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

RAFAEL DESCIO TRINETO

BLOCKWITNESS: A VERIFICATION SYSTEM WITH FULLY HOMOMORPHIC ENCRYPTION FOR PERMISSIONED BLOCKCHAINS

JOINVILLE
2025

RAFAEL DESCIO TRINETO

BLOCKWITNESS: A VERIFICATION SYSTEM WITH FULLY HOMOMORPHIC ENCRYPTION FOR PERMISSIONED BLOCKCHAINS

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

Supervisor: Maurício Aronne Pillon, Ph. D.

JOINVILLE

2025

Ficha catalográfica elaborada pelo programa de geração automática da Biblioteca Universitária Udesc,

com os dados fornecidos pelo(a) autor(a)

Descio Trineto, Rafael
BlockWitness: A verification system with fully homomorphic
encryption for permissioned blockchains / Rafael Descio Trineto. -- 2025.
98 p.

Orientador: Mauricio Aronne Pillon
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2025.

1. blockchain. 2. permissioned. 3. cryptography. 4. homomorphism. 5. post-quantum. I. Pillon, Mauricio Aronne. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. III. Título.

RAFAEL DESCIO TRINETO

BLOCKWITNESS: A VERIFICATION SYSTEM WITH FULLY HOMOMORPHIC ENCRYPTION FOR PERMISSIONED BLOCKCHAINS

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

Supervisor: Maurício Aronne Pillon, Ph.D.

EVALUATION COMMITTEE:

Maurício Aronne Pillon, Ph.D.
PPGCAP/UDESC (Advisor)

Committee:

Charles Christian Miers, Ph.D.
PPGCAP/UDESC

Diego Luis Kreutz, Ph.D.
UniPampa

Joinville, January, 29, 2025

I dedicate this work to my family, friends, colleagues, and teachers who supported me and provided strength throughout this incredible journey.

ACKNOWLEDGEMENTS

I would like to take a moment to express my heartfelt gratitude to my esteemed supervisor, Prof. Dr. Mauricio Aronne Pillon. Your exceptional guidance has meant so much to me, and I truly appreciate your willingness to accept this research work.

Also, I want to thank all of my professors in the Applied Computing course at the Santa Catarina State University (UDESC). Your dedication and expertise have profoundly impacted my learning journey, and I am grateful for the knowledge you have shared.

My family, especially my dear mother, Vitoria do Santos, and my sister, Rafaela Descio, have always been by my side, supporting me throughout my entire journey.

My appreciation is extended to Dra. Maira Ranciaro for her encouragement and support in the initial stages of my research. Your faith in me set me on the right path.

I am also grateful to my colleagues and professors at USP who generously shared their insights and provided valuable guidance on the themes explored in this work. Your contributions have been invaluable.

Lastly, I want to acknowledge my wonderful colleagues and students at the Santa Catarina State University (UDESC). Your engagement in the research process and support for the development of this dissertation have been truly remarkable. Thank you for being part of this journey with me.

"When a man does not know what harbor he is making for, no wind is the right wind."
(Lucius Annaeus Seneca)

RESUMO

Distributed Ledger Technology (DLT) permissionado está em uso há mais de uma década, e muitos usuários e corporações estão ansiosos para aproveitar seu potencial. Esta pesquisa foca em um requisito comum entre sistemas empresariais que adotam essa tecnologia: a necessidade de controle, análise e monitoramento eficazes de ativos registrados na rede. O objetivo deste estudo é aprimorar as soluções disponíveis por meio desses sistemas, introduzindo uma nova plataforma conhecida como BlockWitness. O modelo de plataforma proposto permite que os membros da rede concedam a agentes externos acesso a operações que envolvam informações registradas, ao mesmo tempo que protegem o conteúdo bruto das transações. Como resultado, os agentes podem monitorar e analisar os dados sem exposição direta a informações confidenciais. Um recurso importante do BlockWitness é o forte foco na segurança e na preparação para o futuro. Para garantir essa segurança, a plataforma emprega métodos criptográficos homomórficos, que permitem o processamento de dados sem descriptografia. Além disso, utiliza técnicas resistentes pós-quânticas, assegurando resiliência contra potenciais vulnerabilidades associadas a avanços na tecnologia de computação quântica.

Palavras-chaves: blockchain, permissionada, privacidade, confidencialidade, criptografia, homomorfismo e post-quantum.

ABSTRACT

Permissioned Distributed Ledger Technology (DLT) has been in use for over a decade, and an increasing number of users and corporations are eager to harness its potential. This research centers on a common requirement among business systems that adopt this technology: the need for effective control, analysis, and monitoring of assets recorded on the network. This study's objective is to enhance the solutions available through these systems by introducing a new platform known as BlockWitness. The proposed platform model permits network members to grant external agents access to operations involving recorded information while safeguarding the raw content of the transactions. As a result, agents can monitor and analyze the data without direct exposure to sensitive information. A key feature of BlockWitness is its strong focus on security and future-proofing. To accomplish this, the platform employs homomorphic cryptographic methods, which allow for data processing without decryption. Furthermore, it utilizes techniques that are considered post-quantum resistant, ensuring resilience against potential vulnerabilities associated with advancements in quantum computing technology.

Key-words: blockchain, permissioned, privacy, confidentiality, cryptography, homomorphism, and post-quantum.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

2FA	Two-Factor Authentication
BFT	Byzantine Fault Tolerance
BlockWitness	BlockWitness - A verification system for permissioned DLT
CCPA	California Consumer Privacy Act
CFS	Completely Fair Scheduler
CFT	Crash Fault Tolerance
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DLT	Distributed Ledger Technology
ERP	Enterprise Resource Planning
EVM	Ethereum Virtual Machine
FAPESC	Foundation for Research and Innovation of the State of Santa Catarina
FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HEIR	Homomorphic Encryption Intermediate Representation
HIPAA	Health Insurance Portability and Accountability Act
HFP	Hyperledger Fabric Platform
HSM	Hardware Security Module
IoT	Internet of Things
IPFS	Interplanetary File System
LabP2D	Laboratory of Parallel and Distributed Processing
LHE	Leveled Homomorphic Encryption
LGPD	Brazilian General Data Protection Law
LTS	Long-term support
LWE	Learning With Errors
MIT	Massachusetts Institute of Technology

MFA	Multi-Factor Authentication
MQ	Message Queuing
NFT	Non-Fungible Token
NIST	National Institute of Standards and Technology
PCI-DSS	Payment Card Industry Data Security Standard
PHE	Partial Homomorphic Encryption
PME	Pattern Matching Engine
PoA	Proof-of-Authority
PoC	Proof of Concept
PoW	Proof-of-Work
PPGCAP	Graduate Program in Applied Computing
PQC	Post-Quantum Cryptography
RLWE	Ring Learning With Errors
SEAL	Simple Encrypted Arithmetic Library
SDK	Software Development Kit
SFA	Single-Factor Authentication
SSH	Secure Shell
SWHE	Somewhat Homomorphic Encryption
TFHE	Fast Fully Homomorphic Encryption over the Torus
TFHE-rs	Fast Fully Homomorphic Encryption over the Torus with Rust
TLS	Transport Layer Security
UDESC	Santa Catarina State University
UniPampa	Federal University of Pampa
USP	University of São Paulo
WAF	Web Application Firewall

CONTENTS

1 INTRODUCTION

In recent years, the popularity of blockchain technology has soared, attributed mainly to introducing new features on the platforms that offer them. One such innovation that emerged in the mid-2010s was permissioned blockchain networks, which require members to gain authorization from a central entity or group of members to access the network. This format has brought the interest of corporations and user groups seeking the benefits of the technology, including reliability, transparency, security, and storage, without exposing themselves to a public environment. As these participants explored the solutions offered, new demands emerged, prompting developers to adapt the platform to meet the requirements of diverse markets.

Originally, blockchain was developed as an alternative to traditional monetary systems, with cryptocurrencies being the primary focus. Perhaps the most famous example is Bitcoin, introduced in a white paper published in 2008 by pseudonymous Satoshi Nakamoto (Nakamoto, 2008), but its origins took place long before, with the Distributed Ledger Technology (DLT) systems in the early 90s (Sunyaev, 2020).

However, blockchain has become more versatile as technology advances, and its applications extend beyond finance. This is evident in the abundance of scientific literature on blockchain usage in transport, healthcare, education, IoT, and other fields, which have outnumbered the blockchain finance literature since 2015 (Sunny et al., 2022). Furthermore, resource management and governance are exploring the potential of blockchain technology (ABOU JAOUDE; GEORGE SAADE, 2019). All of this illustrates how blockchain is continuously growing and changing.

An example application of blockchain technology beyond the financial sector is the project Carbon21, which is being conducted by Brazilian researchers at the University of São Paulo (USP) in collaboration with Santa Catarina State University (UDESC) and various other research groups. The project aims to develop a tokenization platform for environmental preservation and reforestation actions, and it is used as the object of study for this research. The platform is designed as a private solution in which members are part of a permissioned blockchain with centralized governance. Still, as it is an asset regulated by public entities, it must be subject to external agent audits.

To further advance and enhance blockchain technology, it's important to create innovative tools to bolster its functionality while ensuring secure and sustainable growth. This research brings as its object of study a fundamental group of requirements in most business systems: the capacity to regulate, investigate, and oversee assets recorded on the network. Maintaining confidentiality and privacy is paramount, particularly with business assets recorded as blockchain data, since this information may contain sensitive data or competitive advantages for the company. Therefore, companies prioritize privacy, confidentiality, security, and transparency when selecting this technology. The proposed solution presents an additional tool that, coupled with the blockchain's transparency and traceability, enables data monitoring and analysis while preserving privacy and confidentiality.

BlockWitness is a complementary system to DLT that allows the anonymous verification of records using homomorphic encryption. The system offers a modular and scalable architecture that can be attached to existing systems with minimal changes. Its design was conceived to execute it in a separate environment using the communication channels already available.

The initial results were promising; the increase of operational costs are low considering the benefits of its adoption. It was also possible to verify the benefits of the chosen architectural format, the adopted load balancing, and the distribution of search operations with higher computational costs, allowing satisfactory results to be obtained when homomorphic searches were considered.

The following work is distributed as follows: Chapter 2 explains the initial research that gave rise to this work and its motivations. Section 2.5 displays related works found during the research. Chapter 3 describes the basic concepts of homomorphic encryption and introduces the TFHE library. Chapter 4 presents the architecture of the BlockWitness system in detail. Followed by the experimental analysis and results in Chapter 5. Closing this work in Chapter 6, considerations and conclusions are presented.

2 BACKGROUND AND MOTIVATION

Although most DLT/blockchain systems has a reliable consensus protocol, companies and regulators may want to validate and perform an external audition for ledger transactions. As governments of multiple countries make more restricted laws for digital information, blockchain systems must be adapted and provide solutions that permit verification that these laws are being constantly followed and revised.

The complexity of this challenge intensifies with the growing adoption of permissioned blockchains, which are frequently employed by projects seeking to enhance confidentiality. This is particularly evident in financial applications and supply chains, which have increasingly embraced this specific blockchain architecture. A recent report from W3Now.de, which analyses blockchain adoption in the German economy, corroborates this, showing that 49% of survey respondents use public-permissioned blockchains, and 34% use private-permissioned blockchains (W3NOW, 2024).

A practical example is the Brazilian law called Brazilian General Data Protection Law (LGPD), which were built to protect user data from being misused, analogously, it can be compared to the european General Data Protection Regulation (GDPR), where the government is trying to guarantee user data protection through a regulatory tool. That challenge is not easy to achieve, mainly because the government has no control over what companies do with user data. Current technology may not completely help achieve this objective, but it may pave the way to a world of solutions that will do it.

Homomorphic encryption is one promising solution that has the potential to assist governments in safeguarding user data. How 2-factor authentication has increased security when granting systems access (Saini, 2021), homomorphic encryption has the ability to take security information sharing a step further.

The Project Carbon21, develop by the Brazilians research at USP, shows a classic government regulatory scenario, where a common physical asset is treated as a digital asset (Correia, 2024). The subject asset is not fully regulated in many countries, including the country where the project was conceived. The project involves a bigger scenario, considering that a legal and physical person may be involved in transactions, which implies regulations to avoid money laundering, terrorist financing, and transfer of funds.

The BlockWitness system was thought over the aforementioned problem that arouses the necessity of achieving confidentiality and security demanded by regulatory and governance systems that need control and audition over the blockchain transactions without exposing the transaction's confidential data.

Another benefit explored in the BlockWitness is the multi-channel audition. The participant's peers must be registered in the Hyperledger Fabric Platform (HFP) when a channel is created into the blockchain. This implied that to have access to a channel, a user must be a member of that channel. When considering the audition process, that is a problem, especially when there are many regulators. BlockWitness proposes an external component format that may be used to monitor and grant access to data in multiple channels simultaneously, creating a single point of access to blockchain data and controlling who and when data is being accessed.

2.1 BIBLIOGRAPHIC SURVEY

The bibliographic survey aimed to search for the main studies relevant to the selected problems. For this purpose, research was carried out from 07/20/2023 to 10/10/2023 in the bibliographic databases of (Scopus) Google Scholar and IEEE — the research was updated during December 2024 to capture the evolution in the number of citations, utilizing the same criteria and articles selected in the initial round — using the following set of keywords presented in Table 1. To calculate the number of citations and average year of publication, the first ten articles ordered by relevance in the search tool itself were selected. The following criteria were considered for the selection

Table 1 – Keywords and results.

Id	**Sentence**	**Citations Oct/2023**	**Citations Dec/2024**	**Average Year**	**Total Results Count**
1	blockchain consensus attacks	874	1392	2019	77300
2	blockchain consensus audit	733	1645	2020	26700
3	blockchain consensus transparency	820	1430	2019	77400
4	blockchain logging	471	656	2019	28600
5	blockchain logging audit	451	568	2018	17000
6	blockchain logging transparency	430	692	2019	18400
7	blockchain transparency	1178	2216	2020	166000
8	transparency logging	991	1369	2017	23000
9	transparency logging privacy preserving	141	187	2016	26300

Source: The Authors.

of articles:

1. Inclusion Criteria:
    - Articles that add extra auditing/transparency tools;
    - They expose blockchain vulnerabilities that must be audited; and

- Tools that add layers of transparency, privacy, and auditing.

2. Exclusion criteria:

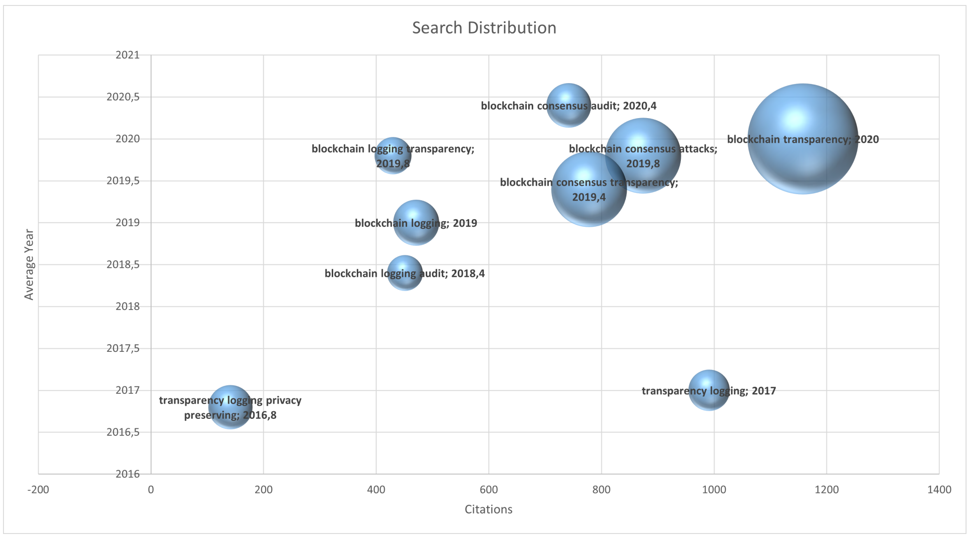
    - Everyone who uses blockchain to do transparent logging;
    - Remove articles that do not mention transparency or logging or blockchain vulnerabilities; and
    - Articles without public or institutional access.

3. Citation Criteria:

    - Maintain transparency tools for project justification; and
    - Maintain tools that mention blockchain auditing.

Figure 1 shows the graphic of the distribution of articles found grouped by year, number of articles and keyword. In addition, a second wave of bibliographic survey was carried

Figure 1 – Distribution of keywords in relation to the year of publication and number of citations.



Source: The Authors.

out to identify the tools chosen to carry out the experimental evaluation. Chapter 4 presents the selected tools.

2.2 BLOCKCHAIN

To achieve the objectives of this research, the use of Blockchain-type DLT was defined. The leading permissioned blockchains were considered according to Table 2. Only the community versions of each platforms were considered in this research.

Therefore, the features available in the corporate versions were not evaluated. All platforms considered are open source, although some have commercial versions. Likewise, they all have the possibility of using smart contracts in their structures.

Hyperledger Fabric Platform (HFP) was selected because it has the main functionalities of the permissioned blockchain and offers an intersectoral solution. Allow interoperability with another blockchain, such as Ethereum, through the Ethereum Virtual Machine Smart Contracts solution or, more broadly, through the Hyperledger Cacti tool. Mainly because the platform supports an event structure ideal for monitoring and analysis, which allows other applications to be alerted when an operation occurs on the blockchain.

The three types of events for which you are allowed to register are:

1. Contract Events - those explicitly emitted by the chaincode developer within a transaction.
2. Transaction Events - Those automatically emitted when a transaction is confirmed after being invoked.
3. Block Events - Those automatically emitted when a block is confirmed.

Receiving information about these events is possible without having to call a transaction on the blockchain. Furthermore, in addition to the possibility of monitoring predefined events, it is also possible to create personalized events (HYPERLEDGER FOUNDATION, 2023b).

In addition, HFP brings the flexibility of supporting a larger set of programming languages compared to other blockchains, allowing more researchers to expand the studies of this research. The languages Java, Go, Typescript and Javascript are currently supported by HFP — Although not officially supported, Python is also used by some.(HYPERLEDGER FOUNDATION, 2023a).

Hyperledger Fabric Platforms (HFPs) is a blockchain solution for companies. It is part of the set of blockchain solutions provided by the Hyperledger Foundation, which, together with 13 other projects, currently forms a broad open community for developing blockchain solutions.

To better understand what HFP is and how it works, it is first necessary to understand the concepts of blockchain and the origin of this technology. Blockchain is one of the three classifications of a Distributed Ledger Technologys (DLTs), along with Directed Acyclic Graph (DAG) and hybrid DLT (Liu; Farahani; Firouzi, 2020), as demonstrated in Figure 2. A DLT is a distributed database structured like a ledger. This means it records the entire history of modifications (also called transactions) to the data

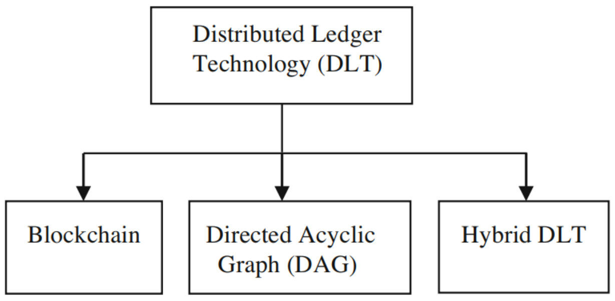
Table 2 – Key features of permissioned blockchains.

Platform	**Industry Focus**	**Smart Contract**	**Open Source**	**Event service**	**Support / Governance**
Hyperledger Fabric	Intersectoral	Yes	Yes	Yes	Linux Foundation
Ethereum	Intersectoral	Yes	Yes	No	Ethereum Developers
Quorum	Financial / Commercial	Yes	Yes	No	Ethereum Developers & JPMorgan Chase
Multichain	Intersectoral	(v1) No, (v2) Yes	Yes & Commercial	No	Coin Sciences
R3 Corda	Financial / Commercial	Yes	Yes & Commercial	(Community) No, (Business) Yes	R3 Consortium

Source: Adapted from (Polge; Robert; Le Traon, 2021).

it stores, and multiple copies of the ledger are available. Each copy is managed by an entity called peer (Capocasale; Gotta; Perboli, 2023). The Figure 3 demonstrates the relationship between distributed database, DLT, blockchain and decentralized DLT. In other words, blockchain can be understood as a distributed ledger, which stores value transactions anonymously and securely, and each new record inserted maintains the traceability of previous records. However, many blockchain innovations seek to take advantage, not of the transfer of value but of the record-keeping capabilities provided by the technology; that is, it offers a new way of storing, using, maintaining, or controlling records (Lemieux, 2016).

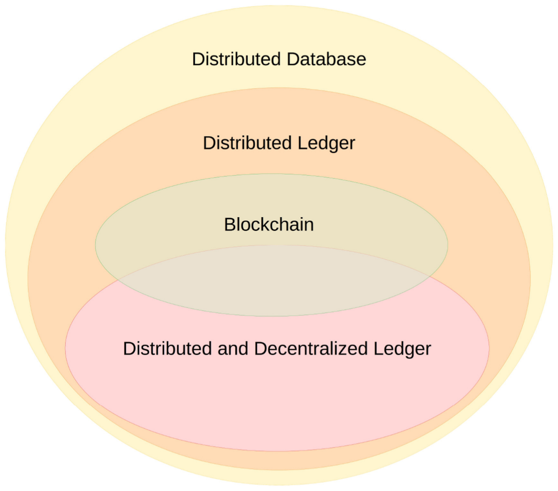
Figure 2 – Classification of DLT.


Distributed Ledger Technology (DLT)
Blockchain
Directed Acyclic Graph (DAG)
Hybrid DLT

Source: (Liu; Farahani; Firouzi, 2020).

Blockchain technology emerged from the Bitcoin whitepaper, suggested by Satoshi Nakamoto in 2008 in the article “A peer-to-peer electronic cash system”, which allows the creation of a peer-to-peer network system in which participants can make payments online to another participant without going through a financial institution (Nakamoto, 2008). The system uses a timestamp server that works with a cryptographic system. Each transaction is stored in a block that, at a pre-determined time,

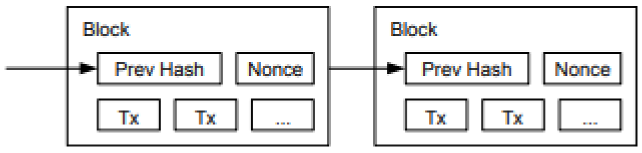
Figure 3 – Relationship between blockchain, distributed, decentralized and distributed ledger; and distributed database.



Source: (Capocasale; Gotta; Perboli, 2023).

receives a date/timestamp and is grouped, recorded, and distributed among network participants. And to implement this timestamp server in a distributed manner in a peer-to-peer network, the network uses a consensus mechanism known as Proof-of-Work (PoW). In this consensus mechanism, network participants compete to solve a random mathematical puzzle. Once solved, the puzzle is transmitted to the rest of the network for verification and, if accepted as valid, recorded in the blockchain network in a temporal chain as a legitimate block recognized by all its participants. To solve the suggested puzzle, each participating node collects transactions, assembles them into a block, and calculates the hash by placing them in a Merkle tree. — Also known as a Binary Hash tree, a Merkle tree is a data structure used in computer science and cryptography applications and can be used to identify changes in sets of distributed system records quickly and efficiently (Bosamia; Patel, 2018).— The engine then examines the nonce values to hash the block that meets the PoW requirements. Each new hash is generated from the previous hash block, ensuring that any changes to previous blocks invalidate the current block. This mechanism requires a high computational demand, reducing the chances of possible attacks (Nakamoto, 2008).

Figure 4 – Structure of a blockchain.



Source: (Nakamoto, 2008).

Initially, blockchain technology was used to transfer electronic money, but from 2013 onwards, with the conception of the Ethereum protocol by programmer Vitalik Buterin, new applications of blockchain technology were allowed, published in a 2014 whitepaper called Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform (Buterin, 2014). The publication presents the concept of arbitrary state transition function, which, through smart contracts, allows a wide range of applications, not just financial ones, theoretically allowing programmers to create any transaction or application using arbitrary contracts. In 2015, the Linux Foundation founded the Hyperledger project to bring Distributed Ledger Technology (DLT) transparency and efficiency to the enterprise market, leveraging the open software model (Lusard et al., 2021). Using the concepts of Smart Contracts and DLT, the Hyperledger project added new concepts, such as permissioned blockchain networks, and began a search to bring new functionalities, such as private transactions and confidential contracts, identification and auditability, interoperability, and portability (Dhillon; Metcalf; Hooper, 2017) According to the creators of HFP (HYPERLEDGER FABRIC, 2023b), for corporate use, it is necessary to consider the following requirements:

- Participants must be identified/identifiable.
- Networks need to be permissioned.
- High transaction throughput performance.
- Low latency of transaction confirmation.
- Privacy and confidentiality of transactions and data pertaining to business transactions.

This concept diverges from the initial purpose conceived by Satoshi Nakamoto for blockchain technology, in which he proposed a model that allows free access for participants without identification and mutual trust between participants. When considering business environment, there is a need for control in conjunction with anonymity, as these are often subject to regulations and legislation. Participants, such as companies competing for the same market, may have similar interests, which necessitates a differentiated level of confidentiality.

The creators of HFP assertively proposed a more controlled model, in which network participants are identified, but the transactions carried out remain confidential. This is one of the differences of blockchains created with HFP, in which the use of permissioned networks is recommended. In other words, the blockchain network operates on a group of known, identified and frequently scrutinized participants, operating under a governance model that allows a certain degree of reliability. In addition to being a permissioned platform, HFP allows for an extra level of confidentiality with its channel architecture and "data privacy" functionality. With the channel structure, network participants can establish subnets where each member can see a particular group of transactions. Therefore, only nodes participating in the channel have access to Smart Contracts and transacted data, preserving the privacy and confidentiality of both (HYPERLEDGER FABRIC, 2023a). The "data privacy" functionality allows you to create collections of private data in which groups of organizations that are channel members can transact data without others being aware of the transaction (HYPERLEDGER FABRIC, 2023d). Although HFP uses a collection of solutions to pursue privacy and confidentiality, the ecosystem is not resistant to post-quantum (Campbell, 2019). [1]. Just like Enterprise Resource Planning (ERP) software, projects aimed at companies need to adapt to the upcoming reality, as an important factor in choosing a business system is longevity and how this can affect the value of the assets of the company that holds them.

...longevity can be viewed as a non-functional quality attribute of an artifact that describes the degree to which the artifact continues to fulfill its purpose for a given interval of time or as long as a defined set of conditions holds. (Proença et al., 2013).

Regarding software developing, maintenance, scalability, compatibility, security, performance, and adaptability are essential to maintain its longevity.

2.3 AUDIT SYSTEMS AND RELIABILITY.

Although distributed ledger technology is built to offer reliability and security, like all data storage systems, this trust depends on the validity of the data stored. This section exploits the importance of audit systems as an extra layer to grant distributed ledgers trustworthiness.

