Integrated Digital Wallet

Master's degree in Computer Engineering – Mobile Computing

Tiago dos Santos Simões

Leiria, September 2024

Integrated Digital Wallet

Master's degree in Computer Engineering – Mobile Computing

Tiago dos Santos Simões

Internship Report under the supervision of Doctor Anabela Moreira Bernardino, professor at the School of Technology and Management of the Polytechnic Institute of Leiria.

Leiria, September 2024

Originality and Copyright

This internship report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Acknowledgements

First, I would like to thank all my teachers and colleagues who contributed to my personal growth and development as a computer engineer.

I also want to thank WIT Software and my supervisor Sérgio Dinis for the opportunity to work on this project, my technical tutors David Salvador and Bruno Santos, and my business analyst Ana Fernandes for the insights and help provided during the internship.

I am thankful to my internship supervisor Anabela Bernardino for her help, support, and understanding of the delay in the report submission.

Finally, I want to thank my family, girlfriend, and friends for their constant concern and support throughout this academic journey, particularly during this internship period and its report.

Abstract

Recently, blockchains and cryptocurrencies have gained more market share and have attracted a great deal of attention among the worldwide community. Banks, stock exchanges, financial institutions, and large companies are increasingly investing in this extremely lucrative digital market, which every day creates solutions, innovative products, and market infrastructures. It is impossible to interact with this universe without a digital wallet, which stores and transacts each user's assets.

WIT is no different and to assist its entry into this market, it has created this internship opportunity to develop a prototype for a digital wallet that allows operations to be performed on a blockchain while keeping the user's portfolio of digital assets. This implementation had to guarantee proper security and offer a user experience targeted at mainstream users, with the ultimate goal of being later integrated into any specific solution implemented under a blockchain.

This internship began with an extensive first analysis phase which helped to conclude that the real value of a wallet is the way it stores and manages its private keys. This conclusion meant a change of focus and led to a second phase of in-depth analysis to understand the most suitable and secure method for managing keys.

This report describes all the work conducted for ten months at WIT Software, which involves a considerable period of state-of-the-art research into digital wallets and then the design and implementation of the wallet prototype that integrates a Multi-Party Computation method, which has been identified as the most effective way of managing its private keys.

Keywords: digital wallet, key management system, multi-party computation

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

2FA	Two-Factor Authentication
API	Application Programming Interface
AWS	Amazon Web Services
DApps	Decentralised Applications
DB	Database
DeFi	Decentralised Finances
DEX	Decentralised Exchanges
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ERC	Ethereum Request for Comments
ETH	Ethereum
HD	Hierarchical Deterministic
HSM	Hardware Security Module
IDE	Integrated Development Environment
IP	Internet Protocol
IPTV	Internet Protocol Television
JS	JavaScript
KMS	Key Management System
MPC	Multi-Party Computation
NFT	Non-fungible Token
PIN	Personal Identity Number
PoS	Proof of Stake
PoW	Proof of Work
REST	Representational State Transfer
RLP	Recursive Length Prefix
SPA	Single Page Application
TEE	Trusted Execution Environment
UI	User Interface
USB	Universal Serial Bus
UX	User Experience
WEB	World Wide Web

1. Introduction

The following report describes the work developed along with the curricular unit Internship of the master's in computer engineering – Mobile Computing, lectured by the School of Technology and Management of the Polytechnic of Leiria.

The report goal consists of describing the work developed during the WIT Software internship from September 2021 to July 2022.

The first section (section 1.1) describes the goals and motivation of the internship, the second section (section 1.2) details the internship entity and the third section (section 1.3) represents the organisation of this document.

1.1. Goals and Motivation

The world of blockchains and cryptocurrencies has seen huge growth in recent years and these technologies are now permanent and fundamental instruments in the financial market. This popularisation is due to performing these cryptocurrency transactions without having to trust third parties and in a secure way since they are registered on a blockchain where everyone can see them, but no one can change them. As a result, banks, stock exchanges, financial institutions, and large companies are increasingly positioning themselves and preparing to take part in this extremely lucrative digital market, which every day creates solutions, innovative products, and market infrastructures [1].

Knowing the importance of such technologies, we also need to understand how we can interact with them. And this is where digital wallets (or crypto wallets) come in. A digital wallet allows you to store, send, and receive cryptocurrencies. More specifically, they consist of software that stores the public and private keys that enable you to manage your digital currency holdings and interact with blockchain networks. Each wallet can support multiple blockchains, functionalities, security systems, and levels of complexity [2].

Although this may seem like a beneficial thing, the truth is that as well as finding a wallet that completely satisfies our needs, will also lead to difficulties for beginners in the world of

blockchains and the consequent necessity to obtain further information before designing a wallet, especially in a universe where many of the decisions made cannot be reversed.

To answer this, and for WIT to expand in this market, this internship was designed, to consist of prototyping a digital wallet that allows operations to be performed on a blockchain while keeping the user's portfolio of digital assets. The implementation of this prototype should be guided by two fundamental objectives: to guarantee maximum security and to offer the best user experience so that mainstream users can use the wallet. This wallet could later be integrated into any specific solution implemented under a blockchain.

To propose a solution, a state-of-the-art study was conducted of the different digital wallets on the market. Following an analysis of the most influential ones, such as MetaMask, Trust Wallet, Exodus, and Binance Wallet, the key characteristics of each one were identified, both those they had in common and those in which they differed the most.

Once all this analysis had been completed, an initial solution was proposed. It consisted of implementing an API (Application Programming Interface) that could later be integrated into any blockchain application. The interest of WIT was in implementing a custodial wallet, where the owner of the wallet (in this case, WIT) is responsible for managing the private keys, thus removing that responsibility from the user. Taking this into account, it was proposed the use of AWS (Amazon Web Services) services for key management. That was suggested because it was concluded that it is the most important and most complex component of all custodial wallets, being the component that differs the most between them and which leads some to be more reliable and trustworthy than others. By using this service, security in key management was immediately guaranteed and allowed me to focus on implementing the other features.

As one of the aims of this wallet is to later be able to integrate with various blockchain solutions, as well as presenting this proposal to my supervisor and internship tutors, it was also important to discuss it with members from other WIT projects to brainstorm some ideas and find out their opinions on my proposal. At this meeting, it was concluded that we needed to have a clear definition and seek out a solution to manage private keys without relying on external services, as this was precisely the component that could bring the most value to our custodial wallet. There was then a need to redefine what was the main focus of the internship and the project, giving priority to the private key management part to the detriment of other

functionalities. More precisely, this part of key management (as the name suggests) covers all the key management in a cryptographic system, handling its creation, exchange, use, and storage in a secure environment.

After redefining the scope of the project, existing solutions for private key management were analysed and a new solution was proposed. Having reached an agreement on this second phase of the project, several weeks of planning and implementation of the wallet prototype followed, accompanied by my supervisors through the meetings held.

1.2. Host Institution

WIT Software is a Portuguese software development company specialising in the telecom industry and has as clients some of the most reputed companies in the world like Vodafone, Safaricom, Swisscom, Jio, T-Mobile, and many others. WIT has software deployed in 40+ countries and has strong expertise in voice-video messaging over IP (Internet Protocol), mobile money, mobile commerce, IPTV (Internet Protocol Television), secure software, and core telecom services [3].

The company was founded in Coimbra in 2001 as a spin-off of the Instituto Pedro Nunes and from the University of Coimbra. Its headquarters is based in Lisbon, with multiple development facilities in Portuguese cities, like Coimbra, Porto, Leiria, Aveiro, and Belmonte, and an office in the United Kingdom. WIT employs over three hundred employees which are distributed around all the facilities.

1.3. Document Structure

This report is organised into six chapters, where this first chapter makes an introduction describing the goals and motivation of the project alongside a brief presentation of the host institution.

Chapter 2 gives an overview of all the concepts that are relevant in the context of the internship.

Chapter 3 presents and describes the methodology used along with the planning of the internship.

Chapter 4 covers the two research stages. These stages represent a considerable part of the internship and are detailed in this chapter, from the first research stage to the focus shift that led to the second stage.

Chapter 5 reports the implementation and development process. This chapter presents the architecture of the proposed solution for developing a digital wallet using Multi-Party Computation to manage its keys and shows the development versions, the technologies used, and the functionalities implemented.

This report ends with a conclusion in Chapter 6 which sums up the entire report, describing the results and the future work.

2. Background

The purpose of this chapter is to provide context for this project by briefly describing some related concepts and explaining the reason for its growth. Section 2.1 describes blockchain, section 2.2 deals with cryptocurrencies, section 2.3 with their growth, and finally, section 2.4 describes how it is possible to interact with these technologies through wallets.

