UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GESSICA FRANCIÉLE MENDONÇA AZEVEDO

# Enabling Blockchain-based Circuit Allocation in MEICAN

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Lisandro Zambenedetti Granville
Coadvisor: Prof. Dr. Muriel Figueredo Franco

Porto Alegre
July 2025

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profª. Marcia Cristina Bernardes Barbosa
Vice-Reitor: Prof. Pedro de Almeida Costa
Pró-Reitora de Graduação: Profª. Nádya Pesce da Silveira
Diretor do Instituto de Informática: Prof. Luciano Paschoal Gaspary
Coordenador do Curso de Ciência de Computação: Prof. Álvaro Freitas Moreira
Bibliotecário-chefe do Instituto de Informática: Alexsander Borges Ribeiro

## ACKNOWLEDGEMENTS

The completion of this undergraduate thesis was only possible thanks to a network of people whose guidance, inspiration, and support were present at every step of the way.

First, I express my deepest gratitude to my advisors, Prof. Dr. Lisandro Granville and Prof. Dr. Muriel Franco, for their dedication, the discussions that broadened my perspective, and the confidence they placed in my potential. Their technical and theoretical advice was fundamental to this work reaching the maturity it has today.

I also thank Prof. Eder Scheid, who closely followed this project. His meticulous observations and constant encouragement contributed decisively to the refinement of each chapter.

I cannot fail to mention two figures who, despite their distance in time, served as intellectual beacons: Alan Turing, the father of modern computing, and Ada Lovelace, a visionary who saw the creative potential of machines in the 19th century.

To my mother, Maria, a daily example of determination and affection, I offer my eternal gratitude for her unconditional love and for always believing in my potential. To my friends, who celebrated every small victory and offered support in times of doubt, I extend my sincere gratitude.

Finally, I dedicate this work to all of you. May each result presented here reflect the support, patience, and affection I received throughout this journey. Please accept my sincere and profound gratitude.

## ABSTRACT

This work presents the design, implementation, and evaluation of a solution to enable a blockchain-based circuit allocation system in MEICAN, leveraging smart contracts and decentralized storage via IPFS. The solution enhances transparency, auditability, and data immutability in circuit request workflows by registering requests and policy files in a tamper-resistant and distributed manner. The architecture includes a Solidity smart contract for managing requests and approvals, a Node.js API for orchestrating blockchain and IPFS interactions, and user-friendly frontend pages that simulate the workflows of circuit requests, evaluations, and approvals. The evaluation demonstrates the feasibility of using blockchain for inter-domain circuit management while highlighting the trade-offs in processing time introduced by encryption, IPFS uploads, and transaction confirmation on the blockchain. The system successfully ensures integrity and traceability while maintaining decentralization principles, serving as a foundation for future developments in transparent and secure network management solutions. Future work may focus on performance optimizations, integration with alternative public or private blockchains, and the implementation of advanced privacy-preserving mechanisms to improve scalability and confidentiality in production environments.



**Keywords:** Blockchain. Smart Contracts. IPFS. Decentralized Storage. Network Management. Inter-domain Circuits. MEICAN. Transparency. Auditability.

**Habilitando a Alocação de Circuitos Baseada em Blockchain no MEICAN**

## RESUMO

Este trabalho apresenta o projeto, implementação e avaliação de uma solução que possibilita um sistema de alocação de circuitos baseado em blockchain no MEICAN, utilizando contratos inteligentes e armazenamento descentralizado via IPFS. A solução visa aumentar a transparência, auditabilidade e imutabilidade dos dados nos fluxos de solicitação de circuitos, registrando requisições e arquivos de políticas de forma distribuída e resistente a alterações. A arquitetura desenvolvida inclui um contrato inteligente em Solidity para gerenciar solicitações e aprovações, uma API em Node.js responsável pela orquestração das interações com a blockchain e o IPFS, e páginas web intuitivas que simulam os fluxos de requisição, avaliação e aprovação de circuitos. A avaliação realizada demonstra a viabilidade do uso de blockchain no gerenciamento de circuitos interdomínios, evidenciando, porém, os impactos no tempo de processamento decorrentes das etapas de criptografia, upload no IPFS e confirmação de transações na blockchain. O sistema assegura integridade e rastreabilidade dos registros, mantendo os princípios de descentralização, e estabelece uma base para futuros desenvolvimentos em soluções de gerenciamento de redes transparentes e seguras. Trabalhos futuros podem explorar otimizações de desempenho, integração com blockchains públicas ou privadas alternativas e a implementação de mecanismos avançados de preservação de privacidade para aprimorar a escalabilidade e a confidencialidade em ambientes de produção.

**Palavras-chave:** Blockchain. Contratos Inteligentes. IPFS. Armazenamento Descentralizado. Gerenciamento de Redes. Circuitos Interdomínios. MEICAN. Transparência. Auditabilidade.

## LIST OF FIGURES

## LIST OF TABLES

# LISTINGS

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| BGP | Border Gateway Protocol |
| BPM | Business Process Management |
| CBC | Cipher Block Chaining |
| CDN | Content Delivery Network |
| CID | Content Identifier |
| CLI | Command-Line Interface |
| DApps | Decentralized Applications |
| DCN | Dynamic Circuit Network |
| DDoS | Distributed Denial-of-Service |
| DeFi | Decentralized Finance |
| DLT | Distributed Ledger Technology |
| DPoS | Delegated-Proof-of-Stake |
| ENS | Ethereum Name Service |
| EVM | Ethereum Virtual Machine |
| FCN | Federated Coalition Networks |
| HTTP | Hypertext Transfer Protocol |
| HTML | Hypertext Markup Language |
| IBE | Identity-Based Encryption |
| IIoT | Industrial Internet of Things |
| IoE | Internet of Energy |
| IoT | Internet of Things |
| IPFS | InterPlanetary File System |

| | |
|---|---|
| IR | Intermediate Representation |
| ISP | Internet Service Provider |
| IXP | Internet eXchange Points |
| IV | Initialization Vector |
| JSON | JavaScript Object Notation |
| NFT | Non-Fungible Token |
| NFV | Network Function Virtualization |
| NSA | Network Security Appliance |
| NSI | Network Service Interface |
| P2P | Peer-to-Peer |
| PBFT | Practical Byzantine Fault Tolerance |
| PoA | Proof-of-Authority |
| PoS | Proof-of-Stake |
| PoW | Proof-of-Work |
| REST | Representational State Transfer |
| RNP | Rede Nacional de Ensino e Pesquisa |
| RSA | Rivest-Shamir-Adleman |
| SDK | Software Development Kit |
| SDN | Software-Defined Networking |
| SLA | Service Level Agreement |
| VNF | Virtual Network Functions |
| WS-BPEL | Web Services Business Process Execution Language |

**CONTENTS**

# 1 INTRODUCTION

The exposure of computer networks and service virtualization in recent years has significantly increased the importance of systems that simultaneously impose reliability, auditability, and transparency in critical operations (MAURO et al., 2025). The development in interconnectivity among academic and research institutions has prompted a requirement for solutions that can keep network circuits securely and efficiently, specifically when they need to go through multiple administrative domains (ASHAROV et al., 2017). Traditional inter-domain routing protocols, such as the Border Gateway Protocol (BGP), while foundational, exhibit known deficiencies in security, often leading to information leakage about routing policies, and can suffer from slow convergence, impacting efficiency (ASHAROV et al., 2017). Managing resources and enforcing policies consistently across these diverse and autonomous domains presents significant challenges to conventional centralized management paradigms, often creating a "trust deficit." Each domain operates under its own set of policies and security postures, and the reluctance to share sensitive operational data, as seen in Internet eXchange Points (IXPs) (CHIESA et al., 2017), highlights this inherent lack of complete mutual trust.

This environment demands solutions that can provide verifiable security and efficiency guarantees despite administrative boundaries, motivating the exploration of technologies that can establish or operate effectively in trust-minimized settings. Similar challenges are observed in Federated Coalition Networks (FCNs), where entities from multiple nations must cooperate while maintaining control over their respective systems, necessitating enhanced security and resilience in multi-domain management (GONZáLEZ; GARCíA; BAEZA, 2025). In response to the inherent difficulties in ensuring integrity and traceability, especially within decentralized or multi-domain processes, technologies such as blockchain and smart contracts have emerged as compelling solutions (AKTER et al., 2024).

The Management Environment of Inter-domain Circuits for Advanced Networks (MEICAN) project is essentially a system put in place to be able to orchestrate and automate network circuit reservations between institutions (WICKBOLDT et al., 2018). Currently, the process of registration and approval of circuits is done in a centralized manner, which is functional, though it does not provide an immutable mechanism for verification and public auditing. Centralized systems, by their nature, concentrate control and data, which can become a single point of failure and compromise (GOEL; RAHULAMATHA-

VAN, 2024).

The research problem is therefore formulated as follows: How can MEICAN guarantee that every registration, approval and rejection action is permanently recorded, tamper-evident and independently auditable? This problem statement directly addresses the need for a system where all actions are recorded on an immutable ledger, visible to authorized participants, thereby ensuring transparency.

The objective of this work is design, implement and experimentally evaluate a blockchain-backed governance layer for MEICAN that logs each step of the circuit-management life-cycle on an immutable ledger and stores the associated policy artefacts in a verifiable manner, thereby enhancing transparency and trust across heterogeneous administrative domains.

The importance of this study lies in the presentation of a working model that synergistically combines the security provided by the blockchain and the conventional MEICAN workflow. The integration achieved is a major step forward in enabling multi-domain network governance to be more efficient, as it makes the decision-making process auditable, and end-user participation is encouraged by immutability capabilities and greater confidence in data integrity.

The research employed a combination of applied and experimental methodologies using open-source tools for technological developments. The contracts were implemented through Solidity, the backend through Node.js using the ethers.js library, while the frontend was developed with HTML and JavaScript, creating the interfaces to represent the MEICAN system, as interactions with users and administrators were emulated. The blockchain was generated locally with Ganache CLI, which allowed testing of the system in a controlled environment.

The remainder of this bachelor's thesis is structured as follows: Chapter 2 presents the background necessary to understand this work, covering Blockchain, Smart Contracts, the InterPlanetary File System (IPFS), and the MEICAN system architecture and workflows. Chapter 3 discusses related work, exploring the integration of Blockchain with NFV, SDN, IoT, and fog computing, while highlighting existing challenges. Chapter 4 details the proposed Blockchain-based circuit allocation system architecture and its implementation, including the design of the smart contract, API, frontend interfaces, and the encryption and decentralized storage mechanisms. Chapter 5 presents the evaluation and results, analyzing gas consumption, transaction costs, execution times, and discussing the limitations of the system. Finally, Chapter 6 concludes this work, summarizing its

contributions and outlining future research directions to enhance scalability, security, and auditability in multi-domain circuit management using Blockchain technologies.

# 2 BACKGROUND

This chapter presents the concepts required to understand this work. It encompasses the main concepts of Blockchain technology, Smart Contracts, InterPlanetary File System (IPFS), and the MEICAN system. Each of these elements plays a critical role in enabling secure, decentralized, and auditable management of network circuit reservations across multiple administrative domains.

The chapter begins with a conceptual explanation of Blockchain and its fundamental mechanisms, followed by an exploration of Smart Contracts, which allow for automated and trustless execution of predefined agreements. Subsequently, the chapter introduces IPFS, highlighting its decentralized approach to content storage and distribution. Finally, it presents the architecture, operational workflow, and policy management mechanisms of the MEICAN system, which serves as the target environment for the integration proposed in this research.

## 2.1 Blockchain

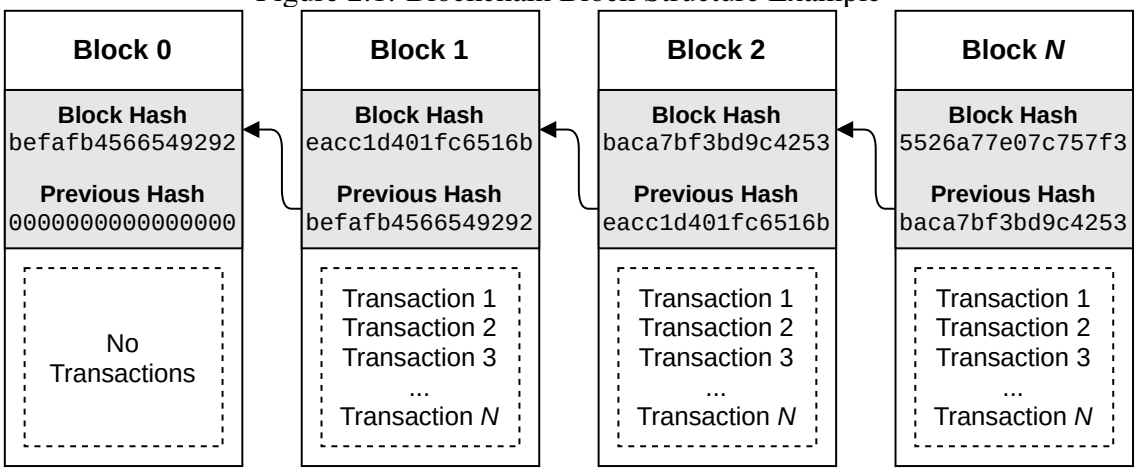
Blockchain, as a type of Distributed Ledger Technology (DLT), operates as a distributed and shared database of records and data, creating a worldwide index for every transaction in a particular market. This system, in its public permissionless deployment, functions as a public and universal ledger between two entities, creating trust mechanisms through direct communication without the need for third-party intermediaries (SCHEID et al., 2021). The blockchain expands through the addition of new blocks, which contain new sets of records. New blocks are added to the blockchain sequentially and in chronological order. When a computer joins the network as a node, which validates and forwards transactions, it receives a complete copy of the blockchain. The blockchain maintains full information about addresses and balances, starting from the genesis block through to the most recent completed block.

