOR et al., 2019), but the demands for applications have different contexts and user realities. Regarding these applications, in general, improvements are sought regarding the efficiency and quality of the technology, but there are concerns related to the computational cost that is necessary for its performance and security. In this context, it is clear that there are several consensus mechanisms responsible for performing various operations among these the blockchain validation process. For private and consortium blockchain models, the consensus mechanisms that stand out are RAFT, Tendermint, pBFT, iBFT and the PoA.

Organizations adopting solutions based on private or consortium blockchain, have used as a practice the creation of their blockchain nodes using VMs in private or public clouds. However, the security of services hosted within an IaaS cloud can be a problem when the blockchain nodes have access from the users of these organizations. For example, a malicious user may accidentally or maliciously cause a DoS attack.

Blockchain networks have a growing concern about issues related to scalability, energy consumption, computational cost and *etc*. Another important issue is the environment in which blockchain nodes are created, especially computing clouds with their virtualization services. In general, the recommended *flavors* settings for creating VMs in private and consortium models vary according to the platform that is applied, it is a valid question for public clouds like AWS and Azure and also for private clouds (Ethereum Foundation, 2021; Linux Foundation, 2018). A platform that stands out for smart contracts and distributed applications is Ethereum, the Ethereum platform has two types of recommendations for generalized and specific environments. In the case of generalized environments, the application of 2 vCPU cores and 4 GB of RAM is indicated, for specific environments 4 vCPU cores and 8 GB RAM are recommended (Ethereum Foundation, 2021).

From these characteristics and based on the real and simple case of a supply chain, illustrated in Figure 7, it is possible to observe an environment of interaction and exchange of messages of the VMs represented in Blockchain network. In this environment, the presence of twelve VMs that maintain the blockchain network is observed, these nodes, or VMs, are responsible for the transaction validation process and insertion of new blocks in the network. Two possible environments are illustrated, in Figures 9 and 10, among several possible scenarios, which portray situations that can occur in real environments.

Figure 9 – Blockchain network with IoT showing error.

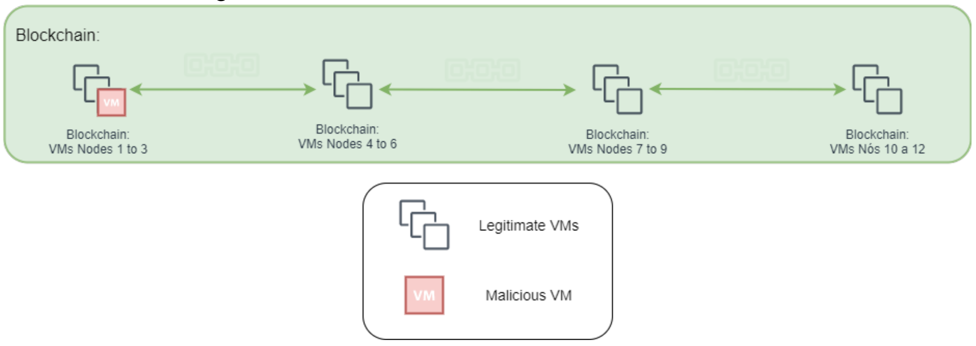


Source: Author.

In the scenario illustrated by Figure 9 an environment with a blockchain network, twelve VMs and IoT devices that send transactions to the blockchain network. This scenario reflects a production environment, in which one of the IoT devices has a failure or an error and sends several transactions per second to the blockchain network. From the occurrence of this situation, the blockchain network does not have the ability to distinguish only the true transactions that are sent by the devices, thus producing an DoS reducing the bandwidth. This sending of a considerable amount of transactions generates an unintentional DoS attack, producing instability for the application and the environment, starting from an unintentional failure.

Figure 10 illustrates the blockchain network environment of Figure 7 with a malicious user. This malicious user has the objective of subverting the system through an DoS, in other words, that the application or some nodes become unavailable making the network unstable or vulnerable. Subversion of the system through an DoS is possible through resource exhaustion, exploiting communication protocol vulnerabilities, and bandwidth reduction, through false-positive transactions. The same issues can be present in malicious applications that run within the network with the same devices.

Figure 10 – Blockchain network with malicious user.


Blockchain:
VM
Blockchain: VMs Nodes 1 to 3
Blockchain: VMs Nodes 4 to 6
Blockchain: VMs Nodes 7 to 9
Blockchain: VMs Nós 10 a 12
Legitimate VMs
VM
Malicious VM

In the first scenario, it is possible to observe two possible occurrences, the first occurrence is when the malicious VM exploits the vulnerabilities of the communication protocol, generating attacks against a certain instance, from which a considerable amount of resources is consumed allowing the VM is subverted and disconnecting from the blockchain network it participates in, leaving it more vulnerable. In the second instance, the malicious instance sends a series of false-positive transactions on the blockchain network, thus leaving it overloaded and making the process of validating licit transactions difficult, which generates instability, overloading the administrator node, if any, and also in the processing. from the Web.

Both scenarios illustrated by Figures 9 and 10 present DoS intentionally or unintentionally, but with similar results. From these possibilities of scenarios and uncertainties regarding the needs of each system, questions regarding the configuration of *flavors* and private/consortium models arise, which are:

- What are the proper instance *flavor* settings?
- What is the number of transactions required to operationalize the technology?
- What criteria/metrics indicate blockchain and/or application issues?

These issues become important for the development of the blockchain network, as the amount of transactions that sectors (*e.g.*, government, industrial, pharmaceutical and financial) perform are different and vary according to related areas. The amount of transactions that a blockchain network supports can change according to the *flavor* settings of the VMs, which vary according to the amount of vCPU, RAM and network-related settings. Thus, through the scenarios presented with the minimum *flavor* settings required by the platform, it is notable that there is a considerable probability that DoS will affect the performance and functionality of the blockchain application.

2.6 CHAPTER CONSIDERATIONS

Computing clouds are a model that allows access to computing resources in a convenient, ubiquitous and scalable way. At this point, actors are involved such as the consumer, who uses the services, the provider, who provides these services and the auditor, who is responsible for the audit issues. In addition, the concepts of the main technologies are revisited, such as virtualization that enables the creation of IaaS environments. The development of applications using VMs is evident, blockchain is one of these applications that benefits from the use of virtualization technology. In this context, it is noted that institutions have applied the technology from VMs. However, further mitigation is needed regarding the use of blockchain parallel to the *flavor* of VM. Finally, different scenarios are presented in which different users, devices or applications can carry out DoS attacks on the blockchain network in a malicious or non-malicious way. These scenarios present the definition of a problem that may occur in blockchain networks. Making it necessary to carry out further mitigations in possible scenarios.

3 REQUIREMENTS & ANALYSIS PROPOSAL

Blockchain technology has considerable versatility, which has attracted interest from various sectors. However, blockchain has some adversities mainly related to its computational cost, application environment and security issues, which vary according to the purpose for which it is applied. What makes it necessary to study the feasibility of which technologies should be applied with the blockchain, taking into account the analysis of the applied model, the consensus algorithm, the characteristics of its *flavor* and issues related to the environment that provide benefits to the technology. .

3.1 REQUIREMENTS DEFINITION

In order to propose a solution to the problem, presented in Section 2.5, it is necessary to survey some prerequisites. In this sense, the following are identified:

- Have access to a private or public computing cloud IaaS, which has VMs with *flavor* settings defined in this work.
- Set up a relevant scenario and with VMs with privileges to collect network traffic and collect information about the instances.

From the adoption of the prerequisites, it is possible to determine the functional requirements (RF), which aim to present the functionalities that the system must perform, and the non-functional requirements (RNF), which present the behavior of the system, to perform of the experiments and solution of the defined problem:

- **RF1:** The environment must allow transactions to be carried out through API or automated mechanisms for the blockchain network;
- **RF2:** The sending of transactions must be collected its quantity and its *hash* for eventual verifications;
- **RF3:** Collect VM metrics such as: processing, memory, networks (TCP and UDP) and transaction latency;
- **RNF1:** The system must provide means of parameterizing the system, allowing it to be adapted to the characteristics of the experiment; and
- **RNF2:** The techniques adopted to capture metrics and transactions should not affect performance.

3.2 RELATED WORKS

In order to conduct the analysis regarding the performance and security in private blockchain networks, a search is carried out in the IEEE, ACM, SciELO, Springer and Elselvier databases with the purpose of identifying works that have a similar scope to the objective of this research. . To conduct the systematic review of the literature, the review of Kitchenham et al. (2009) is applied, and from this a review protocol is established containing inclusion and exclusion criteria, definition of the research bases, keywords and the reading sequence of abstracts and articles.

3.2.1 Review protocol

The protocol established for carrying out the research and delimiting the scope of the research has the following criteria:

- Bases used in the research: IEEE, ACM, SciELO, Springer and Elsevier;
- Samples: conferences, periodicals and books;
- Keyword: "Performance" or "Analysis" and "Blockchain" and "Private";
- Research period: 2015 to 2021;
- Inclusion criteria: Articles and Reviews that contain "Performance" or "Analysis" and "Blockchain" in their title; and
- Exclusion criteria: Articles and reviews that do not contain the word "Blockchain", "Private", "Performance" and/or "Analysis" in their abstract or title.

Searches using the settings established from the criteria presented in the protocol returned a total of 197 works published between 2015 and 2021. From these 197 works that were returned in the research, the first phase of inclusion and exclusion of the work was carried out, which focuses on fulfilling the criteria applied to the title. With the application of exclusion and inclusion criteria in the titles of the works, 174 works were excluded, returning 21 works for analysis of their abstracts. From the reading of the abstracts of the 23 studies that returned from the first phase of exclusion and insertion criteria, 7 works were selected as feedback from the systematic review performed.

3.2.2 Comparison Requirements vs Systematic Review

After carrying out the systematic review and applying the criteria of Subsection 3.2.1, seven works were raised. Table 4 presents a comparison between related works and functional requirements, from Section 3.1.

Table 4 – Related works vs Functional Requirements.

	(DORRI et al., 2017)	(Rouhani; Deters, 2017)	(Pongnumkul; Siripanpornchana; Thajchayapong, 2017)	(Hao et al., 2018)	(Davenport; Shetty; Liang, 2018)	(Vatcharatiansakul; Tuwanut, 2019)	(Chowdhury et al., 2019)	(MONRAT; SCHELéN; ANDERSSON, 2020)
RF1	Yes.	Yes, until 5k transactions.	Yes, until 10k transactions.	Yes, until 10k transactions.	No.	Yes.	No.	Yes.
RF2	No.	No.	No.	No.	No.	No.	No.	No.
RF3	Partially, metrics: Packet overload, time and energy consumption.	Partially, metrics: RAM and latency	Partially, metrics: Latency.	Partially, metric: latency and number of transactions	No.	Partially,metrics: Latency, transfer rate and runtime.	No.	Partially, metrics: Latency and transfer rate.

Source: Author.

The related works found and selected do not address issues related to security or VM with blockchain networks. These have the general objective of performing performance tests related to consensus mechanisms and also to blockchain platforms, Ethereum and Hyperledger.

The works of (Pongnumkul; Siripanpornchana; Thajchayapong, 2017), (Rouhani; Deters, 2017; Hao et al., 2018; Vatcharatiansakul; Tuwanut, 2019; DORRI et al., 2017; MONRAT; SCHELéN; ANDERSSON, 2020) are related to the issue of verifying the performance of blockchain platforms and consensus mechanisms. In these works, several transactions are performed on the platforms and the amount of transactions, memory consumption, latency, transfer rate, processing time and energy consumption are verified. The work of Chowdhury et al. (2019) makes the comparison between different platforms through their characteristics. These related works have no direct connection with the general objective of the present work, but are related to the specific objectives. The choice of these works is due to the intention of obtaining a greater scope for selecting the criteria for carrying out the analyzes of this work.

3.3 PROPOSED ANALYSIS

With the growth of investments in technology, the search for better quality and efficiency of blockchain is growing, especially in the case of private and consortium models. In parallel with the efficiency of the technology is the reduction in the use of computational resources, with the development of different consensus mechanisms. However, it is presented in Section 3.2 that the researches do not focus or aim at issues related to the operationalization of the technology, but rather the validation of the efficiency of blockchain platforms and the developed consensus mechanisms. These works, when related to security and operability issues, only show that there is a need for further studies in the area, especially when related to the environment in which blockchain technology is inserted (TAYLOR et al., 2019; Davenport; Shetty; Liang, 2018; HASANOVA et al., 2019). In this sense, Section 2.5 presents a blockchain environment in a computational cloud of the IaaS type, which has the nodes of a blockchain network created from a VM. This environment is composed of twelve VMs which have as *flavor* 2 vCPU and 4 GB of RAM, and is illustrated from Figures 9 and 10 , already presented, that characterize two scenarios, in which an DoS attack occurs, the first being unintentionally and the second intentionally.

In a real environment where an unintentional DoS attack occurs, there are two situations that can occur:

1. It occurs from the accumulation of several transactions that the platform receives, which causes a high rate of network traffic and, as a result, latency occurs on the network. With the occurrence of latency in the blockchain network, consequently, delays in validation and insertion of new blocks occur.
2. This maintains the same line of thinking as the first situation, but as a result of this latency and reduced network traffic, service unavailability or dropped packets that are queued for network traffic may occur.

The second scenario presented in Section 2.5 is an intentional DoS attack. One of the main features of the private/consortium model is that these blockchain networks are formed by known nodes, whether from the same institution or from partners. Regarding this internal issue, current research indicates that the main culprits for data breaches in computing clouds are caused by people/internal users of institutions that are directly connected to their local network (REPORT, 2019; Zyskind; Nathan; Pentland, 2015).

With these issues in evidence, the second scenario presented in Section 2.5 arises, in which it is an intentional DoS attack. Basically, when an DoS attack occurs intentionally, a malicious user belonging to the network carries out attacks against other healthy nodes in the network in search of consuming their resources, or carries out a direct attack on the network using realizations thus reducing the network bandwidth, and consequently the same implications of carrying out an unintentional DoS occur in this type of attack. The first hypothesis, which deals with a malicious user attacking another integral node of the network, has some behaviors that are expected in the network. The first expected behavior is that the user is able to carry out the process of validating transactions or nodes, but that it occurs slowly, harming the speed and good functioning of the network. The second expected behavior is more offensive, in which, from the consumption of user resources, this user reaches the subversion of the system, in which he becomes unavailable and does not participate in the process, allowing the number of consensus validators to decrease and the realization of other attacks together to be more efficient. In the same case, when there is a network administrator node and it is attacked, there is greater instability in the network, in which its re-organization takes longer.