Commonly used in financial, health, and enterprise management systems, audits are used to expose liability, detect fraud, detect misbehaviors, and ensure the integrity and correctness of information.

According to Kraus, the word "audit" originated in the Roman Empire, where auditors ensured the announcements were communicated correctly (Kraus; Platkus, 2007). The Oxford Dictionary defines audit as "an official examination of business and financial records to see that they are true and correct." (OXFORD LEARNER'S DICTIONARIES, 2024).

The research by Shamoo, Ph.D in his article "Data Audit as a Way to Prevent/Contain Misconduct" reinforces the necessity of good audit processes to avoid

[1] Post-quantum cryptography refers to post-quantum cryptosystems, a group of systems that will presumably remain secure with the advent of large quantum computers (Bernstein; Lange, 2017)

misconduct and fraud (Shamoo, 2013). In his article, he exploits data audit in research scenarios, but the same definition can be extrapolated to different applications. His definition of audit is: “The systematic process by which objective evidence is obtained and evaluated as to assertions about research data and their value to determine the degree of correspondence between those assertions and established or predetermined criteria which can then be communicated to interested parties.”.

Audit systems can be built for different purposes; the most well-known methods are external and internal audits, which may be performed manually or systematically. In critical data and processes, the union of both methods is necessary, especially when the target system has protected data that can be exposed to external participants without warranties. In a controlled environment, where all parts involved are known and the auditors are certificated, the reliability and correctness are built on a contract made between parts, but that frequently demands a lot of standard definitions that must be followed down and verified. ISO standards are the most used guidelines, but different countries and regions have their own laws and rules that must be followed, e.g., for user data, United States of America has Health Insurance Portability and Accountability Act (HIPAA) and California Consumer Privacy Act (CCPA), Europe has General Data Protection Regulation (GDPR), and Brazil has Brazilian General Data Protection Law (LGPD). Beyond that, security standards, like Payment Card Industry Data Security Standard (PCI-DSS) e.g., may be applied according to the system. All these validation barriers that systemic data require make audit systems increasingly necessary.

2.4 SYSTEM LOG

According with Yang Xiao, computer systems have the necessity of log data that represent system actions and the user activities. And many standards and regulations mandate logging mechanism requirements (Zeng et al., 2016). Most blockchain systems have logging system enabled for developers, e.g. Hyperledger Fabric has a complete logging control for peer and orderers (HYPERLEDGER FABRIC, 2024a). But what is system log and with specifications they must cover it is still an open discussion and must be defined according to the system application. In this section, it will be exploiting how logging system are defined and how important it is to build the right solution to have a trustworthy system.

First one, what is a system logging or system log. According to Sabine Canditt, in the article “Aspect Oriented Logging in a Real-World System”, logging is a mechanism to gain information about a running system and is therefore an important part of a system’s infrastructure (Canditt; Gunter, 2002). And logging is an essential concept to guarantee the serviceability of a Data Management System. Systems logging are defined by Lei Zeng as a fundamental feature within the modern computer operating systems because logging may be used through a variety of applications and fashion, such as system tuning, auditing, and intrusion detection systems (Zeng et al., 2016). According with (Gupta, 2005), logging can be defined as a systematic and controlled method for representing the state of an application in a format that is accessible and understandable to humans. A more recent study from (Gu et al., 2023) expands the concept and encompasses more recent methodologies, such as DevOps, which define logging practices as a facilitating tool for the systematic recording of valuable runtime information within software systems. Emphasizing, that this process plays a crucial role in supporting operational tasks, including service monitoring and troubleshooting.

According with (NEW RELIC, 2022) has been seen a 35% year-over-year increase in logging data. Despite systems log being more widely used, they definition are target of misunderstanding and confusion. As depending on definition made by system creators, many systems have different approaches to they log objectively. Basically, when defining a log system, you first need the offer, what the purpose of the logged data, who it is going to consume that data. Secondly, define which information must be provided and how frequently they must be provided. Then, consider who will have access to that data and how they access will be controlled. Finally, defined with data must be available to with members of system and how they will consume that data.

2.5 RELATED WORK

Data verification is a vital process for safeguarding the integrity and security of blockchain systems. The enhancement of privacy, trust, and reliability in emerging technologies depends significantly on this process. While the evaluation of log transparency focused mainly on data verification, a bibliographic survey indicated that this crucial step had been overlooked in the initial study prospecting. In that way, new ways of achieving that verification were considered without being intrusive and avoiding breaking changes in the current blockchain system.

These thoughts bring us to a new research field exploration: Homomorphic encryption. How can homomorphic encryption integrate with emerging technologies to create sustainable systems that support the evolution of computing? That way, our study is trying to evaluate the possibilities of using homomorphic encryption in an audit system for blockchain solutions.

The homomorphic studies, as presented in (Fan et al., 2023), demonstrate how challenging this technology is for our present-day computing power environment. That also turned into a challenge for our project; it is necessary to get used to both technologies, blockchain and homomorphic encryption, without affecting the ongoing blockchain usage.

Several studies focus on offering partial solutions related to the functionality outlined above. A selection of studies published between 2018 and 2023 was reviewed in bibliographic databases, including Scopus, Google Scholar, and IEEE, from July 2023 to October 2023. The results found are discussed below.

In the studies (Li et al., 2019), (Zhang et al., 2020), (Yuan; Wu; Zheng, 2023), and (Behnia et al., 2022), the researchers propose modifications to the blockchain scheme or its mechanisms, i.e., consensus changes, so that they become post-quantum resistant. Still, homomorphism, even if mentioned, is not the focus of the research presented. In articles (Liang et al., 2021), (Yaji; Bangera; Neelima, 2018), (Singh et al., 2021), (Yan; Wu; Sun, 2020), and (Jia et al., 2022), blockchain-based models that use homomorphic cryptography are suggested, which would theoretically allow the implementation of auditing systems, but which are not included as part of the application. It can also be mention the research carried out by (Ahmad; Saad; Mohaisen, 2019) and (Pawar et al., 2021), in which the authors propose a system of record control (log using blockchain for auditing, but do not consider cryptographic solutions for such solutions).

According to the research carried out by(Almeida, 2022) with 11 participants (presented in Table 3), privacy is among the most significant concerns when referring to external audits, which reinforces the purpose of this study.

Table 3 – Main challenges associated with the adoption of blockchain in external auditing.

Obstacles (particularly in auditing)	**Number of respondents who selected each obstacle**
Blockchain vulnerabilities in being targeted by cyber pirates (privacy)	7
Cybersecurity	7
Necessary skills for audit professionals	7
Regulatory barriers	6
Possibility of fraudulent transactions	3
Private Blockchain - Limitations on Disclosure of Financial Information	3
Public Blockchain - Access to information by unauthorized parties	2
Complex accounting assessments	2
Increased risk in accessing data	2
All the disadvantages mentioned	1

Source: Adapted from (Almeida, 2022).

In 2024, a PoC was proposed by (Raj; Kurt Peker; Mutlu, 2024), in the study "Blockchain and Homomorphic Encryption for Data Security and Statistical Privacy" the authors demonstrate that it is possible to apply statistical functions, such as mean, median and variance on encrypted data. The project was built using the Zama blockchain. An equally important project, that serves as the basis for this study also and has a set of Fully Homomorphic Encryption (FHE) solutions, many of them open-source, to be used with Ethereum Virtual Machine (EVM) (ZAMA, 2023b).

An initial study was undertaken to investigate a distributed certificate authority utilizing blockchain technology (Descio-Trineto et al., 2023). While the present research does not employ all the tools applied in the initial study, the previous work provided a foundational framework for the current investigation and facilitated the evolution of the Carbon21 project. This progression introduced technologies such as Interplanetary File System (IPFS), which were subsequently integrated into the Carbon21 system. Furthermore, it prompted critical questions that have shaped the direction of the current study.

2.6 CONSIDERATIONS:

The results of the bibliographic survey, detailed in Table 1, compare citation counts for a group of articles from 2023 to 2024. The findings reveal a noteworthy increase in interest in the subjects discussed in this paper, especially with regard to the keywords "blockchain" and "transparency". This highlights the continuing relevance of the subjects explored in this research within academic publications. It also emphasizes the importance of recognizing that as the adoption of Distributed Ledger Technology (DLT) and Blockchain systems increases, the need for transparent auditing of these systems has become increasingly essential.

Also, privacy concerns are a relevant topic explored in multiple fields of study. General propose systems like DLT are in the target by many researchers, given that achieving privacy is a challenge in distributed and/or shared control systems. The bibliographic survey revealed that while some studies propose modifications to blockchain mechanisms for post-quantum resistance or homomorphic cryptography-based models, just a few has focus on auditing with privacy concerns.

In a global demand for data-driven environment, more regulatory process are being implemented by governments. To meet these requirements, existing systems must offer tools that ensure compliance with regulations, thereby guaranteeing secure access to information while safeguarding sensitive user data from exposure.

In permissioned environments, such as those examined in this study, validation and audit systems can play a crucial role in preventing misconduct and fraud. This reinforce the importance of implementing practical measures to facilitate these efforts.

As defined in the initial objective, homomorphic encryption was a decisive factor in selecting the encryption algorithms to be adopted. In particular, it was necessary to allow be performed on ciphertexts, in other words, maintaining the confidentiality of the information while still performing computational operations on the content of the message.

Thus, it was selected the Fast Fully Homomorphic Encryption over the Torus with Rust (TFHE-rs) algorithm, a variant of the Fast Fully Homomorphic Encryption over the Torus (TFHE) algorithm, which, despite being in the early stages of development, has so far presented itself as a post-quantum-resistant solution. In addition, it has an active community in its development (ZAMA, 2024b) and a wide range of public scientific articles on the subject (TFHE, 2023)

To bridge the gap between verification, transparency, and privacy preservation, this study investigates non-intrusive methods for incorporating homomorphic encryption into blockchain auditing, ensuring compatibility with current systems and avoiding disruptive changes. Furthermore, a transparent systemic solution could potentially address the gap between advancements in hardware, the challenges posed by cryptography, and the systems currently in use.

BlockWitness has the potential of explore homomorphic encryption to create sustainable and secure auditing frameworks for DLT systems, advancing this field and paving the way for future research into practical implementation and computational efficiency improvements.

3 HOMOMORPICH ENCRYPTION

Homomorphism[1] refers to a map that preserves algebraic structures. Homomorphic encryption stands apart from traditional encryption methods by allowing computations to be conducted directly on encrypted data without requiring access to a secret key. The outcomes of these computations remain in encrypted form and can later be decrypted by the owner of the secret key(Gaid; Salloum, 2021). This technique combines properties of mathematical mapping and cryptographic algorithms to support specific homomorphic operations (Dimitoglou; Jim, 2022). The essential principles of homomorphic encryption, along with an overview of encryption algorithms, are discussed in the following sections.

3.1 ENCRYPTION ALGORITHMS

In 1981, theoretical physicist Richard Phillips Feynman presented a new way to model quantum interactions in complex systems in a series of lectures. His proposal was: Build a computer using entangled quantum objects to model the physical object we are interested in (Feynman, 1982). In 1994, Massachusetts Institute of Technology (MIT) physicist Peter W. Shor, in his paper "Algorithms for quantum computation: discrete logarithms and factoring" (Shor, 1994), identified an algorithm that can use quantum computing to break cryptographic systems such as RSA and Diffie Hellman. As demonstrated later, it could be used to break the ECC (Elliptic Curve Cryptography) cryptographic system by extending Short's algorithms (Roetteler et al., 2017). These algorithms are commonly used in public-private key cryptography systems, including Hyperledger Fabric platform systems (Campbell, 2019), which use X.509 encryption by default (HYPERLEDGER FABRIC, 2023c).

An important disclaimer is that although quantum computing has evolved in recent years and by 2024, it is now possible have access to the technology through various cloud computing solutions, such as IBM Quantum (IBM QUANTUM PLATFORM, 2023) and LIQUi (MICROSOFT LIQUI PLATFORM, 2016), as well as academic programs such as that provided by the Quantum Technologies Innovation Centre at the University of Bristol (UNIVERSITY OF BRISTOL, 2021), the computational capacity of current structures is still too small to break the algorithms currently studied (Yan et al., 2022). But as emphasized by National Institute of Standards and Technology (NIST), answering the question of when large-scale quantum computers will be built is a complicated question. Still, currently, many scientists believe that it is basically a question

[1] The word Homomorphic derives from the Greek *homoios* and *morphe*, "same form" BRITANNICA, THE EDITORS OF ENCYCLOPAEDIA (2024)

of engineering and that in the next twenty years large quantum computers will be able to break current encryption (NIST, 2024a).

Currently, there are several lines of research examining and developing possible cryptographic solutions capable of withstanding the potential capabilities of quantum computers. The theoretical basis of post-quantum cryptography can be divided into: Lattice-based, Hash-based, Code-based, Multivariable, and Isogeny of Elliptic Curve. However, it is important to emphasize that quantum cryptography is a more complex process and will require new algorithms and specialized hardware. The cryptography currently developed for this purpose is called Post-Quantum Cryptography (PQC) (also known as quantum-resistant cryptography), which uses current hardware and is already functional but requires further study and development.

3.2 HOMOMORPHIC ENCRYPTION CONCEPT

According to (Chillotti et al., 2020), the literature distinguishes homomorphic encryption schemes into two families: Leveled Homomorphic Encryption (LHE) and Fully Homomorphic Encryption (FHE). In LHE for each function, some parameters can be evaluated homomorphically. In FHE, a single set of parameters allows the evaluation of any function. With this defined (generalized), FHE can be seen as a particular case of LHE. Other literature, such as (Dimitoglou; Jim, 2022), classifies them by the degree and sophistication of homomorphism, that is, by the number of operations that the algorithm allows. These categories are: partially homomorphic encryption (Partial Homomorphic Encryption (PHE)), boundedly homomorphic encryption (Somewhat Homomorphic Encryption (SWHE)) and completely homomorphic encryption (Fully Homomorphic Encryption (FHE)).

3.3 LATTICE

Most of fully homomorphic encryption solutions place reliance on lattice problem. According with (Micciancio, 2012), Lattices are regular arrangements of points in Euclidean space. The most straightforward example of a lattice in n-dimensional space is $\mathbb{Z}^n$, which encompasses all *n*-dimensional vectors with integer entries. More broadly, a lattice is created by applying a nonsingular linear transformation $B \in \mathbb{R}^{d\times n}$ to the integer lattice $\mathbb{Z}^n$. Here, nonsingular means that the linear transformation $\mathbf{x} \mapsto B\mathbf{x}$ from $\mathbb{R}^n$ to $\mathbb{R}^k$ defined by B is injective. This process yields the set:

$$B(\mathbb{Z}^n) = \{Bx : x \in \mathbb{Z}^n\}.$$

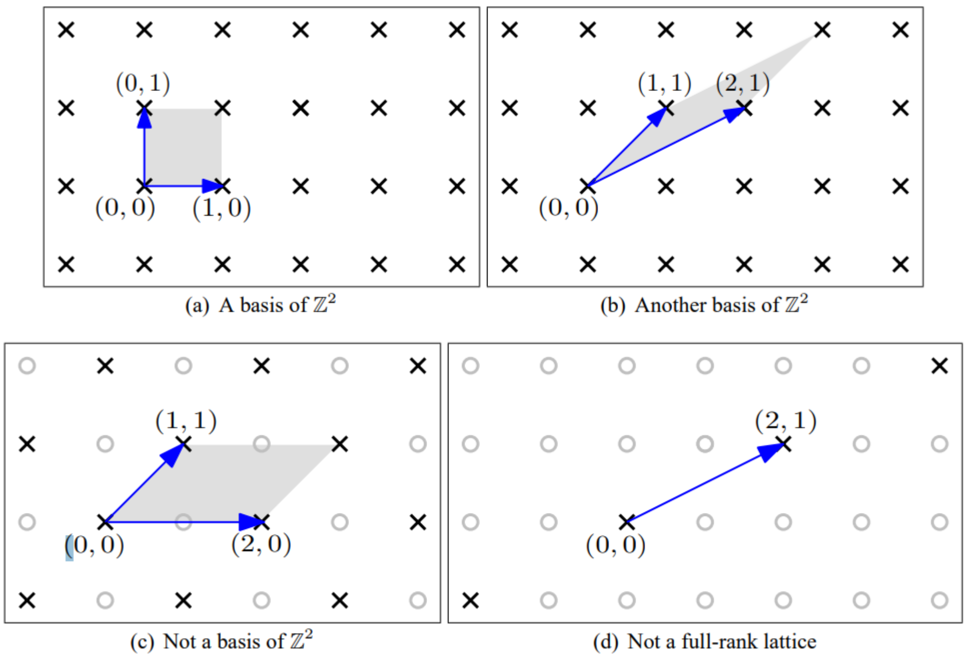
Definition 3.3.1 (LATTICE)**.** Let $B = [b_1, \ldots, b_n] \in \mathbb{R}^m$ given by n linearly independent vectors in $\mathbb{R}^m$. The lattice generated by B is the set

$$\mathcal{L}(B) = \{Bx : x \in \mathbb{Z}^n\} = \left\{\sum_{i=1}^{n} x_i \cdot b_i : x_i \in \mathbb{Z}\right\}$$

of all the integer linear combinations of the columns of B. The matrix B is called a *basis* for the lattice $\mathcal{L}(B)$. The integer n is called *rank* and integer m *dimension* of the lattice. If $n = m$, then $\mathcal{L}(B)$ is called a *full rank lattice*.

In this context, a lattice is defined as a set of points in *n*-dimensional space characterized by a periodic structure. Each point is a vector, and a grouping of vectors in the lattice are called *basis* (Regev, 2004). Figure 5 show 4 examples of lattices, where figure (a) and (b) are basis of $\mathbb{Z}^2$, where $\mathbb{Z}^2$ is the lattice generated by $(1,0)^T$ and $(0,1)^T$, $(1,1)^T$ and $(2,1)^T$ respectively. In contrast, (c) are given by $(1,1)^T$ and $(2,0)^T$ a not basis of $\mathbb{Z}^2$. Finally, (d) is given by $\mathcal{L}((2,1)^T)$, an example of a lattice that is not full.

Figure 5 – Example of some lattice bases.


(0, 1)
(0, 0)
(1, 0)
(a) A basis of $\mathbb{Z}^2$
(1, 1)
(2, 1)
(0, 0)
(b) Another basis of $\mathbb{Z}^2$
(1, 1)
(0, 0)
(2, 0)
(c) Not a basis of $\mathbb{Z}^2$
(2, 1)
(0, 0)
(d) Not a full-rank lattice

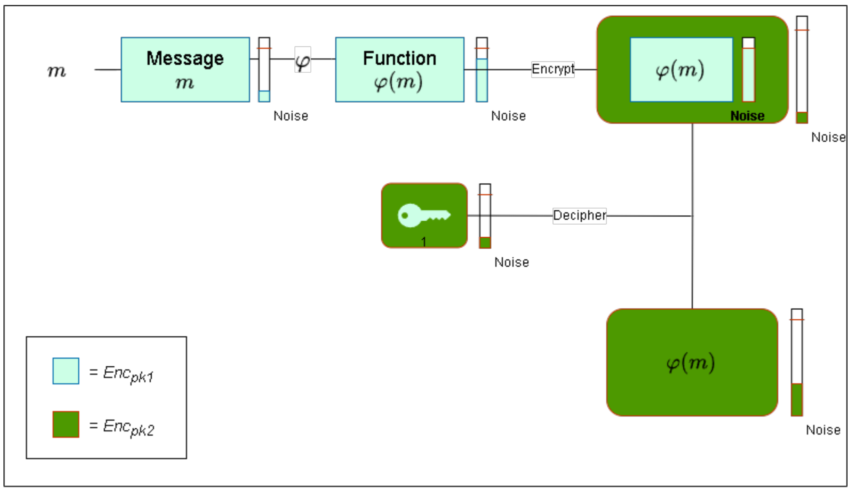
Source: (Regev, 2004).

Based on the lattice-based line of cryptography, in 2005, the concept was introduced by Oded Regev with the Learning With Errors (LWE) (Regev, 2005) problem. This is a presumably difficult computational problem to solve, which makes it useful for use in cryptography.

3.4 FAST FULLY HOMOMORPHIC ENCRYPTION OVER THE TORUS

Fast Fully Homomorphic Encryption over the Torus (TFHE) is a homomorphic encryption scheme, generalized from the FHE system. The security of the scheme is based on the hard lattice problem Learning With Errors (LWE) and its variants, such Ring Learning With Errors (RLWE). As explained by (Chillotti et al., 2020), the efficiency of these schemes depends crucially on the fact that the plaintext computations are expressed in a ring structure, where addition can be inverted, and that this supports sufficient parallelism to fill all computed spaces. This system is based on the inclusion of noise in the message to be transmitted, following the principles of LWE presented by (Regev, 2005). In encryption schemes, noise serves as a controlled element of randomness integrated into the message to enhance security. This addition conceals the message from adversaries while enabling trusted parties, who possess supplementary information, to efficiently remove the noise and recover the original content. However, there exists a critical threshold for the amount of randomness that can be applied; exceeding this limit can result in incorrect decryption outcomes. Hence, noise is characterized as a fragile form of randomness, as an excess can render the message irrecoverable (Hovd, 2017). It also uses the concepts of bootstrapping[2], an operation that allow reduce noise and enlarge homomorphic functions construction. Figure 6 presents the bootstrapping example with the encryption of a message m that underwent an operation $\phi(m)$, in which the vertical bars indicate the noise level at each stage.

Figure 6 – Example of bootstrapping in homomorphic encryption.



Source: The Authors - Derived from (Chillotti, 2022).

The TFHE algorithm has the benefit of allowing operations on an encrypted

[2] For more information, it is recommend reading the article Chillotti et al. (2020)

message, but this comes with a computational cost, which despite having decreased with evolution, its use still depends on the requirements that will adopt it.

According with (Syed et al., 2022), previous research has shown that homomorphic encryption, founded on Ring Learning With Errors (RLWE) and its association with the complex mathematical problem of high-dimensional lattices, is resistant to attacks from current quantum computers. As a result, homomorphic encryption offers greater security compared to RSA and other cryptographic techniques based on elliptic curves.

It is important to acknowledge that numerous open-source projects are investigating the advantages of homomorphic encryption. Reference technology companies in the cloud computing sector, such as Microsoft, Google, and IBM, are dedicating considerable resources to enhance the functionality of homomorphic encryption. Noteworthy initiatives include Microsoft Simple Encrypted Arithmetic Library (SEAL) (MICROSOFT, 2024), Google Homomorphic Encryption Intermediate Representation (HEIR)(GOOG 2024a), and IBM Guardium Data Encryption(IBM, 2024), along with various open-source efforts that provide cloud-based solutions (NIST, 2024b).

3.5 CONSIDERATIONS:

As computer technologies continue to advance, it is essential to regularly assess and adapt data security measures to new scenarios. Most encryption systems rely on mathematical calculations to generate ciphertexts; however, the rise of powerful computing technologies, such as quantum computing, challenges the effectiveness of these existing encryption methods.

Homomorphic encryption schemes represents a significant advancement in cryptographic systems, enabling secure computations to be performed on encrypted data without the need for decryption. This method serves as an effective means of preserving data privacy and facilitates the secure sharing of sensitive information.

Established in lattice-based cryptography and the concept of Learning With Errors (LWE), homomorphic encryption is designed to withstand both classical and quantum attacks, positioning it as a promising solution for the future of data processing.

While the implementation of this encryption can be computationally demanding, recent technological advancements, such as Fast Fully Homomorphic Encryption over the Torus (TFHE), demonstrate notable improvements in both efficiency and scalability.

Distributed Ledger Technology (DLT) systems are recognized for their ability to secure data integrity through a replication model, which tracks any modifications made. Nonetheless, many DLT systems still depend on conventional encryption methods for protecting data. There is a pressing need to research and explore innovative approaches to data security that ensure compliance with relevant laws and regulations, thereby safeguarding long-term integrity.

In light of this, the proposed project seeks to offer an alternative for future researchers, enabling current systems to effectively adapt to the challenges posed by evolving computational technologies.

4 BLOCKWITNESS ARCHITECTURE

BlockWitness was developed as a complementary system to a specific implementation of Distributed Ledger Technology (DLT), namely blockchain. Its primary purpose is to facilitate the auditing of homomorphic data recorded on the blockchain. Figure 7 shows an overview of the system's architecture. For this project, are considered a Hyperledger Fabric Platform (HFP) with the supplied Software Development Kit (SDK) and Gateway API (HYPERLEDGER FOUNDATION, 2023a).

Basically, the system BlockWitness - A verification system for permissioned DLT (BlockWitness) use two communication points, the first one is a client module, which client application can use to invoke to encrypt data. The second one is listener that is connected to the blockchain, responsible to read transactions and write then in a Merke Tree and database.

There are three main structures:

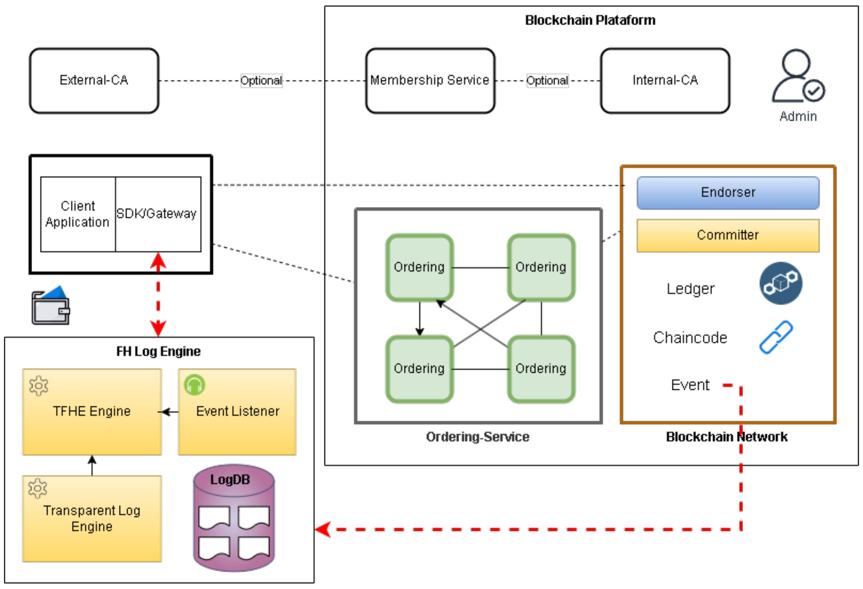
- A library that can be imported by clients and allow user encrypt and communicate with others.
- A Web Service capable of execute BlockWitness actions, such as search, verify and audit data.
- A worker service responsible to execute homomorphic operations and execute long jobs asynchronously.

The project was wrote in two programming languages, Typescript and Rust, and was divide was in a group of components using some of the practices recommends by the language maintainers. It was also follow some guidance provide by the Fast Fully Homomorphic Encryption over the Torus (TFHE) team, using the provided WebAssembly module for encrypt data.