2.1. Blockchain

Blockchain is a decentralised and distributed digital ledger technology that securely records transactions across multiple computers in such a way that the registered transactions cannot be altered retroactively [4]. This technology has emerged in multiple sectors, so it is described below according to its most relevant features for this project's context. These features are:

- Decentralisation: unlike traditional databases that are typically managed by a central authority (like banks or other institutions), blockchain operates across a network of computers (referred to as nodes). This decentralisation makes it more resilient to failures and attacks.
- Immutability: once a transaction is recorded on the blockchain, it is extremely difficult to change. Each block in the chain contains a cryptographic hash of the previous block, along with a timestamp and transaction data. Changing any information in one block would require altering all subsequent blocks, which is computationally infeasible in a large network.
- Transparency: transactions recorded on a blockchain are visible to all participants in the network which helps build trust among them.
- Consensus mechanisms: blockchain networks use consensus algorithms to agree on the validity of transactions. Common mechanisms include Proof of Work (PoW), where miners solve complex mathematical puzzles, and Proof of Stake (PoS), where validators are chosen based on the number of coins they hold and are willing to "stake" as collateral.
- Smart contracts: some blockchains, like Ethereum (ETH), enable the use of smart contracts which are self-executing contracts with the terms of the agreement directly

written into code. They automatically execute when predefined conditions are met, enabling automated processes without intermediaries.

- Cryptography: blockchain uses cryptographic techniques to ensure the security and integrity of data. Each transaction is digitally signed, and the blockchain is secured through hashing algorithms.

Due to these features, blockchain has the potential to revolutionise many industries by providing a more secure, efficient, and transparent way to record and share data. It is used in various sectors such as supply chain management, healthcare, finance, identity verification, voting systems, and more [5]. However, it also faces challenges such as scalability, regulatory concerns, and energy consumption (especially in PoW systems).

2.2. Cryptocurrencies

Cryptocurrencies, such as Bitcoin and Ethereum, are digital or virtual currencies that use cryptographic techniques for secure transactions. They operate on blockchain technology, which helps maintain a decentralised and tamper-proof ledger of transactions. There are also some key concepts regarding cryptocurrencies:

- Digital Nature: cryptocurrencies exist only in digital form, and transactions are conducted electronically.
- Decentralisation: most cryptocurrencies operate on decentralised networks, meaning they are not controlled by any government or financial institution.
- Cryptography: cryptographic techniques secure transactions, control the creation of new units, and verify the transfer of assets.
- Mining: many cryptocurrencies use mining processes (like Bitcoin's Proof of Work) to validate transactions and create new coins.
- Volatility: cryptocurrency prices can be highly volatile, influenced by supply and demand, market sentiment, regulatory news, and other factors.

Knowing already that blockchain is a technology that enables the secure and transparent recording of data, cryptocurrencies are digital currencies that utilise that technology to provide a decentralised financial ecosystem. Together, they have the potential to reshape traditional systems in finance and beyond, offering new avenues for transactions, contracts, and data management [6].

2.3. Blockchain Market Growth

Since its emergence in 2009 with the launch of Bitcoin, the growth of blockchain technology has been significant, driven by the evolution of use cases, technological advancements, and increasing adoption across various industries. One of the greatest milestones in this growth was undoubtedly the launch of Ethereum in 2015. It introduced the concept of smart contracts, allowing developers to build decentralised applications (dApps) on a blockchain. From then on, there were several use cases and new enterprise solutions. Industries started adopting blockchain for supply chain transparency, cross-border payments, identity verification, and healthcare data management, among other applications [7].

A further step that has reinforced the relevance of blockchain is the emergence of Decentralised Finance (DeFi) and Non-Fungible Tokens (NFTs) [7]. Decentralised finance platforms emerged, offering financial services like lending, borrowing, and trading without intermediaries. This growth became a major trend in 2020 and 2021, resulting in billions of dollars in value locked in DeFi protocols. The rise of NFT, which is a unique digital identifier that is recorded on a blockchain and is used to certify ownership and authenticity, brought blockchain to the art and entertainment industries. Artists, musicians, and creators began using NFTs for ownership and monetisation of digital content.

Nowadays, there is continuous innovation, and the recent emergence of the WEB3 concept, which envisages a decentralised internet powered by blockchain, promises even more growth for this technology and it is likely to reshape industries and societal structures in profound ways.

According to this data from Grand View Research [8], the worldwide market for enterprise blockchain technology and related services will grow from an estimated $5.6 billion in 2020 to $13.8 billion by 2025, as we can see in Figure 1. The market hardly existed in 2016-17, hence rapid growth must be following this already rapid start. Growth will be fuelled initially by large organisations looking to transform critical existing operations. Future high levels of growth will be maintained initially by large competitors catching up to rivals, broader use of blockchain by these large organisations, and the expansion of blockchain use into the SME market.

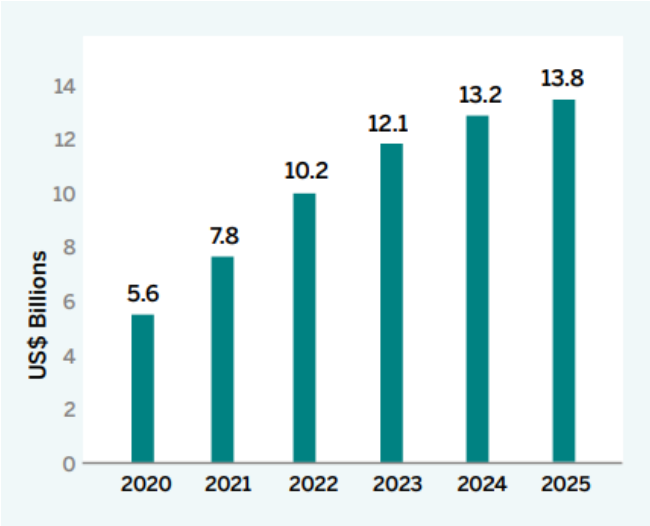

Figure 1 - Enterprise Blockchain Market Growth, 2020-2025 [8]

The paper [8] also estimates the percentage that each sector will have in the market. Figure 2 illustrates how the balance between market segments shifts significantly over the forecast period. While insurance has been the largest sector lately, over time government, legal, and media spending will increase rapidly to take the largest slice of the market share. Although the healthcare segment is showing a great deal of interest in blockchain to improve (among other things) electronic health records, that segment faces steep opposition in the form of market fragmentation, internal resistance to change, and regulatory hurdles. Therefore, although we see healthcare remaining active in this market, its growth will be significantly slower. Like the supply chain (where operating margins are often slim) media faces not only pressure on its bottom line but also major challenges to combat piracy as well as trust issues.

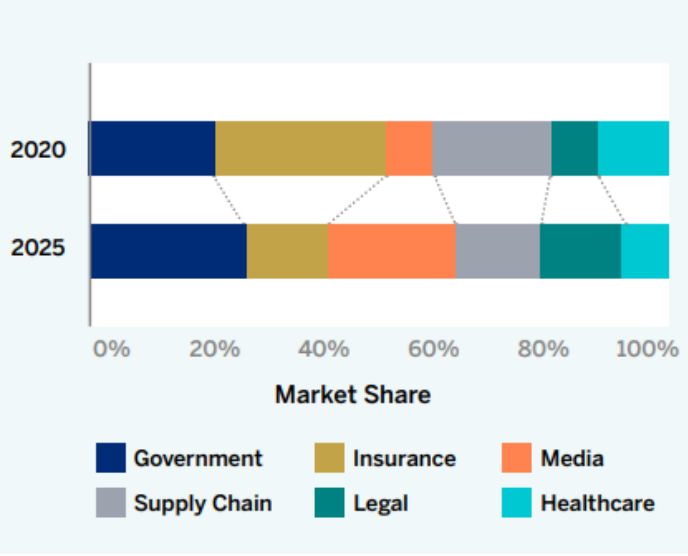

Figure 2 - Blockchain growth by industry sector [8]

2.4. Cryptocurrency Wallet

A cryptocurrency, crypto, or digital wallet, is a digital tool that allows users to store, manage, and transact with cryptocurrencies. Unlike a traditional wallet that holds physical cash, a crypto wallet does not store the coins themselves but rather holds the cryptographic keys that allow users to access and manage their cryptocurrency holdings on the blockchain. From a technical point of view, most crypto wallets can generate one or more pairs of public and private keys. The public key is used to generate addresses, which will be the identifiers of our wallets on a blockchain and are needed to receive payments. However, private keys are confidential keys used to sign and verify transactions.

There are three major crypto wallet categories: software wallets, hardware wallets, and paper wallets. Beginning with paper wallets, they are the safest type of wallet that exists [9]. As the name says, a paper wallet is a piece of printed paper with public and private keys. The paper has a QR code that represents the user keys and can be used for any transactions. The only concern of the user should be to keep that piece of paper and that is why is this type of wallet the safest.

However, for the purpose of this report, it is more relevant to distinguish hardware and software wallets and the primary difference between them lies in how they store the cryptocurrencies and the level of security they provide.