From these hypotheses and expected behaviors, this work proposes to perform a performance and security analysis on a blockchain network, during the execution of a DoS attack. More specifically, this work has the following aspects to be analyzed:

- Analyze the number of transactions per minute that an instance can perform without harming the service;
- Analyze the immutability and integrity of transactions; and
- Establish a relationship between the instance's *flavor* and the intensity of DoS attacks on the monitored computing resources.

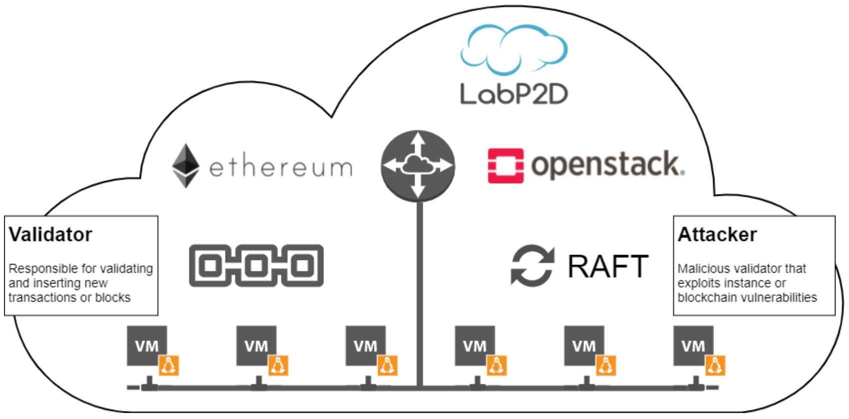
In order to achieve these aspects, the following sets of monitoring are exercised in the applied experiment:

- Instance resource monitoring:
  - Processing: Monitor the percentage of processor occupancy per minute during the attack period;
  - Memory: Monitor the usage of RAM memory occupied per minute, during the attack period;
  - Network traffic: Monitor the volume of traffic on the link during the attack;
  - Number of transactions: Monitor the amount of transactions sent per minute; and
  - Latency: Monitor how long each transaction takes to get from source to destination.

To carry out the proposed analysis, three scenarios are developed, in which, in general, they differ only by the consensus mechanisms that are applied. This choice is due to the fact that the literature shows the importance that a consensus mechanism exerts on a blockchain network, which is a relevant factor for carrying out the experiments. And with that the following scenarios are presented:

- Scenario I - RAFT consensus mechanism: This second topology, illustrated by Figure 12, has six instances with a GNU/Linux Ubuntu 20.04 distribution image. The instances have a *flavor* configuration of 2 vCPUs and 4 GB of RAM, which *flavor* is indicated for generalized projects in Ethereum and applied in this work (Ethereum Foundation, 2021). This scenario uses a private blockchain network with the RAFT consensus mechanism.

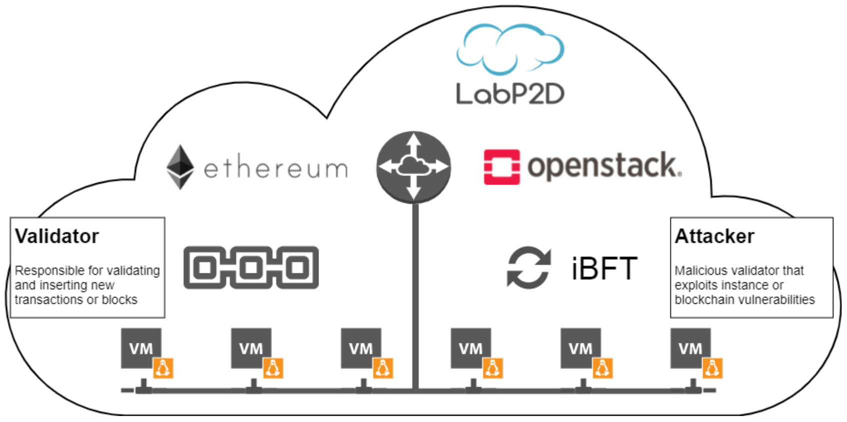
Figure 11 – Architecture Scenario I - RAFT.


LabP2D
ethereum
openstack.
Validator
Responsible for validating and inserting new transactions or blocks
RAFT
Attacker
Malicious validator that exploits instance or blockchain vulnerabilities
VM
VM
VM
VM
VM
VM

Source: Author.

- Scenario II - Consensus mechanisms pBFT: This first topology, illustrated by Figure 11, has six instances with GNU/Linux Ubuntu 20.04 distribution images. The instances have a *flavor* configuration of 2 vCPUs and 4 GB of RAM, which *flavor*is indicated for generalized projects in Ethereum and applied in this work (Ethereum Foundation, 2021). This scenario uses a private blockchain network with the iBFT consensus mechanism.

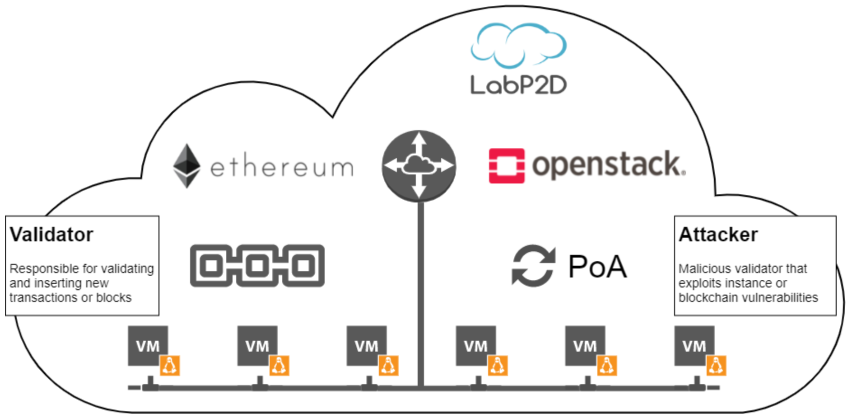
Figure 12 – Architecture Scenario II - iBFT.


LabP2D
ethereum
openstack.
Validator
Responsible for validating and inserting new transactions or blocks
iBFT
Attacker
Malicious validator that exploits instance or blockchain vulnerabilities
VM
VM
VM
VM
VM
VM

Source: Author.

- Scenario III - Consensus mechanism PoA: This second topology, illustrated by Figure 12, has six instances with a GNU/Linux Ubuntu 20.04 distribution image. The instances have a *flavor* configuration of 2 vCPUs and 4 GB of RAM, which *flavor* is indicated for generalized projects in Ethereum and applied in this work (Ethereum Foundation, 2021). This scenario uses a private blockchain network with the PoA consensus mechanism.

Figure 13 – Architecture Scenario III - PoA.


LabP2D
ethereum
openstack.
Validator
Responsible for validating and inserting new transactions or blocks
PoA
Attacker
Malicious validator that exploits instance or blockchain vulnerabilities
VM
VM
VM
VM
VM
VM

Source: Author.

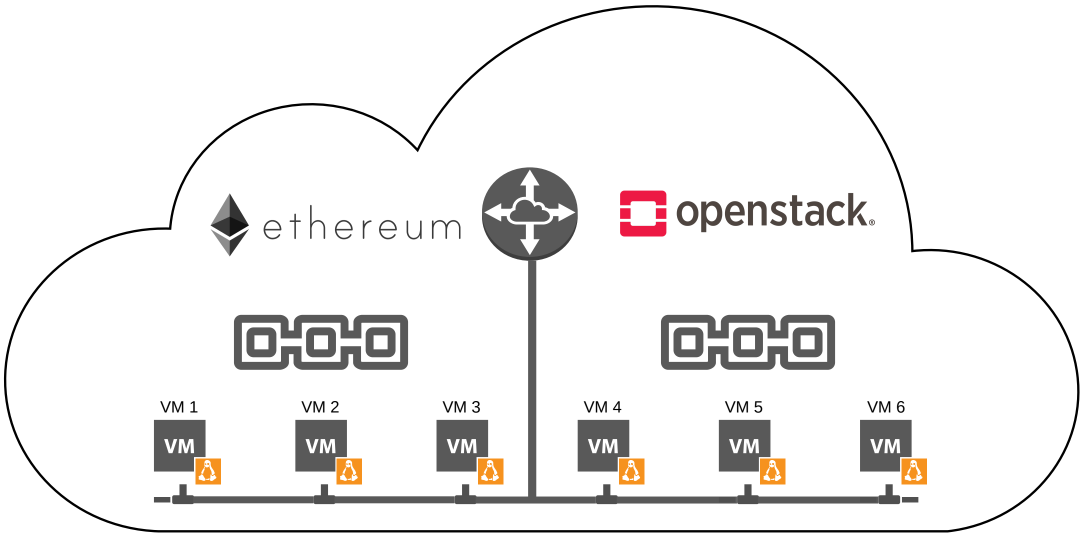
In addition to defining these three scenarios, two hypotheses are also defined: (a) It is an unintentional DoS attack, (b) It is an intentional DoS attack. It is important to point out that in the three defined scenarios none of the VMs have virtual memory resources *swap* enabled, they only use the main memory because there is the objective of depleting their resources. In this sense, the following experiments are performed in each of the scenarios:

- Experiment 1: Perform several transactions through a user/client on the blockchain network, with the objective of reducing network traffic or service unavailability, aiming at identifying, from the monitoring of resources and transactions, the stability of the network in the event of a DoS attack in a non-malicious way.
- Experiment 2: In this experiment, there is a malicious user who sends transactions directly to a node of a blockchain, with the objective of consuming resources from this instance, aiming to identify the stability of the instance and also of the blockchain network in the event of a DoS attack maliciously.

3.4 TEST ENVIRONMENT

The Laboratory of Parallel and Distributed Processing (LabP2D) computing cloud has the IaaS model with the open solution OpenStack in the Ussuri version. This test environment was built as a project in the Tchê Cloud of LabP2D, the approach chosen was the use of VMs with flavor 2 vCPU and 4 GB of RAM and Ethereum platform solution and with private blockchain model. Although the use of a blockchain network is not commonly applied in just one computing cloud, the choice was made due to the practicality of executing the experiments and isolation from other factors (eg, background traffic) that make the analysis more subjective and complex. From this question, the architecture was built through three main components: network, router and VMs (Figure 14). In terms of the applied network architecture, it is an *overlay* network that connects via virtual links and *switches* in OpenStack.

Figure 14 – OpenStack environment scenario.



Source: Author.

From Figure 14 it is possible to observe that the network link starts from the router that has 10.30.30.47/24 as its internal IP. The internal network consists of VMs six instances running GNU/Linux Ubuntu 20.04, with their *IPs* ranging from 10.30.30.0-254. The instances have the Ethereum platform installed. Table 5 presents the relationship between Experiments 1 and 2 performed with the presented architecture in relation to the functional and non-functional requirements presented in Section 3.1.

Table 5 – Experiments *vs.* Requirements.

Requirement	Experiment 1	Experiment 2
RF1	Yes	Yes
RF2	Yes	No
RF3	Yes	Partially, VMs metrics
RNF1	Yes	Yes
RNF2	Yes	Yes

Source: Author.

Table 5 presents a comparison between the functional requirements with Experiments 1 and 2. It is possible to observe that Experiment 1, compared to Experiment 2, has a greater relationship with functional requirements. The issue that differentiates Experiments 1 and 2 is that in the first case it only addresses the issues of exploiting an DoS attack directly related to the blockchain network, which involves its consumption of resources and network traffic of machines equally. While Experiment 2 has the objective of consuming resources directed to a single VM, making different vulnerabilities to be exploited in an instance through the DoS attack.

3.5 ETHEREUM

Ethereum was founded and developed in 2014 by Vitalik Buterin. The platform's main intention is to develop an alternative protocol for creating decentralized applications, providing a different set of tradeoffs that are useful for a wide range of decentralized applications (BUTERIN, 2015a; ETHEREUM, 2021). The platform's focus takes into consideration some issues, e.g., development time, security and the ability of different applications to interact efficiently. The technology has several usage licenses (e.g., GPL3, MIT, and LGPL), which allow changes to its code as well as anyone writing smart contracts and building decentralized applications, allowing them to create custom arbitrary rules (BUTERIN, 2015b). With the development of Ethereum, it was possible to verify the various possibilities of blockchains, which until the creation of Ethereum had a main focus on the financial sector (e.g., cryptocurrencies).

3.5.1 Ethereum accounts

The Ethereum platform is composed of objects, the accounts, which have state transitions and an address, being direct transfers of data between objects. An Ethereum account has four fields (BUTERIN, 2015a):

- *Nonce*: Counter used to ensure that each of these transactions can be processed only once;
- *Ether Base*: Current account balance;
- Contract code, if any; and
- Account Storage.

Regarding the funding of mining and monetization in public blockchain cases, it uses *Ether* (ETHEREUM, 2021). Typically, Ethereum has two types of accounts, (i) contract accounts which are controlled by contract code and (ii) external property accounts which are controlled by private key. These accounts carry out the process of communication, message exchange and transactions.

3.5.2 Messages and Transactions

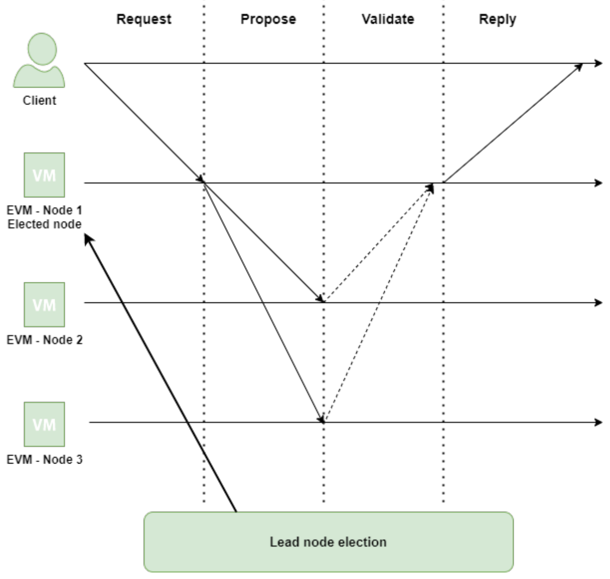
The term transactions, in Ethereum, is used to refer to the signed data packets that store a message to be sent to an account. These transactions have: recipient, sender's identification signature, amount of *Ether*, Data (optional), *STARTGAS* and *GASPRICE*, the latter being the representation of the operational cost of the transactions (BUTERIN, 2015a; ETHEREUM, 2021).