The first blockchain network was Bitcoin, developed by Satoshi Nakamoto, which focuses on the financial market revolution (NAKAMOTO; BITCOIN, 2008). Later, Ethereum came out as an alternative to Bitcoin, with the intent to "*create an alternative protocol for building decentralized applications, providing a different set of tradeoffs that we believe will be very useful for a large class of decentralized applications, with particular emphasis on situations where rapid development time, security for small and rarely*

*used applications, and the ability of different applications to very efficiently interact, are important*" (BUTERIN et al., 2014).

The database for a blockchain contains two basic record types: individual transactions and blocks. A block represents the blockchain's physical structure, as it records recent transactions that become a permanent record in the blockchain database after completion. A new block is added each time a block is completed. The blockchain contains numerous sequential blocks that are connected in a chain-like structure, since each block stores information (*e.g.,* previous block hash) about its predecessor as depicted in Figure 2.1.

Figure 2.1: Blockchain Block Structure Example




Source: (SCHEID, 2022)

In the blockchain, the consensus algorithm is used to solve the problem of trust, which means that if data is deleted from the blockchain, users are able to verify that such data was deleted, ensuring the correctness of the blockchain. Further, it forces that all new insertions must be accepted by everyone following the consensus (SCHEID et al., 2021). To achieve this, a rule (algorithm) must be used that governs the inclusion of new data. A new block can only be considered to have been created (mined) if it also complies with the rule defined by the consensus algorithm, and so the new block is included in the blockchain, making the new data public. Another important point is that the definition of a consensus algorithm means that decisions about what will be inserted into the blockchain do not depend on any centralizing entity, thus ensuring its independence.

Also, through these rules, each user of the network can verify the authenticity of each block contained in the blockchain and verify for themselves the authenticity of all the data entered in it. In this way, no one needs to trust anyone, since this verification makes each user trust the blockchain.

According to (DENG et al., 2022), there are several consensus algorithms, which can be mentioned:

- Proof-of-Work (PoW): This algorithm causes a competition between the computers connected to the blockchain to see who will be the first to find a hash corresponding to the new block, considering the imposed difficulty.
- Proof-of-Stake (PoS): The mining of new blocks by this algorithm occurs through the voting of the network participants who have “digital assets” (from the blockchain that uses it).
- Delegated-Proof-of-Stake (DPoS): This algorithm is similar to PoS, but participants do not vote directly on whether a block is valid or not. (*i.e.*, the participants vote for Delegates and Witnesses, who will vote on their behalf).

## 2.2 Smart Contracts

A smart contract can be understood as a specialized computer program designed to operate on the infrastructure of a blockchain. The concept, although popularized with the advent of blockchain, was introduced by Nick Szabo in the 1990s, who described them as "a set of promises, specified in digital form, including protocols within which the parties perform on these promises" (SZABO, 1996). Its main functionality lies in its ability to self-execute, control, or document events and relevant actions, according to the terms previously agreed upon and encoded in its internal logic. Unlike traditional contracts that require human intermediation for their validation and fulfillment, a smart contract translates the contractual clauses directly into lines of code.

Once the predefined conditions in this code are met through data inputs in the blockchain, the contract autonomously executes the programmed actions. This automatic execution capability is one of the central features, eliminating the need for third-party intervention to ensure compliance with the agreement (BUTERIN et al., 2014). After its deployment in the blockchain network, the code of a smart contract generally acquires the property of immutability, which prevents subsequent changes to the established terms, providing a high degree of security and predictability to interactions. Once deployed on the blockchain, smart contracts are typically immutable, meaning their code cannot be altered. Additionally, the nature of the blockchain provides transparency, as the contractual terms and the execution history can be inspected by network participants, with the level

of access varying according to the type of blockchain. The distributed architecture of the blockchain, where the smart contract is replicated and maintained by multiple nodes, ensures its resilience and availability, while reinforcing the decentralized aspect of contract execution. This decentralization mitigates the dependence on a central authority, since trust is established by the cryptographic integrity and consensus of the network itself. Indeed, smart contracts are often described as "user-defined programs that specify the rules governing transactions, and which are enforced by a network of peers" (DELMOLINO et al., 2016).

The applications of smart contracts are vast and span sectors such as Decentralized Finance (DeFi), where they facilitate loans and exchanges of assets without intermediaries, supply chains, for traceability and verification of authenticity of products, and many other domains that benefit from the secure and transparent automation of agreements. In the context of systems like MEICAN, smart contracts offer a robust mechanism for the immutable recording of transactions, such as circuit requests, their approvals or rejections, and the associated metadata, thus creating an auditable and reliable history, essential for auditing processes.

## 2.3 InterPlanetary File System

IPFS is a decentralized protocol that utilizes a peer-to-peer (P2P) network and is hypermedia-oriented, aiming to transform file storage and sharing into a distributed system. It is the concept of addressing content that makes the system work, with the help of which each file is given a special identifier, which is based on the cryptographic hash. This solution allows the nodes to be the web of the whole system, and all the participating nodes in the network to have their copies of the data simultaneously. It is this approach that further allows the network to be used for hypermedia, which is interconnected and enables efficient and on-time data delivery (BENET, 2014).

Comparable to the way the BitTorrent process is conducted, IPFS enables its users to play the roles of hosts and request network access from each other directly, thus bypassing the traditional centralized server-centric model. For the purpose of carrying the load of the entire network, the system is highly diversified. Each end user plays the part of a network node to store pieces of the entire data, thus contributing to global data storage. The method of storing and sharing is based on the address of the content, namely, the files can be found and downloaded via a distributed hash table. Unlike BitTorrent, IPFS

suggests building a connected network worldwide, meaning that blocks of data having one common hash are interchangeable among various sources, bringing in better content. The main characteristics of IPFS are listed below.

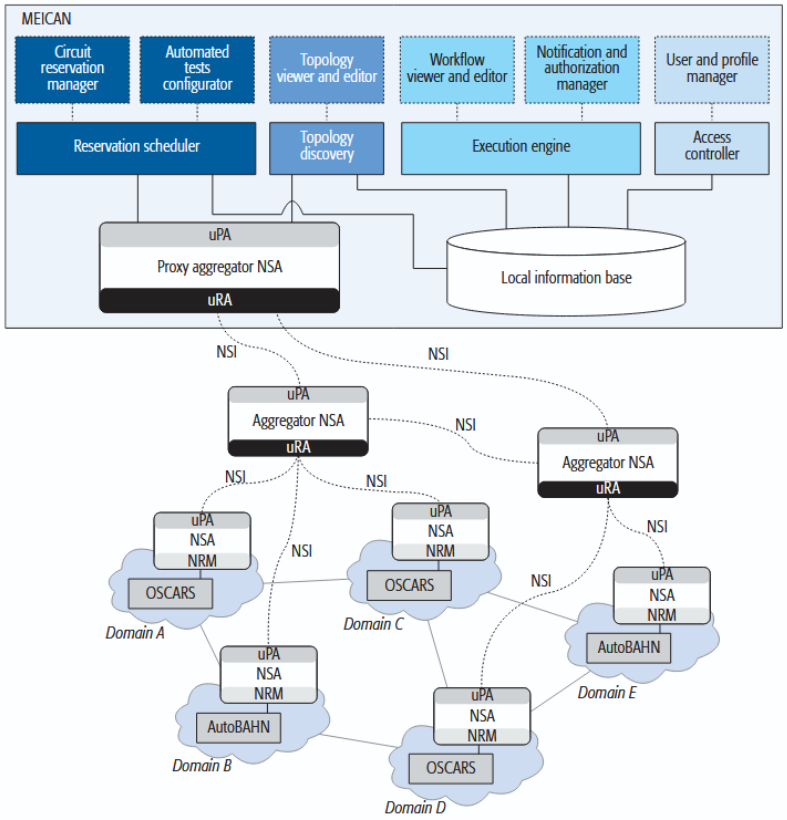
- Data storage based on content addressing: The hash is identical to the author's content, in which the file content is the source of the hash; also, integrity of the data will not be a problem, and the recovery process will become simpler.
- P2P architecture: The decentralized nature of the network, which rules out the need for a centralized server as the conveying channel, thus the eradication of indirect communication becomes the order of the day.
- Versioned file system: The protocol even makes it possible to keep file history versions, thus facilitating the task of change tracking in the file history.
- Integration with decentralized applications: IPFS functions as a storage layer for Decentralized Applications (DApps) and is heavily utilized in blockchain-based solutions and the Web3 ecosystem, so it is the ideal place to store them.

Moreover, IPFS is trying to substitute the typical means through which static web pages are delivered by using gateways that can be accessed through the Hypertext Transfer Protocol (HTTP). These gateways enable users to reach the content located within the IPFS network without having a local client installed, and a public list of the access points is preserved in the project's official repository. IPFS has not only functioned as a substitute for traditional web hosting but has also been a major driving force in sharing scientific data, preserving digital content, and the development of decentralized applications. Therefore, it has been labeled as an integral incipient infrastructure in the evolution of the decentralized internet.

## 2.4 Overview of MEICAN

MEICAN is a system developed to orchestrate and automate the reservation of dynamic network circuits between multiple institutions and administrative domains, ensuring policy compliance, efficient resource allocation, and seamless coordination across the involved networks. MEICAN is composed of different components, as illustrated in Figure 2.2, and its main flow can be summarized in seven stages, as described by (WICKBOLDT et al., 2018).

Figure 2.2: Main Conceptual Components of MEICAN




Source: (WICKBOLDT et al., 2018)

1. Circuit Request: the user accesses a MEICAN web interface, where he can choose the origin and destination points, the desired bandwidth, and the period in which the connection will be required.
2. Booking and Reservation Initiation: MEICAN records the details of the request and initiates the reservation process. This involves generating a unique identifier for the request and internally controlling the states of this reservation.
3. Resource and Path Query: This is when MEICAN, through its Network Security Appliance (NSA) Aggregator Proxy, queries the federated network infrastructure to verify whether there is a viable path with the necessary resources (mainly bandwidth) to fulfill the request. This step involves communication with NSA aggregators from other network domains that make up the path between source and destination.
4. Authorization: once a path with available resources has been identified, MEICAN executes the authorization process, which verifies whether all domains involved allow the establishment of that circuit. This authorization is done through policy workflows previously defined by each participating domain. Such workflows may involve automatic, rule-based decisions or require manual interventions by local

network operators. If the request is not approved in any domain, the reservation is denied. If all authorizations are granted, the process continues.

5. Provisioning and Activation: Once authorization has been approved, the next step is provisioning and activation. Here, MEICAN coordinates the configuration of the circuit on the network equipment of all domains involved, according to the defined path and the requested parameters. The circuit is then activated on the date and time previously specified by the user (WICKBOLDT et al., 2018).
6. Monitoring and Status Update: During the entire period in which the circuit is active, MEICAN monitors and updates the status, maintaining control of the connectivity status. The system monitors possible failures or interruptions and updates its internal records as the circuit status changes (for example, from “active” to “inactive”).
7. Reservation Completion: At the end of the scheduled period, the reservation is finalized. At this stage, the allocated resources are released, the circuit is dismantled, and MEICAN records the request history, allowing operators and users to consult information about the circuit later, if necessary.

### 2.4.1 Workflows

MEICAN applies the policies of each domain in the form of workflows, which are decision flows responsible for defining the authorization rules for creating inter-domain virtual circuits. In previous versions of MEICAN, these workflows were written using the Web Services Business Process Execution Language (WS-BPEL), a standardized language widely used in Business Process Management (BPM) solutions (SANTANNA; WICKBOLDT; GRANVILLE, 2012). This initial approach allowed the description of sequences of tasks, including automated steps and manual interventions, through a service-oriented language.

In the most recent versions, MEICAN has evolved to use its visual workflow language, based on a drag-and-drop graphical interface. This new approach facilitates the modeling and execution of local policies, allowing the definition of elements such as automatic filters, manual interventions, and decision terminals, in addition to offering greater flexibility and extensibility in circuit authorization control (WICKBOLDT et al., 2018).

### 2.4.2 Data Structure

The structural definition of the workflow is stored in JavaScript Object Notation (JSON) format, inside a table in the database. However, the JSON does not contain specific data for each request. It only describes the logic of the flow, indicating which checks will be made and which paths the execution should follow, depending on the results of the decisions.

The JSON representing the workflow contains:

- Workflow name: Identification of the decision logic (*e.g., Manual Approval Workflow*).
- Responsible domain: Indicates the MEICAN domain responsible for the flow.
- List of modules (*nodes*): Each node represents a step in the process, and can be:
- Start node (*New_Request*)
- Automatic decision node (*e.g., Bandwidth, Duration*, etc.)
- Manual decision node (*User* or *Group*)
- End node (*Accept* or *Deny*)

Each module (node) includes:

- `id`: Unique identifier within the workflow.
- `name`: Node type.
- `adjacency`: List of next nodes (the output paths).
- `value`: Parameter required for the decision (for example, minimum bandwidth value, or user responsible for the decision).
- `operator`: Logical operator, used in comparison nodes.