Hardware wallets are physical devices specifically designed to store cryptocurrency offline, providing a secure way to manage private keys. They are not always connected to the internet, the generation and storage of the private keys take place offline which makes them a form of cold storage and generally more secure against online threats. While hardware wallets are secure, they may not be as convenient for frequent transactions compared to software wallets. They require a USB (Universal Serial Bus) or Bluetooth connection to a device for transaction management [9].

Software wallets are applications or software programs installed on your computer or mobile device. They allow users to manage their cryptocurrency transactions online. Software wallets are typically connected to the internet (hot wallets), making them easily accessible for transactions, and their private key storage is guaranteed by encryption methods. Within software wallets, which was the type of wallet requested to be designed for

this internship, there are three other types: mobile wallets, desktop wallets, and web wallets [9].

Web wallets consist of a browser interface that requires no download or installation. They are more practical and convenient, but they are also more dangerous as the private keys are usually managed by third parties.

Desktop wallets are software programs that can be downloaded and run locally on a computer. Although they are not as practical and convenient, they are more secure as the private keys are stored locally and are only managed by the user.

Mobile wallets, on their turn, although similar to desktop wallets, have the particularity of being built for smartphones. They also stand out for the possibility of using QR codes to send and receive cryptocurrencies.

There are already several software wallets on the market today, and what sets them apart from each other are the different security mechanisms they use, the different authentication mechanisms, the different types of key management, support for different blockchains and cryptocurrencies, the different complexity levels and, finally, the different functionalities they offer [9].

Functionalities such as the possibility of exploring dApps, such as the Axie Infinity game, where players can earn and evolve unique characters. Wallets can also offer the possibility of visualising and managing all these characters and other collectables, called NFTs. Access to DeFi where users can lend, borrow, or earn interest on their assets. The purchase of cryptocurrencies with a bank card, the exchange of tokens or cryptocurrencies in a decentralised way, or even security options such as setting a transaction limit.

All these differences that wallets can have, all their possible functionalities, and this inverse relationship between their security and their usability, lead to the difficulty of finding a wallet that will completely satisfy the user's needs. To propose a solution for the prototype of a crypto wallet that would meet WIT's needs, a study of the state of the art of crypto wallets was compiled and is described below.

3. Methodology and Planning

The following sections describe the used methodology (section 3.1) and the different stages and tasks that were conducted along with the internship (section 3.2).

3.1. Methodology

This section describes the methodology used along with the internship, which helped in the research stages and in developing the digital wallet prototype. The chosen methodology for the development process of the internship was based on Agile, which allowed to evolve the platform incrementally from the start of the internship until the end. Agile is a project management methodology characterised by building products using short work cycles that allow for rapid production and constant revision when necessary. A group of seventeen people developed the core of the Agile methodology in 2001 in a written form. The written file was called the Agile Manifest of Software Development, and it enables a ground-breaking mindset on delivering value and collaborating with customers [10].

Agile four main values are expressed as:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

In WIT it is strongly recommended to use Scrum, and multiple teams employ Scrum as their methodology. Scrum is one of the implementations of the agile methodology in which incremental builds are delivered to the customer at the end of each development period, more known as Sprints.

Since the internship was focused on building a proof-of-concept key management system for a digital wallet, the use of a Scrum-like approach would help the development process. Still, since only one developer would compose the team, there was no need to use all the features of a strict Scrum, and it was decided that it would be used a Scrum-based approach, where some elements of the Scrum would be adapted to optimise the development process.

During my internship, this Scrum-based approach consisted of two weekly meetings with my supervisor, tutors, and analyst so I could share feedback on the work conducted and we could define the next tasks until the next meeting.

Additionally, nearly every month a presentation was delivered to the members of other internships, both interns and their managers. This ten-minute presentation included a brief summary of the last session, a demonstration of the developments that had taken place since then, and an indication of the next tasks and where they fit into the overall planning of the internship.

3.2. Planning

The internship lasted nine months and went from the 27th of September 2021 to the 24th of June 2022. Figure 3 shows the tasks that were carried out over the thirty-nine weeks using a Gantt chart.

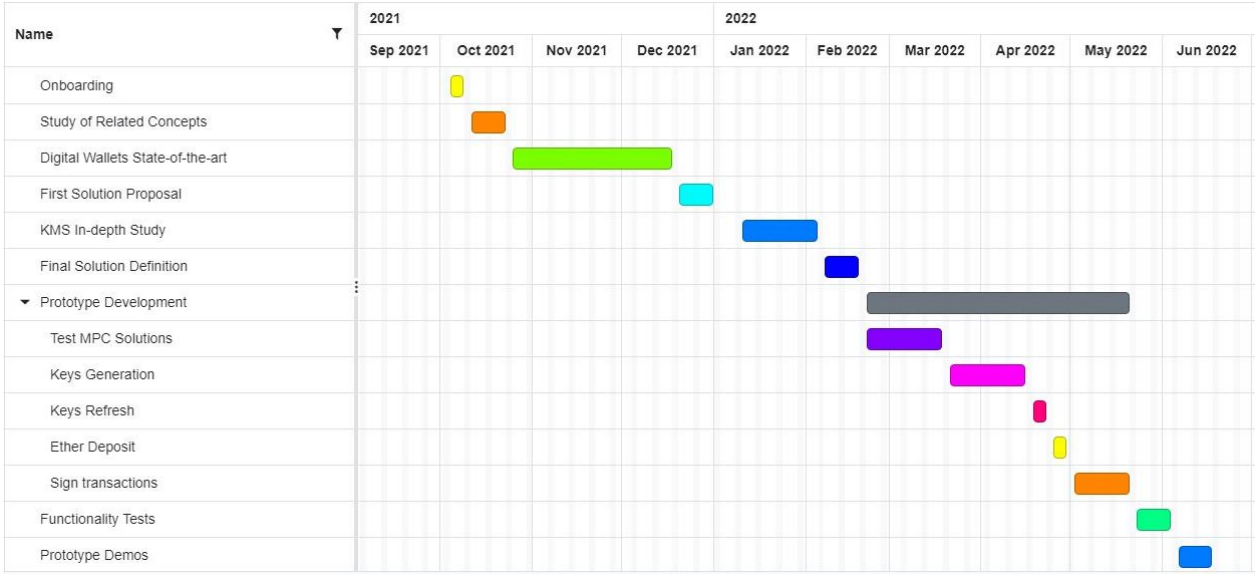

Figure 3 - Gantt Diagram of the Internship Program

Following the initial onboarding and a couple of weeks of studying the concepts related to blockchains and digital wallets, it is possible to split the internship phases into three main groups: the first research stage, the second research stage, and the prototype implementation.

As mentioned above, the first stage consisted of a more comprehensive survey of all the different wallets. During this phase, diverse data was collected from each wallet and divided into four categories: basic properties such as the type of key management, security features

such as the auto-lock timer, transaction features such as the choice of priority, and more global features such as the possibility of collecting NFTs.

This first stage culminated in a solution proposal to implement a prototype of a digital wallet that would guarantee maximum security and store the user's asset portfolio, offering them a favourable user experience. When presenting this proposal at a meeting with several WIT members, it was decided by everyone that the next step would be to redefine the focus of the project and perform a more in-depth study of wallet key management systems.

A second phase of research was then started to determine which key management system best suited WIT's desired solution, offering the maximum security and reliability possible. This research identified that the ideal solution was to use the Multi-Party Computation method.

After all the research was completed, it was only at the end of February that the prototype implementation phase began. This phase lasted until the end of May and included tasks such as testing the various MPC solutions, generating and refreshing keys, depositing Ether in the wallet, and signing transactions.

Finally, functionality tests were also performed, and demonstrations of the prototype were arranged for the final meetings of the internship.

4. Research Stages

The purpose of this chapter is to detail the entire analysis process that took place over several months of the internship, going through different research stages. Section 4.1 reports the first stage, section 4.2 describes the first proposal submitted, and section 4.3 reports the second research stage and final conclusions.

4.1. First Research Stage

This section describes the first research phase of this internship. In this phase, the idea was to identify the different properties that a crypto wallet could have and after analysing and comparing them, a proposal was made for a solution that would guarantee the needs demanded by WIT: making it possible to perform transactions on a blockchain while maintaining a portfolio of the user's digital assets, ensuring their security and offering a user experience suitable for mainstream users.

4.1.1. Definition of the Relevant Properties

As a first step in this phase, a mapping of the most relevant properties of the different wallets was made so that comparisons could be made between all of them. These properties are:

- **Wallet types**

Identification of the wallet type included mobile applications, browser extensions, web applications, desktop applications, and hardware wallets. The latter, although not an option for the development of the prototype, were also important to analyse to find out what they offered differently.

- **Authentication Methods**

There are several authentication methods used in crypto wallets. This research found methods such as the use of passwords that encrypt private keys, the use of biometric data, 12-word passphrases (or seed phrases), Two Factor Authentication (2FA) methods, backup files, and more common methods such as authentication via email or Google.