The Figure 8 gives a overlook in the main project components. The figure is divided in 4 layers, Client Tier, Web Tier, Backend Tier and Workers.

1. The Client Tier layer is presently designed for use in web browsers, with plans to extend its functionality to mobile applications in future iterations. A web browser version was selected as a proof of concept due to its straightforward implementation. This layer comprises a collection of formularies and streams that facilitate user communication with the BlockWitness system.

Figure 7 – BlockWitness Architecture.



Source: The Authors.

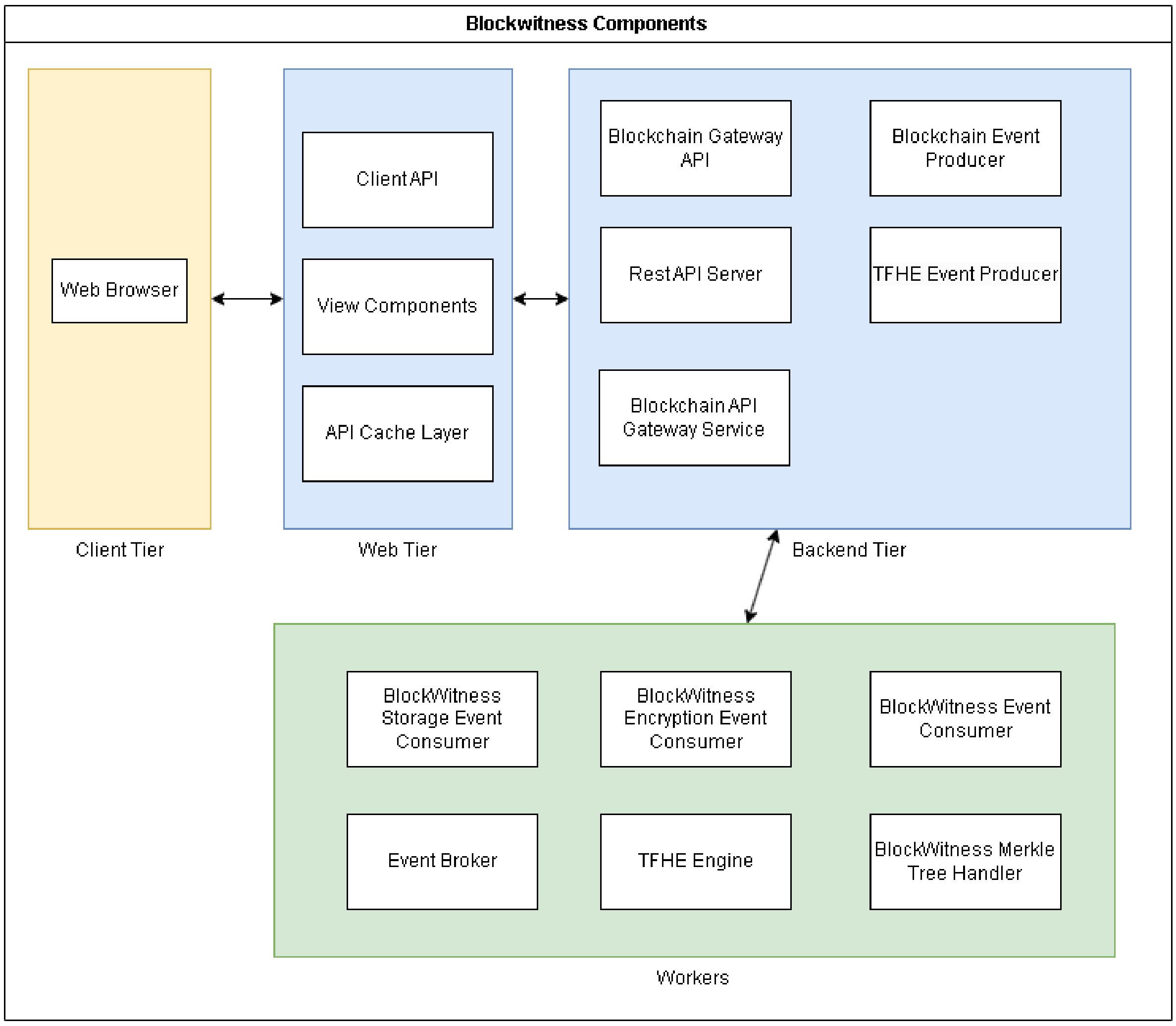
2. The Web Tier layer comprises a range of components and modules specifically crafted to enhance communication between developers and the functionalities of BlockWitness. These components provide an array of methods and functions that developers can employ as required, including verification methods and homomorphic encryption techniques, e.g. built-in operations for signed and unsigned integer formats ranging from 8-bit to 64-bit size.

3. The Backend Tier layer is the main layer responsible for orchestrating all process and operations in the BlockWitness system. It does all the communication between the blockchain, client and peripheral systems.

4. And the least, the Workers layer, a set of services that are mostly built to run asynchronously, listen events, and execute long time jobs.

Communication between asynchronous components is facilitated through a queue component, employing the producer-consumer pattern. The queue component is a Message Queuing (MQ) systems that store and manage messages, connecting producers (publishers) with consumers (subscribers). In this format of distributed messaging, multiple brokers collaborate to deliver messages organized into queues or topics, with producers specifying a topic to ensure accurate routing to consumers (Fu; Zhang; Yu, 2021). Figure 9 presents the 2 main message producers and consumers of the BlockWitness system. The Blockchain Event Producer and TFHE Event Producer are responsible to enqueue the data received from the blockchain and TFHE operations, respectively. A comprehensive overview of the relationship between the produc-

Figure 8 – BlockWitness Components.



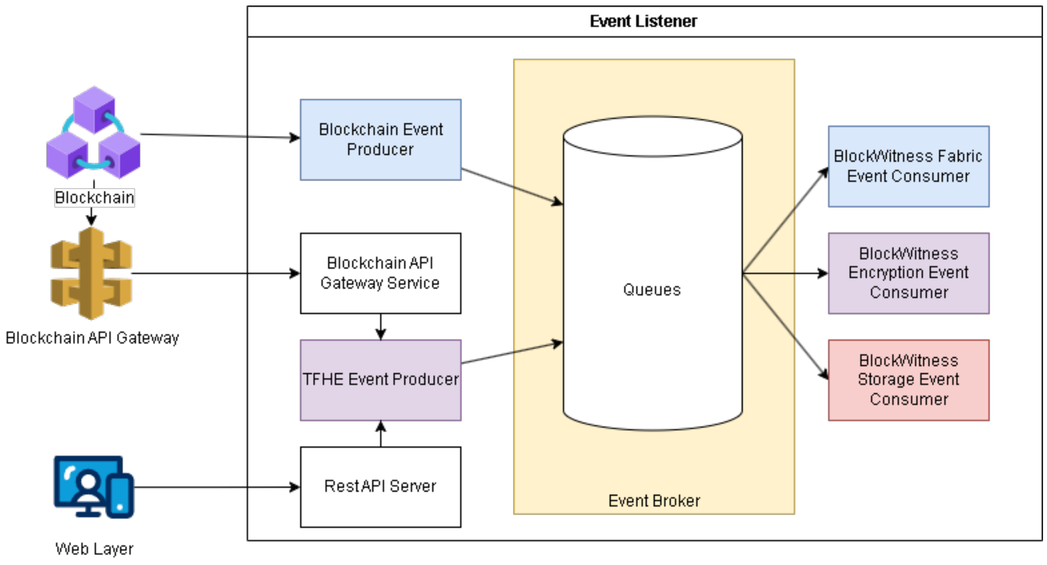
Source: The Authors.

tion and consumption components and the broader system is presented in Figures 10 and 11. The data received will undergo further processing by the BlockWitness Fabric, BlockWitness Encryption, and BlockWitness Storage event consumers, in accordance with the content of the data.

This approach was adopted to meet our objectives, as homomorphic operations can readily surpass the default HTTP web service timeouts. For instance, Nginx and Apache have a 60-second default timeout. Additionally, it enables the execution of multiple operations across various consumers. This selection allows for deploying of multiple consumers in different workers, thereby enabling BlockWitness to process homomorphic operations in parallel without obstructing the client interface.

Another advantage of the queuing system is that it allows consumers to execute processes sequentially, distributing the load among different workers. As the

Figure 9 – BlockWitness Events Architecture.

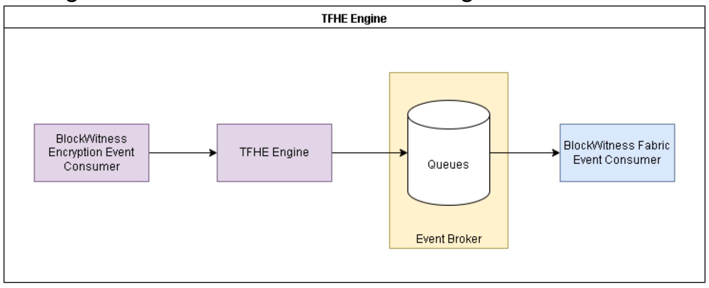

Event Listener
Blockchain
Blockchain API Gateway
Web Layer
Blockchain Event Producer
Blockchain API Gateway Service
TFHE Event Producer
RestAPI Server
Queues
Event Broker
BlockWitness Fabric Event Consumer
BlockWitness Encryption Event Consumer
BlockWitness Storage Event Consumer

Source: The Authors.

consumer pulls a new process according to its availability, the system is prevented from becoming overloaded. An example of this process with BlockWitness is the homomorphic search operation that requires a high amount of processing, overloading the CPU/Graphics Processing Unit (GPU) system and memories at workers. By adopting producer-consumer processing, it is possible to execute several search processes simultaneously distributed across physically separate workers, and execute new demands sequentially as soon as a worker becomes free.

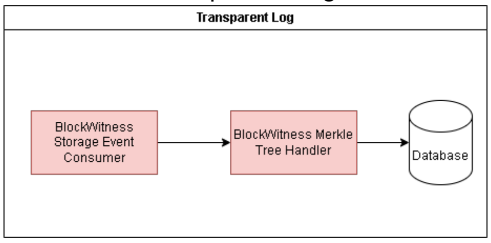
However, this pattern operates asynchronously, making complementary systems necessary so that users can stay informed about the status of the processes being executed.

Figure 10 – BlockWitness THFE Engine Architecture.


TFHE Engine
BlockWitness Encryption Event Consumer
TFHE Engine
Queues
Event Broker
BlockWitness Fabric Event Consumer

Source: The Authors.

Figure 11 – BlockWitness Transparent Logs.


Transparent Log
BlockWitness Storage Event Consumer
BlockWitness Merkle Tree Handler
Database

Source: The Authors.

4.1 CASE STUDY

As previously indicated, the foundation of this project lies in two case studies. The first case study was developed based on an example provided by HFP, while the second was derived from an actual application implemented in the Carbon21 project. The discussions surrounding the sample projects are extensively detailed in (HYPERLEDGER FABRIC, 2024a), although the diagrams associated with this case have been omitted from this section.

The initial use case diagram presented in Figure 12 outlines the operational framework of a Forest Compensation System, referred to as the "Carbon21 System." This system is designed to facilitate processes related to forest conservation and compensation.

The Carbon21's functionality permits users to buy and sell forest compensations and extends into the new transactional domain, both of which include the process for validating the legitimacy of these compensations in accordance with Brazilian standards. The approval process is embedded as extra support within the creation of forest compensations, underscoring the necessity for systematic oversight to sustain integrity and transparency. A unique use case for checking forest compensations is also included, strengthening the system's reliability and allowing participants to verify the compensation's status and compliance.

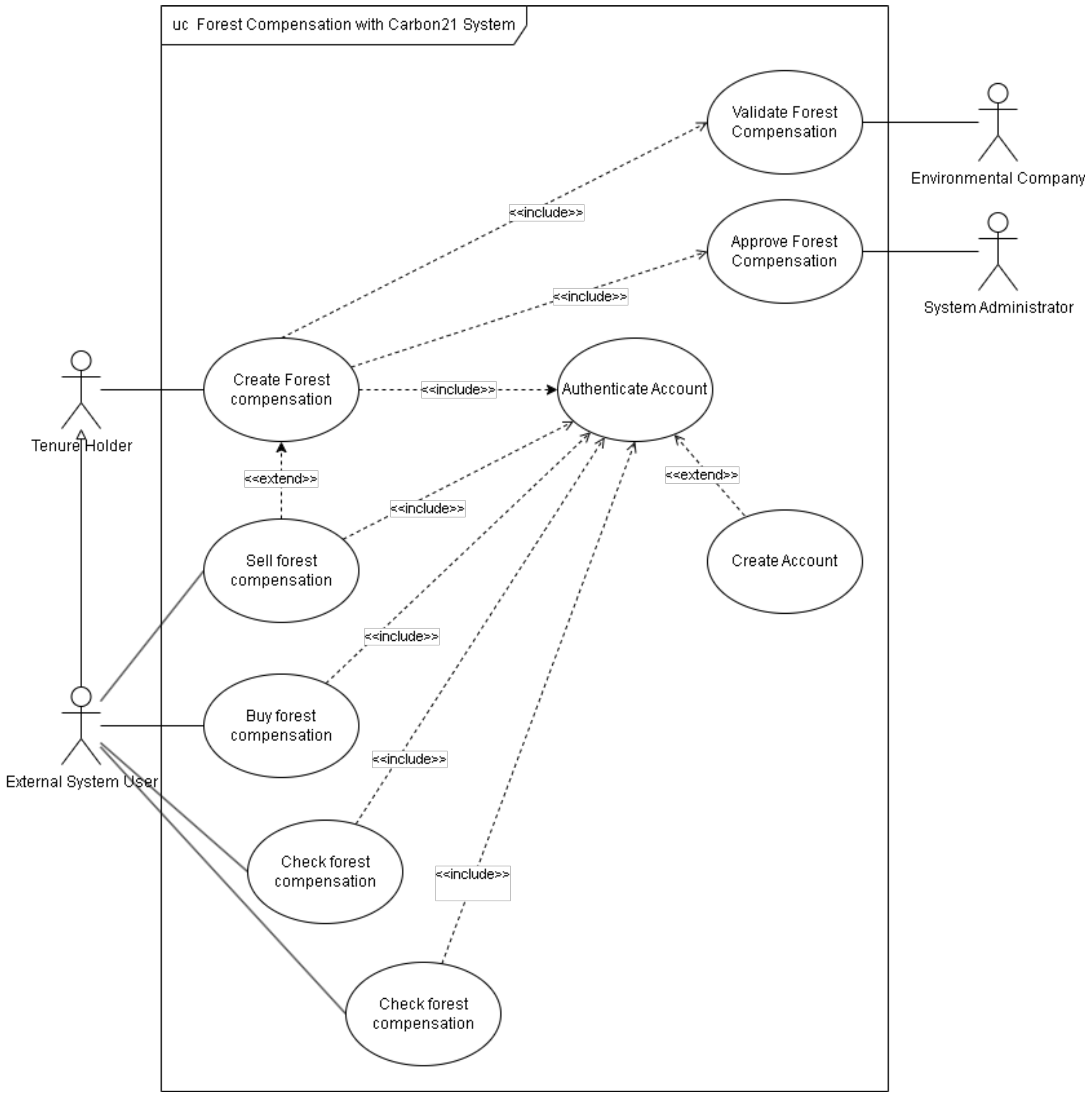
By integrating authentication, validation, and approval mechanisms into its transactional processes, the system achieves a balance between operational flexibility and stringent security and compliance requirements. It embodies an advanced digital infrastructure tailored to support sustainable forest management practices while ensuring trust, transparency, and accountability within compensation transactions. This design is particularly well-suited for carbon offset markets or environmental conservation initiatives, aligning with global efforts to promote sustainability and combat climate change, more information can be found in (Correia, 2024).

The diagram highlights key functionalities, including the creation, buying, selling, and verification of forest compensations, as well as supporting functions such as authentication, validation, and approval of these compensations.

At its core, the system enables the establishment of forest compensations, which require account authentication and the optional creation of accounts. This ensures that only verified and authorized users can engage in the platform's activities.

The Figure 13 use case diagram outlines the functionalities involved in accessing BlockWitness Data, highlighting secure and controlled interactions with encrypted information. The primary use case, "Access Encrypted Data," encompasses various

Figure 12 – Forest Compensation with Carbon 21 System.



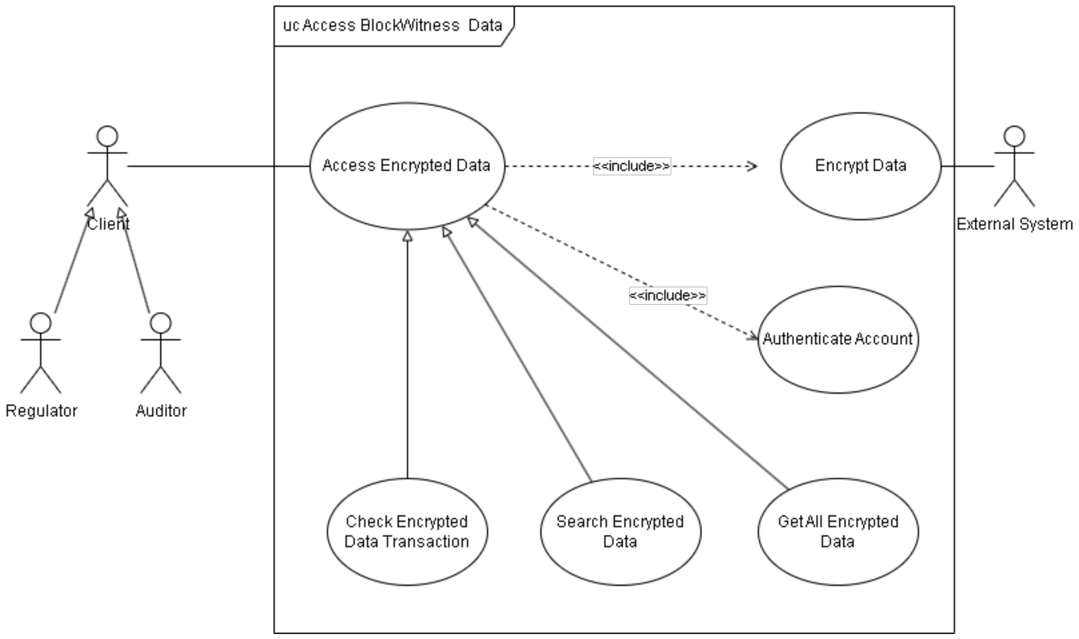
Source: The Authors.

sub-processes and functionalities that focus in data management. The process is intricately linked to the "Authenticate Account" functionality, which is a mandatory step designed to guarantee that only authorized users can access the system information. Furthermore, the inclusion of the "Encrypt Data" function ensures that all accessed data remains secure, reflecting a commitment to confidentiality and integrity.

The system facilitates several operations within the encrypted data environment, including "Search Encrypted Data," "Get All Encrypted Data," and "Check Encrypted Data Transaction." These operations directly support the system primary goal of enabling efficient and secure data access. The "Search Encrypted Data" feature allows users to execute targeted queries, while "Get All Encrypted Data" provides a comprehensive overview of available encrypted assets. At the same time, "Check

Encrypted Data Transaction" serves as a verification mechanism, ensuring that any transaction involving encrypted data is legitimate and adheres to established system standards.

Figure 13 – Access BlockWitness Data.



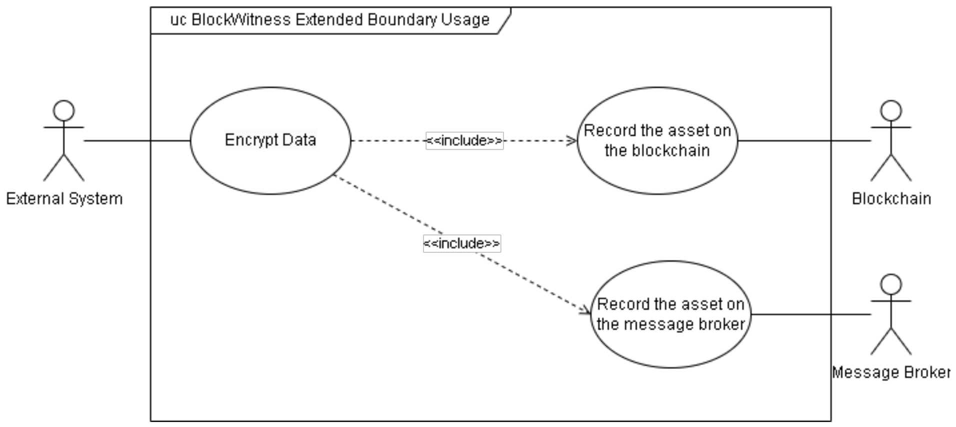
Source: The Authors.

The use case diagram show in the Figure 14 details the extended boundary for the BlockWitness System, emphasizing the integration of encryptions module with the DLT environments. As core functionality is the Encrypt Data process, which guarantees that all assets being managed or recorded are protected through encryption protocols. This is fundamental for preserving the confidentiality and integrity of data and keep consistent with the principles established in the prior use case concerning secure access to encrypted data.

The diagram embody two important extensions to the system defined encryption process: the recording of assets on the blockchain and their recording on the message broker. As seen in Figure 13, this functionalities also underscore the system's dual-recording strategy aimed at enhancing transparency and traceability. By leveraging DLT to record assets on the blockchain, the system ensures immutability and global verifiability. Concurrently, the recording of assets on the message broker allows for efficient data distribution and communication within the system, facilitating real-time updates and interactions among system components and external stakeholders.

This use case complements and expands the functionality described in the Access BlockWitness Data diagram. While the previous use case concentrated on accessing and interacting with encrypted data, the current diagram illustrates the processes involved in securing and distributing such data within an extended operational boundary. The inclusion of both blockchain and message broker recording facilitates a balance between the decentralization and the necessity for efficient data dissemination.

Figure 14 – BlockWitness Extended Boundary Usage.


uc BlockWitness Extended Boundary Usage
External System
Encrypt Data
<<include>>
Record the asset on the blockchain
Blockchain
<<include>>
Record the asset on the message broker
Message Broker

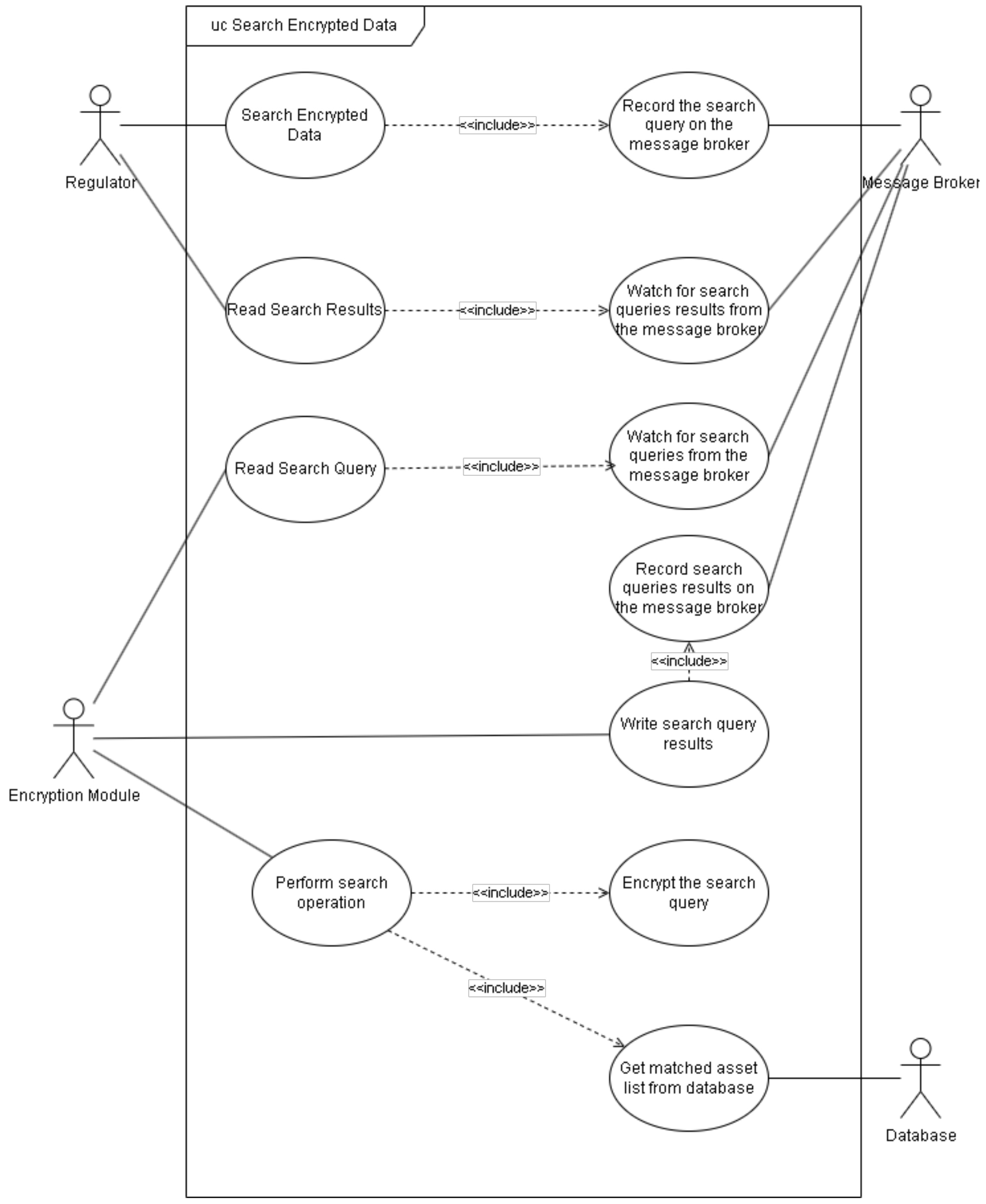
Source: The Authors.

The Figure 15 use case diagram delineates the functionalities of searching encrypted data, emphasizing secure query handling, result retrieval, and integration with a message broker. The primary process initiates with the Search Encrypted Data functionality, which encompasses the encryption of queries, execution of search operations, and retrieval of matched results. To ensure the secure management of sensitive data, the system incorporates critical subprocesses, such as Encrypt the Search Query and Get Matched Asset List from Database. These subprocesses safeguard the confidentiality of the data while facilitating precise retrieval.

An shown in the previous diagram, the same integration with a message broker is adopted here, as evidenced by functionalities including Record the Search Query on the Message Broker and Record Search Query Results on the Message Broker. These components guarantee that search operations are logged and results are distributed efficiently within the system. Supporting operations, such as Watch for Search Query Results from the Message Broker, empower the system to monitor updates, thereby ensuring a streamlined and responsive interaction with the search process.

The Read Search Results and Read Search Query use cases enhance system usability by enabling users to access search-related information in real-time. Furthermore, including of functionalities such as Write Search Query Results ensures that processed data is communicated efficiently back to the user or system. This maintains a feedback loop that keeps users informed about the status of their queries and the resulting data.