The transaction fields *STARTGAS* and *GASPRICE* are crucial to avoid denial of service for Ethereum (BUTERIN, 2015a) services. The application of these fields avoids the occurrence of infinite *loops*, or the sending of several simultaneous transactions with the same information, changing only a few things, avoiding that computational power is wasted.

3.5.3 Blocks

Ethereum platform blocks contain lists of transactions and the latest status. In addition, it has two other values, the block number and its difficulty, all this information stored in the block. Regarding the functioning of the block validation algorithms, they vary according to the defined consensus. In the case of Ethereum, by default, it has three consensus algorithms: PoW, PoA and PoS, the latter in testing processes (ETHEREUM, 2021). However, the platform allows new consensus mechanisms to be implemented (e.g., RAFT, iBFT). The architectures of the RAFT consensus mechanisms are illustrated (Figure 15), iBFT (Figure 16) e PoA (Figure 17).

Figure 15 – Architecture RAFT.

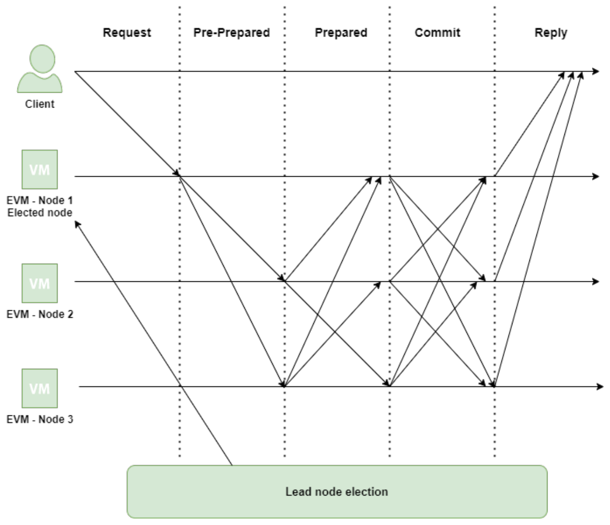


Source: Author.

In Figure 15 it is possible to observe the generic working diagram of the RAFT consensus mechanism in the test environment. In this consensus algorithm, a leader is elected for certain rounds, through a voting process, this leader node is responsible for receiving the transactions, sending them to their peers for the validation process, receiving the response from this validation and ending by inserting the new block in the chain.

Figure 16 illustrates the working diagram of the iBFT consensus algorithm in the test environment. The first step of this consensus algorithm is the election of a leader node for certain rounds, from a voting process, this node is responsible for receiving all transactions and forwarding them to the validation process, after this process all participating nodes , including the leader, perform the validation and everyone returns to all their respective answers, to enter into a consensus and return with the validation and insertion of the next block.

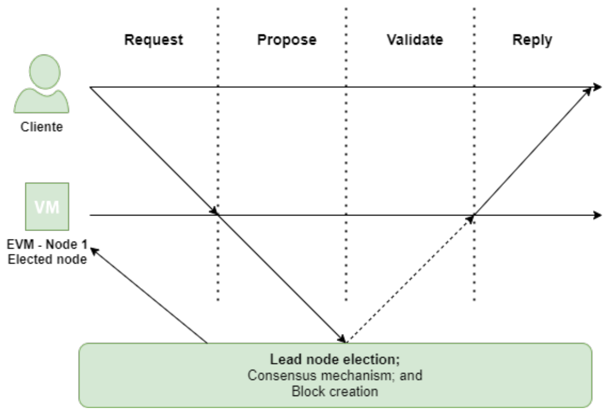
Figure 16 – Architecture iBFT.



Source: Author.

From Figure 17, the architecture of operation of the consensus algorithm PoA in the test environment is illustrated. The first step of this consensus mechanism is the presentation of the identities by the pre-selected authorities, after the presentation, the election for the selection of the leader node is carried out. The leader node is responsible for receiving the requests, and after receiving the requests it sends it to all validating nodes, these nodes validate the requests and return the response to the leader node, which is responsible for inserting the new block into the blockchain chain.

Figure 17 – Architecture PoA.



Source: Author.

3.6 CHAPTER CONSIDERATIONS

In this chapter, the requirements for carrying out the development of the experimentation environment to carry out analyzes regarding the security and performance of private blockchain networks regarding DoS attacks in a IaaS cloud are defined. Together, related works were identified, which did not have the same purpose as this research, but which referenced the criteria selected for carrying out the analysis. Finally, the proposal of this work is defined with the application in three possible scenarios that differ according to the consensus mechanism applied with the Ethereum platform and architecture of the test environment to carry out two experiments in each of these scenarios that use an OpenStack computing cloud.

4 EXPERIMENTS & ANALYSIS OF RESULTS

In Chapter 3 the functional and non-functional requirements are listed, the proposal with its general and specific objectives, and finally the test environments were defined. In the test environment, three scenarios were illustrated, whose main difference is the different consensus mechanisms. From this test environment, the experiments that are presented in this chapter are carried out. The chapter is divided into four sections, Section 4.1 reports the results of Scenario I with RAFT consensus mechanism, Section 4.2 presents Scenario II applying the consensus mechanism acIBFT and Section 4.3 discuss the results of Scenario III with the PoA consensus mechanism. Section 4.4 contains an analysis of the experiments correlating all the presented scenarios.

4.1 SCENARIO I - RAFT

Scenario I is based on the application of the private blockchain with the Ethereum platform, it is composed of six VMs with SO GNU/Linux Ubuntu 20.04 LTS. In this scenario, the RAFT consensus algorithm is used, which is an CFT algorithm. CFT algorithms have a certain degree of resilience in their protocol, allowing the system to correctly reach consensus even with failures in its components, but it does not tolerate Byzantine failures, which are malicious components in the network. In summary, RAFT is a consensus algorithm that is equivalent to the *Paxos* algorithm, in terms of failures and performance, but in a more understandable and easy-to-apply way. As it is an CFT algorithm, it has an unreliable, asynchronous communication, does not tolerate Byzantine faults and has the need for all network nodes to be known (ONGARO; OUSTERHOUT, 2014a). Listing 1 illustrates the block Genesis uses, which has some of these characteristics.

Listing 1 Information about the Genesis RAFT block.

```
 1: {
 2: "alloc": {
    "fe3b557e8fb62b89f4916b721be55ceb828dbd73": {
    "privateKey": "8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63",
    "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
    "balance": "0xad78ebc5ac6200000"
    },
 3: "627306090abaB3A6e1400e9345bC60c78a8BEf57": {
    "privateKey": "c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",
    "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
    "balance": "90000000000000000000000"
    },
 4: "f17f52151EbEF6C7334FAD080c5704D77216b732": {
    "privateKey": "ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f",
    "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
    "balance": "90000000000000000000000"
    }
    },
 5: "coinbase": "0x0000000000000000000000000000000000000000",
 6: "config": {
    "homesteadBlock": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "chainId": 10,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip158Block": 0,
    "maxCodeSizeConfig": [
    {
    "block": 0,
    "size": 35
    }
    ],
    "isQuorum": true
    },
 7: "difficulty": "0x0",
 8: "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
 9: "gasLimit": "0xE0000000",
10: "mixhash": "0x00000000000000000000000000000000000000647572616c65787365646c6578",
11: "nonce": "0x0",
12: "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
13: "timestamp": "0x00"
14: }
```

An interesting feature of the Genesis block, illustrated in Listing 1, is line two that presents the allocation of some nodes. According to the characteristics of the consensus mechanism, there must be at least two predefined nodes, its consensus system supports up to **N/2** of these faults, which is the minimum tolerated for the system to function properly.

It is possible to understand the architecture of the blockchain network, from Listing 1, which allows continuous auditing of the entire process performed by the blockchain. Regarding system investigations, it is necessary to evaluate its performance by exploiting or carrying out attacks, presented in Subsection 4.1.1.

4.1.1 Denial of Service Attack

An DoS is intended to make a system, a machine or a network inaccessible to users who are interested in using it. For successful DoS it is necessary to carry out flood attacks targeting traffic or sending information that exploits the target's vulnerabilities. In either case, DoS deprives legitimate users of the service or resource they expect.

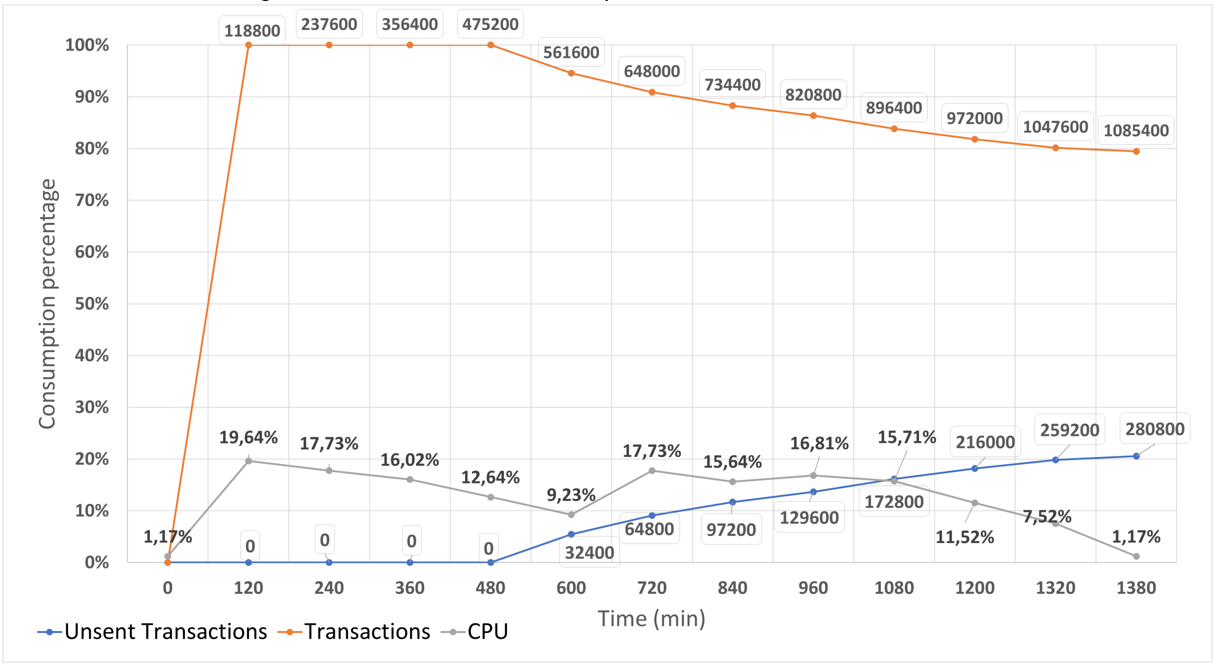
For this work proposal, presented in Section 3.3, it is necessary to carry out two experiments, the first experiment a flood attack with blockchain transactions and the second experiment exploiting vulnerabilities or flooding other ports of a VM. In order to carry out both experiments, it is necessary to identify the services and ports used by the application and the system, in order to choose the best attacks.

Some information is relevant, the first of which is that the Ethereum platform does not have any administrator node and the consensus algorithm selects the leader node through rounds. Another relevant information, that the open ports, are only ports that used by the application and the ssh port, thus reducing the possible vulnerability interfaces, leaving the application optimized and safe. From this information and analysis, the following attacks were selected:Transaction Flood and SSH Flood.

The attacks were initiated from the second experiment, using the SSH Flood, being carried out from the VM-5 and VM-6 instances addressed to VM-1, instances illustrated in Figure 11. During the entire process of the attack, the blockchain network was constantly monitored, all information exchanges were monitored and transactions were carried out constantly, to identify possible instabilities in the network. The attacks were carried out for a period of 180 minutes and throughout this process there were no changes in processing, memory or any instability that could compromise the blockchain network in any way. It is important to note that, as it is a private and controlled network, it is not possible to apply DDoS, making this result already expected.

Regarding the first experiment, the Transaction Flood attack was applied using the API Web3. The Experiment had the participation of three VMs as users/clients, who were responsible for the flood of transactions. During the entire process of this flood attack, the blockchain network was under constant monitoring. Figure 18 and Table 6 illustrate the consolidation of processing data during the attack period.

Figure 18 – Processor Consumption RAFT - *Transaction Flood*.



Source: Author.

Table 6 – Processor Consumption RAFT - *Transaction Flood*.