- **Cryptocurrencies and Tokens**

Each wallet can support one or more cryptocurrencies and tokens. Cryptocurrencies, as described earlier in this document, are digital currencies created to act as a payment method and store of value. Tokens are the digital representation of a real-world financial asset, i.e. an asset that has market value. It could be a coin, an object, a piece of real estate, or even a piece of music. Almost anything can be turned into a token and have its value represented digitally. Tokens are created through smart contracts within Blockchains that have this support and capacity, such as Ethereum and Solana. Several types of tokens are relevant to this research, such as ERC-20 (fungible tokens that can be exchanged for each other), ERC-721 (or NFTs, non-fungible tokens where each one has its unique value), and ERC-1155, which allow fungible and non-fungible tokens to be combined. In addition to the storage of several existing tokens, certain wallets also allow the user to create custom tokens.

- **Key Management**

There are two types of key management: custodial and non-custodial. The foremost factor to consider when comparing custodial with non-custodial wallets is who holds the private key. In the case of custodial wallets, the third party manages the private key. Whereas, in the case of non-custodial wallets, all the blockchain custodian services and responsibilities reside with users. In this internship, the main goal is to propose a prototype of a custodial wallet so that WIT can manage the private keys and ensure their security.

- **Security Attributes**

A digital wallet can have several attributes to make it a safer wallet. Examples include auto-lock timer, Hierarchical Deterministic (HD) settings that help users back up their accounts and reset lost account information, identity check, limits protection, multi-signature transactions that require more than one account to authorise certain transactions, and transactions with extra pins.

- **Transaction Features**

Throughout a transaction, some features will make the whole process easier. Examples have been identified such as price comparison when exchanging tokens, choice of priority on transaction fees, high-volume purchase, slippage protection, non-existence of fees on a cryptocurrency transaction, and the use of QR Codes.

- **Other Global Features**

Other more global features that add value to a wallet have also been identified. It can give the possibility of collecting NFTs, performing decentralised token swaps through Decentralised Exchanges (DEXs), dApps integration, access to DeFi to borrow or earn interest directly, buying crypto with a bank card, withdrawing to a bank, white contacts list, and automatic tokens detection. There are also possibilities to filter the transaction history, configure the network, and learn more through wallet tutorials.

- **Complexity and Target Audience**

Finally, careful attention was also paid to the complexity and target audience of each wallet. It was highlighted which ones have the best User Interface (UI) / User Experience (UX), which ones are more suitable for beginners, which ones require a certain amount of experience, and which ones allow users to choose that complexity.

4.1.2. Existing Digital Wallets

This section lists the wallets that were considered most significant and provides a brief description of each one and where they stand out.

4.1.2.1. MetaMask

MetaMask [11] is one of Ethereum's most used crypto wallets because its user-friendly interface provides quick and easy access to thousands of tokens and decentralised apps (dApps) within the Ethereum network. It is available as a mobile app or as an extension on several browsers and is also notable for its compatibility with other blockchain solutions. Users can add almost any blockchain network to the app, and the wallet fully supports popular Web3 networks, including Polygon, BNB Chain, and Avalanche. Users can also use the wallet to access popular NFT marketplaces like OpenSea and swap a variety of collectables. With MetaMask, the user’s account information is encrypted and stored locally through a key vault, no information ever touches the MetaMask servers, which means users have full control of their private keys.

Figure 4 shows the MetaMask logo and a sample of the browser extension and the mobile app, respectively.

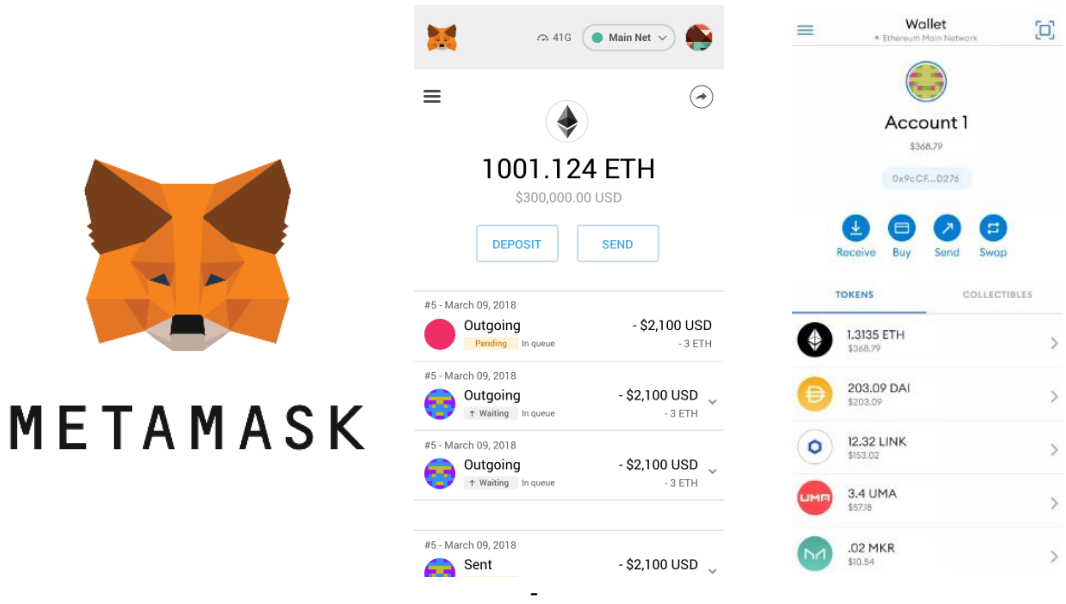

Figure 4 - MetaMask Logo and Samples [11]

4.1.2.2. Enjin Wallet

Enjin Wallet [12] is a cryptocurrency wallet designed for managing and storing various cryptocurrencies and blockchain assets, including Enjin Coin (ENJ) and NFTs. Developed by Enjin, a company known for its blockchain technology and gaming ecosystem, Enjin Wallet offers a user-friendly interface that caters to gamers and crypto enthusiasts alike.

Key features include Multi-Currency Support, NFT management, security features like biometric authentication and built-in secure seed phrase backup, and a built-in Decentralised Exchange (DEX). Figure 5 shows the Enjin logo and two samples of the mobile wallet.

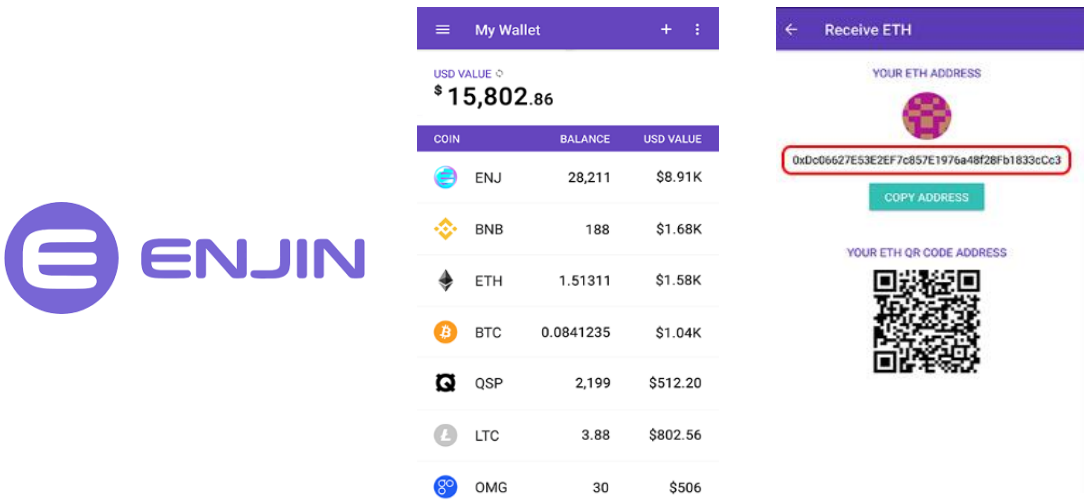

Figure 5 - Enjin Wallet Logo and Samples [12]

4.1.2.3. Trust Wallet

Trust Wallet [13] is one of the most popular wallets because it is the official non-custodial wallet app of Binance, one of the leading cryptocurrency exchanges in the world. One of the wallet's greatest advantages is that it supports over 100 blockchains. This is how it is able to store such a wide variety of digital assets (over ten million coins and tokens, one of the largest numbers on the market).

Trust Wallet is also a great mobile option for NFT and decentralised app enthusiasts. The wallet has a built-in Web3 browser, allowing users to access dApps and blockchain games directly through the app. This feature makes buying NFTs easy, as users can look, purchase, and store tokens using the incorporated decentralised exchange, all without leaving the app. In addition, it offers the possibility of purchasing tokens using a credit card.

Figure 6 shows the Trust Wallet logo and a sample of the mobile wallet main menu.