Figure 15 – Search Encrypted Data.


uc Search Encrypted Data
Regulator
Message Broker
Encryption Module
Database
Search Encrypted Data
Record the search query on the message broker
Read Search Results
Watch for search queries results from the message broker
Read Search Query
Watch for search queries from the message broker
Record search queries results on the message broker
Write search query results
Perform search operation
Encrypt the search query
Get matched asset list from database
<<include>>

Source: The Authors.

The Perform Search Operation use case is integral to the execution of secure search queries, leveraging both encryption and database operations to identify and retrieve relevant assets. This step ensures the search process adheres to stringent data security protocols while delivering accurate results.

Integrating encryption, message broker functionalities, and secure data retrieval processes, the system effectively ensures both data privacy and operational efficiency. The inclusion of real-time monitoring and result communication via the message broker underscores the system's commitment to responsiveness and reliability. This approach is particularly applicable to privacy-sensitive systems, such as blockchain-based data platforms and encrypted cloud storage, where secure, efficient, and traceable search operations are imperative. This reinforces framework's modular and extensible design further enhances its adaptability to evolving security and operational requirements.

Collectively, these diagrams offer a comprehensive and scalable framework for the management of encrypted data in contemporary systems. By integrating encryption, blockchain recording, and message brokering, the BlockWitness System achieves a high degree of security, transparency, and flexibility. This design is particularly applicable to environments that necessitate secure data management and robust audit trails, such as supply chain systems, financial transaction platforms, and privacy-sensitive data ecosystems.

4.2 SYSTEM ANALYSIS

In this section, a descriptive analysis of the BlockWitness system is carried out, illustrating the processes through a group of sequence diagrams.

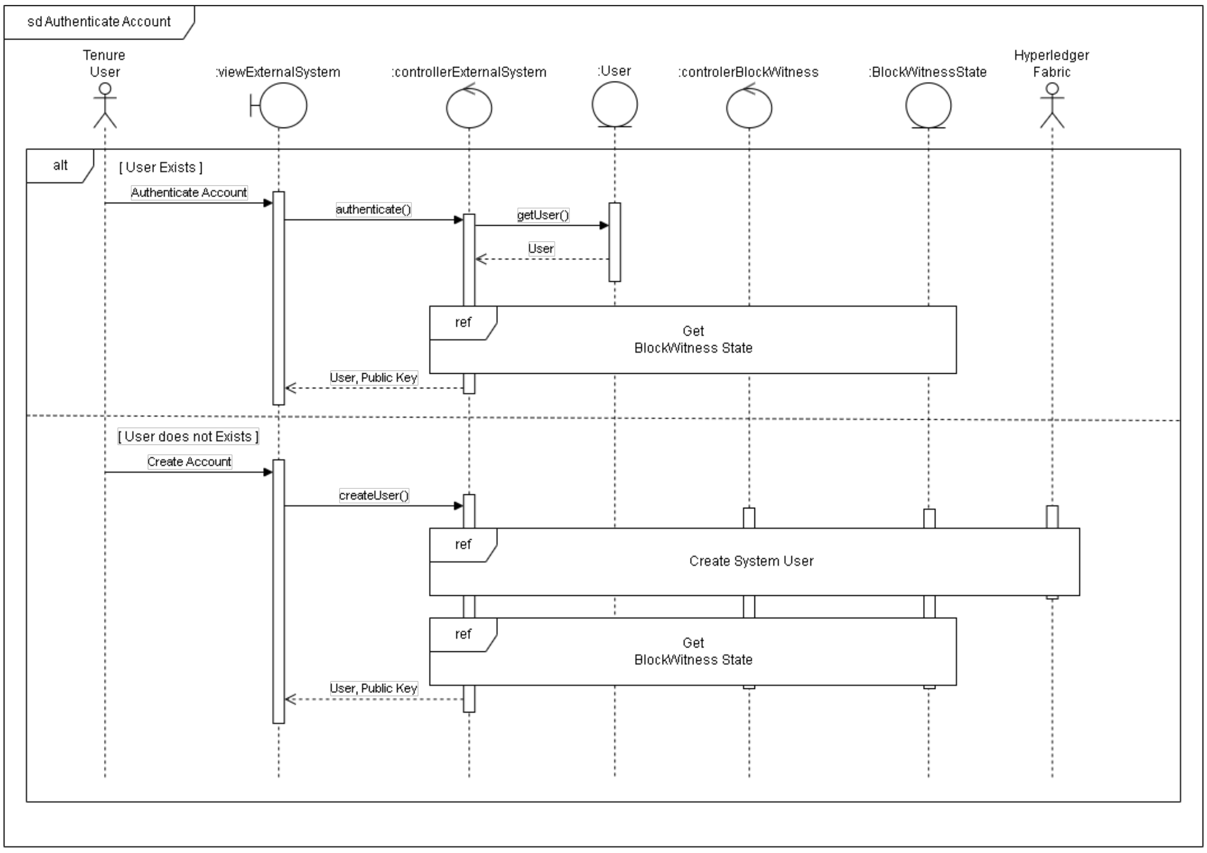
4.2.1 Authenticated Account

The Figure 16 shows a sequence diagram depicts the account authentication process and the steps involved in creating a new account if the user does not exist. The procedure begins with invoking the 'authenticate()' function, where the system seeks to verify the user's credentials through the 'getUser()' function. This function retrieves relevant user information, such as the User ID or Public Key, and checks the system's current state using a module known as BlockWitness State. This mechanism ensures that the user's credentials are validated against the distributed ledger, underscoring the significance of security and reliability. The authentication process is successful if the user exists and the provided details are verified.

When the user does not exist, the process shifts to an alternative workflow for account creation. The system initiates the Create Account procedure, triggering the 'createUser()' function to establish a new user profile. Throughout this process, the system leverages additional referenced modules. The Create System User module is likely responsible for configuring the necessary permissions or cryptographic credentials required for the new user. Meanwhile, the BlockWitness State module is revisited to ensure the seamless integration of the new user into the overall system state. This flow guarantees consistency and adherence to system-level validation standards. Upon successful account creation, the newly generated credentials, such as the Public Key, are returned, completing the account setup process.

The diagram encapsulates a dual-path workflow, where authentication is the

Figure 16 – Authenticated Account Sequence Diagram.



Source: The Authors.

primary objective and account creation is a contingency for non-existent users. Incorporating modular references like BlockWitness State emphasizes decentralized validation and system integrity, reflecting a robust approach to user management in secure environments.

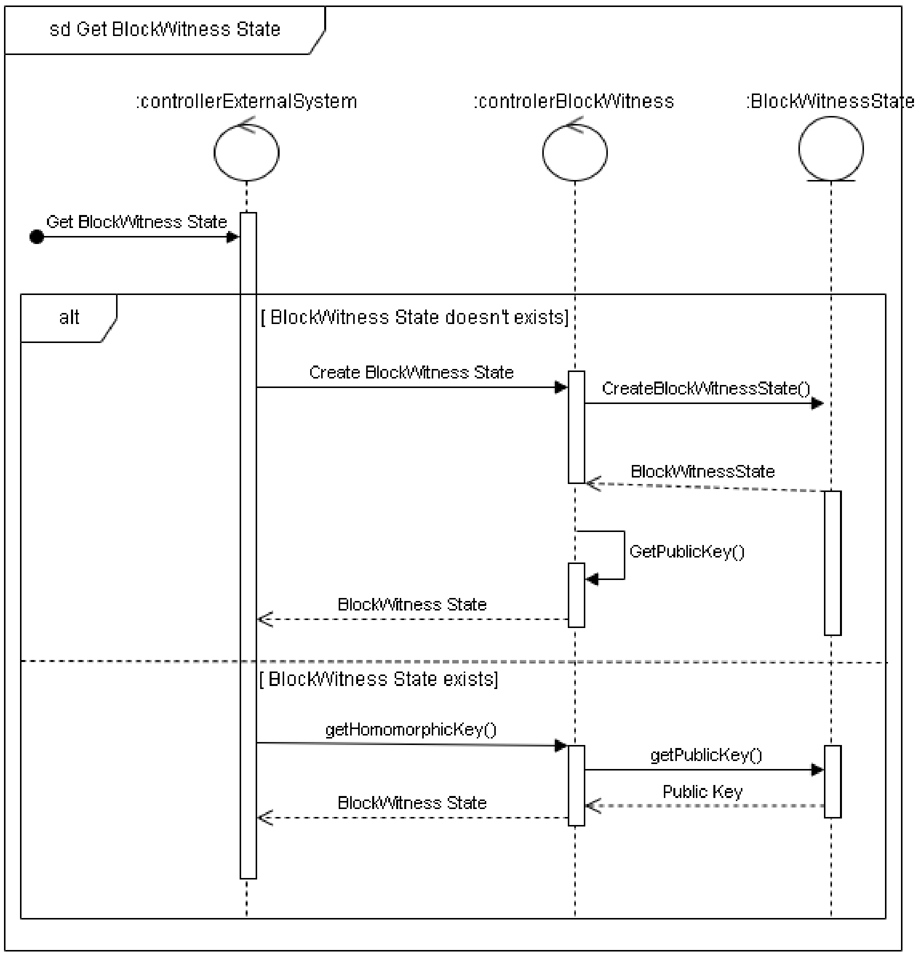
4.2.2 Get State

The sequence diagram represented in Figure 17 illustrates the process for retrieving the BlockWitness State, which is essential for verifying or establishing a system's cryptographic state. The workflow commences with a request to access the BlockWitness State and features an alternative block to accommodate two scenarios: retrieving an existing state or creating a new one if it does not yet exist.

In the first scenario, if the BlockWitness State is available, the system retrieves it directly. The next step involves invoking the 'getPublicKey()' function, which returns the cryptographic public key associated with that state. Conversely, if the BlockWitness State is not available, the system initiates the Create BlockWitness State process by calling the 'CreateBlockWitnessState()' function. This action establishes the necessary cryptographic framework, generating the initial state required for subsequent operations.

Once the state has been established, the workflow transitions to cryptographic key management. The system may call 'getHomomorphicKey()' to manage the parameters for homomorphic encryption, followed by another call to 'getPublicKey()' to retrieve the public key. Both processes rely on the previously established or retrieved BlockWitness State, ensuring consistency and integrity throughout. The public key is returned at the end of the process, indicating that the workflow is complete.

Figure 17 – Get State Sequence Diagram.



Source: The Authors.

4.2.3 Create System User

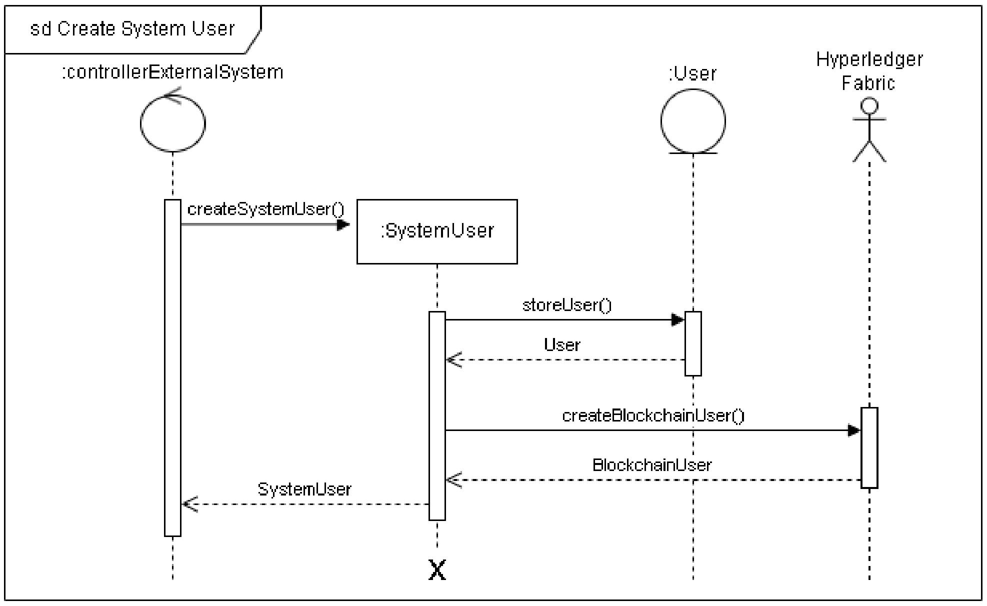
This sequence diagram in Figure 18 illustrates the process of creating a system user, emphasizing the interactions between various components to establish both a system-level user and a blockchain-level user. The process begins with the 'createSystemUser()' function, which initiates the user creation procedure. A new system user object, termed ’SystemUser’, is instantiated and serves as the foundational entity within the system.

The subsequent steps involve storing the user information. The ’storeUser()’ function is invoked to securely save the user’s data within the system’s database or repository. This step is essential for ensuring the persistence and retrievability of user records for future operations.

Following the completion of system-level operations, the process continues with the creation of a blockchain-level user. The ’createBlockchainUser()’ function is called, generating a ’BlockchainUser’ entity that is linked to the system user. This integration establishes a cryptographic identity for the user, facilitating interactions within a blockchain environment. The creation of the blockchain user ensures alignment between the system’s internal user management and its decentralized ledger operations.

The diagram underscores the multi-layered approach to user creation, where system users are interconnected with blockchain-specific identities. This design offers flexibility, enabling users to operate within both centralized and decentralized frameworks. The use of modular function calls, such as ’storeUser()’ and ’createBlockchainUser()’, highlights a structured and extensible architecture that promotes security and scalability.

Figure 18 – Create User Sequence Diagram.



Source: The Authors.

4.2.4 Access Encrypted Data

The sequence diagram presented in Figure 19 describe the procedural framework for accessing encrypted data, emphasizing the crucial aspects of secure retrieval, validation, and search functionalities while upholding both cryptographic and transactional integrity. The workflow initiates with a request denoted as an Access Encrypted Data Transaction. Prior to advancing, the system employs the Authenticate BlockWitness User step to ascertain user validation within the BlockWitness system, ensuring that only authorized individuals are permitted to access the encrypted data.

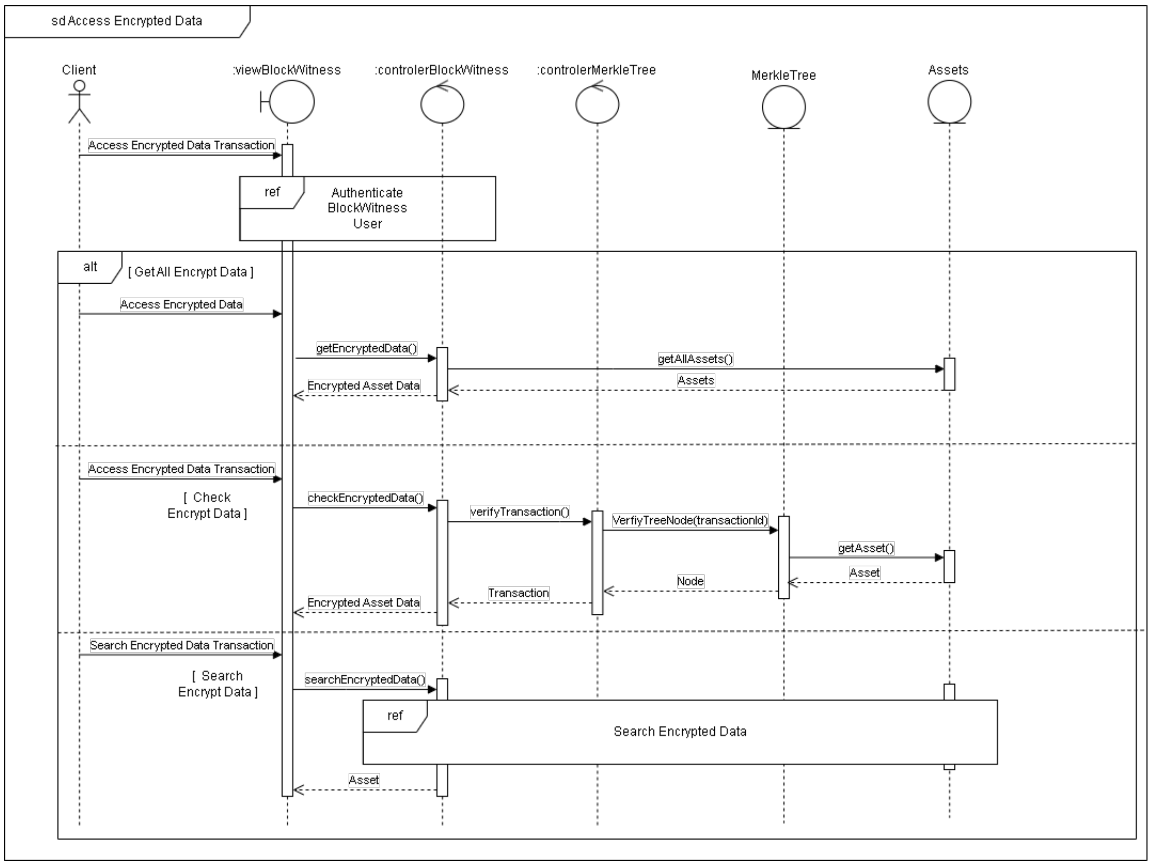
Upon successful authentication, the system engages the getEncryptedData() function to retrieve specific encrypted asset data. Users also have the option to invoke getAllAssets(), which facilitates the retrieval of all available assets, thereby allowing for a broader analytical perspective. This dual functionality provides users with the flexibility to target data or to explore a comprehensive dataset.

Integral to the process are stringent data validation measures that guarantee the integrity of the retrieved information. The checkEncryptedData() function executes an initial consistency assessment of the encrypted data, followed by the invocation of the verifyTransaction() method, which authenticates the legitimacy of the transaction correlated with the data. Additionally, the system reinforces cryptographic integrity through the VerifyTreeNode(transactionId) function, which validates the data contained within a cryptographic tree structure, such as a Merkle tree. Upon the completion of all requisite validations, the getAsset() function retrieves the specific asset associated with the verified transaction.

To facilitate targeted retrieval, the system incorporates specialized search functionality through the searchEncryptedData() function. This function utilizes the Search Encrypted Data module to assist users in locating assets within the encrypted dataset. Upon conclusion of the search process, the results yield the desired asset, effectively finalizing the transaction. This feature enhances user experience by enabling precise identification of specific data amid potentially extensive and intricate encrypted structures.

Overall, the design of the system was built to be a multilayered security approach that amalgamates rigorous authentication protocols, data validation techniques, and flexible data access capabilities. The verification of BlockWitness users ensures that only authorized entities can engage with the system, while the integration of multiple verification layers, including transaction validation and cryptographic node checks, the commitment to data security and integrity. Furthermore, the modular framework, exemplified by components such as Authenticate BlockWitness User and Search Encrypted Data, supports the BlockWitness system's potential for scalability and adaptability in future enhancements.

Figure 19 – Access Encrypted Data Sequence Diagram.



Source: The Authors.

4.2.5 Create Data

Figure 20 sequence diagram outlines the data creation process, emphasizing the interactions among authentication, asset creation, encryption, and storage. The workflow commences with the "Create Data" function, which serves as the initial trigger for the sequence. This action activates the "Authenticate Account" module, responsible for verifying the user's credentials to ensure that only authorized individuals can create new data. This authentication step lays a secure foundation for the subsequent operations.

Upon the user's successful authentication, the system advances to the "Create Asset" phase. The function 'getPartialEncryptedData()' is invoked to generate a segment of the encrypted data. This function collaborates with the "Get BlockWitness State" module, which provides the current state or ensures cryptographic consistency by retrieving or generating the encrypted data essential for asset creation. This verification of state guarantees that the newly created data adheres to the system’s security and integrity standards.

Following the retrieval of the encrypted data, the system executes the function ’CreateAsset()’, which consolidates the encrypted asset information into a complete asset. Subsequent steps involve invoking ’createAssetChain()’, which integrates the new asset into the blockchain structure. This integration ensures the traceability and immutability of the created asset, establishing a secure framework for managing and verifying its existence within the system.

The concluding step involves storing the created asset. The function ’storeAsset()’ commits the asset to the system’s storage infrastructure, ensuring that it is securely saved and can be retrieved for future use. The resultant "Asset Data" constitutes the output of this process, marking the successful creation and storage of the new asset.

This sequence diagram integrates essential components such as user authentication, partial encryption, blockchain integration, and secure storage, underscoring the system’s commitment to ensuring data integrity, confidentiality, and traceability.

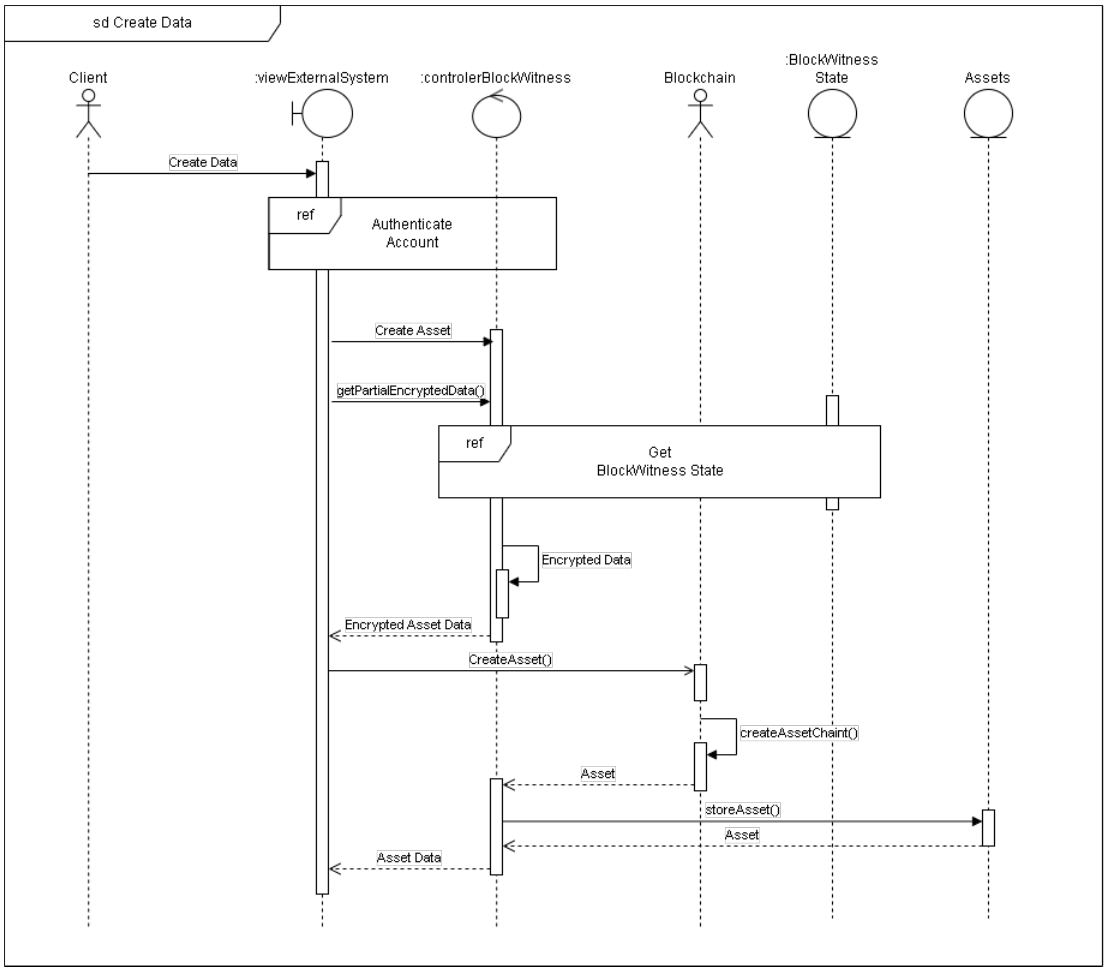
The modular design features distinct functions for asset creation and storage, enhancing both scalability and adaptability. This makes the system particularly well-suited for applications that handle sensitive or cryptographic data.

4.2.6 Search Encrypted Data

The Figure 21 presented a sequence diagram that delineates the methodology for executing a secure search operation on encrypted data, showing the processes of query encryption, operation dispatching, and result retrieval while maintaining user privacy. The procedure starts invoking the ’searchEncryptedData()’ function, which initiates the search operation. This function is the primary entry point for conducting a secure query over encrypted data, thus ensuring confidentiality throughout all phases.

The initial critical step involves encrypting the search query using the ’encryptSearchQuery()’ function. This encryption mechanism safeguards the query’s content, thereby preventing unauthorized access or exposure during its transmission and processing. Following the encryption, the query is forwarded to the search processing system via the ’dispatchSearchOperation()’ function. The search operation was built according with the Pattern Matching Engine (PME) suggest by the TFHE library creators (ZAMA, 2023a), this process uses the structure of regular expressions to perform complex searches in texts. This module functions as a communication intermediary, facilitating the secure transfer of the encrypted query to the backend system for further processing.

Figure 20 – Create Data Sequence Diagram.



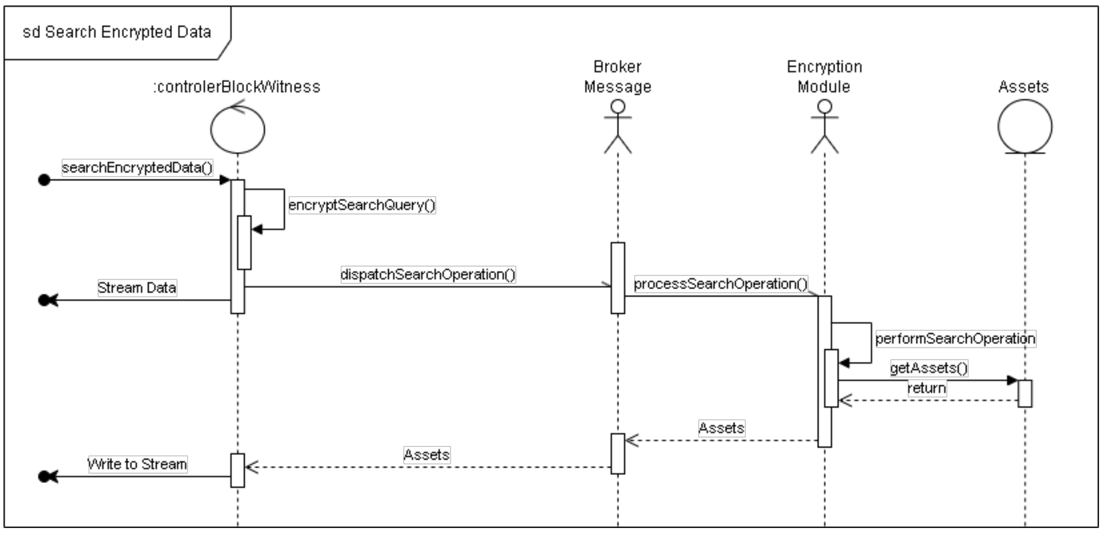
Source: The Authors.