An example of a workflow whose approval process depends on a manual decision by a human operator is presented below in Listing 2.1.

Listing 2.1 – Manual Approval Workflow Example

```
{
  "params": {
    "name": "Manual Approval Workflow",
    "working": {
      "properties": {
        "name": "Manual Approval Workflow",
```

```
        "domains_owner": "example.com"
      },
      "modules": [
        {
          "name": "New_Request",
          "id": 0,
          "adjacency": [1],
          "value": null,
          "operator": null
        },
        {
          "name": "User",
          "id": 1,
          "adjacency": [2, 3],
          "value": "operator@example.com",
          "operator": null
        },
        {
          "name": "Accept",
          "id": 2,
          "adjacency": [],
          "value": null,
          "operator": null
        },
        {
          "name": "Deny",
          "id": 3,
          "adjacency": [],
          "value": null,
          "operator": null
        }
      ]
    }
  }
```

}

In this flow:

1. Every request starts at the *New_Request* node.
2. The flow continues to the *User* node, awaiting a manual decision from the defined operator (in this case, *operator@example.com*).
3. The operator has two options: approve (going to the *Accept* node) or reject (going to the *Deny* node).

During workflow execution, the responsible class loads the workflow JSON and, based on it, accesses the request data in the database. The MEICAN engine then evaluates each node according to the actual request parameters, deciding which node the flow should go to next.

For example, in the *User* node, MEICAN identifies the responsible operator, checks the action taken by him (Approve or Reject), and advances in the flow according to the decision.

This modular and data-driven design allows MEICAN to implement flexible and reusable workflows while maintaining a clear separation between workflow logic and request-specific data. Further details about the MEICAN architecture and source code are available in its public repository at: https://github.com/rnpbr/meican.

### 2.4.3 Application

The MEICAN system has a web application designed to manage Dynamic Circuit Networks (DCNs). Through a web graphical interface, users can request the creation of circuits, specifying source, destination, required bandwidth, and the circuit's uptime. The system allows provisioning to be automatic or dependent on the approval of network administrators, using internal workflows that reflect operational policies (Management Environment of Inter-domain Circuits for Advanced Networks (MEICAN), 2025).

Since version 2, MEICAN has integrated the Network Service Interface (NSI) protocol to serve connection service providers, exemplified by its interaction with the NSI Aggregator of the Rede Nacional de Ensino e Pesquisa (RNP) (Rede Nacional de Ensino e Pesquisa (RNP), 2025) in Brazil, where it acts as a central portal for circuit creation.

Version 3 improved integration with monitoring systems, allowing the monitoring

of circuits and networks directly in the web application (Management Environment of Inter-domain Circuits for Advanced Networks (MEICAN), 2025). This software is the result of a collaboration between RNP and the Federal University of Rio Grande do Sul (UFRGS) (Management Environment of Inter-domain Circuits for Advanced Networks (MEICAN), 2025).

# 3 RELATED WORK

In recent decades, the convergence of emerging technologies such as Blockchain, Software-Defined Networking (SDN), Network Function Virtualization (NFV), and Internet of Things (IoT) has fueled a series of scientific investigations into new architectures and models for efficient, secure, and decentralized network management.

## 3.1 Blockchain and NFV

The use of blockchain in the context of NFV has been widely explored to ensure security, automation, and decentralization in the orchestration and management processes of these functions.(BONDAN et al., 2019) proposed FENDE, a marketplace ecosystem focused on the distribution and lifecycle management of Virtual Network Functions (VNF). Although the proposal does not integrate blockchain directly, it introduces relevant issues pertaining to scalability and innovation in the NFV domain.

In the field of security and reliability, (SCHEID et al., 2019) presented BUNKER, a trusted VNF package repository that uses blockchain to ensure package integrity, thereby obviating the necessity for trusted intermediaries. Subsequently, (SCHEID et al., 2022) proposed VeNiCE, an architecture that automates VNF lifecycle management based on smart contract events, demonstrating practical feasibility with prototypes based on Ethereum and OpenStack.

Other research, such as that of (TASKOU; RASTI; NARDELLI, 2021), introduced frameworks such as NFVChain, which aims at the efficient allocation of energy and financial resources in blockchain-based NFV networks. This model employs optimized algorithms that contribute to cost and latency reduction, although challenges such as computational overhead and scalability persist.

Regarding operational security, (ALVARENGA; REBELLO; DUARTE, 2018) proposed a blockchain-based architecture that ensures the management and migration of VNFs, using Practical Byzantine Fault Tolerance (PBFT) as a consensus protocol. More recently, (CAMILO et al., 2022) developed a distributed and secure system for orchestrating VNFs, employing smart contracts on platforms such as Hyperledger Fabric to manage service-level agreements (SLAs).

To complement these endeavors, (FRANCO et al., 2019) carried out the development of the BRAIN (Blockchain-based Reverse Auction for Infrastructure Supply in

Virtual Network Functions-as-a-Service) concept, which is focused on the reverse auction mechanism to promote competition among infrastructure providers in hosting VNFs. The full architecture of BRAIN is built by combining blockchain and smart contracts components that allow such properties as immutability, auditability, and trust to be achieved in the process of infrastructure choice, predetermined by the final user, such as resource availability, geographic location, and latency requirements. Despite a feasibility demo, the solution faces some practical problems, mostly related to the cost and latency of transactions on public blockchains, such as Ethereum.

## 3.2 Blockchain and SDN

Integrating blockchain with SDN has become a rapidly growing research area. (ALHARBI, 2020) presented a detailed study on the use of the blockchain principle to secure and make the SDN network more efficient. Nevertheless, as the authors note, their study is mainly focused on the theoretical part and is still waiting for actual validation.

(LATAH; KALKAN, 2022) highlighted the BC-Sec-SDN architecture, which aims to distribute power and provide security functions by smart contracts, as well as blockchain-based authentication methods in SDN networks. Similarly, (OKTIAN et al., 2022) envisioned a decentralized system for bandwidth negotiation in SDN edge networks that is more efficient and flexible for all involved users. The paper not only presents the benefits of the approach but also indicates that there are still some challenges to eliminate, such as the high transactions per second in public networks like Ethereum.

The implementation of blockchain-SDN convergence in particular areas, including the Internet of Energy (IoE), has been demonstrated. A concept was provided by (HAYYOLALAM et al., 2024) for a system that uses both blockchain and SDN to conduct energy transactions securely and in a decentralized manner, which will result in the efficient utilization of smart grids and encourage P2P energy trading.

In terms of practical implementations, (KOVACS et al., 2024) demonstrated the feasibility of integrating blockchain into the control of large-scale SDN/NFV networks, focusing on the validation of functions and Internet Service Provider (ISP) environments. However, aspects related to the impact on performance and scalability still necessitate further examination.

### 3.3 Blockchain, SDN and IoT: Synergies and Challenges

The combination of blockchain, SDN, and IoT has demonstrated potential in improving the security, scalability, and reliability of networks. (DORRI et al., 2022) explored timestamp obfuscation methods to protect the privacy of IoT devices in blockchain-based networks, indicating a substantial reduction in the effectiveness of deanonymization attacks.

(TURNER et al., 2023) performed an exhaustive analysis on the integration of blockchain and SDN in IoT networks, identifying six key application areas: security, distributed computing, trust management, access control, privacy, and networking. Although the integration offers substantial advantages, challenges such as latency, scalability, and interoperability still constitute relevant barriers.

In the context of Industry 4.0 and the Industrial Internet of Things (IIoT), (RAHMAN et al., 2022) proposed the DistB-SDCloud architecture, which uses blockchain and SDN to ensure the security of industrial cloud computing environments, especially in protecting against Distributed Denial-of-Service (DDoS) attacks. Similarly, (KUMAR et al., 2024) developed the BCSDNCC framework, enhancing the security and efficiency of IoT networks and cloud computing through the integration of these technologies.

(HAKIRI et al., 2020) proposed an architecture combining blockchain, SDN, and NFV to improve the security of IoT networks, using Proof-of-Authority (PoA) as a consensus mechanism to reduce computational costs and latency, although the centralization inherent in PoA may compromise full decentralization.

### 3.4 Blockchain Applications in Networks and Fog Computing

In addition to the integrations already mentioned, the application of blockchain in fog computing environments has proven to be an emerging trend. (BANIATA; KERTESZ, 2020) conducted a systematic review on the integration of blockchain with fog computing, uncovering opportunities to improve security, traceability, and reliability in several sectors, such as IoT, eHealth, and Industry 4.0. However, scalability remains one of the main challenges due to latency and high energy consumption.

In the healthcare sector, (KAUR et al., 2024) proposed a framework that combines blockchain and the Zero Trust model with fog computing to reinforce security in digital hospital environments. The proposal demonstrated significant gains in intrusion detection

and data integrity, although issues related to computational cost and interoperability with legacy infrastructures still need to be overcome.

## 3.5 Key Findings

The literature analysis underscores the revolutionary potential of the integration between blockchain, SDN, NFV, and IoT, especially with regard to security, automation, and decentralization in network management. However, challenges persist, notably about scalability, latency, and computational costs, which must be thoroughly evaluated in future implementations. These aspects are especially pertinent for initiatives such as the integration of blockchain into MEICAN, whose objective is to improve the management of inter-domain circuits through distributed and secure mechanisms based on smart contracts.

# 4 BLOCKCHAIN-BASED CIRCUIT ALLOCATION

A decentralized solution was developed to provide an audit mechanism for circuit requests managed by the MEICAN system. The architecture of the proposed solution is based on an API that interacts with a Smart Contract previously deployed in a blockchain infrastructure. The smart contract encapsulates essential functionalities, including the registration of data for each circuit request, the registration of the decision to approve or reject a request, and the consultation of the data and current status of a specific request. In order to simulate the primary operational life cycle of MEICAN, web interfaces were additionally implemented. These interfaces allow the registration of new requests, the consultation of data for existing requests, the visualization of requests pending evaluation, and the effective evaluation (approval or rejection) of a request.

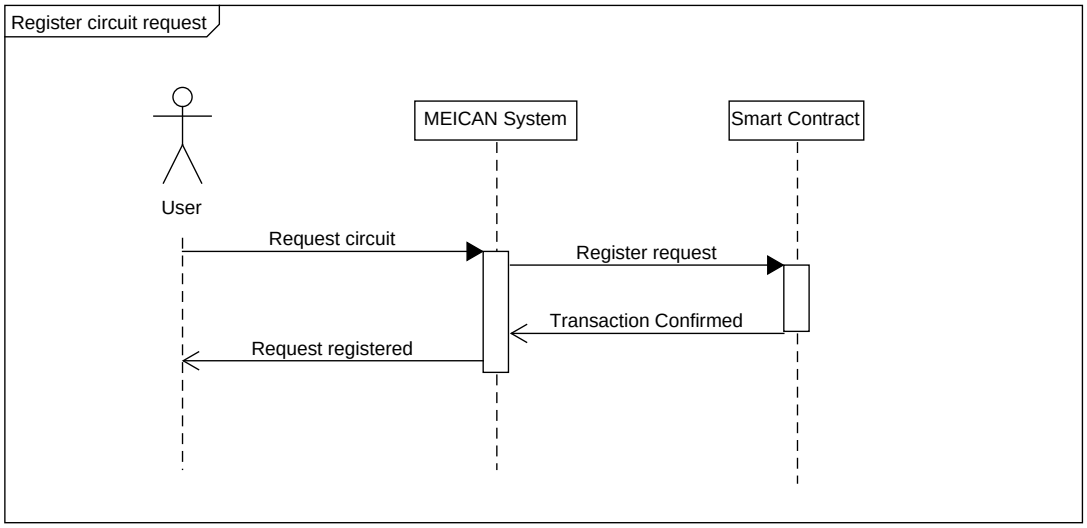
## 4.1 Design

This chapter will present the architecture of the solution, as well as how it was implemented. The solution is based on the main MEICAN flow, which is explained in Section 2.4. The proposal is to add another stage to the flow to increase security and transparency in the process of allocating inter-domain circuits. To achieve this goal, the introduction of blockchain and smart contracts is suitable, since the blockchain ensures a transparent audit process for circuit requests, and the smart contract can automate the registration of these requests in the blockchain. This new stage counts with an interface that performs communication between MEICAN and the smart contract, adding a new layer of security to the circuit request management.

Four essential use cases were defined so that the entire circuit request lifetime is covered. These cases would be:

- Register circuit request: the user enters the circuit request data and sends it to the interface, which makes a function call to the smart contract, so that it registers a pending circuit request in the blockchain. After this process, the blockchain returns whether the transaction was successful, and this response is then propagated to the user. This case is highlighted in Figure 4.1.
- Check user circuits (user view): the user enters their address in the blockchain and sends it to the interface, which makes a function call to query the blockchain

Figure 4.1: Register Circuit Request Sequence Diagram




Source: The Author

through the smart contract. The blockchain returns all circuit requests for the given address.