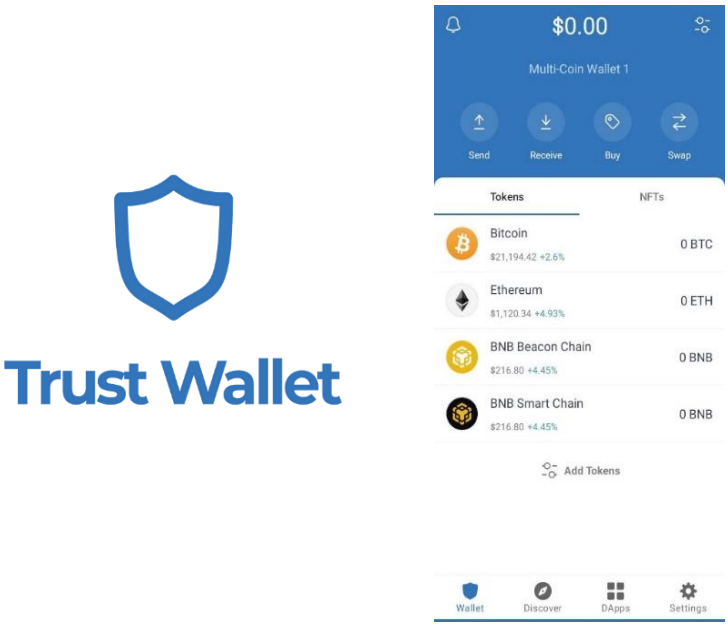

Figure 6 - Trust Wallet Logo and Sample [13]

4.1.2.4. Exodus

Exodus [14] is one of the most widely used desktop wallets on the market due to the speed of its transactions, ease of use, and the varied functionality of its client. One of Exodus wallet's main draws is the number of currencies it supports: more than 335 crypto and NFTs, a larger number than many other hot wallets.

Key features of Exodus include the option for trading fees and confirmation speed and the possibility of earning interest on more than 25 cryptocurrencies. However, funds are inaccessible if users lose their private key and recovery phrase.

Figure 7 shows the Exodus Wallet logo and a sample of the mobile app and the desktop wallet, respectively.

Figure 7 - Exodus Logo and Samples [14]

4.1.2.5. MyEtherWallet

MyEtherWallet (MEW) [15] is a free, open-source, client-side interface for creating and managing Ethereum wallets. It allows users to generate Ethereum wallets, send and receive ETH, and interact with smart contracts on the Ethereum blockchain.

Key features of MEW include new wallet creation, token management, swapping tokens, and integration with hardware wallets.

Figure 8 shows the MEW logo and a sample of its web wallet (browser extension).

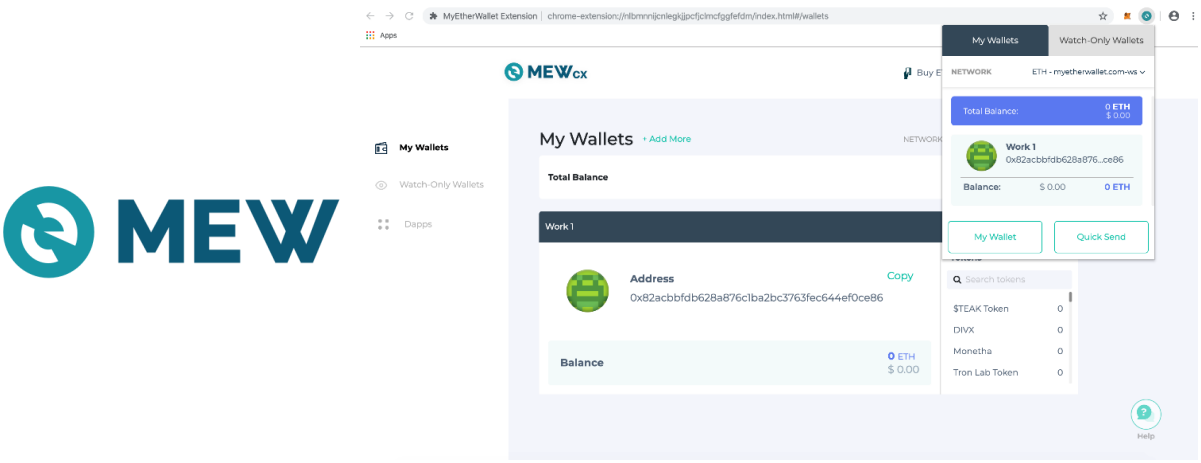

Figure 8 - MyEtherWallet Logo and Sample [15]

4.1.2.6. Coinbase Wallet App

Coinbase wallet [16] app is one of the most popular non-custodial wallets for beginners because it is an intuitive and highly secure wallet backed by a well-known exchange. Coinbase Wallet's user interface was designed to be easy to navigate, with a simple three-tab layout and clearly identifiable functions.

Is important to make a distinction between the Coinbase exchange and the Coinbase wallet. The Coinbase exchange is one of the oldest and most well-known crypto trading platforms. Holding digital assets on the exchange's web wallet makes it easier to trade, but leaves coins exposed to more dangerous cybersecurity threats. The Coinbase wallet may be used without opening an account with the exchange and it's non-custodial, meaning the private key is not in Coinbase's servers. This means you don't need to worry about your currencies being locked for any reason or exposed to a cyberattack on the website.

One of the key features is the use of the Secure Enclave chip available in Android and iOS devices to provide biometric authentication. However, this wallet also stands for having high fees to buy ETH and for its incapacity to store Bitcoin directly.

Figure 9 shows the Coinbase wallet logo and samples of its main menu and its option to show market charts.

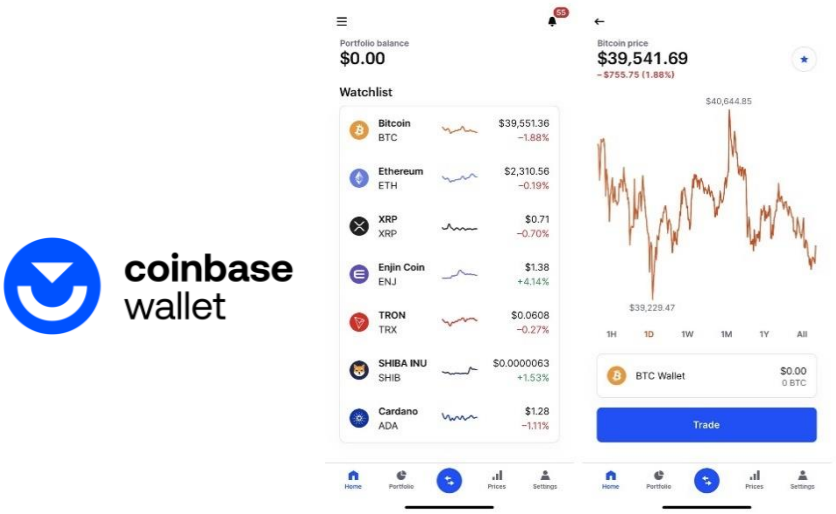

Figure 9 - Coinbase Wallet App Logo and Samples [16]

4.1.2.7. Crypto.com DeFi Wallet

Crypto.com Wallet [17] is one of the most popular DeFi crypto wallets due to its variety of decentralised finance tools, excellent onboarding process, and strong security framework. This wallet has features like one-to-one crypto swaps and tools for users to earn passive income on the crypto they own. Other key features are its several layers of security, including biometric authentication, 2-factor authentication, and Secure Enclave technology.

Figure 10 shows the Crypto.com DeFi wallet logo and a sample of its mobile app.

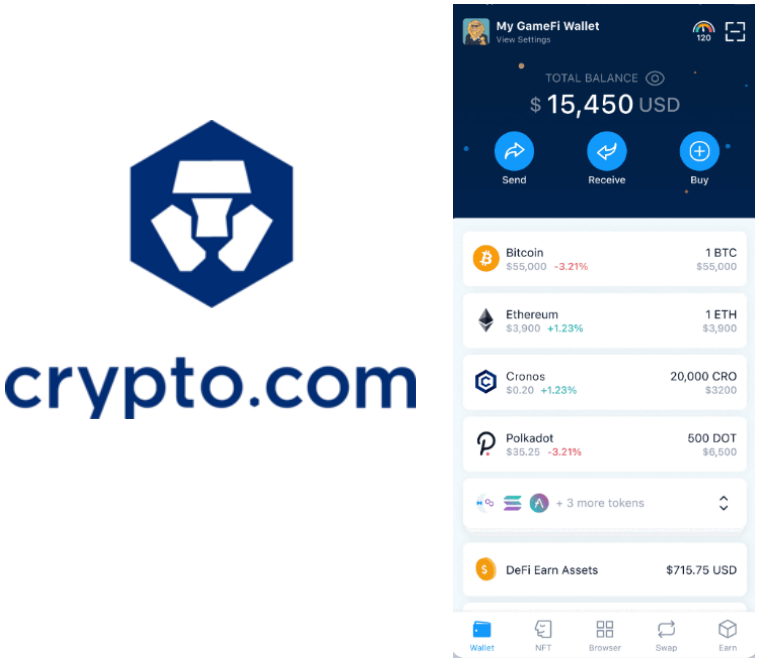

Figure 10 - Crypto.com DeFi Wallet Logo and Sample [17]

4.1.2.8. Guarda Wallet

Guarda Wallet [18] allows users to securely store, manage, and exchange cryptocurrencies and tokens. It is available as a web wallet, desktop application (for Windows, macOS, and Linux), and mobile app (for iOS and Android), making it accessible across multiple platforms. Guarda Wallet is designed to provide an easy and secure way for individuals to manage their cryptocurrency assets without requiring users to give up control of their private keys.