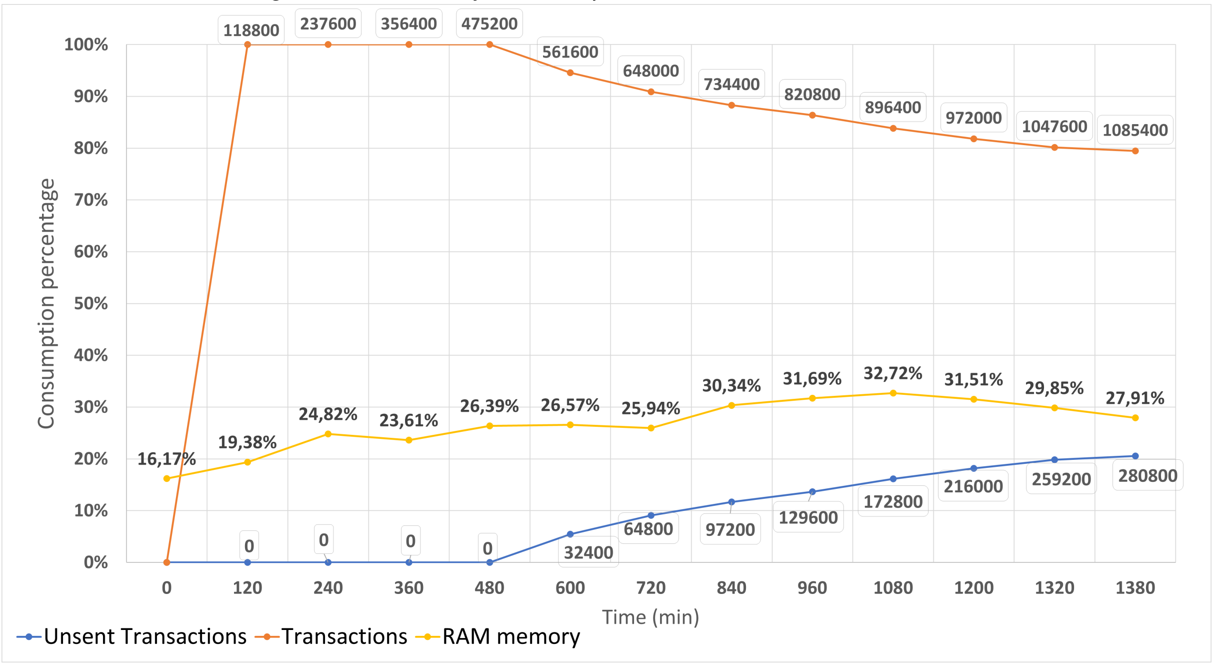
Time/min	0	120	480	600	720	960	1080	1200	1320	1380
Transaction	0%	100%	100%	94,55%	90,91%	86,36%	83,84%	81,82%	80,16%	79,45%
Unsent Transaction	0%	0%	0%	5,45%	9,09%	13,64%	16,16%	18,18%	19,83%	20,55%
Transaction/min	990/min	990/min	990/min	720/min	720/min	720/min	603/min	603/min	603/min	603/min
CPU	1,17%	19,64%	12,64%	9,23%	17,73%	16,81%	15,71%	11,52%	7,52%	1,17%

Source: Author.

From Figure 18 and Table 6 it is possible to observe three main metrics, the first represents the number of transactions carried out and lost by the system, the second represents the number of transactions that are performed per minute and the third represents the system processor consumption during the execution of the Transaction Flood. Regarding the consumption of system processing, the efficiency of the RAFT algorithm is identified as it does not require much computational power, since during the entire execution of the attack the processor remained stable with few changes in consumption during the entire period of the attack. In relation to transactions, it is shown in Table 6, that after 480 minutes of execution of the environment, drops in relation to the response time of the services began. It is important to verify that in the first 480 minutes of the attack, the system received and returned a total of 990 transactions per minute and after this period they were gradually reduced to 720 and 603 transactions per minute.

In Figure 19 the memory consumption during the entire attack period is observed. In the data of Table 7 it is possible to analyze, as a double check, the efficiency of the RAFT algorithm that does not need computational power. In relation to memory consumption, the main peaks of the occurrence of the reduction of the amount of sent transactions are clearer, since the memory consumption in these peaks is reduced.

Figure 19 – Memory Consumption RAFT - *Transaction Flood*.



Source: Author.

Table 7 – Memory Consumption RAFT - *Transaction Flood*.

Time/min	0	120	480	600	720	960	1080	1200	1320	1380
Transactions	0%	100%	100%	94,55%	90,91%	86,36%	83,84%	81,82%	80,16%	79,45%
Unsent Transactions	0%	0%	0%	5,45%	9,09%	13,64%	16,16%	18,18%	19,83%	20,55%
Transaction/min	990/min	990/min	990/min	720/min	720/min	720/min	603/min	603/min	603/min	603/min
Memory RAM	16,17%	19,38%	26,39%	26,57%	25,94%	31,69%	32,72%	31,51%	29,85%	27,91%

Source: Author.

From the results presented in this scenario, the implications of the attack on the performance of the blockchain network are observed. It can be said that the objective of the DoS attack was efficiently achieved. After all, with only three customers carrying out transactions, it is possible to identify in a remarkable way, initially with 990 transactions per minute to 603 transactions per minute, the reduction in the rate of transactions per minute, implying a reduction in the size of the bandwidth. However, as a positive point of these two experiments, it is important to indicate how efficient a blockchain network is in relation to its attack surface and that trust between nodes is essential for the proper functioning of the network. In this way, the need to monitor the network flow during the entire execution process of the blockchain network with the RAFT consensus algorithm becomes essential, in order to avoid or minimize the occurrence of this type of non-malicious attack.

4.2 SCENARIO II - IBFT

Scenario II consists of six VMs with SO GNU/Linux Ubuntu 20.04 LTS, all instances properly aimed at the application of the Ethereum private blockchain and with the API Web3. In this scenario, the consensus algorithm applied to the Ethereum platform is iBFT, which is an algorithm inspired by Castro e Liskov (1999), and is an algorithm BFT. Algorithms BFT, unlike CFT, are more complex and secure algorithms, as in addition to guaranteeing consensus with the failure of network components, they can be applied to hostile systems, with the presence of malicious users . It is important to note that the algorithm iBFT, PoS, pBFT and *et al.* are algorithms BFT, however, they are not completely tolerant to Byzantine faults, mainly compared to PoW, which compensates for its efficiency in computational and energy terms.

Regarding the algorithm iBFT, in addition to having inspirations in pBFT, it also has inspirations in *Clique* and its consensus mechanisms PoA and PoW. Having as main benefits (MONIZ, 2020):

- Immediate block purpose: There is a proposed block for a certain size of the chain, avoiding the occurrence of forks and that transactions can be "undone" once the chain period has passed.
- Reduced time between blocks: The effort required to validate and build a block is significantly reduced, especially compared to PoW, increasing the throughput of the chain.
- Integrity and fault tolerance: This uses a group of known validators, requiring a majority of 66% to insert a new block, avoiding possible fraud. It uses the means of group leadership, the same used in PoA, preventing a malicious node from influencing for a long term.
- Operational Flexibility: Allowing the pool of validators to be modified over time, ensuring that only trusted nodes can participate.

Listing 2, illustrates the Genesis block of the blockchain. The first block of the blockchain network has information about the functioning of the blockchain network and its main participating nodes.

Listing 2 Information about the Genesis IBFT block.

```
 1: {
 2: "config": {
    "chainId": 10,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "istanbul": {
    "epoch": 30000,
    "policy": 0,
    "ceil2Nby3Block": 0
    },
    "txnSizeLimit": 64,
    "maxCodeSize": 0,
    "qip714Block": 0,
    "isMPS": false,
    "isQuorum": true
    },
 3: "nonce": "0x0",
 4: "timestamp": "0x6184131b",
 5: "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000
    f85ad5943091fc1144211b032520b5bd325e9f1825bfdd13b841000000000000000000000000000000
    000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000c0",
 6: "gasLimit": "0xe0000000",
 7: "difficulty": "0x1",
 8: "mixHash": "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
 9: "coinbase": "0x0000000000000000000000000000000000000000",
10:  "alloc": {
    "3091fc1144211b032520b5bd325e9f1825bfdd13": {
    "balance": "0x446c3b15f9926687d2c40534fdb564000000000000"
    },
11:  "09cc465234a231d564d171f70f7afc2507a808a4": {
    "balance": "0x446c3b15f9926687d2c40534fdb564000000000000"
    },
12:  "b66cf6fba976fe2172fa4f243b4953d8ca51e863": {
    "balance": "0x446c3b15f9926687d2c40534fdb564000000000000"
    },
13:  "35f5f71b95f1a305eb2791624c876976ba2946a2": {
    "balance": "0x446c3b15f9926687d2c40534fdb564000000000000"
    },
14:  "8de65dc62606bf3f232bde48dc9765163330ee3d": {
    "balance": "0x446c3b15f9926687d2c40534fdb564000000000000"
    },
15:  "e849e1364a5b7eb4919c2784dfccc5be66aceaf6": {
    "balance": "0x446c3b15f9926687d2c40534fdb564000000000000"
    }
    },
16:  "number": "0x0",
17:  "gasUsed": "0x0",
18:  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
19: }
```

In Listing 2, some of these features are presented. The first relevant feature is the need to have at least four predefined nodes to initialize the Genesis block. The system allows the proper functioning to occur keeping **N = 3F + 1** which is the minimum tolerated, represented between lines 10 to 15.

With this information presented in Listing 2, it is possible to understand and verify the state and architecture of the blockchain network, allowing continuous auditing of the entire process carried out by the chain. Regarding system investigations, it is necessary to evaluate its performance by carrying out attacks or exploiting a vulnerability, carried out in Subsection 4.2.1.

4.2.1 Denial of Service Attack

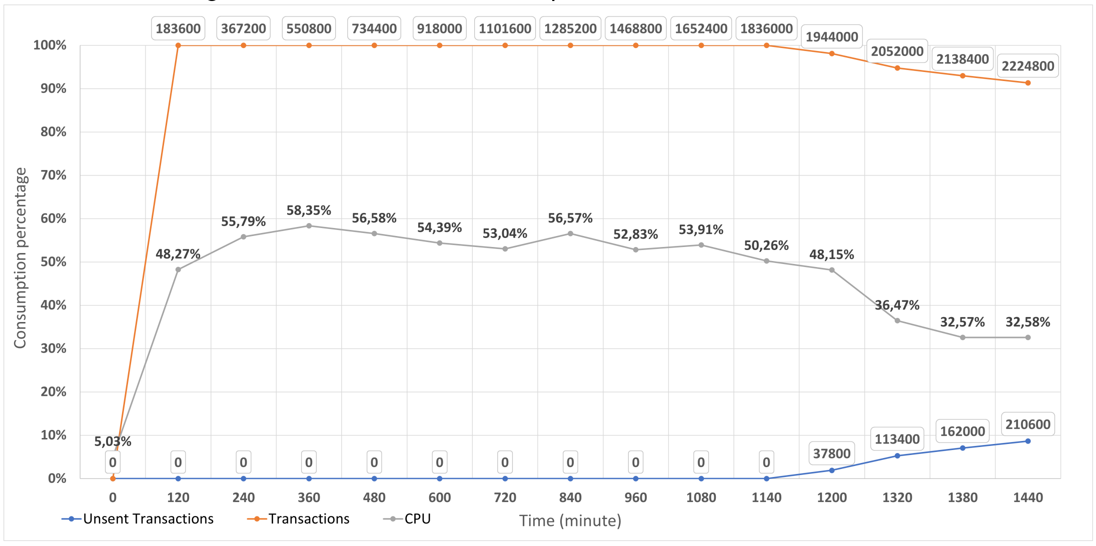
Following the work proposal (Section 3.3), for Scenario II it is necessary to carry out two experiments, the first through a flooding attack with blockchain transactions and the second experiment exploring vulnerabilities or flooding open ports of a VM. With the experiments determined, it is necessary to identify the information about the application to identify the best way of exploration.

For this experiment some information is relevant, previously mentioned the Ethereum blockchain does not have any centralizing node, applying algorithm for selecting a new leader after a certain period. Another relevant information is that it is an algorithm BFT which is more resistant to possible attacks DoS having a more stable behavior. After presenting this information, and using the exploration of communication protocols, the following attacks were selected: Transaction Flood and SSH Flood.

For the second experiment, the SSH Flood attack was selected, being carried out from VM-5 and VM-6 to VM-1. During the entire execution of the attack, monitoring of the blockchain network, processing and memory of the instance was carried out, in order to identify possible oscillations in the network. The attack ran for around 300 minutes. During this entire process, no system changes were detected, either in processing or memory, and no instability was identified in the blockchain network, as the attack packets were quickly discarded. It is important to highlight that, as it is an internal and controlled environment, it is not possible to execute a DDoS on a large scale, making the result expected.

In the first experiment, the Transaction Flood attack was performed using the API Web3 by the clients. In this experiment, three clients were used, responsible for carrying out the attack. During the entire period of execution of the attack, constant monitoring of the blockchain, network, processor and its memory was carried out. From Figure 14 and Table 8, the data collected on processing consumption during the period of the attack is illustrated.

Figure 20 – Processor Consumption IBFT - *Transaction Flood*.



Source: Author.

Table 8 – Processor Consumption IBFT - *Transaction Flood*.

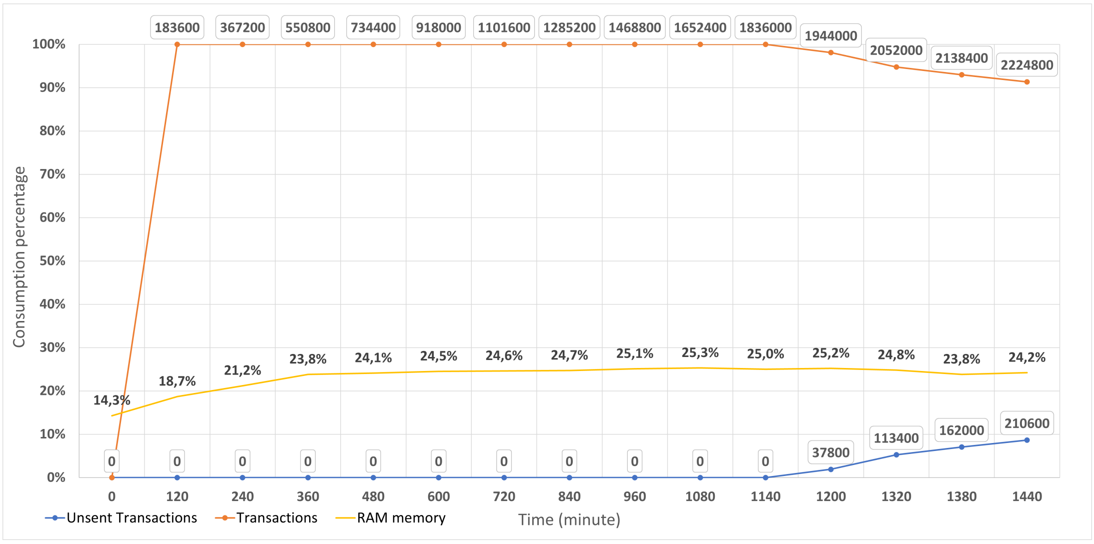
Time/min	0	360	720	1080	1140	1200	1320	1380	1440
Transactions	0%	100%	100%	100%	100%	98,09%	94,76%	92,96%	91,35%
Unsent Transactions	0%	0%	0%	0%	0%	1,91%	5,25%	7,04%	8,65%
Transactions/min	1530/min	1530/min	1530/min	1530/min	1530/min	900/min	900/min	720/min	720/min
CPU	5,03%	58,35%	53,04%	53,91%	50,26%	48,15%	48,15%	32,57%	32,58%

Source: Author.