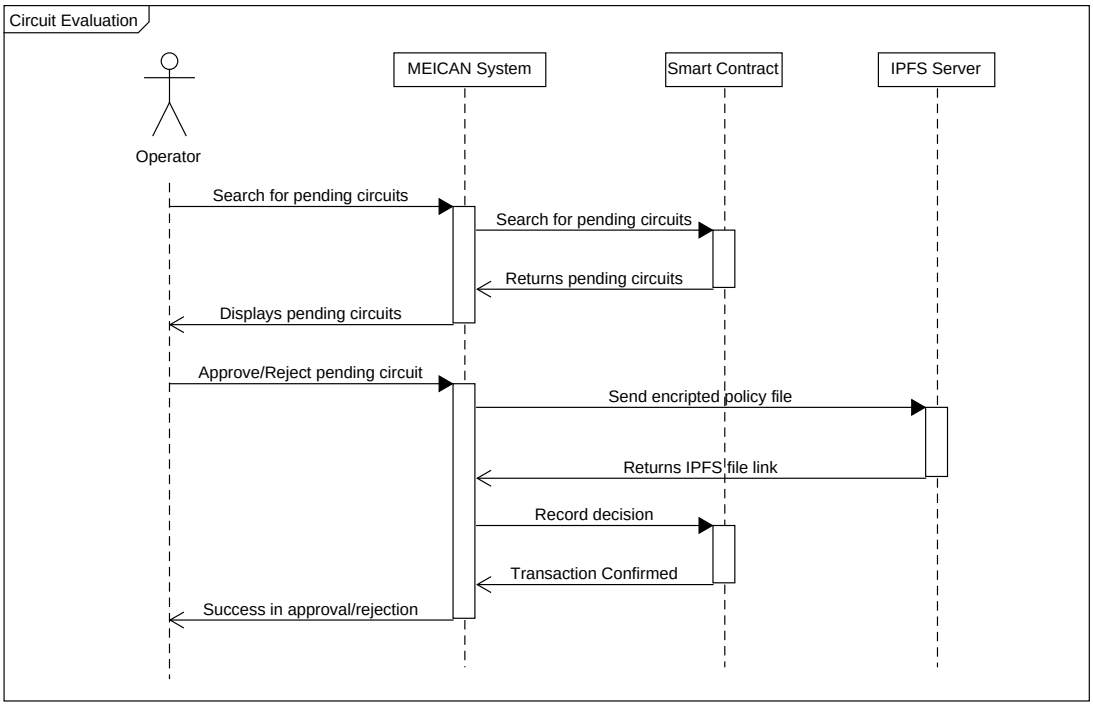
- Check pending circuits (operator view): The interface tells the smart contract to perform a query for all requests with pending status. The blockchain, upon receiving the request, returns all pending circuit requests registered in the blockchain.
- Circuit evaluation (operator view): When viewing pending circuit requests, the operator can proceed to evaluate a request, which consists of deciding whether this request will be approved or rejected, in addition to sending the policy used for that decision. This decision is also recorded in the blockchain, along with the policy. This case is illustrated in Figure 4.2.

Both Check user circuits and Check pending circuits cases have a diagram very similar to that of Figure 4.1, where the only changes are the contents of the messages between the user, the MEICAN system and the Smart Contract.

### 4.1.1 Policies

Inter-domain circuit allocation in MEICAN involves the interaction between multiple administrative domains, each with its own authorization policies and control rules. Due to the sensitive nature of these policies, it is essential to ensure that each MEICAN instance maintains autonomy to decide which policies it wants to expose and how it in-

Figure 4.2: Circuit Evaluation Sequence Diagram




Source: The Author

tends to make them available. In this context, two alternatives for storing these policies on the blockchain were discussed, seeking to balance privacy and auditability requirements.

#### *4.1.1.1 Alternative 1: Recording Policy Hashes on the Blockchain*

In this approach, instead of storing the full content of the policies on the blockchain, each MEICAN instance generates a cryptographic hash of the corresponding rules. This hash is then recorded on the blockchain and can be used later for integrity verification purposes, without the need to publicly disclose the content of the policies. The original policy document remains stored off-chain, on a secure server or in a distributed repository.

Each MEICAN instance defines its authorization policy in an external file, which is later stored on a decentralized service, such as IPFS. A cryptographic hash is generated from the policy content. The generated hash, along with the link to the file in IPFS, is then stored in a smart contract on the blockchain. This mechanism allows any interested entity to validate that the file has not been altered by comparing the hash stored on the blockchain with a new hash generated from the original file. If necessary, an operator can access the policy link and verify its content, maintaining the trust and integrity of the process.

The advantages:

- Privacy: No sensitive information is publicly exposed.
- Auditability: The hash allows the authenticity of the policy to be verified without revealing its content.
- Efficiency: Significantly reduces storage costs on the blockchain.

The disadvantages:

- Reliance on secure external storage: The integrity and availability of policies depend on external repositories to store the original documents.

In addition, each MEICAN instance can define the users authorized to access the policy content. The link stored in IPFS contains the encrypted policy file, and only the requesting user, in possession of their private key, will be able to decrypt it and access the policy content.

#### *4.1.1.2 Alternative 2: Complete Registration of Policies on the Blockchain*

In this method, authorization policies are stored directly on the blockchain, with the content protected by Identity-Based Encryption mechanisms (IBE) or by means of private keys shared between trusted domains.

Before being registered on the blockchain, policies are encrypted, ensuring that only authorized entities — usually other trusted domains — can decrypt and consult the information. The blockchain, in turn, acts as an immutable repository, ensuring that any attempts to modify the policies are detectable and require consensus among network participants.

The advantages:

- Maximum auditability: Enables complete and verifiable registration of policies, with the history of changes preserved on the blockchain.
- Access control: Only previously authorized domains can view the content of the policies.
- Increased trust between domains: Rules can be shared securely, reducing the risk of divergent interpretations.

The disadvantages:

- Higher computational cost: The application of cryptographic mechanisms adds

complexity and demands greater processing capacity.

- Risk of private key compromise: If a domain loses its private key, it may be unable to access its registered policies.
- High storage cost: Since policies are fully registered on the blockchain, there is a significant increase in gas consumption in public blockchains, impacting the cost of operation.

#### *4.1.1.3 Policy Storage Method*

Considering the advantages and disadvantages of both alternatives, the first alternative was chosen; thus, it has a minor computational and storage cost than the second alternative, has more scalability, and it can also increase the trust between the domains.

The solution developed for this work assumes that all circuit requests will be evaluated manually by an operator (*i.e.*, there is no automatic integration between the blockchain and the authorization workflow yet). However, for future work, a more advanced integration is planned, allowing the decision-making process regarding circuits to be automated based on the information recorded in the blockchain.

### 4.1.2 Encryption Scheme

Given that the policies of each MEICAN instance contain sensitive data, a hybrid encryption scheme was adopted to ensure the confidentiality of these files while guaranteeing that access would be restricted exclusively to the user who initiated the circuit request. Symmetric encryption was used to protect the actual content of the policy file. This method employs the same secret key for both encryption and decryption, offering high processing speed and efficiency when handling large volumes of data (MENEZES; OORSCHOT; VANSTONE, 2018). To control access to the symmetric key itself, asymmetric encryption was applied. This method relies on a pair of mathematically related keys: a public key, which is used to encrypt the symmetric key, and a private key, which remains solely in the possession of the requesting user and is required for decryption. This ensures that only the intended recipient can retrieve and use the symmetric key to access the file contents (WILLIAM, 2017).

The combination of these techniques allows the system to benefit from the performance advantages of symmetric encryption for data protection, while also leveraging

the security and access control features provided by asymmetric encryption (KATZ; LINDELL, 2014).

## 4.2 Implementation

The proposed architecture, presented in Figure 4.3, integrates the MEICAN system with blockchain technology and decentralized storage via IPFS, ensuring greater transparency, security, and auditability in the management of inter-domain circuit requests.

Users and operators interact with MEICAN through a web interface, where they make requests and evaluate circuits. These operations are managed by the internal Request Manager module, responsible for controlling the flow of requests within the MEICAN infrastructure. An API layer implements functionalities for managing requests, incorporating an encryption module, integration with IPFS, and the Ethereum blockchain. Before being sent to IPFS, the policy file and relevant metadata undergo an encryption process, ensuring the confidentiality of the stored data.

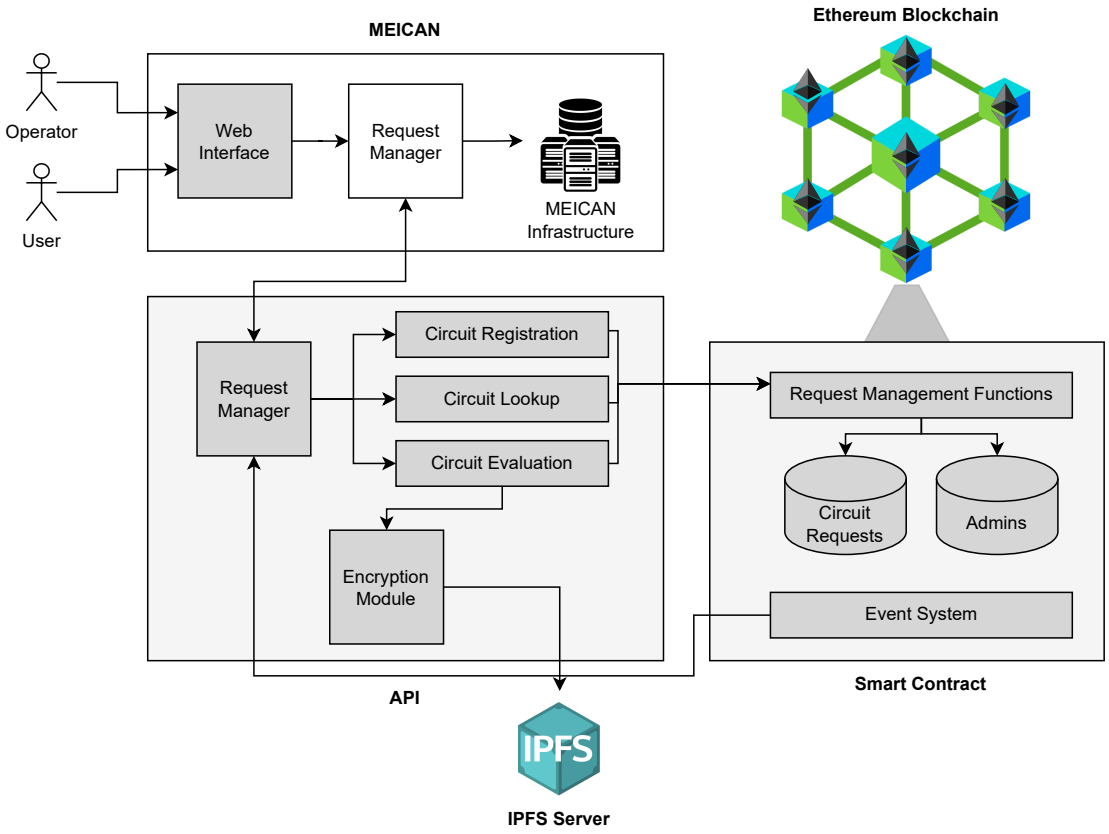
The smart contract, implemented on the blockchain, is responsible for the immutable record of requests, storing all the information recorded in the circuit request, in addition to the hash of the policy file, the IPFS link to this file, and the approval or rejection status. In addition, an event system allows MEICAN, via API, to monitor changes in the status of requests in real time.

With this architecture, the proposed solution offers a secure and distributed environment for storing data and recording administrative decisions, reinforcing trust between domains in the circuit authorization process. The elements presented in this architecture will be better discussed throughout this section. The complete source code of the implemented solution is available for public consultation in the following GitHub repository: https://github.com/gmazevedo/meican-smart-contract-api.

### 4.2.1 Smart Contract

The Smart Contract was developed using the Solidity programming language, which is predominant in development for the Ethereum platform and compatible platforms. Internally, the contract defines the following main data structures:

Figure 4.3: Solution's Proposed Architecture




Source: The Author

- CircuitRequest: Structure that represents a circuit request on the blockchain, storing its fundamental attributes.
- Requests: Mapping (of the mapping type in Solidity) that stores the registered CircuitRequest instances, using a unique identifier as a key.
- CircuitParams: Data structure designed to store the detailed parameters associated with each circuit request.

In addition, the contract emits the following events, which are crucial for auditing and tracking transactions on the blockchain:

- CircuitRequested: Emitted when a new circuit request is registered.
- CircuitApproved: Emitted when a circuit request is approved.
- CircuitRejected: Emitted when a circuit request is rejected.

The functions implemented for registering and querying data on the blockchain are as follows:

- requestCircuit(params CircuitParams): Responsible for registering a new circuit

request on the blockchain, receiving the detailed parameters of the request.

- approveCircuit(requestId uint256): Allows the approval of a specific circuit request, identified by its requestId, updating its status to "Approved".
- rejectCircuit(requestId uint256): Allows the rejection of a specific circuit request, identified by its requestId, updating its status to "Rejected".
- getCircuitRequest(requestId uint256) returns (CircuitRequest memory): Retrieves and returns the details of a specific circuit request, based on its requestId.

#### *4.2.1.1 Access Control Mechanism in the Smart Contract*

An access control system was implemented in the Smart Contract to ensure that only authorized entities (contract administrators) can perform the sensitive functions of approving (`approveCircuit()`) or rejecting (`rejectCircuit()`) requests. Similarly, only the original creator of the contract (*i.e.*, owner) has the prerogative to add or remove addresses from the list of administrators.