In addition to a user-friendly interface, it also has key security features such as encrypted backup and password protection, to ensure the safety of users' funds.

Figure 11 shows the Guarda wallet logo and web and mobile app samples.

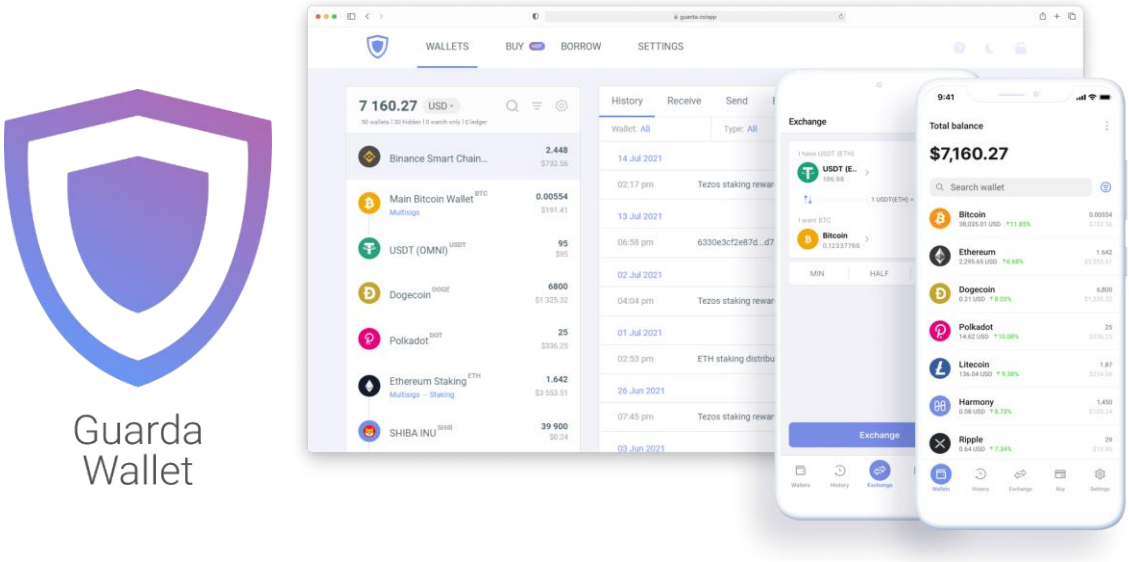

Figure 11 - Guarda Wallet Logo and Samples [18]

4.1.2.9. Binance Exchange

Binance [19] is a cryptocurrency exchange previously based in Hong Kong and is one of the absolute giants in the industry. It launched in July 2017 and has then really developed into being the undisputed leader when it comes to the trading volumes on crypto exchange.

Binance operates as a custodial exchange, meaning that it holds users' funds in wallets for them. Having control of the keys is important to ensure their security. For this purpose, it has features including multi-signature technology and Hardware Security Models (HSM) for private key storage, and it always educates its users on security best practices like 2FA.

Figure 12 shows the Binance exchange logo and a sample of its mobile app.

Figure 12 - Binance Exchange Logo and Sample [19]

4.1.2.10. Coinbase Exchange

Coinbase [20] is another one of the leading cryptocurrency exchanges in the world. Besides having a mobile wallet, as seen above, Coinbase Exchange also has a built-in custodial wallet, and its features are similar to Binance. They have common features like multi-signature technology and HSMs.

Figure 13 shows the Coinbase exchange logo and a sample of its web app.

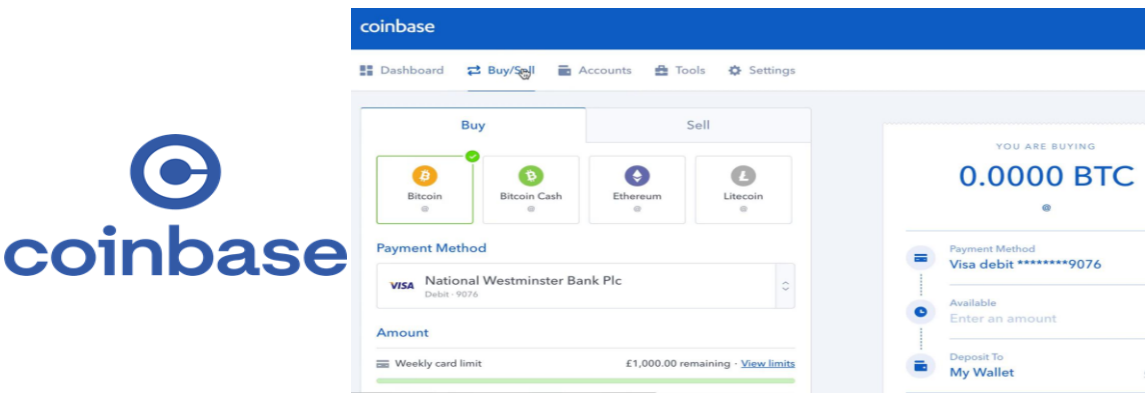

Figure 13 - Coinbase Exchange Logo and Sample [20]

4.1.2.11. Dapper Wallet

Dapper Wallet [21] is a digital wallet designed to manage and store NFTs and cryptocurrencies, primarily associated with the Dapper Labs ecosystem. Dapper Labs is the company behind popular blockchain projects such as NBA Top Shot, CryptoKitties, and Flow, a blockchain designed for fast and secure transactions tailored for NFTs and gaming. This web wallet is a custodial one and has security features like identity checks. Even so, it stands out more for its features of collecting NFTs from the NBA TopShots app and having integration with dApps on the Flow blockchain.

Figure 14 shows the Dapper logo and a sample of its web app.

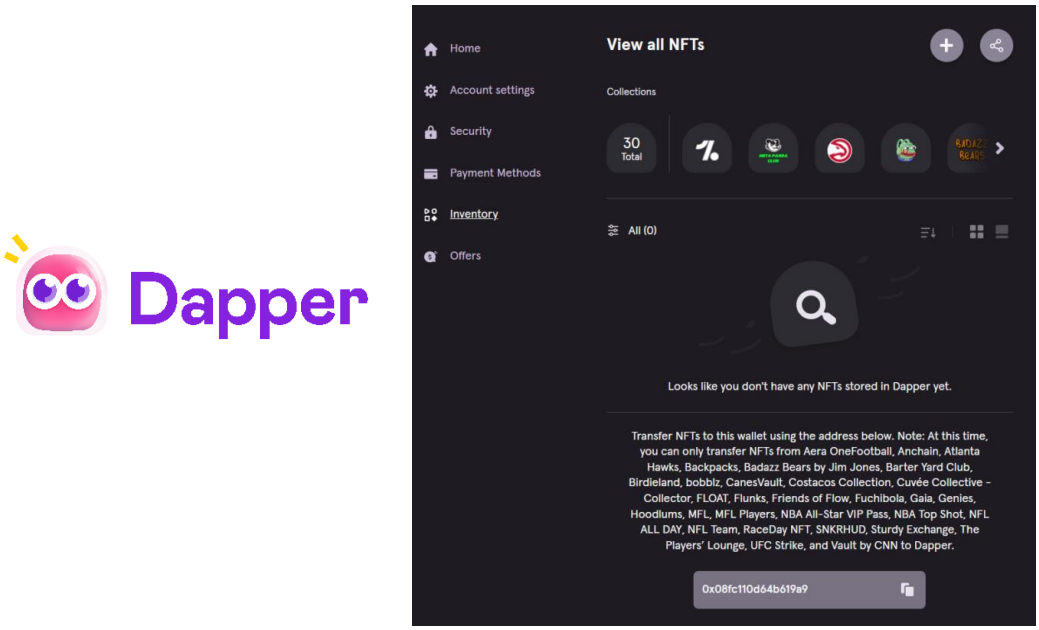

Figure 14 - Dapper Logo and Sample [21]

4.1.2.12. Sorare Wallet

Sorare Wallet [22] is the official digital wallet used within the Sorare ecosystem, which is a blockchain-based fantasy sports platform. Sorare allows users to collect, buy, sell, and trade digital player cards, which are certified as NFTs on the Ethereum blockchain.

When a user creates a Sorare account, a key pair (public and private key) is generated. The private key is encrypted and securely stored in a secure server managed by Sorare. When users access their wallets, they may need to authenticate themselves using a secure method such as a password, email verification, or other forms of authentication.

Figure 15 shows the Sorare logo and a sample of its web app wallet.