From Figure 20 some points are highlighted. The first is related to the transactions that were performed and lost by the system, the second point is the consumption of the processor during the attack. Regarding the use of processing, it is important to note the difference between a consensus algorithm CFT compared to BFT, due to issues of greater complexity and security, the use of computational power is greater. However, it is observed that until time 18 there was great stability on the part of the processing usage, but as the lost transaction rate grew, the processor usage was reduced. In a consensus algorithm BFT, the reduction of processing, below the usual, means the reduction of the processing speed of the transactions, allowing the gradual increase of the queue of the communication process.

Regarding transactions, observed mainly in the data in Table 8, the existing stability of the consensus algorithm iBFT against the Transaction Flood attack is observed. In the data presented, we observed that only after 1200 minutes of the attack, they began to show transaction losses. Another relevant information is that while the processing remained stable at an average rate above 50%, the transaction process went well, but with the gradual decline and in a short time, there was a reduction in the sending of transactions in parallel.

Figure 21 – Memory Consumption IBFT - *Transaction Flood*.



Source: Author.

Table 9 – Memory Consumption IBFT - *Transaction Flood*.

Time/min	0	360	720	1080	1140	1200	1320	1380	1440
Transactions	0%	100%	100%	100%	100%	98,09%	94,76%	92,96%	91,35%
Unsent Transaction	0%	0%	0%	0%	0%	1,91%	5,25%	7,04%	8,65%
Transactions/min	1530/min	1530/min	1530/min	1530/min	1530/min	900/min	900/min	720/min	720/min
Memory	14,30%	23,80%	24,60%	25,3%	25,00%	25,20%	24,80%	23,80%	24,20%

Source: Author.

In Figure 21 and Table 9 it is possible to observe the memory consumption during the attack period. From these data it is possible to perceive the efficiency of this algorithm BFT in comparison with other algorithms (e.g., PoW) reaffirming the non-need of a significant computational power to obtain a consensus that is tolerant and safe. In relation to memory, it proved to be stable throughout the attack period, with no change.

From the results obtained in the experiments through Scenario II, the implications of a DoS in the Blockchain network are observed. It is evident from the results that the objective of the Transaction Flood attack was efficiently achieved. Tables 8 and Table 9 indicate that after 1140 minutes there were considerable losses in the rate of transactions per minute, which initially were at 1530 transactions per minute, reaching 720 transactions per minute. However, the main positive point of this scenario is the efficiency presented by an algorithm BFT with low use of computational power and with guarantees compatible with other more complex consensus algorithms and with greater use of computational power. Another important point is that this consensus algorithm was applied in a private network, with a reduced attack interface, not being recommended for public models. In this way, it becomes necessary as a good security practice, the constant monitoring of the traffic received by the blockchain network, in order to prevent malicious or non-malicious clients from causing damage to the blockchain network.

4.3 SCENARIO III - POA

Scenario III consists of the application of six VMs with SO GNU/Linux Ubuntu 20.04 LTS, all instances contained the Ethereum blockchain application and API Web3. The PoA consensus mechanism is applied to the Ethereum platform, but with a difference compared to Experiments 1 and 2, this mechanism is implemented as standard in Ethereum, being part of core business for private blockchain networks Ethereum. PoA is an BFT consensus algorithm, in which network validators are not asked to solve difficult mathematical problems, but rather stake their reputation.

This consensus mechanism is not recommended for use in public blockchain models, as there are requirements that there is a minimum trust between peers that are pre-selected. However, the algorithm presents an entire process that is safe, reliable and highly scalable, not requiring the use of high computational power, requiring as a safety factor: confirmation of the real identity of the validator, mathematical tests with difficulty to guarantee the compromise of the system and a fair, standardized system for validators, ensuring that all nodes go through the same process. The Listing 3, presents some characteristics mentioned about PoA.

Listing 3 Genesis PoA block information.

```
 1: {
 2: "config": {
    "chainId": 1515,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "clique": {
    "period": 10,
    "epoch": 30000
    }
    },
 3: "nonce": "0x0",
 4: "timestamp": "0x61a84c31",
 5: "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000
    1709ed747eb5c1b77d590fe92e7f81bbe67bbf5f28970eba44ba8a1da0738da5dc04cb9120506ae3436
    7f93383fd7e73e7d9b5996a3ed6b84cf4dffbe80e684202599254421924 6821cb96c9eab8e4bf
    000000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000",
 6: "gasLimit": "0x47b760",
 7: "difficulty": "0x1",
 8: "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
 9: "coinbase": "0x0000000000000000000000000000000000000000",
10:  "alloc": {
    "1709ed747eb5c1b77d590fe92e7f81bbe67bbf5f": {
    "balance": "0x200000000000000000000000000000000000000000000000000000000000000"
    },
    "28970eba44ba8a1da0738da5dc04cb9120506ae3": {
    "balance": "0x200000000000000000000000000000000000000000000000000000000000000"
    },
    "4367f93383fd7e73e7d9b5996a3ed6b84cf4dffb": {
    "balance": "0x200000000000000000000000000000000000000000000000000000000000000"
    },
    "e80e684202599254421924 6821cb96c9eab8e4bf": {
    "balance": "0x200000000000000000000000000000000000000000000000000000000000000"
    }
11: },
12: "number": "0x0",
13: "gasUsed": "0x0",
14: "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
15: "baseFeePerGas": null
16: }
```

From Listing 3 it is possible to make some observations. The first one is on line two in "clique", which are the definitions used by PoA for the validation system, which aim to ensure that there is no "selfish validation" or that only one node is responsible for the validation, ensuring that if this node has vested interests, it is not kept in your domain. The second important configuration occurs on line ten that presents the sealing nodes, in the case of PoA a minimum of three sealers are required for the system to function properly. With this information, Listing 4, presents the relevant information about a transaction that was performed.

Listing 4 PoA Transaction Information.

```
eth.getTransaction ("0x8a21461f7b3b8a56e83eea191a0e2873c6c445ac1bb2d9afcd7b4d901375f4e7" )
{
blockHash: "0x568c439edecb22220118e71d74c6f7e3b4530f406d1006e7ff65b0232273a2f9",
blockNumber: 214,
from: "0x4367f93383fd7e73e7d9b5996a3ed6b84cf4dffb",
gas: 100000,
gasPrice: 10000,
hash: "0x8a21461f7b3b8a56e83eea191a0e2873c6c445ac1bb2d9afcd7b4d901375f4e7",
input: "0x",
nonce: 20,
r: "0x29a5e2ce3cabb05791e3eb78c8218da463e57fb5bf461bf5a6e33e53cc5f8d41",
s: "0x456a3723dd985b8021f72cdf6bcd20e1ad2463b1c991854d941e7d4c63b0d852",
to: "0x4367f93383fd7e73e7d9b5996a3ed6b84cf4dffb",
transactionIndex: 19,
type: "0x0",
v: "0xbf9",
value: 1
},
```

In Listing 4 it is possible to verify one of the transactions carried out from VM-3 to VM-4. The information presented in this transaction makes us understand the idea of an accounting ledger, allowing the verification of transactions and the block in which it is inserted. It is important to note that in the PoA consensus algorithm, from the portfolio information, it is possible to obtain more information about the other participating nodes, which is one of the problems alleged by many that does not guarantee the anonymity of the participants.

With the information presented in the Listings 3 and 4, it is possible to understand and find out about the state and architecture of the blockchain network. Allowing continuous auditing of the entire approved process inserted in the chain, through the consensus mechanism. That said, in relation to the system, it becomes necessary to evaluate its performance from carrying out attacks on its network or exploiting its vulnerabilities, which is carried out in Subsection 4.3.1.

4.3.1 Denial of Service Attack

Continuing the proposal of this work for Scenario III, two experiments are also carried out. The first experiment by applying a blockchain transaction flooding attack and the second experiment by exploiting possible platform vulnerabilities. Thus, it becomes necessary to analyze application characteristics to identify the best ways of exploitation.

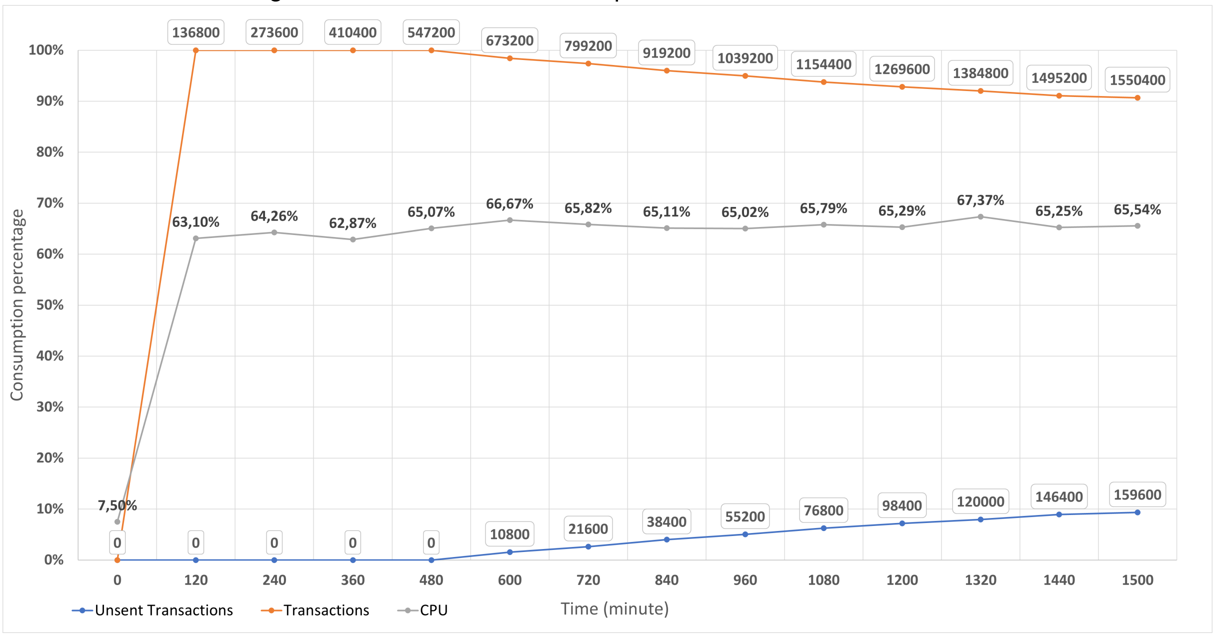
Some information is relevant for carrying out this experiment, the first of which is that Ethereum does not have a centralized node, applying in its consensus algorithm (Listing 3) conditions that exist at least two or three different nodes on the network. This consensus mechanism is also a BFT being more resistant by nature, especially against DoS attacks or malicious miners. From this information, the attacks Transaction

Flood and SSH Flood were selected for application.

For the second experiment, the SSH Flood attack is applied, being carried out from VM-3 and VM-4 towards VM-1. During the entire attack process, the scenario was monitored in general, with the aim of identifying possible fluctuations in the blockchain network or the resources of the instances. During the period of execution of the attack, the monitoring did not present any anomaly or alteration of the system, either in relation to the application or the computational resources. It is important to point out that indoor environments are usually more controlled, consequently these attacks are not so efficient.

Regarding the first experiment, the Transaction Flood is executed using the Web3 API by the clients. In the experiment, three clients were used, responsible for the flood of transactions. During the entire process of executing the attack, there was monitoring of the environment, the blockchain network and the resources of the instances. Figure 22 and Table 10 show the data that was collected from processor consumption during the execution of the attack.

Figure 22 – Processor Consumption PoA - *Transaction Flood*.



Source: Author.

Table 10 – Processo Consumption PoA - *Transaction Flood*.

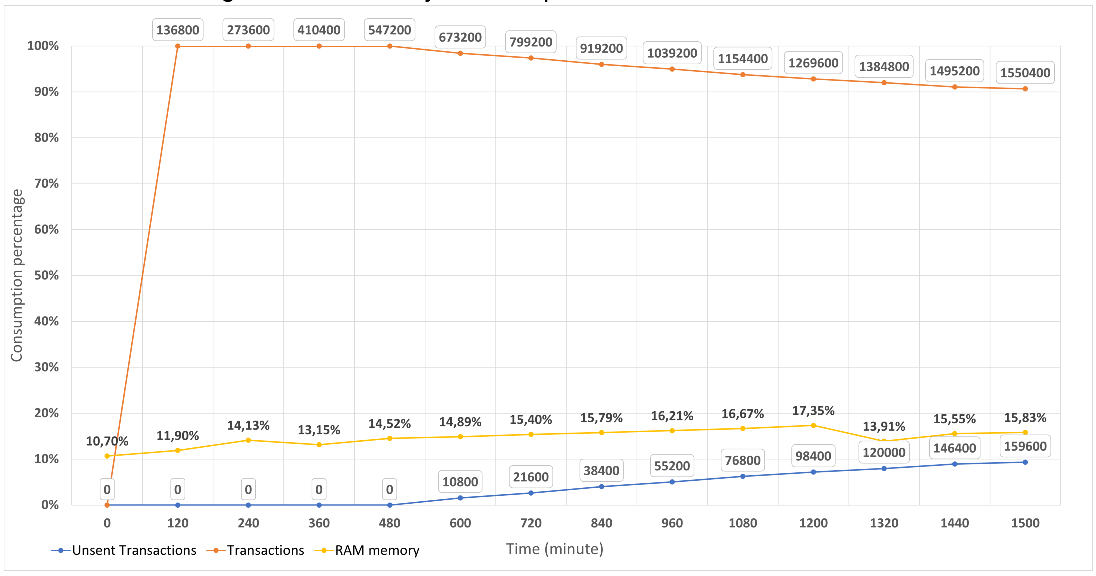
Time/min	0	360	600	840	1080	1200	1320	1440
Transactions	0%	100%	98,42%	95,99%	93,76%	92,81%	92,02%	91,08%
Unsent Transactions	0%	0%	1,57%	4,01%	6,24%	7,19%	7,97%	8,92%
Transactions/min	1140/min	1140/min	1050/min	1000/min	960/min	960/min	960/min	920/min
CPU	7,50%	62,87%	66,67%	65,11%	65,79%	65.29%	67,37%	65,25%

Source: Author.