This control is implemented through an internal mapping (admins) that associates blockchain addresses with a boolean value, indicating the administrator status. The `onlyAdmin` function modifier restricts the execution of the annotated function exclusively to addresses registered as administrators. Similarly, the `onlyOwner` modifier ensures that only the creator of the contract can perform specific functions, such as managing administrators. The `addAdmin(address newAdmin)` and `removeAdmin(address admin)` functions are used to manage the list of administrators.

This mechanism aims to allow the contract creator (potentially a MEICAN operator) to delegate the evaluation capability to operators of the various MEICAN instances by adding their respective blockchain addresses to the administrator mapping.

#### *4.2.1.2 Blockchain Infrastructure Configuration*

For the development, testing, and local deployment of the smart contract used in this solution, a private blockchain environment was configured using the Truffle and Ganache tools.

Truffle is one of the most widely used frameworks for developing DApps on the Ethereum platform. It offers a complete environment for compiling, deploying, testing, and managing smart contracts, in addition to providing a simplified interface for interact-

ing with the local or public blockchain (ConsenSys Software Inc., 2024b).

The configuration was performed through the truffle-config.cjs file, in which a local network environment called development was defined. This configuration establishes the connection with an Ethereum client running on the local machine (`localhost`), using the default port 8545 and accepting the Ethereum Mainnet `network_id`. The relevant configuration snippet is presented in Listing 4.1.

Listing 4.1 – Development Environment Configuration

```
development: {
    host: "127.0.0.1", // Localhost address
    port: 8545, // Ganache default port
    network_id: "1", // Ethereum Mainnet identifier
}
```

This configuration allowed the deployment and testing of smart contracts in an isolated and controlled manner, without the need to expose transactions to a public network.

Ganache was used as the local Ethereum client, operating as a private development blockchain. It offers a fully developer-controlled simulation of the Ethereum network, allowing the creation of dummy accounts with ether balances, block manipulation, and detailed real-time transaction monitoring (ConsenSys Software Inc., 2024a).

Listing 4.2 – Truffle Compiler Configuration

```
compilers: {
    solc: {
        version: "0.8.21",
        settings: {
            optimizer: {
                enabled: false,
                runs: 200
            },
        viaIR: true
        }
    }
}
```

With Ganache, it was possible to test features such as logging circuit requests,

approving or rejecting requests, and storing on-chain metadata, before any possible migration to a public or production network. In the Truffle configuration file, the Solidity compiler version to be used was also defined, ensuring compatibility with the source code of the developed smart contracts. This is presented in Listing 4.2.

This configuration ensures that the Solidity code is compiled with version 0.8.21, with the optimizer disabled and with the use of Intermediate Representation (IR) enabled, which contributes to greater efficiency in generating bytecode for the Ethereum Virtual Machine (EVM) (Ethereum Foundation, 2024b).

### 4.2.2 User Interaction

The developed solution presents a simple and functional web interface, focused on the interaction of two user profiles: the circuit requester and the MEICAN operator. These pages are intended to simulate how MEICAN would interact with the API that communicates with the smart contract. The main pages of the application are described below.

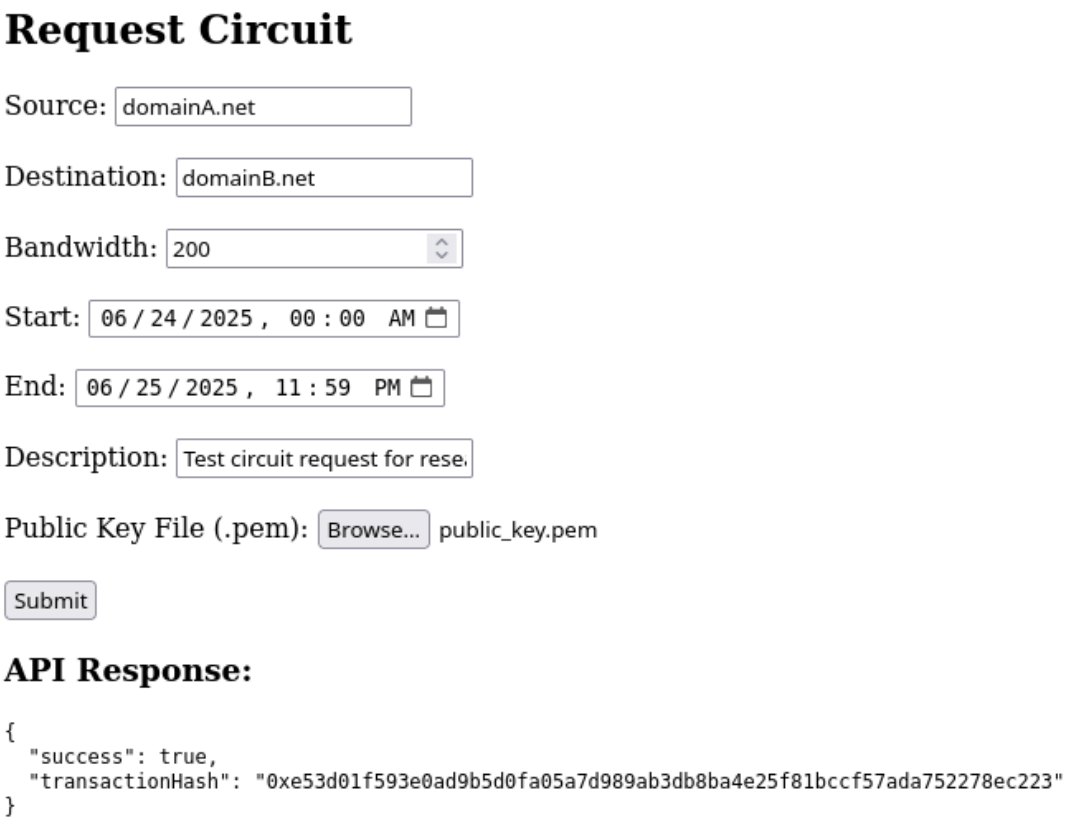
#### *4.2.2.1 Circuit Request*

In the Circuit Request interface, the user can fill in the data related to the desired circuit, including origin, destination, required bandwidth, activation period, description, and public key (in a .pem file). After sending the request, it is registered on the blockchain, with the parameters stored in the smart contract. The hash of the request transaction is displayed as feedback to the user, validating the success of the operation. This interaction is illustrated in Figure 4.4.

#### *4.2.2.2 Query Circuits by Address*

On the Query Circuits by Address page, any user can query the requests made by their address (wallet). The interface allows entering the address and viewing all associated requests, with status, policy hash (when available), and IPFS link (also when available). This provides traceability and auditability for all interactions with the system. This interaction is presented in Figure 4.5.

Figure 4.4: Circuit Request Web Page




Source: The Author

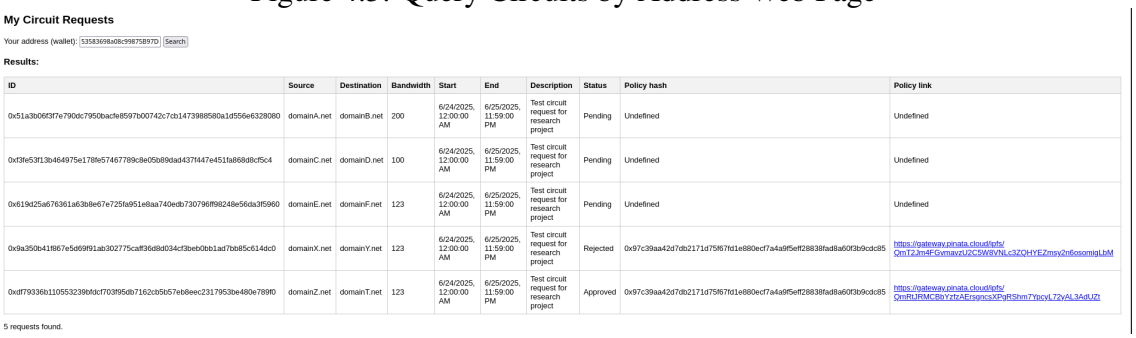
#### *4.2.2.3 View Pending Circuits*

The View Pending Circuits page is dedicated to operators (users with administrative permissions on the MEICAN instance). This interface lists all requests with a "Pending" status, displaying data such as ID, source, destination, bandwidth, activation period, and description. Each row has an Evaluate button, which redirects to the individual request evaluation page. This interaction is presented in Figure 4.6.

#### *4.2.2.4 Circuit Request Evaluation*

On the Circuit Request Evaluation page, the operator sees the request ID, the requester's public key, and can select a policy file to upload. This file is encrypted with the requester's public key, stored on IPFS via Pinata, and the resulting link is registered on the blockchain with the hash of the content. The operator then decides whether to Approve or Reject, and the request status is updated accordingly. This interaction is shown in Figure 4.7.

Figure 4.5: Query Circuits by Address Web Page



My Circuit Requests

Your address (wallet): 53583698a08c99875B97D Search

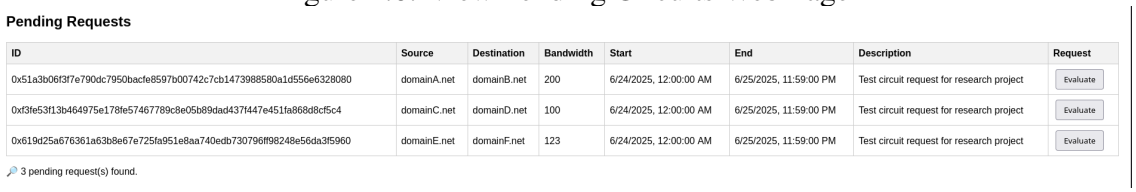
Results:

| ID | Source | Destination | Bandwidth | Start | End | Description | Status | Policy hash | Policy link |
|---|---|---|---|---|---|---|---|---|---|
| 0x51a3b06f3f7e790dc7950bacfe8597b00742c7cb1473988580a1d556e6328080 | domainA.net | domainB.net | 200 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Pending | Undefined | Undefined |
| 0xf3fe53f13b464975e178fe57467789c8e05b89dad437f447e451fa868d8cf5c4 | domainC.net | domainD.net | 100 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Pending | Undefined | Undefined |
| 0x619d25a676361a63b8e67e725fa951e8aa740edb730796ff98248e56da3f5960 | domainE.net | domainF.net | 123 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Pending | Undefined | Undefined |
| 0x9a350b41f867e5d69f91ab302775caff36d8d034cf3beb0bb1ad7bb85c614dc0 | domainX.net | domainY.net | 123 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Rejected | 0x97c39aa42d7db2171d75f67fd1e880ecf7a4a9f5eff28838fad8a60f3b9cdc85 | https://gateway.pinata.cloud/ipfs/QmT2JtnfFGvmavzU2CSWRVNLc3ZQHYEZmsy2n6osomigLbM |
| 0xdf79336b110553239bfdcf703f95db7162cb5b57eb8eec2317953be480e789f0 | domainZ.net | domainT.net | 123 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Approved | 0x97c39aa42d7db2171d75f67fd1e880ecf7a4a9f5eff28838fad8a60f3b9cdc85 | https://gateway.pinata.cloud/ipfs/QmRtJRMCBbYztzAErsgncsXPgRShm7YpcyL72yAL3AdUZt |

5 requests found.

Source: The Author

Figure 4.6: View Pending Circuits Web Page



Pending Requests

| ID | Source | Destination | Bandwidth | Start | End | Description | Request |
|---|---|---|---|---|---|---|---|
| 0x51a3b06f3f7e790dc7950bacfe8597b00742c7cb1473988580a1d556e6328080 | domainA.net | domainB.net | 200 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Evaluate |
| 0xf3fe53f13b464975e178fe57467789c8e05b89dad437f447e451fa868d8cf5c4 | domainC.net | domainD.net | 100 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Evaluate |
| 0x619d25a676361a63b8e67e725fa951e8aa740edb730796ff98248e56da3f5960 | domainE.net | domainF.net | 123 | 6/24/2025, 12:00:00 AM | 6/25/2025, 11:59:00 PM | Test circuit request for research project | Evaluate |

3 pending request(s) found.

Source: The Author

## 4.3 API

The API is responsible for communication between the MEICAN system, the blockchain via smart contract, and the IPFS decentralized storage platform, through the Pinata service. It was implemented using Node.js and the Express.js library.