Within the backend system, the "BlockWitness Encryption Module" and the 'processSearchOperation()' function are tasked with managing the encrypted query. This framework ensures the secure handling of sensitive data during search operations. This function orchestrates the internal logic necessary for processing the query and subsequently invokes the 'performSearchOperation()' function. The latter accesses the underlying encrypted data storage or indexing mechanism to identify relevant assets that correspond to the search criteria. The 'getAssets()' function is then employed to retrieve the results, ensuring that only the encrypted matching assets are obtained and prepared for delivery.

The concluding phase involves streaming the retrieved data back to the requesting entity. The Write to Stream operation returns the result to the user, ensuring that the results are transmitted securely and efficiently while preserving the data's encryption. This final step makes the process's end-to-end security, maintaining the confidentiality of both the query and the results throughout their entire lifecycle.

The modular design presented in functions such as 'encryptSearchQuery()', 'dispatchSearchOperation()', and 'processSearchOperation()' facilitates scalability and adaptability across various system architectures, as will be seen in the next chapters. Moreover, the proposed implementation of encryption at every stage adheres to best practices in data security and privacy-preserving technologies. This renders the workflow applicable across various domains, including encrypted databases, DLT systems, and environments that handle privacy-sensitive data.

Figure 21 – Search Encrypted Sequence Diagram.



Source: The Authors.

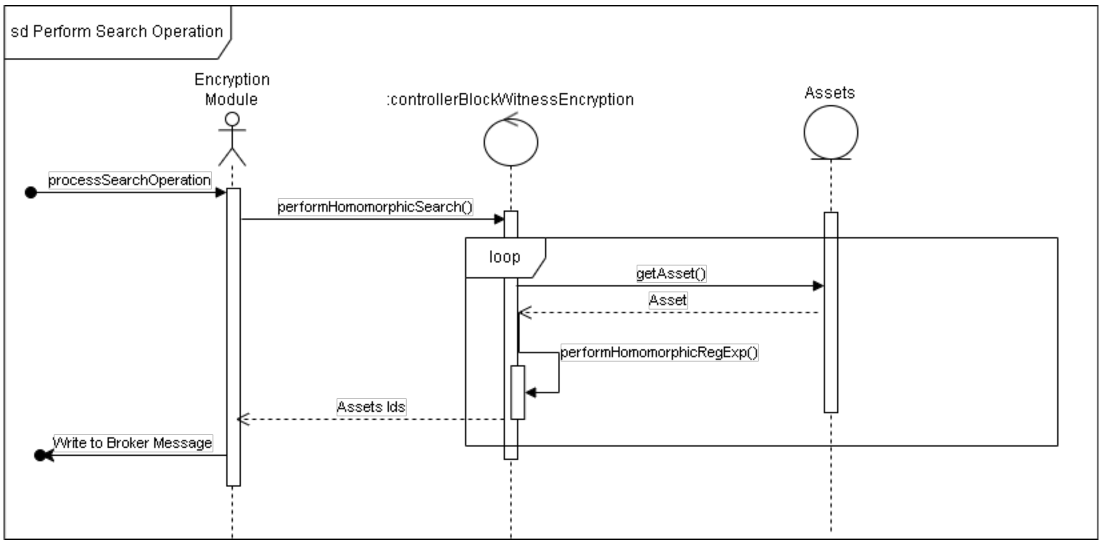
4.2.7 Perform Search Operation

The sequence diagram illustrated in Figure 22 shows the process of performing a secure search operation, emphasizing the use of homomorphic encryption to process queries and retrieve relevant assets while preserving privacy. The workflow begins with the 'processSearchOperation()' function, which orchestrates the search logic and prepares the system for operations on encrypted data.

Next, the sequence involves invoking the 'performHomomorphicSearch()' function. This function utilizes homomorphic encryption techniques in the same manner from the previous methods, allowing computations to be carried out directly on encrypted data without the need for decryption. This ensures that sensitive data remains secure throughout the search process. The function identifies potential matches by evaluating the encrypted data against the encrypted query parameters.

The workflow then enters a loop to retrieve and validate assets that match the search criteria. Within this loop, the 'getAsset()' function retrieves individual encrypted assets corresponding to the query. Following that, the 'performHomomorphicRegExp()'

Figure 22 – Perform Search Operation Sequence Diagram.


sd Perform Search Operation
Encryption Module
:controllerBlockWitnessEncryption
Assets
processSearchOperation
performHomomorphicSearch()
loop
getAsset()
Asset
performHomomorphicRegExp()
Assets Ids
Write to Broker Message

Source: The Authors.

function applies homomorphic regular expression matching to the encrypted assets. This step enables advanced pattern-matching capabilities while keeping the data encrypted, ensuring that results are derived without compromising security.

Once the loop completes and all matching assets are identified, the system consolidates the results and prepares them for transmission. The final step, "Write to Broker Message," securely delivers the results back to the requester in a brokered message format, in the specific case, throwing back to same queue system. This ensures a reliable and secure channel for conveying the encrypted search results.

The process elucidates an advanced approach to secure data retrieval utilizing homomorphic encryption. The incorporation of homomorphic search and pattern-matching techniques highlights the system’s focus to preserving data privacy throughout the process. Featuring a modular and iterative design, the system includes functions such as ’performHomomorphicSearch()’ and ’performHomomorphicRegExp()’, which demonstrate its flexibility and scalability. This methodology is particularly well-suited for environments where sensitive data must remain encrypted, such as privacy-preserving databases, encrypted cloud storage solutions, and secure blockchain platforms. The implementation of homomorphic encryption ensures that the system complies with the highest data security standards while offering sophisticated search and retrieval capabilities.

4.3 SECURITY ANALYSIS

An analysis of the adopted technologies, systems and networks was realized in order to identify potential threats and develop ways to mitigate these threats. The threat model presented in Figure 23 highlights the main vulnerability points of the application, as well as the types of attacks, threats, possible security controls and mitigation measures.

In our studies, a group of threat actors were identified, as shown in 23, these actors are:

- Malicious User: This category encompasses internal actors within an organization who may intentionally or unintentionally cause security breaches.
- Internal Data Center Attacker: Represents individuals with physical or network access to data centers who could exploit their position to execute attacks.
- External Attacker: External attackers do not have authorized access to an organization's systems but attempt to breach security from outside.
- Software Provider: Vulnerabilities introduced by third-party software providers that might be exploited.

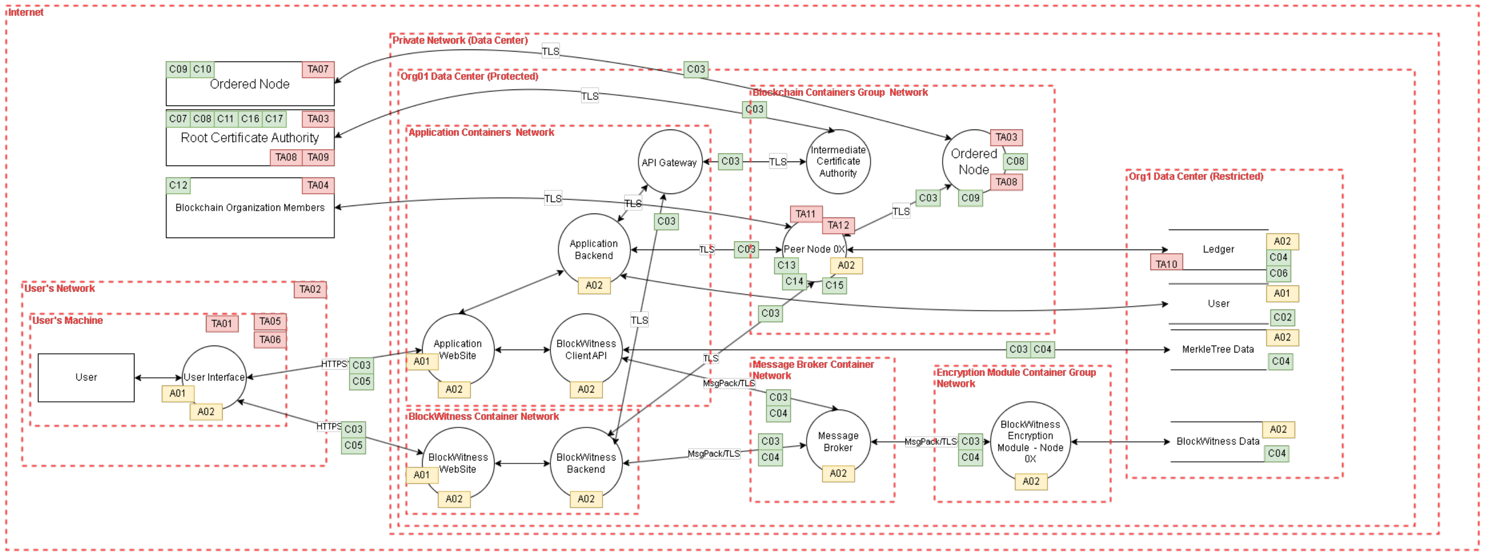
Considering these actors, some attacks and threats were identified with the potential to harm the functioning of the BlockWitness system and the main application, as shown in figure. These are:

- Misused, Leaked: A incidents where sensitive information is misused or accidentally leaked due to insufficient security controls.
- Man-in-the-middle: An attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other.
- Compromised Root Authority: This threat involves the compromise of a root certificate authority, which could undermine the security of Transport Layer Security (TLS) certificates across the internet.
- Compromised Organization: Indicates a broader security breach within an organization, possibly impacting multiple systems and data.
- Phishing Attacks: A typical cyber attack where attackers deceive individuals into providing sensitive data by masquerading as a trustworthy entity in digital communications.

- Stolen Credentials: A incidents where user credentials (like usernames and passwords) are stolen, typically through hacking or phishing.
- Sybil Attacks: A type of attack on a network service where the attacker subverts the service by creating a large number of pseudonymous entities.
- Private Key Theft: Involves the unauthorized access and theft of private cryptographic keys, which are crucial for securing communications.
- Fake Certificate Issuance: Pertains to the creation and distribution of fraudulent digital certificates, which can be used to spoof the websites or intercept secured communications.
- Unauthorized Ledger Modification: A unauthorized changes made to a blockchain ledger.
- Smart Contract Vulnerabilities: Weaknesses within the code of smart contracts deployed on blockchain networks. Common issues include reentrancy attacks and reliance on untrusted contract inputs.
- Unauthorized Chaincode Execution: Unauthorized or unintended execution of chain code, which could potentially lead to security breaches or data manipulation.

The correlation between the actors and the types of attacks and threats is presented in Table "Threat Actors " of the figure 23.

Figure 23 – BlockWitness Threat Model.


Internet
Private Network (Data Center)
Org01 Data Center (Protected)
Blockchain Containers Group Network
Application Containers Network
BlockWitness Container Network
Message Broker Container Network
Encryption Module Container Group Network
Org1 Data Center (Restricted)
User's Network
User's Machine
User
User Interface
Ordered Node
Root Certificate Authority
Blockchain Organization Members
API Gateway
Intermediate Certificate Authority
Application Backend
Application WebSite
BlockWitness ClientAPI
BlockWitness WebSite
BlockWitness Backend
Peer Node 0X
Message Broker
BlockWitness Encryption Module - Node 0X
Ledger
MerkleTree Data
BlockWitness Data
TLS
HTTPS
MsgPack/TLS

Assets	
ID	Description
A01	User Credential
A02	Application Data

Threat Actors	
ID	Description
Malicious user	TA01, TA02, TA05, TA06
Internal Data Center Attacker	TA02, TA07, TA08, TA09, TA10, TA12
External Attacker	TA03, TA04, TA07
Software Provider	TA03, TA04, TA09, TA10, TA11

Security Controls	
ID	Description
C01	Authentication, 2FA
C02	Password Hashing
C03	Encrypted Link
C04	Cryptographic Hashing
C05	MFA
C06	Append-Only Ledger
C07	HSM
C08	Certificate Signing Policies
C09	Identify Verification
C10	Permissioned Network
C11	Role Based Access Control
C12	WAF
C13	Endorsement policies
C14	Chaincode Authentication
C15	Formal Verification, testing and audits
C16	Key rotation
C17	Access Monitoring

Threat/Attacks	
ID	Description
TA01	Misused, leaked
TA02	Man-in-the-middle
TA03	Compromised Root Authority
TA04	Compromised organization
TA05	Phishing attacks
TA06	Stolen Credentials
TA07	Sybill attacks
TA08	Private Key Theft
TA09	Fake certificate issuance
TA10	Unauthorized ledger modification
TA11	Smart contract vulnerabilities
TA12	Unauthorized Chaincode execution

Source: The Authors.

Considering these threats, in our proof of concept some security controls have already been adopted, among which we can mention:

- Single-Factor Authentication (SFA): The user provides a password to identify himself.
- Password Hashing: The security practice of converting a password into a format that cannot be easily reversed.
- Encrypted Link: Use of encryption to secure links, for data transmission. Common examples are the adoptions of secure protocols like Transport Layer Security (TLS) 1.3 and Secure Shell (SSH) 2.0.
- Cryptographic Hashing: Securing data by transforming it into a fixed-size hash that cannot be easily reversed, in our case protecting part of the data with homomorphic encryption.
- Append-Only Ledger: Used in blockchain, indicating a record-keeping system where entries can only be added, not deleted or altered.
- Certificate Signing Policies: Defined policies to creating and distributing digital certificates that confirm the identity of entities.
- Identity Verification: The process of verifying the identity of a person or entity. E.g., used by the ordering service to identify nodes.
- Permissioned Network: The blockchain network where a central authority controls access.
- Chaincode Authentication: The authentication mechanisms for smart contracts in blockchain networks.
- Endorsement Policies: In blockchain these define the rules for how transactions must be validated before being committed to the ledger.

Other measures may be adopted to increase BlockWitness and main application security as indicated in the threat model, namely:

- Two-Factor Authentication (2FA): It might enhances security by requiring two distinct forms of identification. E.g., a user might enter a password and then a code sent to their phone.
- Multi-Factor Authentication (MFA): Uses two or more verification methods from different categories of credentials, significantly increasing security.

- Hardware Security Module (HSM): A physical computing device safeguards and manages digital keys for strong authentication.
- Role-Based Access Control: A method of regulating access to computer or network resources based on the roles of individual users within an enterprise.
- Web Application Firewall (WAF): A security system specifically for monitoring, filtering, and blocking data packets traveling to and from the web application.
- Formal Verification, Testing, and Audits: Processes to ensure systems operate correctly and securely, according to formal specifications.
- Key Rotation: A practice of regularly changing encryption keys to enhance security.
- Access Monitoring: The tracking and recording of actions regarding resource access, ensuring compliance and security.

The threats and attacks listed only consider the basis of the BlockWitness architecture, which despite mentioning them, does not explore the attacks and vulnerabilities of the blockchain network. Important to mention, the version 2.5 of HFP used in this study uses the Crash Fault Tolerance (CFT) ordering service and is known to be susceptible to a subset of nodes behaving maliciously. In new version 3.0, HFP adopts a Byzantine Fault Tolerance (BFT) ordering service based on the SmartBFT consensus library (Barger et al., 2021). The platform's creators suggest considering the BFT orderer for achieving true decentralization. This is especially important if up to a third of the parties operating the orderers may be untrustworthy, either due to malicious intent or compromise. (HYPERLEDGER FABRIC, 2024b).

4.4 CONSIDERATIONS:

The BlockWitness system is a promising solution that combines blockchain technology with homomorphic encryption to improve security and auditability. By integrating client-side data encryption, efficient data logging utilizing Merkle Trees, and asynchronous operations through a queue-based system, it guarantees reliable performance even under significant computational loads. The provided architecture, is designed for modularity and scalability, supports parallel processing, making it well-suited for applications such as secure data searches and environmental compensation systems like Carbon21. Furthermore, the system's integration with blockchain and message brokers promotes both transparency and real-time communication, effectively addressing key requirements in decentralized data ecosystems.

While the suggested system excels in managing intensive operations such as homomorphic searches, its dependence on asynchronous processes may pose challenges in maintaining user interactions and are necessary additional mechanisms for status updates which can result in computational overload. This design choice underscores the trade-offs between operational efficiency and real-time feedback. Overall, BlockWitness may provide a structured approach to secure data management, with valuable applications in industries that demand transparent and reliable systems, including financial, health, supply chain management and environmental sustainability initiatives.

5 EXPERIMENTAL ANALYSIS

A Proof of Concept (PoC) was developed to verify the concept's feasibility. It was also integrated into another project created with blockchain technology. The PoC development took over two years and is available in <https://bitbucket.org/patrocineinc/blockwitness> (It may be necessary to request access). The first version was developed considering the default sample provided by Hyperledger Fabric Platform (HFP), using the provided smart contracts (chaincodes), metadata, and asset formats.

A second version was produced using the Carbon21 project in a more complex environment with more structured assets and metadata. This allowed the project to affirm his goals to achieve flexibility and adaptability to different applications and situations. Furthermore, it offers the opportunity to experience the blockwitness system in a real-world context.

5.1 PROOF OF CONCEPT ARCHITECTURE

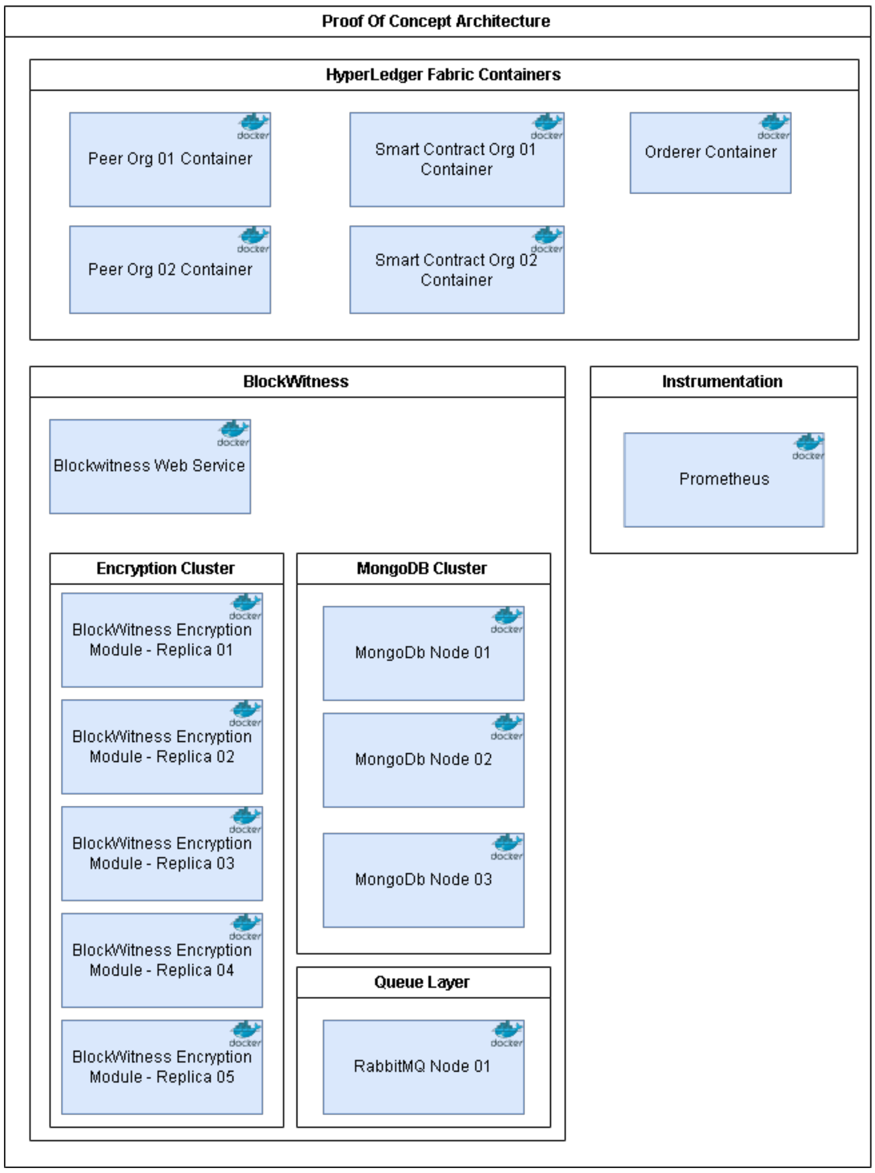
The PoC architecture and organization were created considering the two aforementioned scenarios. Details of the select components and technologies used are described in this section.

Figure 24 demonstrates the initial version of the BlockWitness project using the HFP sample as the base project. The system was built around the provided solution, developing a front-end web solution without modifying the chain code structure and asset metadata. The developed part included a library responsible for sending and receiving blockchain data and communicating with the BlockWitness encryption module. Figure 25 represents the second phase developed over the Carbon21 project. In this case, the front end needed to be changed to attach the BlockWitness client library.

The first architecture scenario shown in Figure 24 are:

- Peer Org Containers (Org 01 and Org 02): These are core nodes that validate and endorse transactions, acting as the backbone of the distributed ledger.
- Smart Contract Containers: Each organization hosts its own chaincode (smart contract) in isolated containers to execute business logic.
- Orderer Container: Central to the consensus mechanism, ensuring consistent transaction ordering and block creation.
- BlockWitness Web Service: It likely functions as an intermediary interface for accessing blockchain data or conducting customized operations.

Figure 24 – Proof of concept architecture.

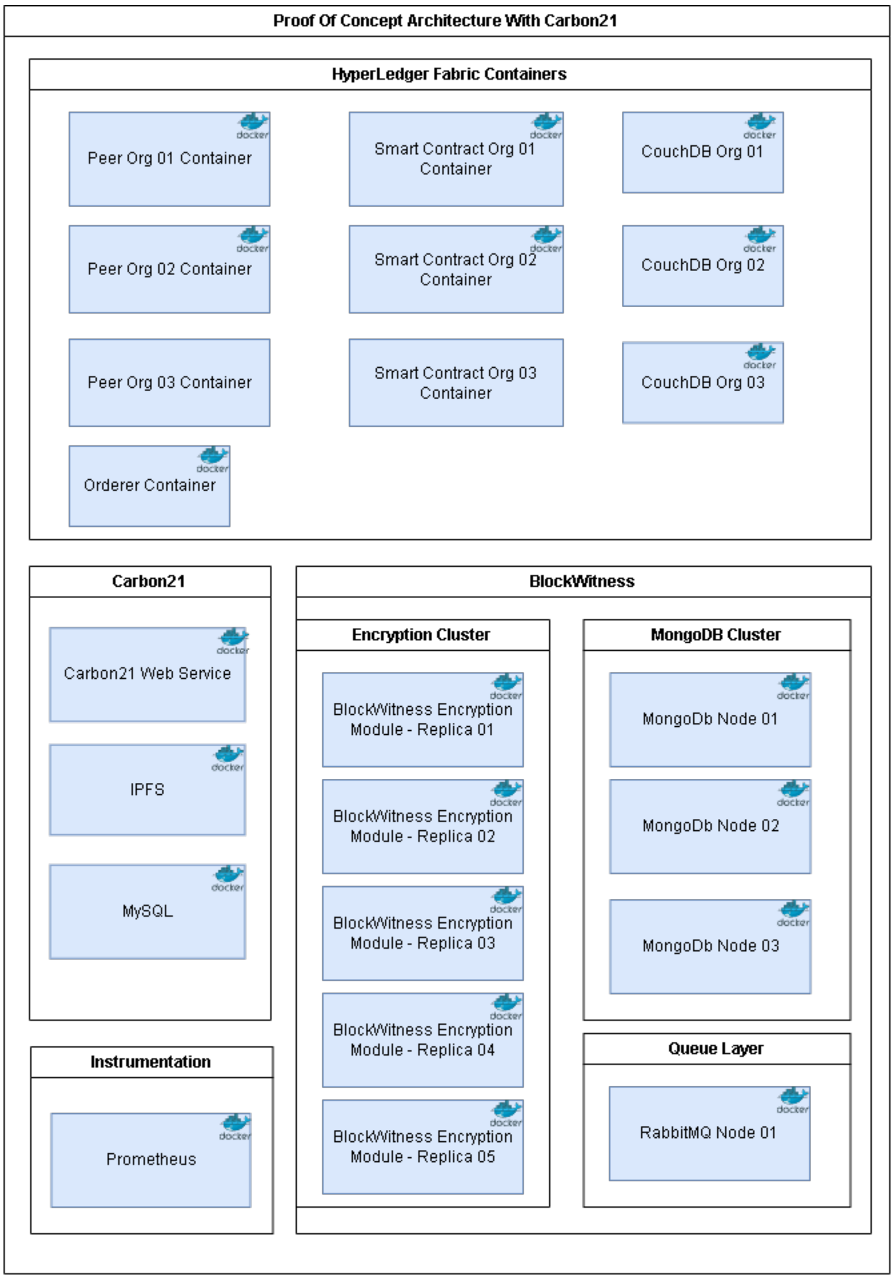

Proof Of Concept Architecture
HyperLedger Fabric Containers
Peer Org 01 Container
Smart Contract Org 01 Container
Orderer Container
Peer Org 02 Container
Smart Contract Org 02 Container
BlockWitness
Blockwitness Web Service
Instrumentation
Prometheus
Encryption Cluster
BlockWitness Encryption Module - Replica 01
BlockWitness Encryption Module - Replica 02
BlockWitness Encryption Module - Replica 03
BlockWitness Encryption Module - Replica 04
BlockWitness Encryption Module - Replica 05
MongoDB Cluster
MongoDb Node 01
MongoDb Node 02
MongoDb Node 03
Queue Layer
RabbitMQ Node 01

Source: The Authors.

- Encryption Cluster: A dedicated set of Blockwitness Encryption modules (replicas) for cryptographic operations, such as search operations.
- MongoDB Nodes: Three nodes configured to manage data storage.
- RabbitMQ Node: A message broker for asynchronous communication.
- Prometheus: A monitoring tool for performance metrics and system health.

It is important to highlight that for the version of HFP adopted, the default consensus algorithm employed is the Crash Fault Tolerance (CFT), which operates as a Proof-of-Authority (PoA) model. Specifically, the orderer service is implemented using the Raft algorithm. This decision is not expected to influence the outcomes, as the architecture employed decouples the blockchain from the BlockWitness system.

Figure 25 – Proof of concept architecture with Carbon21.


Proof Of Concept Architecture With Carbon21
HyperLedger Fabric Containers
Peer Org 01 Container
Smart Contract Org 01 Container
CouchDB Org 01
Peer Org 02 Container
Smart Contract Org 02 Container
CouchDB Org 02
Peer Org 03 Container
Smart Contract Org 03 Container
CouchDB Org 03
Orderer Container
Carbon21
Carbon21 Web Service
IPFS
MySQL
BlockWitness
Encryption Cluster
BlockWitness Encryption Module - Replica 01
BlockWitness Encryption Module - Replica 02
BlockWitness Encryption Module - Replica 03
BlockWitness Encryption Module - Replica 04
BlockWitness Encryption Module - Replica 05
MongoDB Cluster
MongoDb Node 01
MongoDb Node 02
MongoDb Node 03
Queue Layer
RabbitMQ Node 01
Instrumentation
Prometheus
docker

Source: The Authors.