In Figure 22 some points are highlighted, the first one is in relation to the consumption of processing during the attack, the second point is in relation to the transactions sent and those lost during this process. Regarding the processing consumption during the attack, the first perception that is observed is that it is an algorithm BFT, since its consumption is high due to the mathematical calculations that require greater computational power, but which remained stable throughout the attack execution process. Another issue observed is that after 600 minutes of the attack, transaction losses began, however, there was no change in processing consumption, which represents that the amount of transactions that were being lost did not impact the system as much, the keeping safe and stable.

Regarding transactions, it is observed mainly in Table 10 the stability of the consensus algorithm PoA against a Transaction Flood attack. In the data presented, it is observed that after 600 minutes of execution, data losses began, but in a much more contained way compared to Scenarios I and II. Presenting that, even though transaction loss rates were presented early on, the environment remained stable and with gradual but not alarming reductions.

Figure 23 – Memory Consumption PoA - *Transaction Flood*.



Source: Author.

Table 11 – Memory Consumption PoA - *Transaction Flood*.

Time/min	0	360	600	840	1080	1200	1320	1440
Transactions	0%	100%	98,42%	95,99%	93,76%	92,81%	92,02%	91,08%
Unsent Transaction	0%	0%	1,57%	4,01%	6,24%	7,19%	7,97%	8,92%
Transactions/min	1140/min	1140/min	1050/min	1000/min	960/min	960/min	960/min	920/min
CPU	10,70%	13,15%	14,89%	15,79%	26,67%	17.35%	13,91%	15,55%

Source: Author.

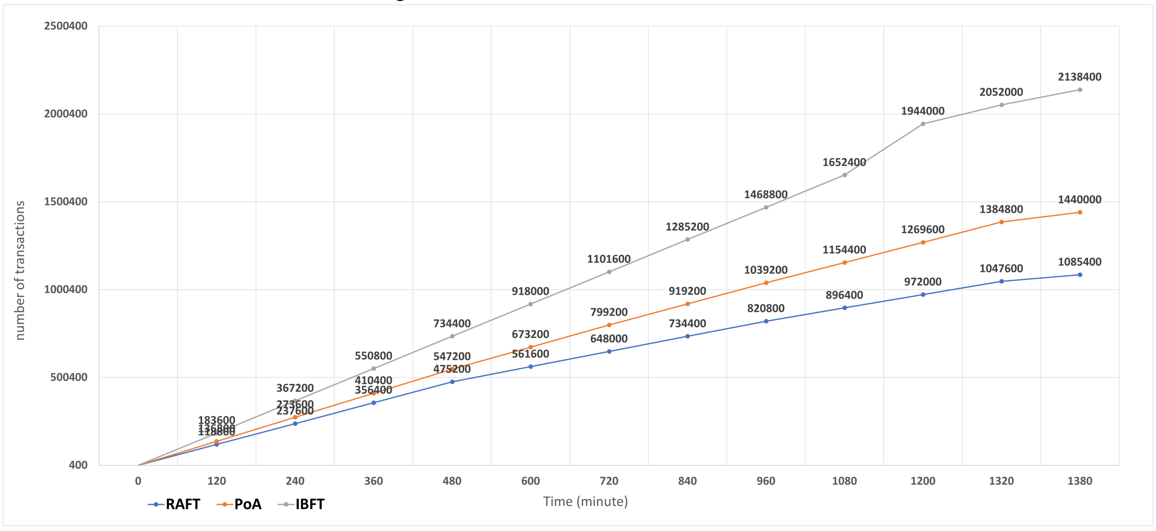
Table 11 and Figure 23 list memory consumption during the execution of the attack. With these data, it is possible to observe the real performance of the BFT algorithm, which does not require significant computing power to work optimally and with good performance in this regard. Regarding memory, it remained constant throughout the execution of the attack, without any anomalies.

With the results obtained from the experiments applied in Scenario III, the implications of an DoS in the Blockchain network are observed, which were successful in their attempt. It is evident from the results presented that the objective of Transacation Flood was successfully achieved, however, not efficiently. Tables 11 and 10 show that after 600 minutes the blockchain network presented lost transactions, but unlike Scenario II, which is also a BFT algorithm, the transactions per minute were gradually reduced from 1140, 1050, 1000, 960 and 920, in comparisons the same amount of transactions were lost in both, with only difference in their time. However, the main point in this experiment is related to the efficiency obtained by the algorithm, which, even with all the attacks suffered, remained stable and with few problems being dealt with in the long term. Another important point is that private blockchain networks have a smaller attack interface, reducing the possibilities of attack combinations for the benefit of a user. Indicating that it is necessary as a good security practice, constant monitoring of the blockchain network, so that the problem is mitigated.

4.4 RESULTS ANALYSIS

This section presents the compilation of results obtained through Scenarios I, II and III according to the test plan (Section 3.3). The analyzes are separated by issues related to transactions and issues of consumption of computational resources. It becomes relevant to identify relationships between the mode of operation of the consensus algorithms and the use of resources that are allocated to VM instances. It is important to note that the instance flavor used in the experiments is with 2 vCPUs, 4Gb RAM and a network interface with no speed limitation. In this way, network operations between VMs allocated on the same host can reach up to 23Gb/s throughput, if they were allocated on different hosts they would be limited by the throughput of the network interface physical, ie, 1Gb/s. All VMs used in the experiment were allocated to the same host to ensure that the number of requests was not affected/restricted by the network throughput. Figure 24 illustrates the total number of transactions that were carried out in each of the scenarios.

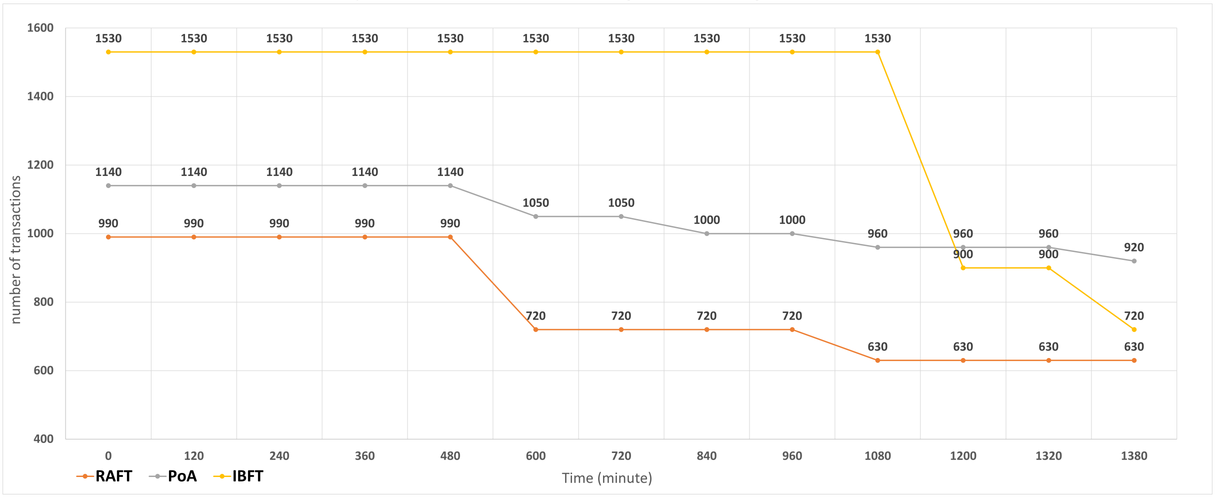
Figure 24 – Total transactions sent.



Source: Author.

In Figure 24 it is possible to observe the comparison of the number of total transactions that were sent per hour during the entire Transaction Flood attack process. Figure 24 details the comparison of Scenario I (RAFT), Scenario II (iBFT) and Scenario III (PoA). In these data it is possible to observe a positive performance of the consensus algorithm iBFT, which is an algorithm BFT, and even with greater consumption of resources, during the process it was the one that best obtained results in terms of quantity of transactions. The consensus algorithm PoA, in relation to the total of transactions, had a good growth, maintaining an intermediate growth. On the other hand, the RAFT consensus algorithm, which is an CFT algorithm, presented the worst results in terms of total sent and executed transactions. To observe in a more granular way, Figure 25 illustrates the behavior of these scenarios per minute.

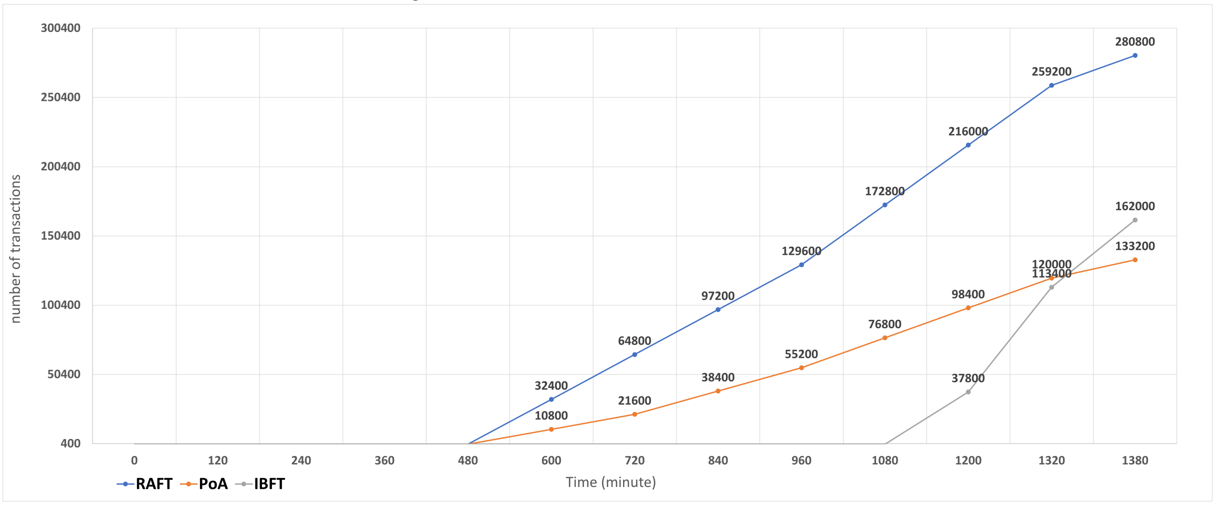
Figure 25 – Transactions performed per minute.



Source: Author.

Figure 25 shows the comparison between all these scenarios in the comparison of sending per minutes, in their given time. Regarding the scenario with the RAFT algorithm, since the execution of Transaction Flood, it had the worst performance in sending transactions per minute, starting with 990 transactions per minute and ending at time 23, with a sending rate of 630 transactions per minute. The consensus algorithm iBFT, from the beginning, revealed, compared to the others, a higher rate of sending transactions, this being 1530 transactions per minute. However, with the loss of processing and its high transaction cache, these numbers started to drop, indicating a greater percentage of decrease in transaction rates per minute, reaching 720 transactions per minute at the end of the 1380 minutes of execution of the attack. On the other hand, the consensus algorithm PoA showed the best behavior in relation to the amount of transactions sent per minute, which started with 1140 transactions, and after 600 minutes of execution of the attack, delays or non-sending of transactions began, but it maintained a good behavior reaching the end of 1380 minutes with a rate of 920 transactions per minute, higher than RAFT and iBFT. For a compilation of the total amount of transactions that were lost, Figure 26 illustrates this scenario.

Figure 26 – Total unsent transactions.



Source: Author.

Table 12 – Total unsent transactions.

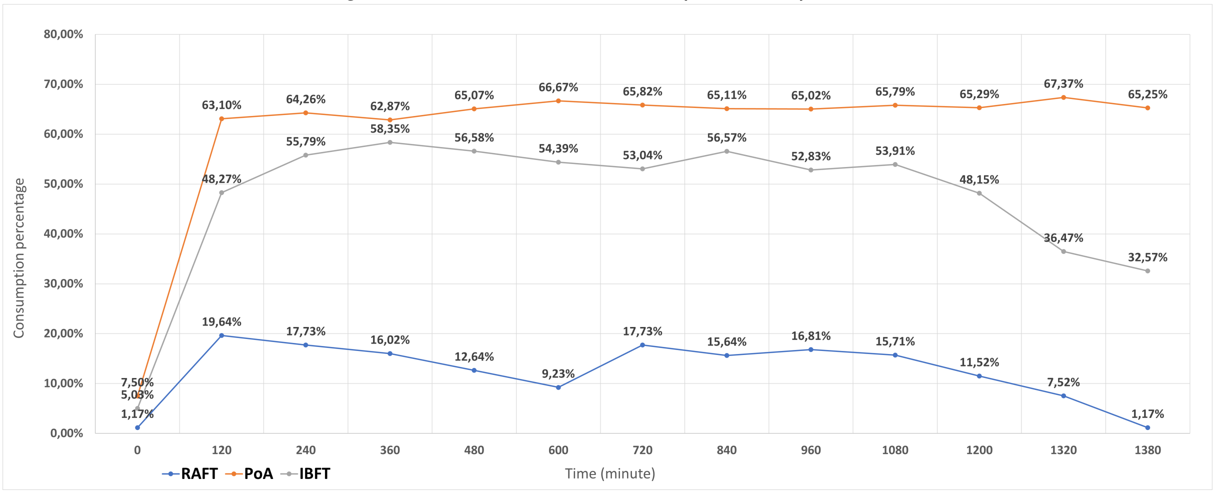
Time(min)	0	360	600	840	1080	1200	1320	1380
RAFT	0%	0%	5,45%	11,68%	16,16%	18,18%	19,83%	20,55%
iBFT	0%	0%	0%	0%	0%	1,91%	5,25%	7,04%
PoA	0%	0%	1,57%	4,01%	6,24%	7,19%	7,97%	8,48%

Source: Author.