### 4.3.1 Connection to the Blockchain

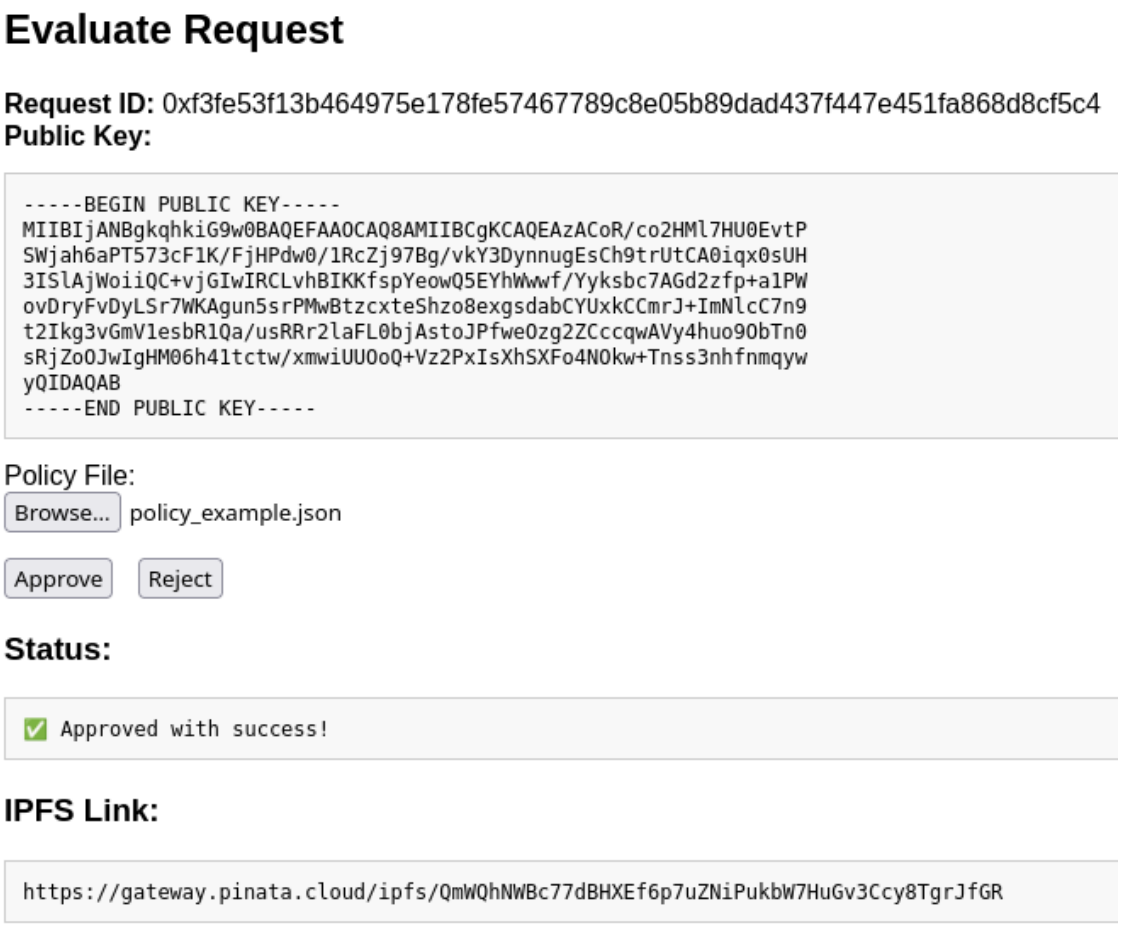
The system connects to the Ethereum blockchain through a remote node, whose address is specified in the environment variables. Authentication and authorization of transactions on the blockchain are performed with a private key also defined in the execution environment. The backend interacts directly with a previously deployed MEICAN smart contract, using the Application Binary Interface (ABI) stored in a local JSON file. The contract manages inter-domain circuit requests.

### 4.3.2 Implemented Functionalities

The code provides the following Representational State Transfer (REST) API routes:

- Circuit Request (`/requestCircuit`): Receives the parameters of the circuit

Figure 4.7: Circuit Request Evaluation Web Page




Source: The Author

request, including technical details (origin, destination, bandwidth, period, etc.) and the public key of the requesting user, registering the request directly on the blockchain.

- Circuit Approval or Rejection (`/approveCircuit` and `/rejectCircuit`): Allows the system administrator to approve or reject circuit requests. In this step, a previously encrypted policy file is processed and its hash is calculated using the Keccak-256 cryptographic hash function, standard in Ethereum (WOOD et al., 2014). This hash, along with a link to the file hosted on IPFS, is recorded on the blockchain, ensuring integrity and traceability.
- Circuit Query by User (`/getUserCircuit`): Allows a user to query all circuit requests that he or she has submitted, based on events stored on the blockchain.
- Pending Request List (`/getPendingCircuits`): Retrieves all circuit requests still in a pending state, allowing for their subsequent analysis and approval or rejec-

tion decision.

- Query by Request ID (`/getRequestById`): Allows you to get the details of a specific circuit request based on its unique identifier.
- Upload Policy Files to IPFS (`/uploadPolicy`): Uploads policy files (already encrypted on the frontend) to IPFS via Pinata. After uploading, the backend returns the generated public link, which is later stored in the smart contract.

### 4.3.3 Policy Encryption

For this work, a hybrid encryption approach was implemented to ensure the protection of policy files associated with circuit requests. This technique combines the strengths of symmetric and asymmetric encryption, resulting in a secure and efficient solution for handling sensitive data (MENEZES; OORSCHOT; VANSTONE, 2018) (STALLINGS, 2017). The process is executed directly in the evaluator's browser, ensuring that the original content of the file is never transmitted or stored in clear text during communication with the server.

Encryption begins with the local generation of a random 256-bit key, which will be used with the Advanced Encryption Standard (AES) algorithm in Cipher Block Chaining (CBC) mode to encrypt the content of the policy file. To ensure variability between executions, a 128-bit Initialization Vector (IV) is also generated. The file is then converted to a memory buffer and encrypted with the symmetric key and IV using the `crypto.subtle.encrypt` API, which offers native support for cryptographic algorithms in the browser environment (MDN Web Docs, 2024).

Once the file has been encrypted, the symmetric key must be protected so that only the requester of the circuit request can access it. To do this, asymmetric encryption is applied: the symmetric key, converted to base64, is encrypted with the requester's Rivest-Shamir-Adleman (RSA) public key, previously obtained from the blockchain. This process is done with the `JSEncrypt` library, widely used for RSA operations on the client side (TIDWELL, 2024). Since asymmetric encryption is not suitable for large volumes of data, its use is limited to protecting the symmetric key (STALLINGS, 2017).

After encryption, both the encrypted file and the metadata required for its recovery (including the IV and the encrypted AES key) are organized into a .zip package, dynamically created with the `JSZip` library (KNIGHTLEY, 2024). This package is sent to

the backend, which transfers it to IPFS via the Pinata service (BENET, 2014) (Pinata Technologies Inc., 2024b).

The choice of a hybrid model is justified by the practical limitations of asymmetric encryption algorithms such as RSA, which have restrictions on the size of the data that can be directly encrypted, in addition to presenting higher computational costs for large files (RIVEST; SHAMIR; ADLEMAN, 1978). On the other hand, symmetric encryption algorithms such as the AES are more efficient for processing significant volumes of data (DAEMEN; RIJMEN, 2002). Thus, asymmetric encryption remains solely responsible for protecting the symmetric key, ensuring access control to the encrypted content.

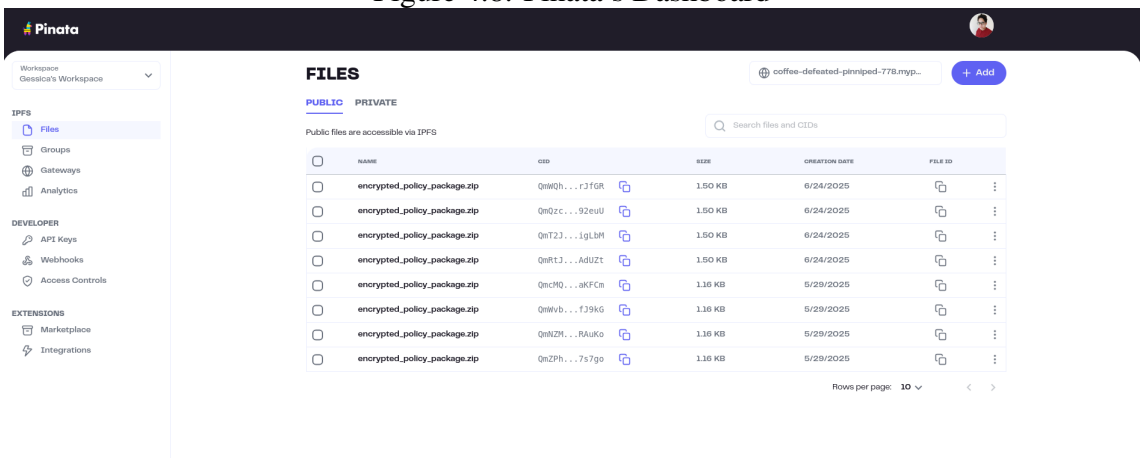
## 4.4 IPFS Server

One of the requirements of this work was to store policy files in a decentralized and accessible way, without overloading the blockchain with large volumes of data. To meet this requirement, we adopted the use of IPFS in conjunction with the Pinata service.

Pinata is a commercial platform that provides file hosting and management services on IPFS. Its main function is to ensure the persistence and availability of content on the IPFS network. While any user can add files to IPFS, there is no native guarantee that the nodes on the network will keep these files stored permanently. Pinata solves this problem by offering a pinning service, which keeps files pinned and permanently accessible on its nodes on the network (Pinata Technologies Inc., 2024b). In addition, Pinata provides a RESTful API that allows applications, like our API, to perform programmatic file uploads, query IPFS links, and monitor the status of hosted content.

The Node.js backend was configured to send encrypted policy files to Pinata, using the Pinata Software Development Kit (SDK). After uploading, Pinata returns the corresponding Content Identifier (CID), which is used to generate a public link to access the file, usually in the format: https://gateway.pinata.cloud/ipfs/{CID}. This link is then stored on the blockchain along with the file hash, ensuring that any future changes to the file would invalidate the recorded hash. Using IPFS and Pinata therefore provides decentralized distribution, content integrity, and continuous availability, without overloading the blockchain network with large file storage. The Pinata's dashboard is presented in Figure 4.8.

Figure 4.8: Pinata's Dashboard




Source: (Pinata Technologies Inc., 2024a)

# 5 EVALUATION AND RESULTS

This chapter presents the evaluation of the implemented system, focusing on the analysis of gas consumption, transaction costs, and execution times of the developed solution. The assessment covers both blockchain transaction costs and the performance of the integrated workflow, including encryption, file uploads to IPFS, and interactions with the smart contract. Through this evaluation, it is possible to understand the practical operational behavior of the system, identify potential bottlenecks, and establish a baseline for future scalability and optimization of the proposed architecture.

## 5.1 Blockchain Transactions

During the analysis of performance and gas cost of the implemented solution, it was essential to distinguish between the two main types of functions in a Solidity smart contract: read-only functions (*view*) and state-changing functions (*non-view*).

Functions that perform changes in the blockchain state, such as insertions or modifications in *mappings*, *structs*, or any permanent storage variables, require the creation of a transaction to be executed. The following functions of the developed contract fall into this category:

- `requestCircuit()`
- `approveCircuit()`
- `rejectCircuit()`
- `addAdmin()`
- `removeAdmin()`

Each execution of these functions involves actual gas consumption since it affects the blockchain's storage. The gas cost is directly proportional to the computational complexity of the function and the volume of data being modified (BUTERIN et al., 2014).

On the other hand, functions declared as *view* are designed exclusively to query data already stored on the blockchain without performing any state modification. These functions can be called by external applications (*off-chain*), such as web frontends or monitoring scripts, without the need to send a transaction. This means there is no gas consumption or financial cost when performing such queries (Ethereum Foundation, 2025). The function `getCircuitRequest()` is an example of a *view* function and,

for this reason, is not included in the cost calculation presented in Table 5.1. This function allows querying the details of a specific circuit request, returning all fields of the `CircuitRequest` *struct*. As it is read-only, its execution is free when performed *off-chain* via `call()` methods (using tools like Truffle or Ethers.js).

### 5.1.1 Gas Cost Calculation

To estimate the financial cost of executing the smart contract functions on a public Ethereum network, it is necessary to convert gas consumption into monetary values, considering the average *Gas Price* and the current exchange rate of *ether* (ETH) against the US dollar (USD).

#### *5.1.1.1 Cost Calculation in Ether (ETH)*

The cost of each transaction in ETH is calculated by multiplying the number of gas units consumed by the transaction by the current *Gas Price* on the network, as shown in Equation 5.1 (Etherscan, 2025).

$$CostinETH = GasUsed \times GasPrice \tag{5.1}$$

Considering that the *Gas Price* is usually expressed in Gwei, and knowing that $1\ Gwei = 10^9\ wei$ and $1\ ETH = 10^{18}\ wei$, the equation can be detailed as:

$$CostinETH = GasUsed \times \left(\frac{GasPrice(Gwei)}{10^9}\right) \times \left(\frac{1}{10^9}\right) \tag{5.2}$$

For this study, a *Gas Price* of $0.43\ Gwei$ was considered, representing the average value observed on the Ethereum Mainnet in June 2025 (Etherscan, 2025).

#### *5.1.1.2 Conversion to US Dollars (USD)*

To convert the obtained ETH cost to USD, the average exchange rate of *ether* for the same period was used, estimated at approximately $\$2,273$ per ETH (CoinMarketCap, 2025). The formula used is shown in Equation 5.3.

$$CostinUSD = CostinETH \times ETHPrice(USD) \tag{5.3}$$

*5.1.1.3 Results*

Table 5.1 presents the gas consumption values, the estimated cost in ETH, and the approximate cost in USD for each contract function, considering the Ethereum Mainnet scenario with a *Gas Price* of $0.43\ Gwei$.

Table 5.1: Gas Consumption and Estimated Cost per Function on Ethereum Mainnet (Gas Price = 0.43 Gwei)

| **Function** | **Gas Used** | **Estimated Cost (ETH)** | **Estimated Cost (USD)** |
|---|---|---|---|
| `requestCircuit()` | 277,152 | 0.000119176 | $0.27 |
| `approveCircuit()` | 100,734 | 0.000043316 | $0.10 |
| `rejectCircuit()` | 100,770 | 0.000043331 | $0.10 |
| `addAdmin()` | 47,791 | 0.000020550 | $0.05 |
| `removeAdmin()` | 25,700 | 0.000011051 | $0.03 |

The simulation was conducted with both the *Chain ID* and *Network ID* set to 1, replicating the official identifiers of the Ethereum Mainnet. This configuration does not turn the local environment into a public network but allows transaction execution behavior and gas calculation to be similar to those observed in production (ConsenSys Software Inc., 2024a).