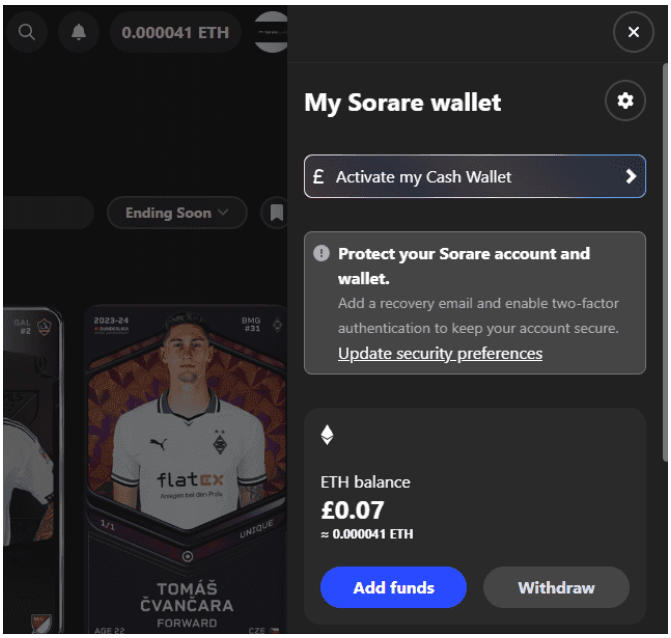

Figure 15 - Sorare Wallet Logo and Sample [22]

4.1.2.13. Ledger Nano X

Ledger [23] is one of the most well-known brands in the crypto space, with hardware wallets that are a popular choice among crypto enthusiasts. Its products stand out for using a Secure Element component, a type of chip often seen on passports, credit cards, and payment systems, to provide an extra layer of security.

Ledger Nano X key features include Bluetooth connectivity, large storage capacity, USB-C connection, multi-currency support, and firmware updates. Figure 16 shows the Ledger Wallet logo and its Nano X device.

Figure 16 - Ledger Nano X and its logo [23]

4.1.2.14. Trezor Model T

The Trezor Model T [24] is a leading hardware wallet known for its robust security features and user-friendly interface. The Model T features a full-colour touchscreen, which makes it easy to navigate through menus, enter your PIN, and confirm transactions directly on the device. This intuitive interface enhances the overall user experience.

Key features include multi-currency support, a highly secure processor to protect private keys, 24-word recovery seed phrases, and cross-platform compatibility.

Figure 17 shows the Trezor Wallet logo and its Model T device.

Figure 17 - Trezor Model T and its logo [24]

4.1.3. Wallets Comparative Tables

Once all these wallets have been analysed and information collected, it is important to compare them across all their components.

In this section, the information collected is divided into 4 tables: one with basic properties, one with security features, one with transaction features, and one with other global features.

4.1.3.1. Basic Wallet Proprieties Comparative Table

Table 1 shows a comparison of the basic properties of each wallet. This table examines types, authentication methods, crypto and tokens supported, key management, and complexities.

Table 1 - Basic wallet proprieties comparative table

Wallet	Type	Auth Methods	Crypto and Tokens	Key Management	Complexity
MetaMask	Mobile Browser Ext.	Password Biometric Passphrase	ETH, BNB, ERC-20, NFT, ERC-1155, Custom Token	Non-Custodial	Suitable for all experience levels
Enjin Wallet	Mobile	Password Biometric Passphrase	BTC, ETH, BNB, ERC-20, NFT, ERC-1155, Custom Token	Non-Custodial	For Beginners, Focus on UI/UX
Trust Wallet	Mobile	Password Biometric Passphrase	BTC, ETH, BNB, ERC-20, NFT, ERC-1155, + 1600	Non-Custodial	Suitable for all experience levels, Focus on UI/UX
Exodus	Mobile Desktop	Password Biometric Passphrase Backup File	BTC, ETH, BNB, ERC-20, NFT, ERC-1155, + 150	Non-Custodial	For Experts, Focus on UI/UX
My Ether Wallet	Mobile Browser Ext.	Password Biometric Passphrase	BTC, ETH, BNB, ERC-20, NFT, ERC-1155	Non-Custodial	For Beginners
Coinbase Wallet	Mobile Browser Ext.	Password Biometric Passphrase	BTC, ETH, BNB, ERC-20, NFT, ERC-1155, + 200	Non-Custodial	For Beginners, Focus on UI/UX
Crypto.com DeFi Wallet	Mobile	Password Biometric Passphrase	BTC, ETH, BNB, ERC-20, NFT, ERC-1155, + 200	Non-Custodial	For Experts, Focus on UI/UX
Guarda Wallet	Mobile Browser Ext. Web Desktop	Password Biometric Passphrase Backup File	BTC, ETH, BNB, ERC-20, NFT, ERC-1155, + 400K	Non-Custodial	For Experts
Dapper Wallet	Web	Email Google 2FA	BTC, ETH, BCH, DAI	Custodial	For Beginners
Binance	Mobile Web	Email Google 2FA	BTC, ETH, BNB, LTC, ERC-20, + 100	Custodial	Has Complexity Option
Coinbase	Web	Email 2FA	BTC, ETH, BNB, LTC, ERC-20, + 30	Custodial	For Experts
Ledger Nano X	Hardware Mobile Desktop	Password Passphrase	BTC, ETH, BNB, LTC, ERC-20, + 1800	Non-Custodial	For Experts
Trezor Model T	Hardware	Password Passphrase	BTC, ETH, BNB, LTC, ERC-20, + 1200	Non-Custodial	For Experts

By analysing Table 1 it is possible to conclude that:

- There is a major investment in web (app or browser extension) and mobile wallets, with lots of companies betting on both components.
- Companies that develop wallets tend to leave the private keys up to the users of the wallet, thus guaranteeing their total anonymity. In other terms, there is a greater focus on non-custodial wallets.
- The few cases of custodial wallets are usually associated with wallets that are part of exchanges, and in these cases, it is natural for companies to want to take responsibility for the private keys.
- In non-custodial wallets, the most used authentication methods are those that do not require user data to be stored in a third-party system. For example, the use of encrypted passwords, biometric data, and passphrases.
- In custodial wallets, the usual authentication methods are the ones we are most used to seeing in our daily lives, such as email, Google, and the use of 2FA.
- Regarding blockchains and tokens, it's clear that there's a big priority on guaranteeing the support of the leading ones on the market, meaning that they are multi-currency wallets.
- There are wallets with all kinds of complexity, but there is no doubt that many companies have tried to develop a wallet that offers a good UI/UX to the user so that they don't need to have a lot of experience using this type of technology.

4.1.3.2. Security Features Comparative Table

Table 2 shows a comparison of the security features of each wallet. This table considers whether the wallet has an auto-lock timer, Hierarchical Deterministic settings, identity check, limits protection, multi-signature option, and transaction pin.

By analysing Table 2 it is possible to conclude that:

- Some features are only explored in non-custodial wallets, such as the auto-lock timer and HD settings.
- Some features are very common in custodial wallets, such as the use of identity checks and the multi-signature option.
- The transaction pin feature is used in almost all cases.

Table 2 - Security features comparative table

Wallet	Auto-Lock Timer	HD Settings	Identity Check	Limits Protection	Multi-signature	Transaction Pin
MetaMask	☑	☑	☒	☒	☑	☒
Enjin Wallet	☑	☑	☒	☑	☒	☒
Trust Wallet	☑	☑	☒	☒	☒	☑
Exodus	☒	☑	☒	☑	☒	☑
My Ether Wallet	☑	☑	☒	☒	☒	☑
Coinbase Wallet	☑	☑	☒	☒	☑	☑
Crypto.com DeFi Wallet	☑	☑	☒	☑	☑	☑
Guarda Wallet	☒	☑	☒	☒	☑	☑
Dapper Wallet	☒	☒	☑	☒	☒	☒
Binance	☒	☒	☑	☑	☑	☑
Coinbase	☒	☒	☑	☑	☑	☑
Ledger Nano X	☒	☑	☒	☒	☒	☑
Trezor Model T	☒	☑	☒	☒	☒	☑

Table 3 shows a comparison of the transaction features of each wallet. This table considers whether the wallet has a price comparison option, a priority choice, accepts high volume purchases, has slippage protection, has no fees on transactions, and has a QR Code option.

Table 3 - Transaction features comparative table

Wallet	Price Comparison	Choice of Priority	High-Volume Purchases	Slippage Protection	No fees on transactions	QR Code for transactions
MetaMask	☑	☑	☑	☑	☒	☑
Enjin Wallet	☑	☑	☒	☒	☒	☑
Trust Wallet	☑	☒	☒	☑	☑	☑
Exodus	☑	☒	☒	☒	☒	☑
My Ether Wallet	☑	☑	☒	☑	☒	☑
Coinbase Wallet	☒	☒	☑	☒	☒	☑
Crypto.com DeFi Wallet	☒	☑	☑	☒	☒	☑
Guarda Wallet	☑	☑	☑	☑	☒	☑
Dapper Wallet	☒	☒	☑	☒	☑	☒
Binance	☒	☒	☑	☒	☒	☒
Coinbase	☒	☒	☑	☒	☒	☒
Ledger Nano X	☑	☒	☑	☒	☒	☒
Trezor Model T	☑	☒	☑	☒	☒	☒

By analysing Table 3 it is possible to conclude that:

- Price comparison functionality is associated with non-custodial wallets.
- Few mobile wallets support high-volume purchases.
- Custodial wallets have few transaction-related features.
- As expected, there is a strong investment in the use of QR codes by mobile wallets.