The Table 12 presents in a more visible way the number of transactions that were not carried out during the entire attack process. It is observed that the RAFT consensus algorithm actually had the worst performance, as it failed to perform more than 20% of the total transactions at the end of the experiment. On the other hand, the BFT consensus algorithms showed greater consistency in relation to the sending of transactions. However, it is clear that after time 20 of the execution of the attack, the consensus algorithm iBFT no longer had such resistance to the continuity of attacks and that PoA in this regard proved to be more stable and with greater control of the occurrences.

Regarding issues related to the consumption of computational resources, for the best selection of flavor from VMs, an analysis was carried out on the consumption of two resources: (i) processor and (ii) memory. For processing consumption, Figure 27 illustrates the comparison during the Transaction Flood attack. Figure 27 presents a comparison of processing consumption in Scenarios I, II and III.

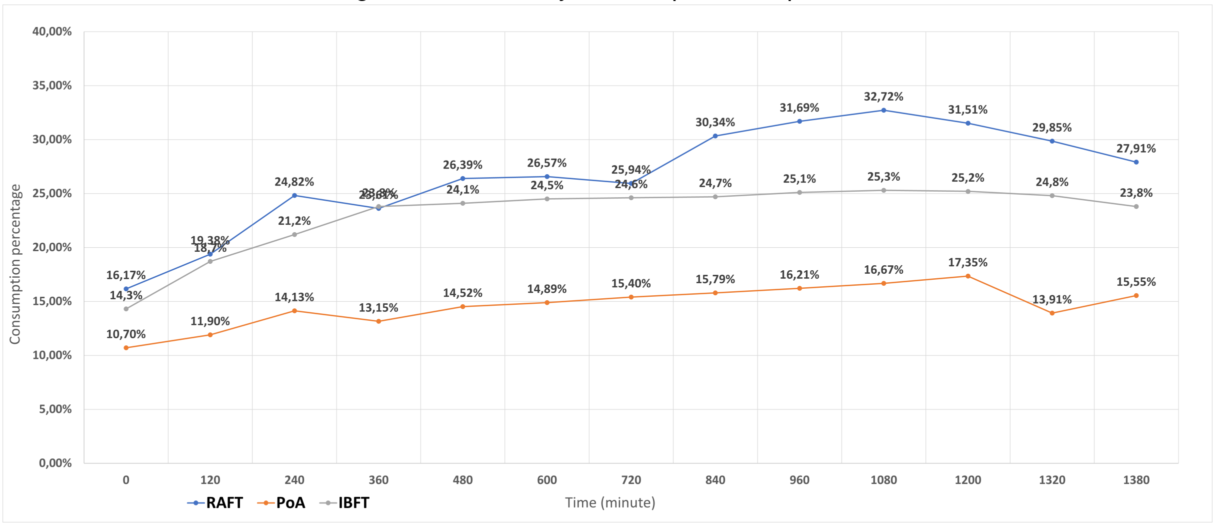
Figure 27 – Processor consumption comparison.



Source: Author.

From Figure 27, it is possible to make some observations; (i) the CFT algorithm has considerably low processing consumption during the validation for the insertion of its blocks; and (ii) algorithms based on BFT, because they are more complex and require greater computational power, make greater use of the processor. With a focus on these two algorithms BFT, it is notable that in contrast with Table 12, the consensus algorithm iBFT remained stable up to 1080 minutes, with an average 50% processing, but from the moment the flood occurred the processing dropped reasonably, representing the final discrepancy that occurred with your transaction rate. On the other hand, the algorithm PoA proved to be stable throughout the monitoring, representing the stability of the system during the occurrence of the flood, and allowing the transaction rates to drop gradually. The relationship between memory consumption and scenarios is illustrated in Figure 28.

Figure 28 – Memory consumption comparison.



Source: Author.

Consensus algorithms that are based on the principles of BFT, in general, do not require a high memory consumption and this can be observed in the algorithms PoA and iBFT. In the case of the algorithms PoA and iBFT, it is possible to verify that during the entire process of monitoring the VMs memory remained stable, without expressive scales of growth or even worrying reductions, which could reach cause problems for the platform. However, in CFT consensus algorithms, in general, the memory is the one that most oscillates during the execution of these algorithms. The RAFT Algorithm, presented these oscillation contrasts during the execution of the Transaction Flood, not compromising in general the good functioning of the instance but demonstrating that as the losses happened the memory consumption increased. Finally, Table 13 presents the opinion of all this information that was collected, being separated between the consensus algorithms, the flavor of the VMs and the number of transactions that can be supported by the instance without catastrophically compromising application functionality.

Table 13 – General aspects of flavor in relation to the consensus algorithms tested.

Consensus	RAFT	iBFT	PoA
Processor	900 Transactions/min 20% de 2 vCPU	1500 Transactions/min 55% de 2 vCPU	1100 Transactions/min 67% de 2 vCPU
Memory	900 Transactions/min 33% de 4GB RAM	1500 Transactions/min 25% de 4GB RAM	1100 Transactions/min 17% de 4GB RAM
Transactions	900 non-continuous transactions; e 700 continuous transactions	1500 non-continuous transactions; e 900 continuous transactions	1000 continuous transactions

Source: Author.

With the results of Table 13, it is important to highlight some issues: (i) when consensus mechanisms BFT are applied, it is necessary to understand the needs of blockchain applications and according to the complexity and amount of transactions occur in parallel with the highest processing consumption, so this analysis of the application is indicated to choose the best flavor; and (ii) CFT algorithms usually require more memory than BFT applications, so it is recommended that this analysis be done so that the *flavor* choice is the best possible. Furthermore, concerns about security and the proper functioning of environments and their applications have become increasingly worrisome, especially with the increasing number of vulnerabilities and attackers. To expose it more comprehensively, Table 14 brings a comparison between consensus mechanisms, resources and transactions at the time DoS attacks started to affect blockchains and at the end of the attack.

Table 14 – Flavor (2 vCPU, 4GB RAM, 1 Gbps network): Consensus mechanisms regarding transactions/resources and DoS.

Metrics	RAFT	iBFT	PoA
DoS start time	600	1200	600
End of DoS	1380	1380	1380
Start DoS Transactions rate	720/min	900/min	1050/min
Transaction rate at the end of DoS	630/min	720/min	920/min
Percentage of processor use at the start of DoS	9,23%	48,15%	66,67%
Percentage of processor use at the end of DoS	7,52%	32,58%	65,54%
Percentage of memory usage at the start of DoS	26,57%	25,2%	14,89%
Percentage of memory usage at the end of DoS	29,86%	24,2%	15,83%

Source: Author.

The Table 14 establishes a relationship between the flavor of the instance and the intensity of DoS attacks, in relation to the computational resources used. The results presented have a comparison from the beginning of the occurrence of DoS attacks to the final monitoring, in relation to the scenarios and their respective consensus mechanisms. Among the data, the first highlight is for the RAFT algorithm which, like PoA, had the beginning of DoS after 600 minutes of execution, revealing a greater fragility in the face of these attacks in relation to the transactions, but not about the use of your resources. The consensus algorithm iBFT presented the best performance in relation to the amount of transactions sent and the time, of 1050 minutes, for the beginning of the occurrences of DoS, but after the beginning of the floods there was a reduction of its processing causing the biggest differential in transaction fees. The PoA presented, in general, the best performance in the relation of the resources of its flavors vs DoS, remaining in balance, mainly in the use of its resources, with the smallest differential of transaction fees. From these assessments, it is important that good security practices in a private blockchain network are applied, with monitoring of network traffic, transactions and processing being carried out on a constant basis. In addition, that all participating users and validators of the network have the minimum trust that is necessary for the proper functioning of the network.

4.5 CHAPTER CONSIDERATIONS

In Chapter 4 the results of the experiments listed by Section 3.3 were described. The results obtained from Experiments 1 and 2 applied in Scenarios I, II and III were presented together with an analysis of the relationship of the scenarios with their consensus mechanisms and the flavors of the VMs. Another issue presented in the analysis was the behavior of blockchain chains when executing an DoS, indicating that a simple attack on private/consortium models can affect the performance and the entire chain process. Finally, it is possible to perceive that the private/consortium models have a much smaller interface of vulnerabilities, but that these vulnerabilities can affect the proper functioning of the chain, and depending on the application's need, a greater study of the number of transactions is necessary before defining it. the instance's flavor.

5 CONSIDERATIONS & FUTURE WORK

Blockchain technology is in constant adoption by organizations, researchers and developers. This fact is due to the characteristics and technologies that are used in blockchains to meet the requirements of the applications in which they are used. The benefits of using a private or consortium blockchain in terms of economy, data management, auditing and security are many, as can be seen in the different applications mentioned and referenced in Subsection 2.3.3. Another important issue is the ease of creating nodes of a blockchain in computing clouds, through the use of virtualization.

In this context, there is a concern about issues related to the security of the application and its development environment. It is important to note that in the related works there are mentions of security needs and concern for the environment, but all of them performed tests with the focus of concern on issues related to the performance of the blockchain application. Making it clear the need to develop analyzes focused on the performance and security of private and consortium blockchains in the event of a possible attack, whether malicious or not, and that criteria for analysis can be identified with the well-defined characteristics of the instances and the environment .

The analysis proposed for this work is based on a work carried out previously (MIERS et al., 2019), but which only involved security issues on Multichain and Ethereum platforms and without concern for the environment and the performance issues of VMs. The current work maintains its concern related to blockchain security issues, including issues of the environment and the application being developed in a computational cloud, allowing it to be possible to perform the evaluation of the performance of VMs, during the occurrence of some device failure or an attack by a malicious user.

Regarding the current work, with the main objective of performing a performance and security analysis of a blockchain network, during an DoS attack, some scopes have already been previously selected. The first definition, throughout the presentation of the fundamentals (Chapter 2), is a computing cloud environment in which each node participating in the platform is an instance of VM all with standardization of SO and same characteristics as flavor. The second definition is in relation to the blockchain model, the private one being chosen, making the blockchain network partially decentralized. Another issue that was defined is in relation to the possible platforms to be applied, the Ethereum platform was chosen, as it is widely spread and allows different consensus mechanisms to be applied in a friendly way. The last definition, and the most important, is in relation to the choice of consensus mechanisms, because there is a wide range of application possibilities, being necessary to understand their objectives in contrast to the application, being chosen the RAFT consensus mechanisms, iBFT and PoA, being consensus mechanisms BFT and CFT.

Based on these definitions, in relation to the environment, two experiments were carried out, defined in Section 3.3. The analyzes carried out reinforce the concern about the need for monitoring and security of a private blockchain network. The results of the experiments showed that an DoS, whether malicious or not, results in noticeable damage to the blockchain network, allowing the exploitation of other vulnerabilities through a combination of attacks.

Regarding Experiment 2, which involved a malicious user, which aimed to exploit possible application vulnerabilities to consume instance resources. In the scope of this attack there was a small change in which instead of a malicious user, two users were used, with the objective of optimizing the attack. Regarding the results of this experiment in the three scenarios, the blockchain network in general proved to be stable when trying to explore other ports. And as for the computational resources, flavor, from all instances, also did not present relevant changes that could somehow generate general context problems with the application. The reason this experiment did not show great effects is because it is a private and limited network, not allowing DDoS to be applied or for external tools to be applied.

As for Experiment 1, which was a Transaction Flood which could be malicious or non-malicious, as shown in Figures 9 and 10, it was evident that in all scenarios there was heavy traffic on their networks. In Scenario I with the RAFT consensus mechanism, the blockchain platform shows a drop in its transaction rate after 10 hours of the attack, indicating network congestion. Scenario II applying the iBFT consensus mechanism, the blockchain platform presents greater resistance with this consensus mechanism, accepting higher transaction fees per minute, however, from the 20th hour of the Transaction Flood was presented congestion in the network, resulting in a decrease in processing and generating inefficiency in transactions, resulting in an DoS. With Scenario III and its consensus mechanism PoA, in this consensus mechanism the platform proved to be very resistant to DoS, not being the most efficient of all in terms of number of transactions per minute obtained, but dealing with if in loss of transactions it proved to be the most efficient.

In general, in relation to consensus mechanisms, it is noticeable that with the application of good security and implementation practices, the problems presented can be minimized. Regarding the choice of flavor of VM, it is recommended that a feasibility study be carried out, so that a safe choice can be made in relation to the computational resources needed for the application. However, as much as there are some vulnerabilities, the use of blockchain technology has grown and shown to institutions and companies that the investment and development of it, as well as its methods, produces numerous benefits to its users.

As for the scope presented in the Qualification for the Defense of the dissertation, there were relative changes mainly in the choice of the blockchain platform and in the consensus mechanisms. Previously, the platform chosen was the Hyperledger Fabric, the choice was due to all the growth and expansion that it has presented, however, when expanded to applications of other consensus mechanisms, it tends to be quite complex in which several related works pointed out this difficulty. and inefficiency of consensus algorithms. It is important to note that the Hyperledger platform has several allowed consensus algorithms, but for each of these algorithms a different tool is applied, changing the core, which would directly influence the search results. With this information, a new research was carried out along with the possible consensus mechanisms and chosen as a replacement for the Hyperledger Fabric the Ethereum platform, and the RAFT, iBFT and PoA consensus mechanisms.

Finally, in relation to the general and specific objective of this work, they were fully completed. Despite the difficulties presented during the process of applying the consensus mechanisms, the experiments clearly showed that there is a need for choice in the selection of the instance *flavor* that is the most suitable, and also on the application needs for the decision of the chosen consensus mechanisms. In addition, with the growth of possible applications with the technology, the need for more in-depth studies and greater interfaces for exploiting internal vulnerabilities becomes clearer.

5.1 PUBLICATIONS

The following publications have already been made:

- Análise de mecanismos para consenso distribuído aplicados a Blockchain - XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSEG). Setembro/2019. Capítulo de Livro 50 Pags.
- Análise dos métodos para consenso distribuído aplicados à tecnologia Blockchain – XIX Escola Regional de Alto Desempenho da Região Sul (ERAD/RS) 2019. Abril/2019. Resumo estendido.
- Análise de segurança e desempenho de redes blockchains privadas/consorciadas quanto aos ataques DoS internos. XVIII Escola Regional de Redes de Computadores (ERRC) 2020. Novembro/2020. Artigo completo.
- Análise de Segurança dos Mecanismos de Consenso no pBFT usando Multichain e PoW usando Ethereum Aplicados em Redes Blockchain Privadas/Consórcio. Computer on the Beach 2020. Setembro/2020. Artigo completo.