The parameters used for the testing environment were as follows:

- Chain ID: 1
- Network ID: 1
- Block Gas Limit: $30,000,000$ gas units per block
- Gas Price: $0.43$ Gwei

During the tests conducted in the Ganache environment, each smart contract function was called in isolation for precise gas consumption measurement. The results obtained indicate that the `requestCircuit()` function presented the highest consumption, directly reflecting the complexity of the operation, which involves creating a new data structure and emitting an event.

Simple state update functions, such as `addAdmin()` and `removeAdmin()`, showed significantly lower consumption, while the functions `approveCircuit()` and `rejectCircuit()`, which involve multiple storage updates and event emissions, registered intermediate consumption levels.

These values serve as a basis for future cost projections, should the contract be

deployed on a public network such as the Ethereum Mainnet.

## 5.2 Performance Analysis

Figure 5.1 presents the average execution times of the main functions of the developed API, calculated from 20 test runs, considering the end-to-end execution of circuit-registration flows, request evaluation, and policy storage. These times reflect the practical behavior of the implemented system, which integrates the web pages and API with the smart contract on the blockchain and decentralized storage via IPFS.

In this work, the Ethereum mainnet block-mining time (approximately 12 seconds per block, according to (Ethereum Foundation, 2024a)) was not taken into account. Instead, we adopted Ganache's default behavior, which insta-mines a block as soon as a transaction arrives. Although this interval is dramatically shorter than the real mainnet cadence, it still introduces overhead to the MEICAN solution: Every state-changing function call must be wrapped into a transaction, sent to the local node, included in the newly generated block, and only then can its receipt be returned.

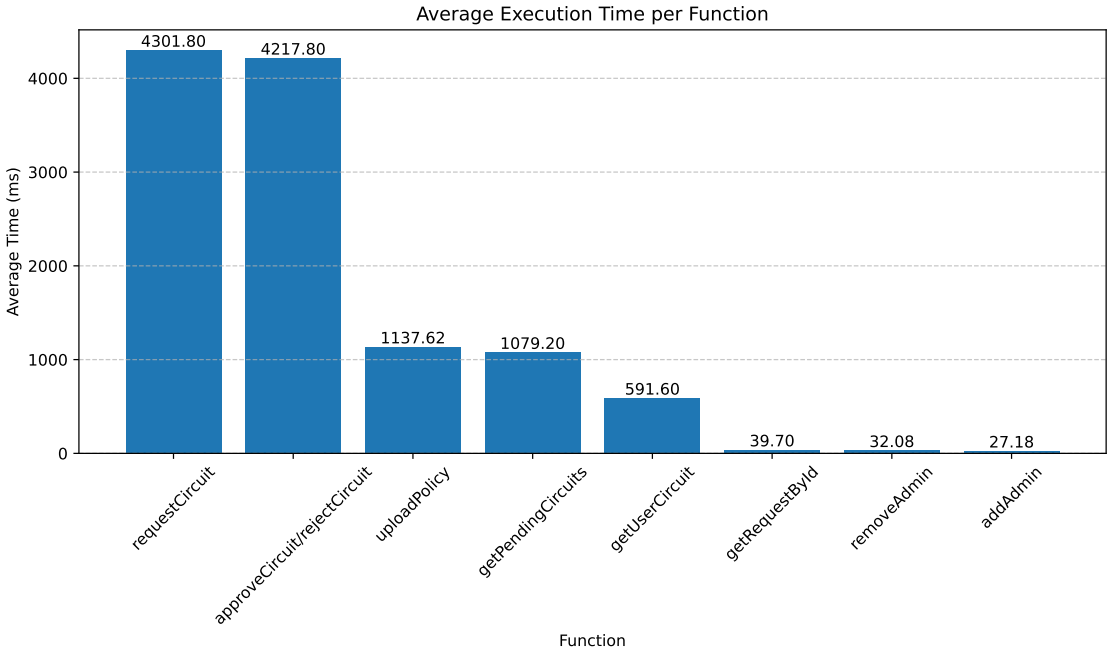
The highest times observed correspond to the `requestCircuit`, `approveCircuit`, and `rejectCircuit` functions, with average values above four seconds. These times are justified by the complexity of the involved workflow, which includes file reading operations, hybrid encryption in the browser, compression into ZIP format, and subsequent upload of the file to IPFS. Additionally, when approving or rejecting a circuit, a transaction is executed on the smart contract to register the file hash and the link generated on IPFS, which adds time due to the mining and block confirmation process on the blockchain network.

Functions that presented intermediate times, such as `uploadPolicy` and `getPendingCircuits` has a processing load associated with uploading files to Pinata and querying smart contract events to identify pending requests. In the case of policy uploads, the backend receives the compressed file, performs the upload via the Pinata SDK, and returns the generated IPFS link to the client, while the search for pending requests involves filtering the contract logs and mapping them to the JSON structure used by the operator interface.

Other functions, such as `getUserCircuit`, `getRequestById`, `addAdmin`, and `removeAdmin`, showed significantly lower execution times, ranging from tens to a few hundred milliseconds. This occurs because these functions only perform data reads

from the smart contract or simple mapping update transactions, without encryption operations or file uploads, resulting in lower gas usage and lower computational complexity.

Figure 5.1: Average Execution Time per Function




Source: The Author

This demonstrates how the execution time of the functions is directly associated with the technical flow adopted in our solution, being influenced mainly by client-side encryption stages, file upload to IPFS, and interaction with the smart contract on the blockchain. In practical applications, this analysis makes it possible to identify system bottlenecks, highlighting the need for optimizations in the encryption stages and in the decentralized storage process if the system needs to be scaled for production environments with a higher volume of requests and multiple simultaneous users.

## 5.3 Discussion and Limitations

The development of the proposed solution highlighted several practical and architectural aspects that need to be critically assessed to guide future improvements and potential production deployments. While the implemented architecture successfully demonstrates the feasibility of using blockchain and decentralized storage to register, evaluate, and store inter-domain circuit requests, it also exposes inherent limitations related to scalability, cost, and operational complexity. Additionally, the choice of underlying technologies, such as Ethereum for smart contracts and IPFS for decentralized file storage,

imposes constraints that require careful consideration regarding their impact on performance, usability, and maintainability of the system in real-world environments.

The following sections present a detailed discussion of the identified limitations, their implications for system scalability and efficiency, and the possible adaptations of the proposed model to alternative public or private blockchains, as well as the operational considerations associated with decentralized file storage using IPFS.

### 5.3.1 Blockchain's Limitations

Despite its transformative potential, blockchain technology presents several limitations that must be considered when defining architectures and solutions based on distributed records. One of the main limitations of blockchain is scalability. Public networks, such as Ethereum and Bitcoin, have a limited capacity of transactions per second, which restricts their use in applications with high transactional demand (VUKOLIć, 2016). This is due to the distributed consensus process, which guarantees security but adds latency and reduces throughput.

Another significant limitation is the cost of storing and executing smart contracts. Each operation consumes "gas", a unit that represents computational effort. Storing large volumes of data directly on the blockchain is financially unfeasible, which motivates hybrid approaches with external storage, such as IPFS (BENET, 2014).

Immutability is a central feature of blockchain, ensuring integrity and auditability. However, it also poses a risk: incorrect or malicious data recorded cannot be changed or removed. This requires special attention to validating information before recording and defining governance mechanisms to mitigate errors (ZHENG et al., 2018). Also, in public blockchains, all transactions and contracts are visible to any participant in the network. This can be a limitation for applications that require confidentiality, such as contracts with sensitive data or strategic information. Solutions such as homomorphic encryption, zk-SNARKs, and permissioned blockchains have been proposed to mitigate this problem, but they still present technical and performance challenges (MIERS et al., 2013).

The lack of consolidated standards makes interoperability between different blockchains difficult. Applications that need to interact with multiple networks or integrate data from legacy systems face compatibility barriers, which limit adoption in more complex corporate environments (HARDJONO; LIPTON; PENTLAND, 2019). Although the proposed solution was evaluated using the Ethereum Mainnet parameters and

simulated in a local Ganache environment, the architectural model and implementation strategy are adaptable to other blockchain platforms.

### 5.3.2 Limitations of Using IPFS Servers

IPFS is a decentralized P2P technology designed for immutable file storage and distribution. However, despite its advantages in terms of integrity and censorship resistance, the use of IPFS presents some important operational limitations that need to be considered in distributed systems that require high availability and confidentiality.

By default, the IPFS network does not guarantee that files will remain available after they have been inserted. This is because IPFS nodes only store data temporarily unless it is explicitly "pinned" by a node (BENET, 2014). To ensure continuous availability, it is necessary to use external pinning services such as Pinata or Web3.Storage. Also, the IPFS performance can be negatively impacted when the requested content is stored on a few nodes or nodes with low availability. File retrieval latency tends to be higher compared to traditional servers, especially for files that are not widely replicated (BENET, 2014).

Another critical challenge is data privacy. Since files are publicly accessible via their CID hash, anyone with this identifier can retrieve the content. To mitigate this issue, hybrid encryption techniques (a combination of AES and RSA) are often adopted before uploading, ensuring that only authorized recipients can access sensitive content (TIDWELL, 2024).

Using IPFS gateways is essential to keep files active and accessible on the network. While Pinata is widely used for its intuitive interface and API support, there are other options with different approaches and business models, which can be seen in Table 5.2.

Table 5.2: Comparison between IPFS gateways

| Gateway | Features | Pricing Model |
|---|---|---|
| **Pinata** | Intuitive interface, pinning support via REST API | Free up to 1GB; paid plans (Pinata, 2024) |
| **Web3.Storage** | Storage on Filecoin, operated by Protocol Labs | Free up to 5GB (Web3.Storage, 2023) |
| **NFT.Storage** | Optimized for Non-Fungible Tokens (NFTs), also uses Filecoin for retention | Free (NFT.Storage, 2023) |
| **Infura IPFS** | ConsenSys' high-performance gateway, integrates with Ethereum | Free with limits, paid plans (Infura (by ConsenSys), 2024) |
| **Fleek** | Static web hosting support with Content Delivery Network (CDN) + IPFS + Ethereum Name Service (ENS) | Freemium (Fleek, 2023) |

IPFS offers a viable alternative for decentralized storage, but it imposes limitations in terms of persistence, performance, and confidentiality. The use of pinning gateways is necessary to ensure availability, and the choice between services such as Pinata, Web3.Storage, or Fleek, should consider the volume of data, retention time, and available budget. Also, encryption of content before upload is recommended, especially in systems that handle sensitive information.

### 5.3.3 Deployment in Other Blockchains

Given the modularity and interoperability characteristics of smart contracts, the solution can be deployed in other public blockchains that support smart contract execution and the EVM, such as Binance Smart Chain (BSC), Avalanche C-Chain, or Polygon (Binance Smart Chain, 2025; Avalanche, 2025; Polygon Technology, 2025). These networks offer compatibility with Solidity, often with the advantage of lower transaction fees and higher transaction throughput when compared to the Ethereum Mainnet (WOOD, 2021).

However, deploying in a different public network requires adjustments to configuration files, as well as consideration of network-specific parameters such as Chain ID, gas price, and block gas limits. Additionally, developers must evaluate the trust, decentralization level, and community support of each network to ensure that they meet the required

security and reliability standards for production use.

An alternative approach is the deployment of the solution in a private blockchain, tailored to the specific needs of an organization or consortium. Frameworks such as Hyperledger Besu and Quorum allow the creation of private Ethereum-compatible networks, where the cost of gas can be artificially set to zero or adjusted according to internal governance rules (Hyperledger Foundation, 2025; ConsenSys Quorum, 2025).

Private blockchains offer advantages such as greater control over transaction validation, enhanced privacy, and customized performance tuning (CACHIN, 2016). However, these benefits come at the cost of reduced decentralization and a higher responsibility for infrastructure maintenance.

In scenarios where the primary objective is internal process auditing, access control, or data immutability within a single organization or restricted group of stakeholders, private blockchain deployment becomes a viable and often recommended option. The flexibility of the developed smart contract and the reliance on widely adopted standards (*e.g.*, Solidity and the EVM) ensure that the solution remains portable across different blockchain environments. The choice between public and private deployment should consider factors such as transaction costs, required throughput, scalability, governance, and the desired level of decentralization.

# 6 CONCLUSION AND FUTURE WORK

This work presented the design, implementation, and evaluation of a solution to enable a blockchain-based circuit allocation system in MEICAN, leveraging smart contracts for transparent and auditable registration of inter-domain circuit requests and using IPFS for decentralized policy storage. The integration of these technologies into the MEICAN workflow demonstrated the feasibility of building a secure and immutable system capable of improving governance, auditability, and end-user confidence in circuit reservation processes. Thus, this work demonstrated the practical viability of leveraging blockchain and decentralized storage for circuit management in multi-domain networks, setting a foundation for future research on scalable, secure, and auditable network management systems aligned with the principles of decentralization and trust minimization.