In the second architecture scenario, several notable changes have been made. A third organization has been added, and the HPC storage system has been replaced by CouchDB, with one instance allocated for each organization. Additionally, two complementary tools have been introduced: IPFS and MySQL Database. These differences in the system are illustrated in Figure 25:

- Carbon21 Web Service: It likely serves as an interface or middleware that enables interaction between blockchain systems and off-chain systems. It also provides an interface for handling operations such as user authentication and asset consultation.

- CouchDB: Serves as a state database for storing world states in a queryable format, replacing the default LevelDB storage system for HFP.

- MySQL: A relational database for structured data storage.

- Interplanetary File System (IPFS): A distributed file system for storing and sharing large or immutable data.

As the blockchain selected in both scenarios was HFP in a permissioned configuration, the results are focused on the complementary developed system and not the blockchain projects themselves. More information from the blockchain projects, such as architecture details and benchmarking, can be found in the research made by the Carbon21 team and available at: https://github.com/Carbon-21/4orgs-ERC1155 As well, currently, the main challenges with homomorphic encryption are the computation effort and increased data size, so the selected metrics focus on analyzing parameters related to this challenge.

In contrast to the default example provided with HPF, which employs LevelDB (GOOGLE, 2024b), the Carbon21 system utilizes CouchDB(APACHE, 2024) for the purpose of blockchain data persistence. Furthermore, the Carbon21 system incorporates a MySQL database to ensure the persistence of its core system's data.These differences may not impact captured results, as the blockchain record times are being disregarded.

Although Figure 25 does not explicitly depict this aspect, it is essential to consider the architecture of the web system. For the HFP example developed in this project, Next.js is utilized, whereas the Carbon21 system adopts a different approach by using Express.js (EXPRESS, 2024) with EJS for templating (EJS, 2024).

Results were generated using Prometheus (PROMETHEUS, 2024) and Grafana K6 (GRAFANA K6, 2024), a group of open-source tools that makes viable monitor server stats and run load testing. Two custom-built metrics were built with the K6 Browser module (GRAFANA, 2024) and Chromium (THE CHROMIUM PROJECT, 2024) to measure the transaction creation and verification time, varying both the number of users and the payload sizes. For rust functions, benchmarking uses the provided time function from rust language, 'std::time::Instant' (RUST LANG, 2024). All other metrics were set up using the default K6 metrics.

5.2 TEST PLAN

This section exposes how the test plan was developed according to the initial set objective. To validate the aim of making it feasible to extend homomorphic encryption to everyone who already has a blockchain system running, the first test set was built to consider whether it is possible to add homomorphic encryption in asset creation. So, for this purpose, the first strategy was to capture the whole creation process time and see how much this would impact the entire end-user experience.

The second strategy concerns the verification process; the main question is how much time a verifier will spend and how much this will affect the original system in computational cost and stakeholder experience. These metrics are more complex to develop and complete, so it is necessary to have a clear view of who will be involved in the process and get the benefits of the verification processes. Figure 13 shows the main involvement participants for the initially defined case; it was consider the regulator and auditors in this strategy.

The third part aimed to test the system's feasibility and the number of resources necessary to expand an existing system, so it considered the whole extra structure willing to support the complementary system that BlockWitness demands.

A fourth strategy was considered: user acceptance, an integral part of a system that extends to another. The threaded case shows two important users: the developer and the final user. However, that aspect should be considered in further experiments because of time and resource limitations. The Future Works section will discuss this aspect more. Based on this strategy, test plans present in Tables 4, 5, 6 and 7 where defined.

Table 4 – Test Plan 01- Asset Creation.

Attribute	**Details**
Test Plan ID	01
Test Name	Asset Creation
Test Strategy	Capture the end-to-end time spent to create an Asset.
Test Criteria	A test that has less than 10% time overload.
Test Deliverable	A graph and a report analysis.
Risk Analysis	Network buffers, and other uncontrolled parameters may impact the final result.
Resource Requirements	BlockWitness Web Service, MongoDB, RabbitMQ, Blockchain HFP, K6, k6 Browser Module, Chromium
Dependencies	Systems were set up at an early stage.

Source: The Authors.

Table 5 – Test Plan 02- Asset Verification by hash.

Attribute	Details
Test Plan ID	02
Test Name	Asset Verification by hash
Test Strategy	Capture the verification time in the Merkle Tree
Test Criteria	A verification time lasting 1 seconds or less.
Test Deliverable	A graph and a report analysis.
Risk Analysis	Different browsers, network buffers, and other uncontrolled parameters may impact the final result.
Resource Requirements	BlockWitness Web Service, MongoDB, RabbitMQ, Blockchain HFP
Dependencies	Systems were set up at an early stage.

Source: The Authors.

Table 6 – Test Plan 03 - Search Asset.

Attribute	Details
Test Plan ID	03
Test Name	Search Asset
Test Strategy	Capture the search time
Test Criteria	A search time lasting 10 seconds or less.
Test Deliverable	A graph and a report analysis.
Risk Analysis	Different browsers, network buffers, and other uncontrolled parameters may impact the final result.
Resource Requirements	BlockWitness Encryption Module, MongoDB, RabbitMQ
Dependencies	Systems were set up at an early stage.

Source: The Authors.

Table 7 – Test Plan 04 - System Performance Measurement.

Attribute	Details
Test Plan ID	04
Test Name	System Performance Measurement
Test Strategy	Get computational costs running a non-functional test using performance and loading tests.
Test Criteria	CPU average usage lower than 70%, Memory usage less than 90%.
Test Deliverable	A graph and a report analysis.
Risk Analysis	Different browsers, network buffers, and other uncontrolled parameters may impact the final result.
Resource Requirements	Prometheus
Dependencies	Systems were set up at an early stage.

Source: The Authors.

Software details and versions are exposed in the next section in Table 8.

5.3 TESTBED

The tests were performed on an infrastructure available at the Parallel and Distributed Processing Laboratory (LabP2D) from UDESC University. The utilized server has two Intel Xeon Silver 4216 processors, each with 16 physical cores and 32 threads, totaling 64 threads. It also has 96 GB of RAM and 960 GB of disk storage. For graphical and remote management, it uses an ASPEED Graphics Family GPU. The operating system installed is Ubuntu 20.04.6 Long-term support (LTS).

All the infrastructure was built using docker containers, with docker-compose version 1.28.5 (DOCKER, 2024a) and docker version 27.1.2 (DOCKER, 2024c). Fig. 24 shows all containers used to execute the test plan of section 5.2. Table 8 displays a list of the main software dependencies required for the Blockwitness environment.

Table 8 – Hyperledger and BlockWitness Related Dependencies.

Category	**Details**
Hyperledger	Fabric: v2.5.0
BlockWitness Web Module	TypeScript: 5 Next.js: 14.1.4 Node-TFHE: 0.9.1 React: 18 MerkleTreeJS: 0.3.11 AMQPLib: 0.10.4 Hyperledger/Fabric-Gateway: 1.5.0 MsgPack: 3.0.0
BlockWitness Encryption Module	Rust: 1.78.0 TFHE: 0.10.0 AMQPRS: 1.6.1 Tokio: 1 Bincode: 1.3.3 RMP_Serde: 1.3.0
For Both	RabbitMQ: 3.13 MongoDB: 6
For Testbed	Prometheus: 2.32.1 K6: 0.54.0 K6 Browser: 1.8.5 Chromium: 131.0.6778.85

Source: The Authors.

The distributed ledger structure uses the version 2.5.0 of Hyperledger Fabric, establishing it as the foundational blockchain platform. The TypeScript section outlines essential libraries and frameworks, including TypeScript v5, Next.js v14.1.4, React v18, along with additional libraries such as node-tfhe for cryptographic operations and merkletreejs for working with Merkle Trees. The encryption module, written in Rust, includes key dependencies such as Rust v1.78.0, tfhe 0.10.0 for homomorphic encryption, tokio for asynchronous runtime, and both bincode and rmp_serde for serialization. Finally, shared resources consist of RabbitMQ v3.13 for message brokering and MongoDB version 6 for database storage, ensuring effective communication and data management throughout the system. This well-structured setup facilitates scalability, efficiency, and security in the system's overall operation.

5.4 RESULTS

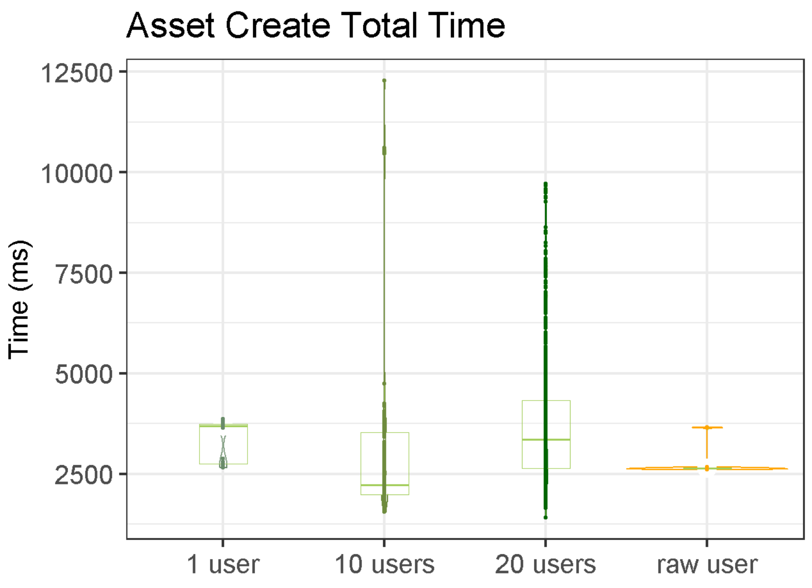
This chapter presents the results found during the study. The tests performed are in accordance with the test plan presented in Chapter 5 Section 5.2, and the testbed in accordance with Section 5.3.

5.4.1 Results Test 01

The initial test was carried out in accordance with Test Plan 01, aiming to measure the time required to create an asset, from the moment the user initiates the call to the point at which the call is confirmed. This encompasses both the web service process and the encryption process. For this test, it was excluded the time taken for key creation, as this operation is performed only once during each setup.

In this testing scenario, a total of 1,600 tests were generated and organized into three distinct rounds. The first round involved one user, the second round included ten simultaneous users, and the third round featured twenty simultaneous users. Furthermore, an additional round was conducted using a raw asset, meaning the tests were performed without the encryption process. Each user participated in 50 iterations, resulting in the total number of tests reported.

Figure 26 – Asset create time.



Source: The Authors.

As can be seen in Figure 26, the result times were consistent while varying the number of simultaneous users, as a comparative the *1 raw user* refers to an asset creation without cryptography. The violin and the box-plot graph shows the archived results. The follow statistics result are shown in Table 9.

Table 9 – Asset create statistics.

Users	**Mean(ms)**	**Median(ms)**	**Standard Deviation(ms)**	**Maximum**	**Minimum**
1 raw user	2714.24	2635.35	278.99	3654.30	2606.80
1 user	3312.35	3691.85	501.24	3868.40	2647.10
10 users	2776.67	2215.70	1378.21	12280.40	1552.00
20 users	3680.89	3347.35	1475.43	9720.10	1408.10

Source: The Authors.

As the number of users grows, it can be notice a gradual increase in the standard deviation. For this scenario, it had been examined a single instance of the web container, where all users were accessing the resource simultaneously, meaning the outcome presented was as expected. For future research, it would be beneficial to distribute the load across multiple instances using load balancing. Nevertheless, the median asset creation times reveal that, in certain cases, the same creation time could be achieved with and without encryption.

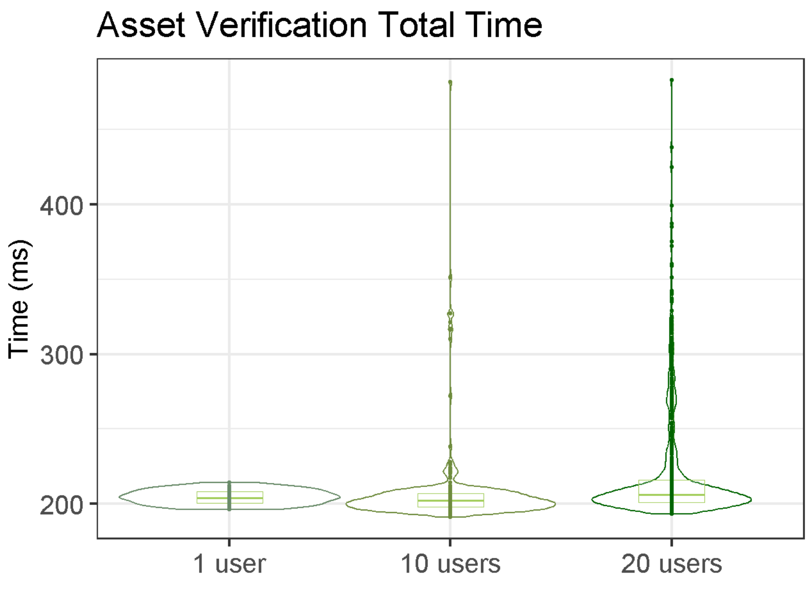
What is clear is that it cannot define a result based on a single user. The results with the largest number of users presented better averages. This result comes from the way the web server and browser manage caching and reuse of components. The Next.js framework incorporates a built-in caching mechanism aimed at improving application performance and reduce costs (NEXT.JS, 2024a). This mechanism influences both the handling of data requests and the rendering of components within the web browser. A more comprehensive study is needed to isolate and evaluate the impact of each of these components across various scenarios.

The main point studied in this test was the difference between environments with and without encryption, and also the scalability of the process. Therefore, it can be seen that there is no major impact on this process, and the encryption process did some considerable increase in the result, but not enough to be considered impactful. Homomorphic encryption with one user represents a 22.04% increase in the mean response time compared with the process without encryption (raw data). This results in an increase of 598 ms in the end user's response time. The differences in results ranged from a minimum of 40 ms (1.55% of the lower value) to a maximum of 214 ms (5.86% of the lower value), respectively.

5.4.2 Results Test 02

In the context of the tests conducted under Test Plan 02, the results were consistent with those obtained in the previous testing phase, indicating a stable performance across various samples. However, a notable disparity becomes evident as the number of users increases, as can be seen below.

Figure 27 – Asset verify time.



Source: The Authors.

For this test case was used the same input data and number of users from Test Plan 01 scenario, but with removal of the extra scenario compound by the raw data.

This way, a total of 1,550 tests were generated. Figure 27 and table 10 demonstrates the results found.

Table 10 – Asset verify statistics.

Users	Mean(ms)	Median(ms)	Standard Deviation(ms)	Maximum	Minimum
1 user	204.46	204.00	4.89	214.00	196.00
10 users	206.30	202.00	24.51	482.00	191.00
20 users	218.80	206.00	34.50	483.00	193.00

Source: The Authors.

The mean values show a small variation of 14.34 ms between samples, which represents approximately 7.01% of the lower value. When analyzing the medians, the variation is 4 ms, representing approximately 1.98% of the lower value. Therefore, it can be infer that the increase in the number of simultaneous users did not represent a significant increase in asset verification times.

It is crucial to consider specific information related to the creation and verification of time recordings. Throughout all tests, the web service was restarted, and it is important to acknowledge that there was no optimization of the process. While this aspect was considered in the tests, it was not fully addressed. Additional exploration is warranted, as well recommended by the creators of the Next.js framework that is being utilized (NEXT.JS, 2024b).

The use of memory remains at an average value during the variation in the number of users, but seeing spikes at certain times, considering that all users were connected to the same entry point, it may be related to the singular use of a web server. Future studies should explore a load balance and distribution in this regard.

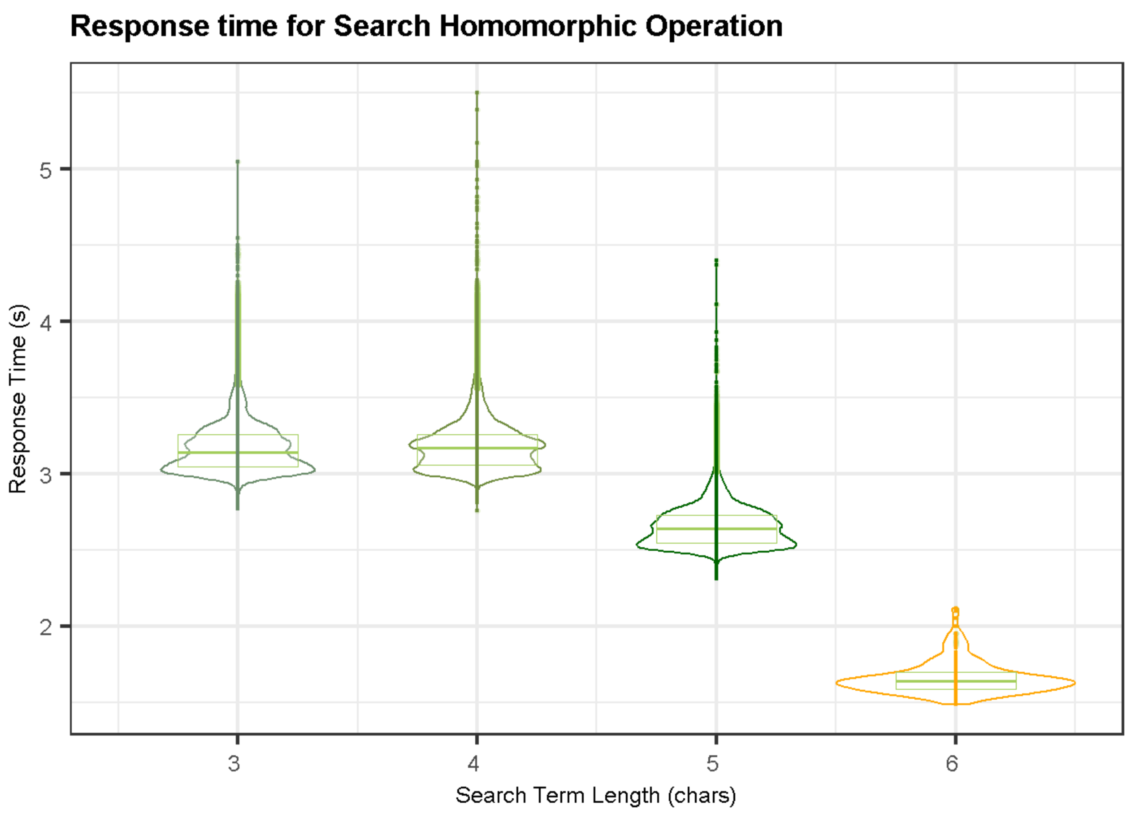
In the tests it was verified that the encryption time did not affect the system significantly, in a way that affected the end user experience, even when multiple users performed simultaneous operations. More detailed observations can be found in the results of memory and Central Processing Unit (CPU) consumption from Test Plan 04.

5.4.3 Results Test 03

The search results were somewhat unexpected at the first glance. As the number of characters used in the test was increased, better response times were found in the results. Figure 28 shows on the x axis the evolution of 3 to 6 characters for search term query, and the operation response time in the y axis. Details about the results obtained for this operation can be seen in the table 11.

This operation was performed in 5 instances divided into containers that were running the encryption module of the BlockWitness system. While some peaks were noted in the execution times, the medians demonstrated consistent results, with the standard deviation remaining quite stable across samples, ranging from a minimum of 100 ms to a maximum of 300 ms. This variation reflects 0.66% and 1.81% of the shortest average time recorded, respectively.

Figure 28 – Search time for homomorphic operation.



Source: The Authors.

However, according to the implemented algorithm (more details were shown in Chapter 4), some factors has to be considered to achieve this result. The use of the cache during execution, that is, previous results are reused and optimizes performance, and also the way the programming language (Rust) adopts and processes regular expression patterns, the basis of the algorithms used by Fast Fully Homomorphic Encryption over the Torus (TFHE). Although, the most relevant factor is the implementation of the search engine itself. The encrypted content is segmented and organized into branches according to the regular expression used during the search. In this particular test, wildcards were not used, reducing the number of branches for longer search terms and, consequently, fewer executions in the search process. This resulted in a reduction in the resulting times for searches with 5 and 6 characters.

Table 11 – BlockWitness Search Operation Statistics.

Search Term Length	Mean (s)	Median (s)	Standard Deviation(s)	Maximum (s)	Minimum (s)
3	3.17	3.14	0.18	5.05	2.77
4	3.19	3.17	0.17	5.50	2.76
5	2.66	2.64	0.14	4.40	2.31
6	1.66	1.64	0.11	2.12	1.49

Source: The Authors.

5.4.4 Results Test 04

In Test Plan 4, the results are categorized into two distinct groups: the first group encompasses memory usage metrics, while the second group analyzes CPU consumption data.

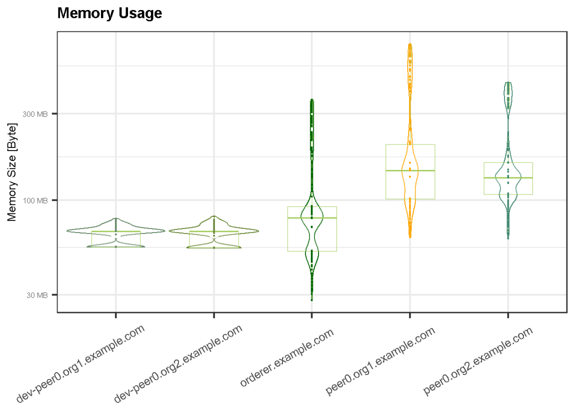
The tests were executed using the containers' default settings, without any additional configurations to restrict memory and CPU usage. However, it is worth noting that Docker provides options to limit both CPU and memory consumption for containers. Users can define the maximum amount of memory a container may utilize, enable swap space to transfer excess memory to disk and set kernel memory. Furthermore, you can specify which CPUs or cores a container is permitted to access, establish a CPU Completely Fair Scheduler (CFS) quota, and apply other advanced configurations to control CPU usage (DOCKER, 2024b).

All metrics collected were obtained using the Prometheus software tool, as described in Section 5.3, that is, the values expressed here are relative to the memory and CPU usage of the specified docker containers. Regarding the application's memory and CPU usage, three groups of processes were analyzed:

- HFP Container: The group of Hyperledger Fabric Platform (HFP) sample containers.
- BlockWitness Container: The group of BlockWitness system containers.
- Services Container: The group of auxiliary services container, in this case, MongoDB and RabbitMQ instances.

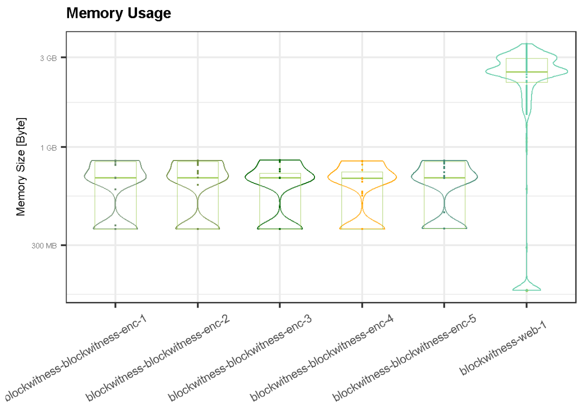
CPU usage was collected using the metric *container_cpu_user_seconds_total* from Prometheus. The calculation used to generate the graphs presented forms the difference between successive values of each sample. For the memory metrics was used the *container_memory_usage_bytes* from the same tool.

Figure 29 – HFP memory usage.



Source: The Authors.

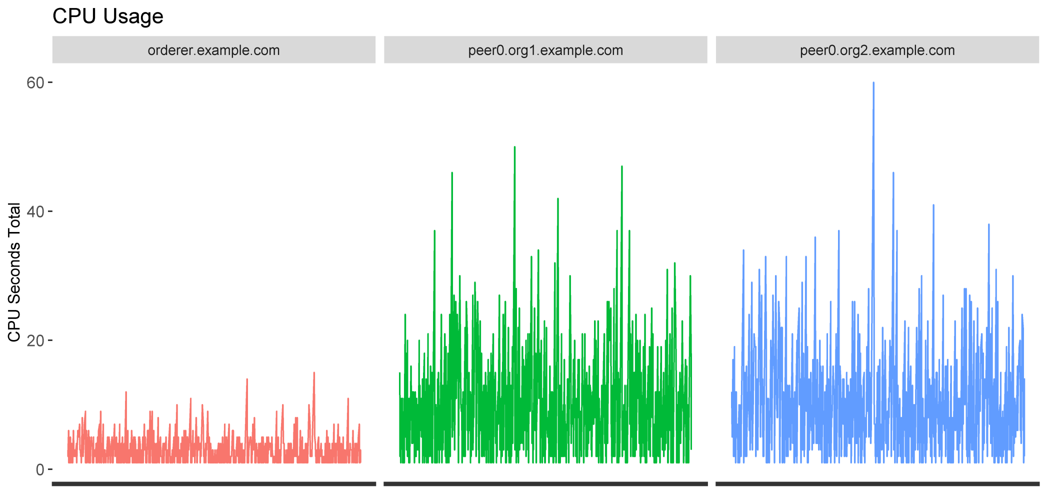
Figure 30 – BlockWitness memory usage.



Source: The Authors.

Firstly, the memory and CPU usage of the containers used by HFP was evaluated. Containers starting with *'dev'* prefix represent Smart Contract Containers (Chaincode Containers) for the organizations involved. The demand for these containers was minimal, leading to very low memory consumption. As there were no changes to smart contracts throughout the BlockWitness testing process, there was no significant activation of these containers. Consequently, the CPU usage results were considered insignificant and were disregarded. As all communication with the blockchain was carried out directly in organization one (org1), the results can be seen in the peer0.org1 results, which show the highest memory and CPU usage. In any case, the usage of the peer container from peer organization two did not show a big difference in memory usage, despite a small difference can be noticed with higher maximum from CPU usage from peer organization one, as can be observed in Table 15.

Figure 31 – BlockWitness - Fabric CPU usage.



Source: The Authors.

For the tested version of HFP, the consensus algorithm used was CFT, as mentioned previously in Chapter 4. This choice has not affect the results of the Block-Witness system, but it must be considered that changes in the adopted consensus may bring different results in the orderer service container.

The results for HFP containers are displayed in Tables 15 and 12, and Figures 31 and 29.

Table 12 – HFP Containers Memory Usage Statistics.