- Análise de redes blockchain baseadas em Hyperledger do tipo privada/consorciada quanto a ataques DoS internos – XIX Escola Regional de Alto Desempenho da Região Sul (ERAD/RS) 2020. Maio/2020. Resumo estendido.
- Analysis of an Ethereum Private Blockchain Network Hosted by Virtual Machines Against Internal DoS Attacks - Advanced Information Networking and Applications (AINA 2022). April/2022. Artigo Completo.

5.2 FUTURE WORKS

Through this work, it was possible to understand more about private/consortium blockchain applications from VMs in computing clouds, in particular the Ethereum platform. As future work, it is interesting that the research be extended to comparisons with other sizes of flavors and also the use of other platforms with their consensus mechanisms within computational clouds. For example, the comparison of Hyperledger Fabric with the RAFT consensus mechanism and Ethereum with the RAFT consensus mechanism, in order to evaluate the behavior of these instances against different platforms and consensus algorithms, in order to have greater visibility of the selection of the best flavor with the platform and application.

Another research point is related to network throughput, limiting the speed of the network interface can work as a mitigation mechanism so that so many requests do not arrive. In this context, it can be explored how different types of interface associated with the node can affect performance, knowing that the network speed in containerized applications used Docker and Kata Containers have expressive throughput and latency.

BIBLIOGRAPHY

AMAZON. **Amazon EC2 tipos de instâncias - AWS**. 2020. Library Catalog: aws.amazon.com. Disponível em: <https://aws.amazon.com/pt/ec2/instance-types/>.

AZURE, M. **Singapore Airlines transforms customer loyalty with blockchain on Azure**. 2019. Library Catalog: customers.microsoft.com. Disponível em: <https://customers.microsoft.com/en-gb/story/singapore-airlines-travel-transportation-azure>.

Blockpharma. **Blockpharma Solution**. 2021. Disponível em: <https://www.blockpharma.com/>.

BUI, T. Analysis of docker security. **CoRR**, abs/1501.02967, 2015. Disponível em: <http://arxiv.org/abs/1501.02967>.

BUTERIN, V. A next-generation smart contract and decentralized application platform. In: . [S.l.: s.n.], 2015.

BUTERIN, V. **On Public and Private Blockchains**. 2015.

CASTRO, M.; LISKOV, B. Practical byzantine fault tolerance. In: **Proceedings of the Third Symposium on Operating Systems Design and Implementation**. USA: USENIX Association, 1999. p. 173–186.

Chowdhury, M. J. M. et al. A comparative analysis of distributed ledger technology platforms. **IEEE Access**, v. 7, p. 167930–167943, 2019.

COSTELLO, K.; RIMOL, M. **Gartner Forecasts Worldwide Public Cloud End-User Spending to Grow 18% in 2021**. 2020. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in

DALEY, S. **How Using Blockchain in Healthcare Is Reviving the Industry's Capabilities**. 2021. Disponível em: <https://builtin.com/blockchain/blockchain-healthcare-applications-companies>.

Davenport, A.; Shetty, S.; Liang, X. Attack surface analysis of permissioned blockchain platforms for smart cities. In: **2018 IEEE International Smart Cities Conference (ISC2)**. [S.l.: s.n.], 2018. p. 1–6.

DORRI, A. et al. Blockchain for iot security and privacy: The case study of a smart home. In: **2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerffCom Workshops)**. [S.l.: s.n.], 2017. p. 618–623.

E-Estonia. **Building blocks of e-estonia**. 2021. Disponível em: <https://e-estonia.com/solutions/>.

Energy Web. **The Energy Web Chain**. 2021. Disponível em: <https://energyweb.org/wp-content/uploads/2019/05/EWF-Paper-TheEnergyWebChain-v2-201907-FINAL.pdf>.

ESSBAUER, S.; SCHMIDT, M. **BMW Group uses Blockchain to drive supply chain transparency**. 2020. Disponível em: <https://www.press.bmwgroup.com/global/article/detail/T0307164EN/bmw-group-uses-blockchain-to-drive-supply-chain-transparency?language=en>.

ETHEREUM. **Ethereum Whitepaper**. 2021. Disponível em: <https://ethereum.org>.

Ethereum Foundation. **Nodes and Clients - Ethereum**. [S.l.], 2021. Disponível em: <https://ethereum.org/en/developers/docs/nodes-and-clients/>.

GREENSPAN, G. **MultiChain Private Blockchain – White Paper**. 2015. Disponível em: <www.multichain.com/download/MultiChain-White-Paper.pdf>.

Hao, Y. et al. Performance analysis of consensus algorithm in private blockchain. In: **2018 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.: s.n.], 2018. p. 280–285.

HASANOVA, H. et al. A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. **International Journal of Network Management**, v. 29, n. 2, p. e2060, mar. 2019. ISSN 10557148. Disponível em: <http://doi.wiley.com/10.1002/nem.2060>.

HEINRICH, T.; OBELHEIRO, R. **Caracterização de ataques DRDoS usando Honeypot**. [S.l.]: Dissertação - PPGCA - Universidade do Estado de Santa Catarina, 2019.

IBM. **Carrefour and Nestlé Partner with IBM to Extend Use of Blockchain to New Food Categories**. 2019. Disponível em: <https://www.ibm.com/blogs/think/2019/04/tracing-your-mashed-potatoes-on-ibm-blockchain/>.

IBM. **Módulos do IBM Food Trust**. 2020. Disponível em: <https://www.ibm.com/br-pt/blockchain/solutions/food-trust/modules>.

IDC. **New IDC Spending Guide Sees Strong Growth in Blockchain Solutions Leading to $15.9 Billion Market in 2023**. 2019. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS45429719>.

IZMAYLOV, M. et al. A practical application of blockchain for the travel industry. p. 1–24, 2021. Disponível em: <https://4454jm4bovib1sa6vrtflbew-wpengine.netdna-ssl.com/assets/docs/Factom_Whitepaper_v1.2.pdf>.

JANSEN, W.; GRANCE, T. Guidelines on security and privacy in public cloud computing. National Institute of Standards & Technology, Gaithersburg, MD, 2011. Disponível em: <"https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf">.

JOSHI, A.; HAN, M.; WANG, Y. A survey on security and privacy issues of blockchain technology. **Mathematical Foundations of Computing**, v. 1, p. 121–147, 01 2018.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering – a systematic literature review. **Information and Software Technology**, p. 7–15, 2009.

KULKARNI, P. Getting your hands dirty with containers. p. 1–45, 2016. Disponível em: <"https://www.cse.iitb.ac.in/~puru/courses/spring19/cs695/refs/containers-manual-prashanth.pdf">. Acesso em: 21 set. 2016.

LAMPORT, L. The part-time parliament. **ACM Trans. Comput. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 16, n. 2, p. 133–169, maio 1998. ISSN 0734-2071. Disponível em: <https://doi.org/10.1145/279227.279229>.

LIN, I.-C.; LIAO, T.-C. Survey of blockchain security issues and challenges. **International Journal of Network Secutiry**, Taiwan, 2017.

Linux Foundation. An introduction to hyperledger. p. 33, 2018. Disponível em: <https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf>.

LIU, C. et al. Studying gas exceptions in blockchain-based cloud applications. **Journal of Cloud Computing**, v. 9, n. 1, p. 35, dez. 2020. ISSN 2192-113X. Disponível em: <https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-020-00176-9>.

LIU, F. et al. NIST Cloud Computing Reference Architecture. **NIST Special Publication**, p. 35, 2011.

MELL, P. M.; GRANCE, T. Sp 800-145. the nist definition of cloud computing. National Institute of Standards & Technology, Gaithersburg, MD, 2011.

Merkle, R. C. Protocols for public key cryptosystems. In: **IEEE Symposium on Security and Privacy**. [S.l.: s.n.], 1980. p. 122–134. ISSN 1540-7993.

MIERS, C. et al. Análise de Segurança para Soluções de Computação em Nuvem. In: **SBRC 2014 Minicursos**. [S.l.: s.n.], 2014.

MIERS, C. et al. Análise dos métodos para consenso distribuído aplicados à tecnologia blockchain. In: **SBSeg 2019 - Minicursos**. USP - São Paulo: [s.n.], 2019. cap. 3, p. 1–49. Disponível em: <https://sbseg2019.ime.usp.br/minicursos.pdf>.

MOGULL, R. et al. Csa security guidance for critical areas of focus in cloud computing v4.0. Cloud Security Alliance's, 2017. Disponível em: <"https://cloudsecurityalliance.org/guidance/#_overview">.

MONIZ, H. **The Istanbul BFT Consensus Algorithm**. 2020.

MONRAT, A. A.; SCHELéN, O.; ANDERSSON, K. Performance evaluation of permissioned blockchain platforms. In: **2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)**. [S.l.: s.n.], 2020. p. 1–8.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. International Journal of Network Secutiry, 2008. Disponível em: <"https://bitcoin.org/bitcoin.pdf">.

NEEDHAM, M. **Global Spending on Blockchain Solutions Forecast to be Nearly $19 Billion in 2024, According to New IDC Spending Guide**. 2021. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS47617821>.

NIST. **FIPS 180-4: Secure Hash Standard (SHS)**. Gaithersburg, MD, USA, 2015.

ONGARO, D.; OUSTERHOUT, J. In search of an understandable consensus algorithm. In: **Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference**. USA: USENIX Association, 2014. (USENIX ATC'14), p. 305–320. ISBN 9781931971102.

ONGARO, D.; OUSTERHOUT, J. In Search of an Understandable Consensus Algorithm. p. 18, 2014.

PANIZZON, G. et al. A Taxonomy of container security on computational clouds: concerns and solutions. **Revista de Informática Teórica e Aplicada**, v. 26, n. 1, p. 47–59, abr. 2019. ISSN 21752745. Disponível em: <https://seer.ufrgs.br/rita/article/view/RITA-VOL26-NR1-47>.

PLURASIGHT. **Blockchain Architecture**. 2017. Disponível em: <https://www.pluralsight.com/guides/blockchain-architecture>.

Pongnumkul, S.; Siripanpornchana, C.; Thajchayapong, S. Performance analysis of private blockchain platforms in varying workloads. In: **2017 26th International Conference on Computer Communication and Networks (ICCCN)**. [S.l.: s.n.], 2017. p. 1–6.

R3. **Blockchain in Banking: Use Cases and Applications**. 2021. Disponível em: <https://www.r3.com/customers/banking/>.

R3. **Blockchain Use Cases in Digital Identity**. 2021. Disponível em: <https://www.r3.com/customers/digital-identity/>.

REINMUELLER, J. **Blockchain spotlight: Singapore Airlines - KPMG Global**. 2018. Disponível em: <https://home.kpmg/xx/en/home/insights/2018/09/blockchain-spotlight-singapore-airlines-fs.html>.

Renault Group. **XCEED: a new blockchain solution for Renault plants in Europe**. 2021. Disponível em: <https://www.renaultgroup.com/en/news-on-air/news/xceed-a-new-blockchain-solution-for-renault-plants-in-europe/>.

REPORT, S. **Funcionários são responsáveis por nove em cada dez violações de dados na nuvem**. 2019. Disponível em: <http://www.securityreport.com.br/overview/funcionarios-sao-responsaveis-por-nove-em-cada-dez-violacoes-de-dados-na-nuvem/>.

RODRIGUES, B. et al. Blockchain and Smart Contracts – From Theory to Practice. In: **Tutorials of IEEE International Conference on Blockchain and Cryptocurrency**. Seoul, South Korea: IEEE Computer Society Press, 2019. p. 31. Disponível em: <https://files.ifi.uzh.ch/CSG/staff/rodrigues/extern/publications/CNSM18-Tutorial.pdf>.

Rouhani, S.; Deters, R. Performance analysis of ethereum transactions in private blockchain. In: **2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)**. [S.l.: s.n.], 2017. p. 70–74.

SABAHI, F. Secure Virtualization for Cloud Environment Using Hypervisor-based Technology. **International Journal of Machine Learning and Computing**, p. 39–45, 2012. ISSN 20103700. Disponível em: <http://www.ijmlc.org/show-29-73-1.html>.

SHAHID, A. et al. Blockchain-based agri-food supply chain: A complete solution. **IEEE Access**, v. 8, p. 69230–69243, 2020.

SINGAPORE Exchange Case Study. 2020. Library Catalog: aws.amazon.com. Disponível em: <https://aws.amazon.com/solutions/case-studies/singapore-exchange-case-study/>.

SOMOROVSKY, J. et al. All your clouds are belong to us: Security analysis of cloud management interfaces. In: **Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop**. New York, NY, USA: Association for Computing Machinery, 2011. (CCSW '11), p. 3–14. ISBN 9781450310048. Disponível em: <https://doi.org/10.1145/2046660.2046664>.

SWAN, M. Blockchain: Blueprint for a new economy. O'relly, 2015.

TASCA, P.; TESSONE, C. A taxonomy of blockchain technologies: Principles of identification and classification. **Ledger**, v. 4, n. 0, 2019. ISSN 2379-5980. Disponível em: <http://www.ledgerjournal.org/ojs/index.php/ledger/article/view/140>.

TAYLOR, P. J. et al. A systematic literature review of blockchain cyber security. **Digital Communications and Networks**, fev. 2019. ISSN 2352-8648. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2352864818301536>.

Vatcharatiansakul, N.; Tuwanut, P. A performance evaluation for internet of things based on blockchain technology. In: **2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST)**. [S.l.: s.n.], 2019. p. 1–4.

VECHAIN. **Automotive Passport Solution**. 2021. Disponível em: <https://vechain.com/solution/logistics>.

VECHAIN. **Logistics Solution**. 2021. Disponível em: <https://vechain.com/solution/car>.

WU, H. et al. Network security for virtual machine in cloud computing. **Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference On**, p. 4, 11 2010.

YAGA, D. et al. **Blockchain technology overview**. Gaithersburg, MD, 2018. NIST IR 8202 p. Disponível em: <https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf>.

Zyskind, G.; Nathan, O.; Pentland, A. . Decentralizing privacy: Using blockchain to protect personal data. In: **2015 IEEE Security and Privacy Workshops**. [S.l.: s.n.], 2015. p. 180–184.