Throughout the implementation, a modular architecture was developed, including a smart contract written in Solidity, a Node.js API for blockchain and IPFS interactions, and a user-friendly frontend to simulate user and operator actions within MEICAN. The evaluation demonstrated that while blockchain and IPFS add a layer of transparency and security, they also introduce processing overheads, such as encryption latency, IPFS upload delays, and blockchain transaction confirmation times. Nevertheless, the system successfully ensured data integrity, traceability, and decentralized control in inter-domain environments.

Future work can advance in several directions to enhance the system's capabilities and practicality for production environments. Firstly, optimizing encryption and IPFS upload workflows could reduce execution times, enabling the system to handle higher volumes of requests efficiently. Secondly, deeper integration between the smart contract and MEICAN's policy workflows could automate approval decisions, reducing manual interventions while maintaining auditability. Thirdly, exploring deployment on alternative blockchains, such as Polygon or Binance Smart Chain, could reduce transaction costs and improve scalability, while testing private blockchain deployments with Hyperledger Besu or Quorum would enable internal governance for organizations requiring stricter control.

Furthermore, improvements in IPFS content management, including redundancy strategies and automated re-pinning mechanisms, can be explored to increase data availability without relying heavily on centralized pinning services. Further research could also investigate advanced privacy-preserving mechanisms, such as zero-knowledge proofs, to address confidentiality concerns while maintaining transparency. Additionally,

the frontend can be enhanced with new pages for smart contract administrator management and a circuit request evaluation viewer, allowing all system users to track and monitor the status of requests that have already been evaluated.

# REFERENCES

AKTER, S. et al. Enhancing data integrity and traceability in industry cyber physical systems (icps) through blockchain technology: A comprehensive approach. **Available at SSRN 5041397**, 2024.

ALHARBI, T. Deployment of blockchain technology in software defined networks: A survey. **IEEE Access**, IEEE, v. 8, p. 9146–9156, 2020.

ALVARENGA, I. D.; REBELLO, G. A.; DUARTE, O. C. M. Securing configuration management and migration of virtual network functions using blockchain. In: IEEE. **NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium**. [S.l.], 2018. p. 1–9.

ASHAROV, G. et al. Privacy-preserving interdomain routing at internet scale. **Cryptology ePrint Archive**, 2017.

Avalanche. **Avalanche C-Chain Documentation**. 2025. <https://docs.avax.network/>.

BANIATA, H.; KERTESZ, A. A survey on blockchain-fog integration approaches. **Ieee Access**, IEEE, v. 8, p. 102657–102668, 2020.

BENET, J. Ipfs-content addressed, versioned, p2p file system. **arXiv preprint arXiv:1407.3561**, 2014.

Binance Smart Chain. **Binance Smart Chain Documentation**. 2025. <https://docs.bnbchain.org/>.

BONDAN, L. et al. FENDE: Marketplace-Based Distribution, Execution, and Life Cycle Management of VNFs. **IEEE Communications Magazine**, IEEE, v. 57, n. 1, p. 13–19, 1 2019.

BUTERIN, V. et al. A next-generation smart contract and decentralized application platform. **white paper**, v. 3, n. 37, p. 2–1, 2014.

CACHIN, C. Architecture of the hyperledger blockchain fabric. **Workshop on Distributed Cryptocurrencies and Consensus Ledgers**, 2016.

CAMILO, G. F. et al. A blockchain-based system for secure and distributed virtual network functions orchestration. In: IEEE. **ICC 2022-IEEE International Conference on Communications**. [S.l.], 2022. p. 347–352.

CHIESA, M. et al. Sixpack: Securing internet exchange points against curious onlookers. In: **Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies**. [S.l.: s.n.], 2017. p. 120–133.

CoinMarketCap. **Ethereum Price Chart (ETH/USD)**. 2025. <https://coinmarketcap.com/currencies/ethereum/>.

ConsenSys Quorum. **Quorum Documentation**. 2025. <https://docs.goquorum.com/>.

ConsenSys Software Inc. **Ganache Documentation**. 2024. <https://archive.trufflesuite.com/docs/ganache/>.

ConsenSys Software Inc. **Truffle Documentation**. 2024. <https://archive.trufflesuite.com/docs/truffle/>.

DAEMEN, J.; RIJMEN, V. **The design of Rijndael**. [S.l.]: Springer, 2002.

DELMOLINO, K. et al. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In: CLARK, J. et al. (Ed.). **Financial Cryptography and Data Security**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. p. 79–94. ISBN 978-3-662-53357-4.

DENG, X. et al. A survey of blockchain consensus algorithms. In: **2022 International Conference on Blockchain Technology and Information Security (ICBCTIS)**. [S.l.: s.n.], 2022. p. 188–192.

DORRI, A. et al. Device identification in blockchain-based internet of things. **IEEE Internet of Things Journal**, IEEE, v. 9, n. 24, p. 24767–24776, 2022.

Ethereum Foundation. **Proof-of-stake (PoS)**. 2024. <https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/>.

Ethereum Foundation. **Solidity Compiler Documentation**. 2024. <https://docs.soliditylang.org/en/v0.8.21/>.

Ethereum Foundation. **Ethereum Gas and Fees**. 2025. <https://ethereum.org/en/developers/docs/gas/>.

Etherscan. **Ethereum Gas Price Tracker**. 2025. <https://etherscan.io/gastracker>.

Fleek. **Developer Documentation**. 2023. <https://docs.fleek.co/>.

FRANCO, M. F. et al. BRAIN: Blockchain-based Reverse Auction for Infrastructure Supply in Virtual Network Functions-as-a-Service. In: **IFIP Networking Conference (IFIP Networking 2019)**. Warsaw, Poland: [s.n.], 2019. p. 1–9.

GOEL, A.; RAHULAMATHAVAN, Y. A comparative survey of centralised and decentralised identity management systems: Analysing scalability, security, and feasibility. **Future Internet**, MDPI, v. 17, n. 1, p. 1, 2024.

GONZáLEZ, J. Álvaro; GARCíA, A. M. S.; BAEZA, V. M. **Blockchain-Enabled Management Framework for Federated Coalition Networks**. 2025. Available from Internet: <https://arxiv.org/abs/2503.09666>.

HAKIRI, A. et al. A blockchain architecture for sdn-enabled tamper-resistant iot networks. In: IEEE. **2020 Global Information Infrastructure and Networking Symposium (GIIS)**. [S.l.], 2020. p. 1–4.

HARDJONO, T.; LIPTON, A.; PENTLAND, A. Towards a design philosophy for interoperable blockchain systems. **arXiv preprint arXiv:1805.05934**, 2019. Available from Internet: <https://arxiv.org/abs/1805.05934>.

HAYYOLALAM, V. et al. Synergistic integration of blockchain and software-defined networking in the internet of energy systems. In: IEEE. **2024 6th International Conference on Blockchain Computing and Applications (BCCA)**. [S.l.], 2024. p. 420–427.

Hyperledger Foundation. **Hyperledger Besu Documentation**. 2025. <https://besu.hyperledger.org/>.

Infura (by ConsenSys). **Infura IPFS Gateway Documentation**. 2024. <https://docs.infura.io/infura/networks/ipfs>.

KATZ, J.; LINDELL, Y. Private-key encryption. In: **Introduction to Modern Cryptography**. [S.l.]: CRC Press, 2014. p. 77–82.

KAUR, N. et al. Securing fog computing in healthcare with a zero trust approach and blockchain. 2024.

KNIGHTLEY, S. **JSZip: Create, Read and Edit .zip Files with JavaScript**. 2024. <https://stuk.github.io/jszip/>. Accessed: 2025-06-18.

KOVACS, R. et al. Practical implementation of a blockchain-enabled sdn for large-scale infrastructure networks. **Applied Sciences**, MDPI, v. 14, n. 5, p. 1914, 2024.

KUMAR, S. et al. Bcsdncc: A secure blockchain sdn framework for iot and cloud computing. **International Research Journal of Multidisciplinary Technovation**, v. 6, n. 3, p. 26–44, 2024.

LATAH, M.; KALKAN, K. When sdn and blockchain shake hands. **Communications of the ACM**, ACM New York, NY, USA, v. 65, n. 9, p. 68–78, 2022.

Management Environment of Inter-domain Circuits for Advanced Networks (MEICAN). **Homepage**. 2025. <https://github.com/rnpbr/meican>.

MAURO, M. D. et al. Reliability and availability in virtualized networks: A survey on standards, modeling approaches, and research challenges. **arXiv preprint arXiv:2503.22034**, 2025.

MDN Web Docs. **SubtleCrypto.encrypt() - Web APIs | MDN**. 2024. Accessed: 2025-06-18. <https://developer.mozilla.org/en-US/docs/Web/API/SubtleCrypto/encrypt>.

MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. **Handbook of applied cryptography**. [S.l.]: CRC press, 2018.

MIERS, I. et al. Zerocoin: Anonymous distributed e-cash from bitcoin. **IEEE Symposium on Security and Privacy**, IEEE, p. 397–411, 2013.

NAKAMOTO, S.; BITCOIN, A. A peer-to-peer electronic cash system. **Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf**, v. 4, n. 2, p. 15, 2008.

NFT.Storage. **API and Documentation**. 2023. <https://nft.storage/docs>.

OKTIAN, Y. E. et al. Blockchain-powered bandwidth trading on sdn-enabled edge network. **IEEE Access**, IEEE, v. 10, p. 114024–114039, 2022.

Pinata. **Pricing and Plans**. 2024. <https://www.pinata.cloud/pricing>.

Pinata Technologies Inc. **Pinata Dashboard**. 2024. <https://app.pinata.cloud/ipfs/files>.

Pinata Technologies Inc. **Pinata Documentation**. 2024. <https://docs.pinata.cloud/quickstart>.

Polygon Technology. **Polygon Developer Documentation**. 2025. <https://polygon.technology/docs/>.

RAHMAN, A. et al. On the integration of blockchain and sdn: Overview, applications, and future perspectives. **Journal of Network and Systems Management**, Springer, v. 30, n. 4, p. 73, 2022.

Rede Nacional de Ensino e Pesquisa (RNP). **Homepage**. 2025. <https://www.rnp.br>.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, ACM New York, NY, USA, v. 21, n. 2, p. 120–126, 1978.

SANTANNA, J. J. C. de; WICKBOLDT, J. A.; GRANVILLE, L. Z. A bpm-based solution for inter-domain circuit management. In: IEEE. **2012 IEEE Network Operations and Management Symposium**. [S.l.], 2012. p. 385–392.

SCHEID, E. J. **An Intent-based Blockchain-agnostic Interaction Environment**. Thesis (PhD) — Universität Zürich, Zürich, Switzerland, 07 2022.

SCHEID, E. J. et al. VeNiCE: Enabling Automatic VNF Management based on Smart Contract Events. In: **IEEE Conference on Local Computer Networks (LCN 2022)**. Edmonton, Canada: [s.n.], 2022. p. 98–105.

SCHEID, E. J. et al. BUNKER: a Blockchain-based trUsted VNF pacKagE Repository. In: **Conference on the Economics of Grids, Clouds, Systems, and Services (GECON 2019)**. Leeds, UK: Springer, 2019. p. 1–9.

SCHEID, E. J. et al. Blockchains and Distributed Ledgers Uncovered: Clarifications, Achievements, and Open Issues. In: **Advancing Research in Information and Communication Technology**. Cham, Switzerland: Springer, 2021, (IFIP AICT Festschrifts, v. 600). p. 289–317.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 7th. ed. [S.l.]: Pearson, 2017.

SZABO, N. Smart contracts: building blocks for digital markets. **EXTROPY: The Journal of Transhumanist Thought,(16)**, v. 18, n. 2, p. 28, 1996.

TASKOU, S. K.; RASTI, M.; NARDELLI, P. H. Energy and cost efficient resource allocation for blockchain-enabled nfv. **IEEE Transactions on Services Computing**, IEEE, v. 15, n. 4, p. 2328–2341, 2021.

TIDWELL, T. **JSEncrypt: JavaScript RSA Encryption Library**. 2024. <https://github.com/travist/jsencrypt>. Accessed: 2025-06-18.

TURNER, S. W. et al. A promising integration of sdn and blockchain for iot networks: A survey. **IEEE Access**, IEEE, v. 11, p. 29800–29822, 2023.

VUKOLIć, M. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. **International Workshop on Open Problems in Network Security**, Springer, p. 112–125, 2016.

Web3.Storage. **Documentation**. 2023. <https://web3.storage/docs>.

WICKBOLDT, J. A. et al. Meican: Simplifying dcn life-cycle management from end-user and operator perspectives in inter-domain environments. **IEEE Communications Magazine**, v. 56, n. 1, p. 179–187, 2018.

WILLIAM, S. **Cryptography and Network Security - Principles and Practice, 7th Edition**. Pearson Education India, 2017. ISBN 9789353942564. Available from Internet: <https://books.google.com.br/books?id=AhDCDwAAQBAJ>.

WOOD, G. **Polkadot: Vision for a Heterogeneous Multi-chain Framework**. [S.l.: s.n.], 2021. <https://polkadot.network/Polkadot-lightpaper.pdf>.

WOOD, G. et al. Ethereum: A secure decentralised generalised transaction ledger. **Ethereum project yellow paper**, v. 151, n. 2014, p. 1–32, 2014.

ZHENG, Z. et al. An overview of blockchain technology: Architecture, consensus, and future trends. **2017 IEEE International Congress on Big Data (BigData Congress)**, IEEE, p. 557–564, 2018.