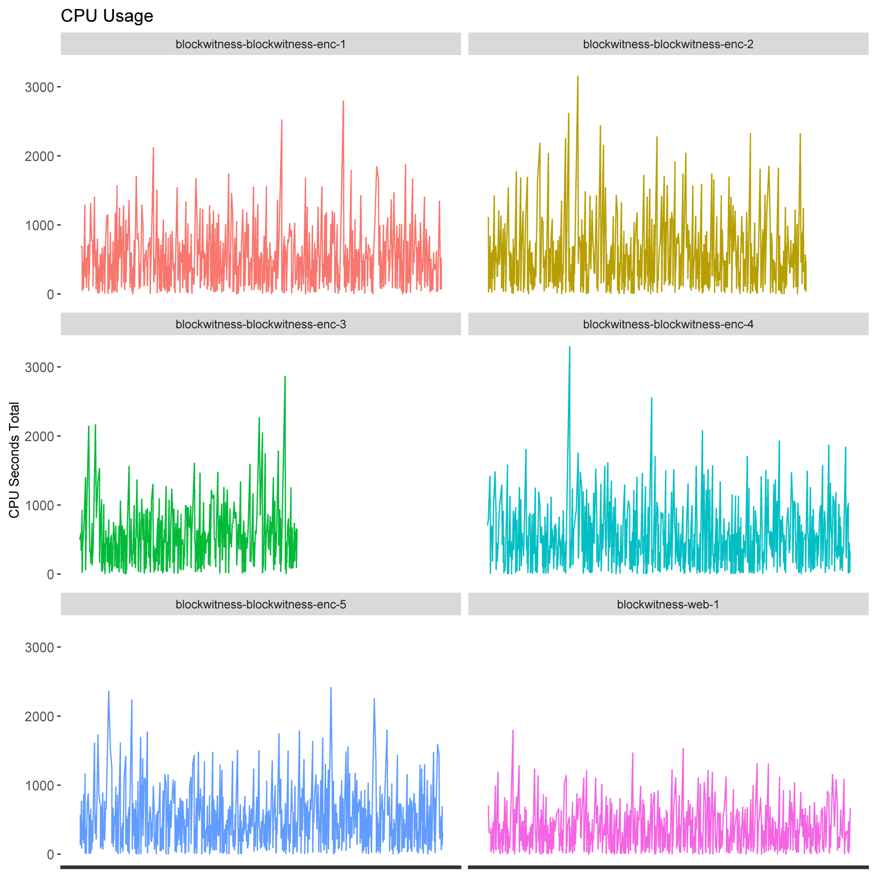
Container Service	Mean (MB)	Median (MB)	Standard Deviation(MB)	Maximum (MB)	Minimum (MB)
dev-peer0.org1	64.84	67.37	6.91	78.85	54.98
dev-peer0.org2	64.93	67.36	7.44	81.136	54.53
orderer	106.99	79.74	79.89	356.35	28.01
peer0.org1	217.10	145.42	173.49	723.89	62.44
peer0.org2	177.24	132.99	105.96	443.56	61.37

Source: The Authors.

For the second group, the memory and CPU use of the BlockWitness system was evaluated. For the Web system, the results already mentioned above can be observed, with high memory and CPU usage by this container. As there was no optimization and better distribution of this load, it presented moments of overload, compared to other containers. In any case, the medians and standard deviation presented low values compared to when the maximum value reached in memory and CPU usage is observed. Similar means, medians, and standard deviations were obtained in memory usage among all containers

The best results memory for BlockWitness modules were found in the encryption module, representing on average 27.50% of memory usage of the web module. This outcome results from the manner in which the system was distributed and load-balanced via queue consumption, employing the producer-consumer model. Conversely, when examining CPU usage, the situation changes; the web module utilized, on average, only 64.86% of the CPU time when compared to the encryption module. This behavior was expected, given that the encryption modules are responsible for traversing the stored assets and executing homomorphic search operations on each resultant item.

Figure 32 – BlockWitness CPU usage.



Source: The Authors.

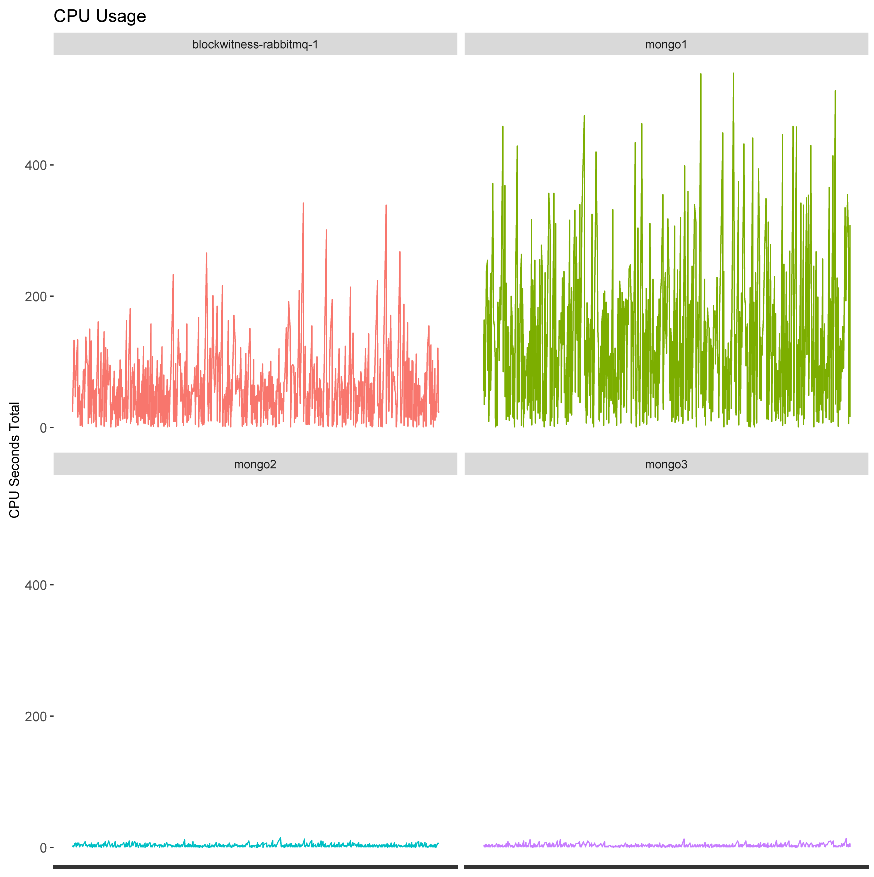
Also, there is a small variation in CPU usage between the encryption modules containers, which is a result of the way the distributed system was designed, with no communication between the containers to guarantee a homogeneous distribution of the load. The chosen architectural framework facilitates a first-come, first-served consumption of the queue, while also minimizing inter-process communication. This design decision reflects the trade-off between operational efficiency and the communication overhead as long as the crash fault tolerance. The results are shown in Table 15 and 13, as long as the graphs in Figure 32 and 30.

Table 13 – BlockWitness Containers Memory Usage Statistics.

Container Service	**Mean (MB)**	**Median (MB)**	**Standard Deviation(MB)**	**Maximum (MB)**	**Minimum (MB)**
Encryption-1	636.02	687.15	189.63	849.49	366.58
Encryption-2	637.83	686.87	189.55	847.20	366.63
Encryption-3	627.90	685.30	183.57	850.37	366.51
Encryption-4	627.90	685.02	183.21	844.03	366.08
Encryption-5	633.15	686.47	185.96	849.13	368.58
Web-1	2300.03	2517.85	925.90	3553.80	172.24

Source: The Authors.

Figure 33 – MongoDB and RabbitMQ CPU usage.



Source: The Authors.

The third group is made up of auxiliary services, which, even though they are not optimized, influence the results in the general use of the BlockWitness system. As seen in the results presented in the Tables 15 and 14; and the graph present in Figures 33 and 34, the two most requested containers during all tests were the Rabbit queuing system and the main MongoDB database container.

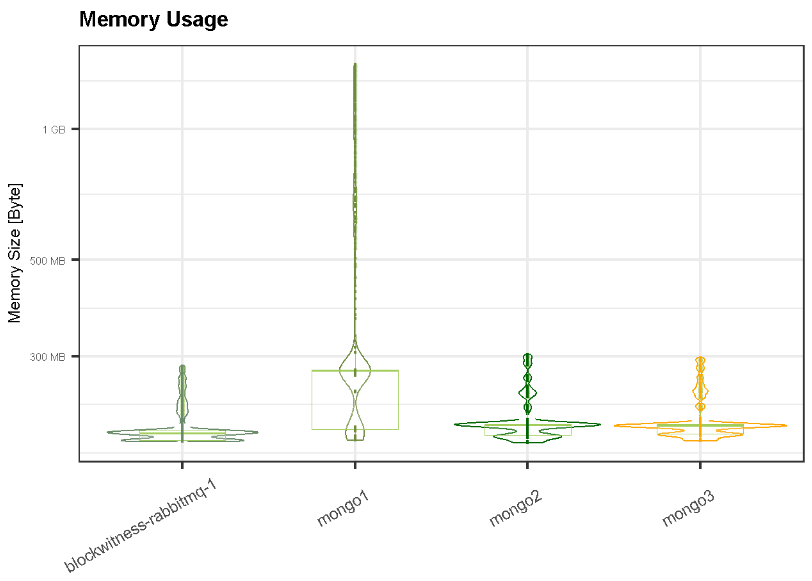
Table 14 – MongoDB and RabbitMQ Memory Usage Statistics.

Container Service	**Mean (MB)**	**Median (MB)**	**Standard Deviation(MB)**	**Maximum (MB)**	**Minimum (MB)**
rabbitmq-1	206,82	199.75	21.49	285.35	190.91
mongo1	376.17	278.34	282.97	1408.90	192.22
mongo2	214.80	208.90	25.93	302.52	189.57
mongo3	214.67	207.85	24.60	297.82	191.91

Source: The Authors.

The auxiliary replicas from MongoDB, added as a security precaution, maintained low memory and CPU usage throughout all tests. The message broker container maintained a behavior consistent with its function, with some spikes in CPU usage when demanded more consistently, but with memory usage as low as expected.

Figure 34 – MongoDB and RabbitMQ memory usage.



Source: The Authors.

Table 15 – CPU Usage Statistics.

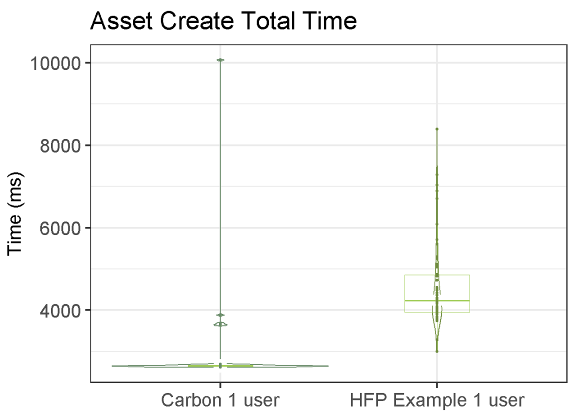
Container Service	Mean (s)	Median (s)	Standard Deviation(s)	Maximum (s)	Minimum (s)
orderer	2.50	2.00	1.92	15.00	1.00
peer0.org1	7.86	6.00	7.28	50.00	1.00
peer0.org2	8.07	6.00	7.27	60.00	1.00
Encryption-1	406.00	306.00	381.38	2796.00	3.00
Encryption-2	424.73	265.50	459.80	3155.00	4.00
Encryption-3	444.92	357.00	404.31	2865.00	4.00
Encryption-4	405.77	278.50	404.22	3296.00	4.00
Encryption-5	400.93	288.50	389.60	2415.00	4.00
Web-1	271.81	171.0	278.99	1794.00	2.00
mongo1	98.04	63.50	98.51	540.00	1.00
mongo2	2.67	2.00	2.13	15.00	0.00
mongo3	2.57	2.00	2.09	14.00	0.00
rabbitmq-1	43.36	29.00	44.79	342.00	1.00

Source: The Authors.

5.4.5 Results Test Carbon21

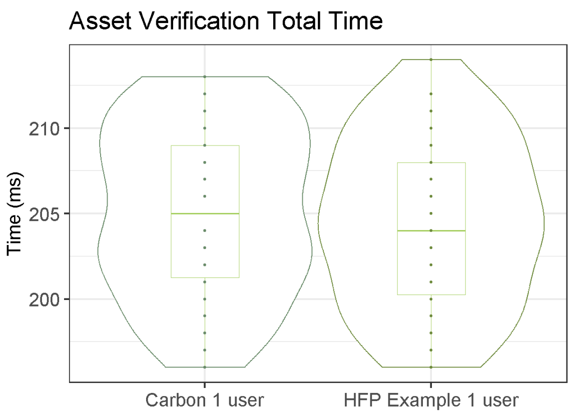
The results related to the use of the Carbon21 system and the blockchain used in these tests were suppressed as they are available in studies carried out in (Correia, 2024). In this way, the same test plan from Subsections 5.4.1, 5.4.2, 5.4.3 were reproduced, i.e, for these scenarios, the focus was to verify the use of the BlockWitness system with Carbon21 assets. To achieve these objectives, homomorphic encryption was applied in the CPF(*Cadastro de Pessoas Físicas* or Natural Persons Register) field, which is defined as having 11 numeric digits. The remainder of the Carbon21 system payload was maintained as per the original definition.

Figure 35 – Comparative between Carbon21 data and HFP Sample data for Asset Create Operation.



Source: The Authors.

Figure 36 – Comparative between Carbon21 data and HFP Sample data for Asset Verify Operation.



Source: The Authors.

Table 16 – Comparative between Carbon 21 data and HFP Sample data for Asset Create Operation.

Scenario	Mean (ms)	Median (ms)	Standard Deviation(ms)	Maximum (ms)	Minimum (ms)
Carbon21 1 User	2884.67	2650.95	1077.86	10071.40	2615.50
HFP Example 1 User	4594.89	4233.95	1090.56	8391.60	3003.70

Source: The Authors.

Table 17 – Comparative between Carbon21 data and HFP Sample data for Asset Verify Operation

Scenario	Mean (ms)	Median (ms)	Standard Deviation(ms)	Maximum (ms)	Minimum (ms)
Carbon21 1 User	204.32	205.00	5.32	227.00	197.00
HFP Example 1 User	204.46	204.00	4.89	214.00	196.00

Source: The Authors.

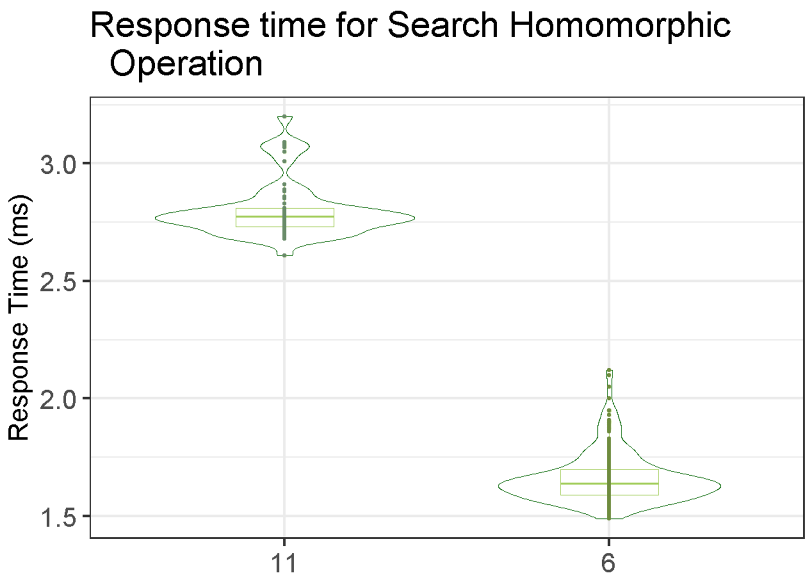
Table 18 – Comparative between Carbon21 data and HFP Sample data for Asset Search Operation.

Search Term Length	Mean (s)	Median (s)	Standard Deviation(s)	Maximum (s)	Minimum (s)
6 (HPC Sample payload)	1.66	1.64	0.11	2.12	1.49
11 (Carbon21 payload)	2.80	2.78	0.12	3.20	2.61

Source: The Authors.

In order to establish a comparable scenario, the supplementary database and authentication layers utilized by the Carbon21 system were suppressed. This adjust-

Figure 37 – Comparative between Carbon21 data and HFP Sample data for Search Term Operation.



Source: The Authors.

ment was made while preserving the payload size and maintaining the coupling with the BlockWitness library. The results found are presented in Tables 16, 17, 18 and graphs in Figures 35, 36, 37.

Surprisingly, the asset creation time showed lower means and medians when compared to the environment using the HFP example. Although the longest time was also in the test with the Carbon21 payload. The behavior observed in the results may be associated with the adoption of a more streamlined and lightweight web system solution, with the aforementioned adoption of Express.js in the Carbon21 system. Despite that, the standard deviation numbers were practically equal, with 1078 ms for Carbon21 and 1091 ms for HFP example. Considering that this test evaluate the total recording time of the asset, there was no isolation of components in the asset creation test and it is known that the only component that varied was the Web service, can be infer that repeating the scenario using the same web service and the same payloads, the response time should show minimal differences. Another indication of this is that in tests with 10 and 20 users of the Subsection 5.4.1, means and medians were very close to the results found with tests the Carbon21 environment.

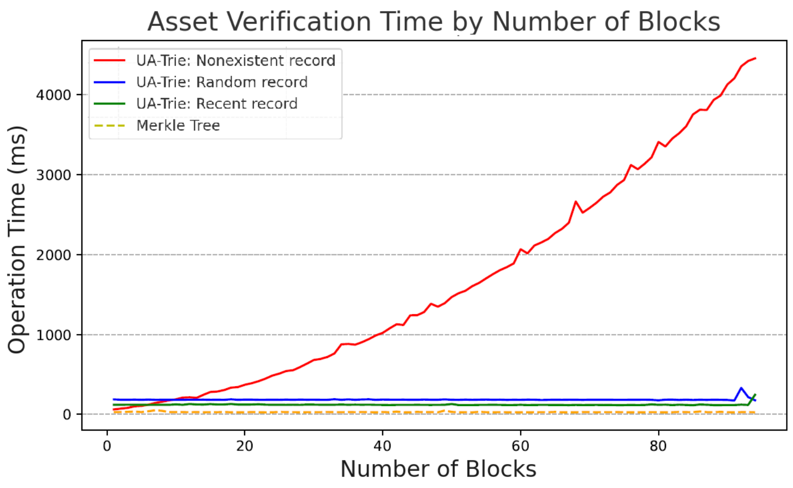
The asset verification test presented very close numbers with a slight advantage in metrics in favor of the load test of the HFP example.

For the homomorphic search tests, the results were as expected, with a slight increase in execution time when compared to the results found in Subsection 5.4.3. This behavior should occur naturally, as a greater load is applied to comparing terms in the test with Carbon payload. However, there is a consistency between the results, with the standard deviation very close to that found in other tests performed previously.

5.5 EXTENDED STUDIES:

As long this study was conducted, two other studies derived from at Santa Catarina State University (UDESC). Both developed in the UDESC CCT department. The first one seeking for a better approach in the verification system, replacing the Merkle Tree with an abstract-trie developed especially for this architecture. The study called Unified Abstract-Trie: An Alternative to Merkle Tree for BlockWitness System was conducted by Arthur José Budal da Silva, an undergraduate student in the Computer Science course (Silva, 2024). The results shown that the Merkle Tree still presents itself as the best solution for the current scenario. The Figure 38 demonstrate the comparison between the two solutions.

Figure 38 – BlockWitness CPU usage.



Source: (Silva, 2024).

As can be seen, the Merkle Tree had the more consistent response time for data verification compared with the devolved new structure. Notwithstanding, in search of new created records the new structure proved to be more efficient (Silva, 2024).

Table 19 – Table of libraries and their functionalities.

Library	Support for arithmetic and boolean functions	Implements bootstrapping	Schemes
MS SEAL	No	No	BGV, BFV, CKKS
HEaan	No	Yes	CKKS
TFHE	Yes	Yes	TFHE
HElib	Yes	Yes	BGV, CKKS
FV/NFLib	No	No	BFV
Palisade	Yes	Yes	BGV, BFV, FHEW, TFHE, CKKS
Concrete	Yes	Yes	TFHE
Lattigo	No	Yes	BFV, CKKS
FHEW	No	Yes	FHEW
OpenFHE	Yes	Yes	BGV, BFV, FHEW, TFHE, CKKS

Source: (Bertelli, 2024).

The second study looked for other homomorphic encryption algorithms. Student Luis Eduardo Bertelli wrote the study called Performance Analysis of Homomorphic Cryptography in BlockWitness (Bertelli, 2024), comparing different schemes and libraries for the proposed encryption operations, the principal founds are show in Table 19.

Table 20 – Performance comparison of homomorphic encryption operations.

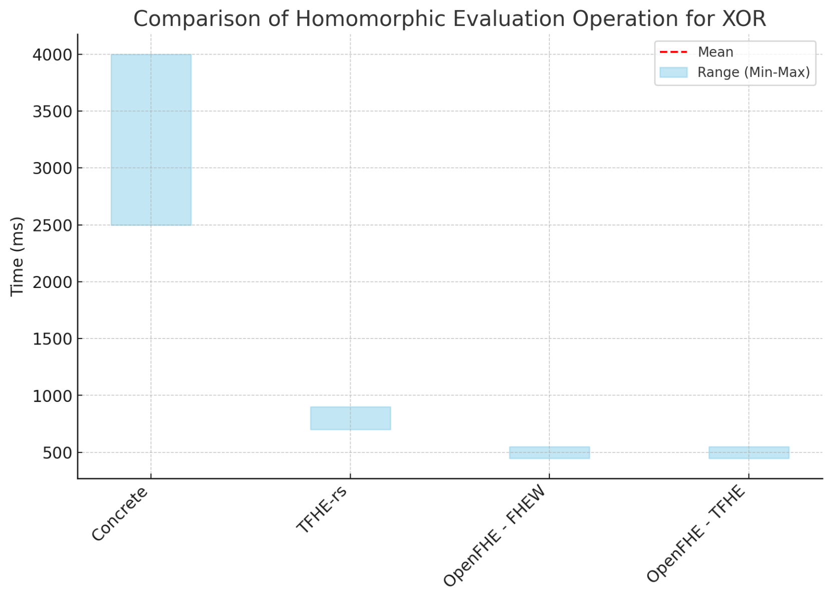
Operation	Concrete	TFHE-rs	OpenFHE - FHEW	OpenFHE - TFHE
Total time (ms)	32233.80	1352.50	6381.00	5269.60
Key return DB	21.80	21.40	20.30	22.00
First Deserialization	11532.79	0.08	3010.70	2490.90
First Circuit Assembly	1114.01	N/A	N/A	N/A
Data Encryption	30.92	2.10	21.90	24.00
Serialization	1.41	0.11	0.11	0.11
Second Deserialization	11970.56	0.85	3049.50	2495.90
Second Circuit Assembly	1170.21	N/A	N/A	N/A
Homomorphic Operation	2924.17	800.07	158.70	146.30
Data Decryption	4.06	0.45	0.01	0.01

Source: (Bertelli, 2024).

Although with the time available it was not possible to carry out all the comparisons, and some library had a better performance in some operations, e.g. XOR operation as show in the Figure 39. Between the compared libraries, the library chosen in this study demonstrated the best average result, considering the multiple operations it supports. The comparison result can be seen in the Table 20.

In conclusion, although there are certain advantages associated with utilized solutions such as OpenFHE (OPEN FHE PROJECT, 2024), the library TFHE remains the optimal choice when considering the architecture design and mode of distribution of the subject system.

Figure 39 – Homomorphic operation comparison between libraries.



Source: (Bertelli, 2024).

For more information, both studies are available at <https://pergamumweb.udesc.br/acervo/164849>.

5.6 SECURITY CONSIDERATIONS:

As a work in constant progress and evolution, the modules tested in this work must be evaluated in terms of security, because when it deals with encryption, new ways of attack are constantly tested. The threat model presented in Section 4.3 can be used as a guide for future researchers implement security control measures and exploit vulnerability of the proposed architecture.

In our proposals no means of attack were used, however it is recommend reading and following several studies that have demonstrated the strengths and weaknesses of the TFHE system. Among the main ones can be mention (Chaturvedi et al., 2022), (Walter, 2022) and specially (ZAMA, 2024a) and the used security estimator used from (Albrecht; Player; Scott, 2015) updated and available at <https://github.com/malb/lattice-estimator>.

5.7 CONSIDERATIONS:

The system proved to be valid, and through the results it is possible to observe that with a planned architecture for a distributed system capable of distributing the computational load between subsystems operating as if they were a single element. This architectural format allows that even with current hardware and its limitations, it may prepare existing Distributed Ledger Technology (DLT) systems to support more advanced cryptographic systems that require greater computational power, as in our example with TFHE.

Furthermore, the experiments carried out in complementary studies demonstrate that the choice of tools and technologies adopted in this project were assertive, but that more in-depth studies must be carried out in different scenarios, before a more comprehensive adoption.

Additionally, the findings from the Carbon21 System were favorable and reinforce the efficacy of the BlockWitness system in ongoing projects that implement DLT. The metrics observed were closely aligned with the results obtained from the tests conducted using the example provided by HFP.

6 CONCLUSION

The adoption of enterprise blockchain technology has significantly influenced the pursuit of confidentiality, compliance, and legal regulations. Most of the permissioned blockchains provide effective solutions that address many of these requirements. However, the audit and verification process can be improved by adding solutions to verify liability, detect fraud, detect misbehaviors, and ensure the integrity and correctness of information without exposing the confidentiality of the data.

The research conducted in these projects, along with data collected from other surveys that motivate this study, demonstrates a growing interest in permissioned blockchain and auditable systems within the academic and business sectors. This underscores the need to expand studies that integrate these concepts.

This project integrates a range of technologies and concepts to develop a novel approach for auditing and verifying assets recorded in Distributed Ledger Technology (DLT), focused mainly on permissioned blockchains. By introducing an additional layer of encryption and verification, it facilitates operations on anonymous data. With the objective of enhance the confidentiality and security of the recorded information while still permitting access to external agents. The technologies chosen to achieve the objective comprised blockchain and homomorphic encryption, supplemented by a tree data structure and a distributed system. To facilitate the integration of these technologies, a web system was developed utilizing a traditional message queuing system and a non-relational database.

Besides the fact that the homomorphic encryption concept is a relatively old technology and is being studied by multiple academic institutions, many challenges must be overcome. Even after 20 years of studies, the technology has recently been adopted on a larger scale, but it is still in the early stages. Open projects such as Zama, used as the basis of study in this research, bring a new breath to the technology, evolving technology and providing practical solutions for daily use. Also, big tech companies recently released their solutions with homomorphic encryptions, e.g., Microsoft Seal, Google HIER, and IBM Guardium Data Encryptions, which provide easier access and visibility to the broad public.

It's hard to tell the reasons for this behavior; adopting new technology depends on many factors. Most studies demonstrate that complex concepts are more challenging to accept widely(Skare; Riberio Soriano, 2021). Cryptographic encryptions are a complex field of research, and homomorphism is even more complicated. The bibliography research showed fewer research groups are interested in amplifying and deeper security homomorphic encryption. Ideally, everyone who shares information in a distributed system should worry about free access to the unknown third party to their information. According to the initial bibliographic research, financial and medical researchers are the precursors investing more resources and time in this encryption format.

The integration of new technologies with established theoretical frameworks offers a fresh perspective on addressing the challenges posed by emerging blockchain use cases. BlockWitness system architectural design allowed the non-intrusive integration of homomorphic encryption with blockchain, resulting in minimal overhead during the asset creation process within the target system. When comparing the creation process time for one user without encryption (raw data) to that with homomorphic encryption, was noted a 22.04% increase in the median response time. However, for the current scenario, this results in an increase of only 598 ms in the end user's response time. Furthermore, when examining the minimum values of the operation, was found a difference of just 1.55%, or 40 ms.

Additionally, the audit and verification system was developed in a decoupled manner, proving to be effective in facilitating searches and verifying encrypted assets without the need for decryption. The distributed model implemented in the cryptography module of the BlockWitness system showed consistent results among workers, with an average memory usage variation of 1.56% and an average CPU usage variation of 9.89%. Nevertheless, When evaluating the results of the cryptographic search module system in comparison to other systems, it becomes evident that this particular system necessitates substantially greater computational resources. In the scenario tested, it consumed 55 times more CPU resources than the blockchain system.

This research's results reinforced that computer resource costs are one of the principal challenges to ensuring that more researchers and developers embrace homomorphic technology. Homomorphic operations have a computational cost not yet covered by commonly used hardware. Blockchain technology also experienced the same challenge at its beginning and had to be adapted to get more and more spread. Many blockchains have changed how consensus occurs; the most widely publicized, known as "Merge" from Ethereum in 2022, has altered how computation resources were used to execute blockchain consensus, changing the consensus model from proof-of-work to proof-of-stake.

The main challenges encountered in this project may be common to other researchers, factors such as difficulty in finding updated documentation, sudden changes in the libraries used, scarce theoretical foundation and lack of replicable results. Even considering that the Zama project is an extensive source of information and has a good number of published articles, the Fast Fully Homomorphic Encryption over the Torus with Rust (TFHE-rs) library, used in this study, lacked documentation and portability.

This research simplifies new users adoption of homomorphic technology. What sets this project apart is its ability to integrate and use homomorphism encryption in environments that were not originally configured for this type of technology. This is particularly useful in scenarios with infrastructure constraints or limited resources, such as Internet of Things (IoT) devices, legacy systems, or low-cost solutions.

The system's try an innovative approach involves transferring the complex computational load, which would traditionally be carried out centralized and locally, to a complementary service. This service operates as an partitioned backend, handling the heavy homomorphism calculations while the end-user environment remains in a separated functional environment.

6.1 FUTURE WORK

Despite its benefits, the system has several areas that could be enhanced for improvement.

- Optimizing User-Side Operations: An in-depth and advanced study can be carried out to better balance the computational load used in the Web service.
- System overheads: Considering the BlockWitness system as a complementary system, the target system overhead is minimized, but in any case, the amount of additional resources that must be allocated to each application must be considered. A greater number of real applications must be tested to expand the studies carried out.
- Security and reliability: The library adopts Fast Fully Homomorphic Encryption over the Torus (TFHE) and has shown satisfactory results. However, as with all encryption systems, more attack formats must be explored. The same applies to the BlockWitness system, which, even though it adopts an advanced verification system, still has unique verification points.

Thus, some opportunities for future work were identified during the execution of these studies. Expansion and improvements to the BlockWitness system:

- Carry out tests with other real-world systems besides the Carbon21 system.
- Conduct homomorphic search using additional schemas, including Boolean and integer operations.
- Improve the asset creation system by implementing a more robust creation interface.

- Compare other homomorphic search algorithms with TFHE.
- Develop new auditing functionalities specifically tailored to the case studies outlined in Section 6.3.
- Develop and implement a homomorphic encryption module that utilizes hardware with superior capabilities compared to existing solutions and conduct an analysis of Graphics Processing Unit (GPU) performance.

Incorporate BlockWitness functionalities into the DLT system:

- Add encryption system directly to target DLT systems.
- Perform homomorphic operations in smart contracts and chaincodes.
- Apply the concepts of the BlockWitness system to other DLT formats besides blockchain, e.g. Directed Acyclic Graph (DAG) and Holochain.

6.2 IDENTIFIED BENEFITS AND DIFFERENCES OF THIS SYSTEM

- Facilitated Adoption: Users with little experience in advanced technologies can quickly integrate their solutions without needing in-depth technical knowledge.
- Light Infrastructure: There is no need for heavy updates or adaptations to existing devices or systems. It is ideal for operations on low-capacity devices or limited connectivity networks.
- Compatibility with Existing Solutions: It works in harmony with already-developed systems, eliminating the need for rewriting or significant modifications to the base code.
- Scalability and Sustainability: It allows for expanding use without affecting local performance, making the system sustainable in the long term.

6.3 SUGGEST APPLICATION SCENARIOS

This study delineates several promising application scenarios in which the proposed system is positioned to deliver substantial value:

- Financial Sector: Analysis of encrypted data in legacy banking systems.
- Healthcare: Processing sensitive medical data on portable medical devices.
- IoT: Use in sensor devices with limited processing capacity.

- Regulated systems: Systems that require external verification by regulatory bodies.

6.4 FINAL CONSIDERATIONS

This project implements homomorphism in a simple manner, allowing audit systems to be possible in DLT systems without compromising data confidentiality, as well as overcoming a technological barrier and transforming into a practical and accessible tool, promoting technological inclusion and operational efficiency. The developed Proof of Concept (PoC) points to a future where advanced technologies, such as homomorphic encryption, can be democratized and used on a large scale, even in initially harsh environments.

BIBLIOGRAPHY

ABOU JAOUDE, J.; GEORGE SAADE, R. Blockchain Applications – Usage in Different Domains. **IEEE Access**, v. 7, p. 45360–45381, 2019. ISSN 2169-3536. Available on: <https://ieeexplore.ieee.org/document/8656511/>.

AHMAD, A.; SAAD, M.; MOHAISEN, A. Secure and transparent audit logs with BlockAudit. **Journal of Network and Computer Applications**, v. 145, p. 102406, nov 2019. ISSN 10848045. Available on: <https://linkinghub.elsevier.com/retrieve/pii/S1084804519302401>.

ALBRECHT, M. R.; PLAYER, R.; SCOTT, S. **On the concrete hardness of Learning with Errors**. 2015. Cryptology ePrint Archive, Paper 2015/046. Available on: <https://eprint.iacr.org/2015/046>.

ALMEIDA, A. G. d. O. P. **O impacto da Blockchain em Auditoria Externa**. Tese (Dissertação de Mestrado) — Universidade do Porto, 2022.

APACHE. **Couch DB**. 2024. Available on: <https://couchdb.apache.org/b>. Accessed: 12.2.2024.

BARGER, A. et al. **A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric**. 2021. Available on: <https://arxiv.org/abs/2107.06922>.

BEHNIA, R. et al. Lattice-Based Proof-of-Work for Post-Quantum Blockchains. In: . [s.n.], 2022. p. 310–318. Available on: <https://link.springer.com/10.1007/978-3-030-93944-1_21>.

BERNSTEIN, D. J.; LANGE, T. Post-quantum cryptography. **Nature**, v. 549, n. 7671, p. 188–194, sep 2017. ISSN 0028-0836. Available on: <https://www.nature.com/articles/nature23461>.

BERTELLI, L. E. **Performance Analysis of Homomorphic Cryptography in Block-Witness**. 2024.

BOSAMIA, M.; PATEL, D. Current Trends and Future Implementation Possibilities of the Merkel Tree. **International Journal of Computer Sciences and Engineering**, v. 6, n. 8, p. 294–301, aug 2018. ISSN 23472693. Available on: <http://www.ijcseonline.org/full_paper_view.php?paper_id=2691>.

BRITANNICA, THE EDITORS OF ENCYCLOPAEDIA. **"homomorphism"**. Encyclopedia Britannica, 2024. Available on: <https://www.britannica.com/science/homomorphism>. Accessed: 14.1.2024.

BUTERIN, V. A next-generation smart contract and decentralized application platform. **Etherum**, n. January, p. 1–36, 2014. ISSN 19233299. Available on: <http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf>.

CAMPBELL, R. Transitioning to a Hyperledger Fabric Quantum-Resistant Classical Hybrid Public Key Infrastructure. **The Journal of the British Blockchain Association**, v. 2, n. 2, p. 1–11, 2019. ISSN 25163949.

CANDITT, S.; GUNTER, M. Aspect oriented logging in a real-world system. **Coady et al.[268]**, 2002.

CAPOCASALE, V.; GOTTA, D.; PERBOLI, G. Comparative analysis of permissioned blockchain frameworks for industrial applications. **Blockchain: Research and Applications**, v. 4, n. 1, p. 100113, mar 2023. ISSN 20967209. Available on: <https://linkinghub.elsevier.com/retrieve/pii/S2096720922000549>.

CHATURVEDI, B. et al. **A Practical Full Key Recovery Attack on TFHE and FHEW by Inducing Decryption Errors**. 2022. Cryptology ePrint Archive, Paper 2022/1563. Available on: <https://eprint.iacr.org/2022/1563>.

CHILLOTTI, I. **TFHE Deep Dive - Part IV - Programmable Bootstrapping**. 2022. Available on: <https://www.zama.ai/post/tfhe-deep-dive-part-4>.

CHILLOTTI, I. et al. TFHE: Fast fully homomorphic encryption over the torus. **Journal of Cryptology**, Springer, v. 33, n. 1, p. 34–91, 2020.

CORREIA, P. H. B. **Carbono 21: Promovendo florestamento utilizando tokenização**. Dissertação (Mestrado) — University of São Paulo, 2024.

DESCIO-TRINETO, R. et al. DCANon: Towards distributed certification authority (CA) with non-fungible token (NFT). In: BAROLLI, L. (Ed.). **Advanced Information Networking and Applications**. Cham: Springer International Publishing, 2023. p. 286–298. ISBN 978-3-031-29056-5.

DHILLON, V.; METCALF, D.; HOOPER, M. The Hyperledger Project. In: **Blockchain Enabled Applications**. Berkeley, CA: Apress, 2017. p. 139–149. Available on: <http://link.springer.com/10.1007/978-1-4842-3081-7_10>.

DIMITOGLOU, G.; JIM, C. Performance Evaluation of Partially Homomorphic Encryption Algorithms. In: **2022 International Conference on Computational Science and Computational Intelligence (CSCI)**. IEEE, 2022. p. 910–915. ISBN 979-8-3503-2028-2. Available on: <https://ieeexplore.ieee.org/document/10216406/>.

DOCKER. **Docker Compose**. 2024. Available on: <https://docs.docker.com/compose/>. Accessed: 12.2.2024.

DOCKER. **Resource constraints**. 2024. Available on: <https://docs.docker.com/engine/containers/resource_constraints/>. Accessed: 12.12.2024.

DOCKER. **Use containers to Build, Share and Run your applications**. 2024. Available on: <https://www.docker.com/resources/what-container>. Accessed: 12.2.2024.

EJS. **Embedded JavaScript templating.** 2024. Available on: <https://ejs.co/>. Accessed: 10.12.2024.

EXPRESS. **Fast, unopinionated, minimalist web framework for Node.js**. 2024. Available on: <https://expressjs.com/>. Accessed: 10.12.2024.

FAN, G. et al. Towards Faster Fully Homomorphic Encryption Implementation with Integer and Floating-point Computing Power of GPUs. In: **2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)**. IEEE, 2023. p. 798–808. ISBN 979-8-3503-3766-2. Available on: <https://ieeexplore.ieee.org/document/10177431/>.

FEYNMAN, R. P. Simulating physics with computers. **International Journal of Theoretical Physics**, v. 21, n. 6-7, p. 467–488, jun 1982. ISSN 0020-7748. Available on: <http://link.springer.com/10.1007/BF02650179>.

FU, G.; ZHANG, Y.; YU, G. A Fair Comparison of Message Queuing Systems. **IEEE Access**, v. 9, p. 421–432, 2021. ISSN 2169-3536. Available on: <https://ieeexplore.ieee.org/document/9303425/>.

GAID, M. L.; SALLOUM, S. A. Homomorphic Encryption. In: . [s.n.], 2021. p. 634–642. Available on: <https://link.springer.com/10.1007/978-3-030-76346-6_56>.

GOOGLE. **HEIR: Homomorphic Encryption Intermediate Representation**. 2024. Available on: <https://heir.dev/>. Accessed: 10.11.2024.

GOOGLE. **Level DB**. 2024. Available on: <https://github.com/google/leveldb>. Accessed: 12.2.2024.

GRAFANA. **Using k6 Browser**. 2024. Available on: <https://grafana.com/docs/k6/v0.54.x/using-k6-browser/>. Accessed: 12.2.2024.

GRAFANA K6. **From metrics to insight**. 2024. Available on: <https://k6.io/>. Accessed: 12.2.2024.

GU, S. et al. Logging Practices in Software Engineering: A Systematic Mapping Study. **IEEE Transactions on Software Engineering**, v. 49, n. 2, p. 902–923, feb 2023. ISSN 0098-5589. Available on: <https://ieeexplore.ieee.org/document/9756253/>.

GUPTA, S. **Pro Apache Log4j**. A-Press, 2005. ISBN 978-1-59059-499-5. Available on: <http://link.springer.com/10.1007/978-1-4302-0034-5>.

HOVD, M. N. **The Handling of Noise and Security of Two Fully Homomorphic Encryption Schemes**. Dissertação (Mestrado) — Norwegian University of Science and Technology, 2017. Available on: <http://hdl.handle.net/11250/2453097>.

HYPERLEDGER FABRIC. **Hyperledger Fabric - Channels**. 2023. Available on: <https://hyperledger-fabric.readthedocs.io/en/latest/channels.html>. Accessed: 12.2.2023.

HYPERLEDGER FABRIC. **Hyperledger Fabric - Introduction**. 2023. Available on: <https://hyperledger-fabric.readthedocs.io/en/release-2.5/whatis.html>. Accessed: 12.2.2023.

HYPERLEDGER FABRIC. **Hyperledger Fabric - Policies**. 2023. Available on: <https://hyperledger-fabric.readthedocs.io/en/release-2.5/policies.html>. Accessed: 12.2.2023.

HYPERLEDGER FABRIC. **Hyperledger Fabric - Private Data**. 2023. Available on: <https://hyperledger-fabric.readthedocs.io/en/release-2.5/private-data/private-data.html>. Accessed: 12.2.2023.

HYPERLEDGER FABRIC. **HyperLedger Fabric**. 2024. Available on: <https://hyperledger-fabric.readthedocs.io/en/release-2.5/>. Accessed: 12.2.2024.

HYPERLEDGER FABRIC. **What's New in Hyperledger Fabric v3.0**. 2024. Available on: <https://hyperledger-fabric.readthedocs.io/en/latest/whatsnew.html>. Accessed: 12.2.2025.

HYPERLEDGER FOUNDATION. **Fabric Contract APIs and Application APIs**. 2023. Available on: <https://hyperledger-fabric.readthedocs.io/en/release-2.5/sdk_chaincode.html#fabric-application-apis>. Accessed: 4.1.2024.

HYPERLEDGER FOUNDATION. **Fabric Gateway v1.4**. 2023. Available on: <https://hyperledger.github.io/fabric-gateway/>. Accessed: 4.1.2024.

IBM. **Guardium Data Encryption**. 2024. Available on: <https://www.ibm.com/products/guardium-data-encryption>. Accessed: 10.11.2024.

IBM QUANTUM PLATFORM. **IBM Quantum**. 2023. Available on: <https://quantum.ibm.com/>. Accessed: 4.1.2024.

JIA, B. et al. Blockchain-Enabled Federated Learning Data Protection Aggregation Scheme With Differential Privacy and Homomorphic Encryption in IIoT. **IEEE Transactions on Industrial Informatics**, v. 18, n. 6, p. 4049–4058, jun 2022. ISSN 1551-3203. Available on: <https://ieeexplore.ieee.org/document/9448383/>.

KRAUS, J. L.; PLATKUS, W. Incorporating continuous improvement principles into EMS auditing strategies. **Environmental Quality Management**, v. 16, n. 4, p. 7–12, jun 2007. ISSN 1088-1913. Available on: <https://onlinelibrary.wiley.com/doi/10.1002/tqem.20138>.

LEMIEUX, V. **Blockchain for Recordkeeping: Help or Hype?** [S.l.], 2016. Volume 1 p. Available on: <https://www.researchgate.net/publication/309414276>.

LI, C.-Y. et al. A New Lattice-Based Signature Scheme in Post-Quantum Blockchain Network. **IEEE Access**, v. 7, p. 2026–2033, 2019. ISSN 2169-3536. Available on: <https://ieeexplore.ieee.org/document/8579537/>.

LIANG, W. et al. Circuit Copyright Blockchain: Blockchain-Based Homomorphic Encryption for IP Circuit Protection. **IEEE Transactions on Emerging Topics in Computing**, v. 9, n. 3, p. 1410–1420, jul 2021. ISSN 2168-6750. Available on: <https://ieeexplore.ieee.org/document/9091234/>.

LIU, X.; FARAHANI, B.; FIROUZI, F. Distributed Ledger Technology. In: **Intelligent Internet of Things**. Cham: Springer International Publishing, 2020. p. 393–431. Available on: <http://link.springer.com/10.1007/978-3-030-30367-9_8>.

LUSARD, A. et al. **An Overview of Hyperledger Foundation**. [S.l.], 2021.

MICCIANCIO, D. **Lecture notes in Lattice Algorithms and Applications**. University of California San Diego, 2012. Available on: <https://cseweb.ucsd.edu/classes/wi12/cse206A-a/lec1.pdf>.

MICROSOFT. **Microsoft SEAL**. 2024. Available on: <https://www.microsoft.com/en-us/research/project/microsoft-seal/>. Accessed: 10.11.2024.

MICROSOFT LIQUI PLATFORM. **Language-Integrated Quantum Operations: LIQUi**. 2016. Available on: <https://www.microsoft.com/en-us/research/project/language-integrated-quantum-operations-liqui/>. Accessed: 4.1.2024.

NAKAMOTO, S. Bitcoin : A peer-to-peer electronic cash system. In: . [s.n.], 2008. Available on: <https://bitcoin.org/bitcoin.pdf>.

NEW RELIC. **2022 State of Logs**. 2022. Available on: <https://newrelic.com/sites/default/files/2022-10/new_relic_2022_state_of_logs_white_paper.pdf>. Accessed: 10.12.2024.

NEXT.JS. **Caching in Next.js**. 2024. Available on: <https://nextjs.org/docs/app/building-your-application/caching>. Accessed: 12.12.2024.

NEXT.JS. **Memory Usage**. 2024. Available on: <https://nextjs.org/docs/app/building-your-application/optimizing/memory-usage>. Accessed: 12.2.2024.

NIST. **Post-Quantum Cryptography**. 2024. Available on: <https://csrc.nist.gov/projects/post-quantum-cryptography>. Accessed: 15.2.2024.

NIST. **Privacy-Enhancing Cryptography**. 2024. Available on: <https://csrc.nist.gov/projects/pec>. Accessed: 25.11.2024.

OPEN FHE PROJECT. **Open FHE**. 2024. Available on: <https://www.openfhe.org/>. Accessed: 12.2.2024.

OXFORD LEARNER'S DICTIONARIES. **Audit noun**. 2024. Available on: <https://www.oxfordlearnersdictionaries.com/definition/english/audit_1>. Accessed: 10.2.2024.

PAWAR, A. et al. BlockAudit 2.0: PoA blockchain based solution for secure Audit logs. In: **2021 5th International Conference on Information Systems and Computer Networks (ISCON)**. IEEE, 2021. p. 1–6. ISBN 978-1-6654-0341-2. Available on: <https://ieeexplore.ieee.org/document/9702378/>.

POLGE, J.; ROBERT, J.; Le Traon, Y. Permissioned blockchain frameworks in the industry: A comparison. **ICT Express**, v. 7, n. 2, p. 229–233, jun 2021. ISSN 24059595. Available on: <https://linkinghub.elsevier.com/retrieve/pii/S2405959520301909>.

PROENçA, D. et al. Longevity as an information systems design concern. **CEUR Workshop Proceedings**, v. 998, p. 73–80, 01 2013.

PROMETHEUS. **From metrics to insight**. 2024. Available on: <https://prometheus.io/>. Accessed: 12.2.2024.

RAJ, R.; Kurt Peker, Y.; MUTLU, Z. D. Blockchain and Homomorphic Encryption for Data Security and Statistical Privacy. **Electronics**, v. 13, n. 15, p. 3050, aug 2024. ISSN 2079-9292. Available on: <https://www.mdpi.com/2079-9292/13/15/3050>.

REGEV, O. Lect 1 Introduction - Lattices in Computer Science - Lecture notes. **Tel Aviv University Lattices in Computer Science**, v. 2, n. c, p. 2015, 2004. Available on: <https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/introduction.pdf>.

REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. In: **Proceedings of the thirty-seventh annual ACM symposium on Theory of computing**. New York, NY, USA: ACM, 2005. p. 84–93. ISBN 1581139608. Available on: <https://dl.acm.org/doi/10.1145/1060590.1060603>.

ROETTELER, M. et al. Quantum resource estimates for computing elliptic curve discrete logarithms. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 10625 LNCS, p. 241–270, 2017. ISSN 16113349. Available on: <https://arxiv.org/abs/1706.06752v3>.

RUST LANG. **Struct Instant**. 2024. Available on: <https://doc.rust-lang.org/std/time/struct.Instant.html>. Accessed: 12.2.2024.

SAINI, M. S. **Comparative Analysis of Top 5, 2-Factor Authentication**. 2021. Available on: <https://era.library.ualberta.ca/items/3e7dba43-fa60-400c-b1f3-fa4279ccea44/view/0ef696f2-5c00-42ba-905b-6b1f310b2be8/Saini_M.pdf>.

SHAMOO, A. E. Data Audit as a Way to Prevent/Contain Misconduct. **Accountability in Research**, v. 20, n. 5-6, p. 369–379, sep 2013. ISSN 0898-9621. Available on: <https://www.tandfonline.com/doi/full/10.1080/08989621.2013.822259>.

SHOR, P. Algorithms for quantum computation: discrete logarithms and factoring. In: **Proceedings 35th Annual Symposium on Foundations of Computer Science**. IEEE Comput. Soc. Press, 1994. p. 124–134. ISBN 0-8186-6580-7. Available on: <http://ieeexplore.ieee.org/document/365700/>.

SILVA, A. J. B. da. **Unified Abstract-Trie: An Alternative to Merkle Tree for Block-Witness System**. 2024.

SINGH, P. et al. Blockchain and homomorphic encryption-based privacy-preserving data aggregation model in smart grid. **Computers & Electrical Engineering**, v. 93, p. 107209, jul 2021. ISSN 00457906. Available on: <https://linkinghub.elsevier.com/retrieve/pii/S0045790621002032>.

SKARE, M.; Riberio Soriano, D. How globalization is changing digital technology adoption: An international perspective. **Journal of Innovation & Knowledge**, v. 6, n. 4, p. 222–233, oct 2021. ISSN 2444569X. Available on: <https://linkinghub.elsevier.com/retrieve/pii/S2444569X21000202>.

SUNNY, F. A. et al. A Systematic Review of Blockchain Applications. **IEEE Access**, v. 10, p. 59155–59177, 2022. ISSN 2169-3536. Available on: <https://ieeexplore.ieee.org/document/9786734/>.

SUNYAEV, A. **Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies**. Springer International Publishing, 2020. 1–413 p. ISBN 9783030349578. Available on: <https://link.springer.com/deleted>.

SYED, N. F. et al. Zero Trust Architecture (ZTA): A Comprehensive Survey. **IEEE Access**, v. 10, p. 57143–57179, 2022. ISSN 2169-3536. Available on: <https://ieeexplore.ieee.org/document/9773102/>.

TFHE. **About TFHE**. 2023. Available on: <https://www.tfhe.com/about>. Accessed: 08.2.2024.

THE CHROMIUM PROJECT. **Chromium**. 2024. Available on: <https://www.chromium.org/Home/>. Accessed: 12.2.2024.

UNIVERSITY OF BRISTOL. **Quantum Technologies Innovation Centre**. 2021. Available on: <https://www.bristol.ac.uk/temple-quarter-campus/research-teaching-and-partnerships/qtic/>. Accessed: 4.1.2024.

W3NOW. **State of Blockchain Adoption in German Economy**. 2024. Available on: <https://www.w3now.de/wp-content/uploads/2024/03/W3NOW-Mini-Report-Englisch.pdf>. Accessed: 12.12.2024.

WALTER, M. **On Side-Channel and CVO Attacks against TFHE and FHEW**. 2022. Cryptology ePrint Archive, Paper 2022/1722. Available on: <https://eprint.iacr.org/2022/1722>.

YAJI, S.; BANGERA, K.; NEELIMA, B. Privacy preserving in blockchain based on partial homomorphic encryption system for ai applications. In: **2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW)**. [S.l.: s.n.], 2018. p. 81–85.

YAN, B. et al. Factoring integers with sublinear resources on a superconducting quantum processor. 2022. Available on: <http://arxiv.org/abs/2212.12372>.

YAN, X.; WU, Q.; SUN, Y. A Homomorphic Encryption and Privacy Protection Method Based on Blockchain and Edge Computing. **Wireless Communications and Mobile Computing**, v. 2020, p. 1–9, aug 2020. ISSN 1530-8669. Available on: <https://www.hindawi.com/journals/wcmc/2020/8832341/>.

YUAN, B.; WU, F.; ZHENG, Z. Post quantum blockchain architecture for internet of things over NTRU lattice. **PLOS ONE**, v. 18, n. 2, p. e0279429, feb 2023. ISSN 1932-6203. Available on: <https://dx.plos.org/10.1371/journal.pone.0279429>.

ZAMA. **Regular Expression Engine with TFHE-rs**. 2023. Available on: <https://www.zama.ai/post/regex-engine-tfhe-rs>. Accessed: 12.2.2024.

ZAMA. **Zama**. 2023. Zama is a cryptography company building open-source homomorphic encryption tools for developers. Available on: <https://github.com/zama-ai>. Accessed: 4.1.2024.

ZAMA. **Security and cryptography**. 2024. Available on: <https://docs.zama.ai/tfhe-rs/get-started/security_and_cryptography>. Accessed: 10.11.2024.

ZAMA. **Zama - TFHE-rs**. 2024. Available on: <https://github.com/zama-ai/tfhe-rs>. Accessed: 16.2.2024.

ZENG, L. et al. Computer operating system logging and security issues: a survey. **Security and Communication Networks**, v. 9, n. 17, p. 4804–4821, nov 2016. ISSN 1939-0114. Available on: <https://onlinelibrary.wiley.com/doi/10.1002/sec.1677>.

ZHANG, X. et al. Post-Quantum Blockchain over Lattice. **Computers, Materials & Continua**, v. 63, n. 2, p. 845–859, 2020. ISSN 1546-2226. Available on: <http://www.techscience.com/cmc/v63n2/38547>.