4.1.3.3. Global Features Comparative Table

Table 4 shows a comparison of the other global features of each wallet. This table considers whether the wallet has NFTs support, decentralised token swaps, dApps integration, access to DeFi, buy crypto with a bank card option, market charts, automatic token detection, and filtered transaction history.

Table 4 - Global features comparative table

Wallet	Collect NFTs	Decentralised Token Swaps	DApps Integration	Access to DeFi	Buy crypto with a bank card	Market Charts	Automatic Token Detection	Filtered Transactions History
MetaMask	☑	☑	☑	☑	☑	☒	☑	☒
Enjin Wallet	☑	☑	☑	☒	☒	☒	☑	☒
Trust Wallet	☑	☑	☑	☑	☑	☑	☒	☑
Exodus	☒	☑	☒	☑	☑	☑	☒	☑
My Ether Wallet	☑	☑	☑	☑	☑	☒	☑	☒
Coinbase Wallet	☑	☑	☑	☑	☑	☑	☒	☑
Crypto.com DeFi Wallet	☑	☑	☒	☑	☑	☒	☒	☒
Guarda Wallet	☒	☑	☒	☑	☑	☑	☒	☑
Dapper Wallet	☑	☒	☑	☒	☑	☒	☒	☒
Binance	☒	☒	☒	☒	☑	☑	☒	☑
Coinbase	☒	☒	☒	☒	☑	☑	☒	☑
Ledger Nano X	☒	☑	☒	☑	☑	☒	☒	☒
Trezor Model T	☒	☒	☒	☒	☑	☒	☒	☒

By analysing Table 4 it is possible to conclude that:

- Collecting NFTs is more common in non-custodial but there is the example of Dapper and Sorare which also offer this possibility and are custodial.
- Decentralised token swaps, dApps, and Defi integration are associated with non-custodial wallets.
- There is a strong trend toward buying crypto with a bank card, market charts, and transaction history.
- It is common for custodial wallets to withdraw money to the bank.

4.2. First Solution Proposal

All this research, corresponding to almost 3 months of the internship, resulted in the first solution proposal. To recap, the aim was to develop a digital wallet that allows transactions on a blockchain, maintains the user's digital asset portfolio, ensures maximum security, provides the best user experience, and can be integrated into any specific solution implemented under a blockchain.

In a nutshell, my first proposal consisted of implementing a REST (Representational State Transfer) API with two main endpoints: one for user accounts, where they can authenticate themselves and manage all their wallets, and another for the wallets themselves, where they can manage their balance, sign transactions, view their transaction history, and have an NFT inventory.

For this API to be able to connect to blockchains, blockchain providers are required. As this solution is intended to be implemented on Ethereum-based blockchains, the Web3 library has been proposed to do this integration via code. Subsequently, if it is necessary to integrate with other blockchains, there are also external APIs that already handle this connection. A database is also essential, to store data such as user information and the assignment of the respective wallets, and also to store the record of transactions made by each one.

Given that this is a proposal for a custodial wallet, the responsibility for managing private keys lies with the wallet's creator. The proposed solution was to use the AWS Key Management Service integrated with its Cloud Hardware Security Module. It was proposed

to use these AWS services because it was concluded that this area of key management is the most crucial and complex component of all the custodial wallets, being the component that differs the most between them and which makes them more reputable and reliable than others. By using these reliable AWS services, even though this key management was being done on their services rather than on WIT's servers, it was possible to guarantee security straight away and have time to implement other features in the wallet.

This API could then be integrated into a mobile or web application and the diagram of the solution proposed is shown in Figure 18.

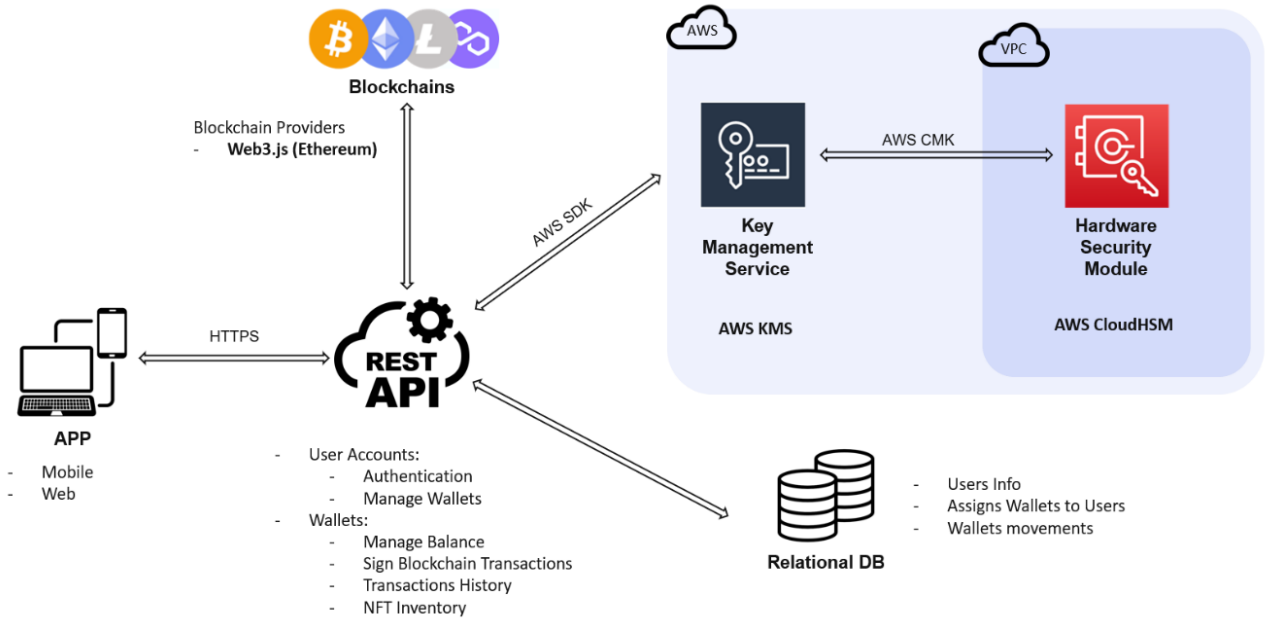

Figure 18 - First solution proposal diagram

In addition to this structure, it was also proposed which features would be implemented based on the analysis performed. Table 5 shows these features and categorises them as Must-Have or Should-Have using the colours blue and green, respectively.

Table 5 - Proposed features

Security Features	Transaction Features	Other Global Features
Email/Password	Perform Transactions	NFTs Collection
2 Factor Authentication	Transactions History	Balance Management
Transactions Pin	On-Ramp Deposit	Show Values in FIAT
Multi-sig Transactions	QR Code (Mobile)	Token Market Graphs
Limits Protection		
Biometric Data (Mobile)		

As already mentioned, this is a wallet that can later be integrated with several blockchain solutions, so it was important to have a meeting with directors and members of other WIT projects to discuss some ideas and get their opinions on my proposal. At this meeting, it was agreed that it was important to have a clear definition and seek out the best solution for managing private keys on a WIT server rather than using AWS services since this management is the crucial component of a custodial wallet and so it was possible to inject real value into my solution and my internship. It was therefore decided that it would be more beneficial to redefine the main focus of the internship, prioritizing private key management over possible features such as managing NFTs or checking transaction history.

4.3. Second Research Stage

This section approaches a new phase of research after changing the focus of the internship, where the goal shifted to choosing the most efficient and secure private key management to be carried out in a WIT environment. Specifically, this part of key management, as the name suggests, covers the entire key management in a cryptographic system, from creating keys to exchanging them, using them, and storing them.

4.3.1. Existing Key Management Solutions

To this effect, it was analysed the existing Key Management Solutions.

4.3.1.1. Local File System

The simplest approach is through the Local File System, where the private keys are generated and stored in an encrypted file where only the owner of that file knows its encryption key. With this method, blockchain transactions are signed locally without any kind of private data being sent over the Internet. This solution has the downside of not having a high level of private key protection, so it was soon discarded for this prototype.

4.3.1.2. Hardware Security Model

Hardware Security Modules (HSMs) are specialised physical devices designed to manage, process, and safeguard digital keys and perform cryptographic operations in a secure manner. HSMs provide a high level of security for sensitive information through hardware-based protection, making them an essential part of data security architecture, especially in enterprise environments. They are widely used across various sectors for

safeguarding sensitive information, complying with regulatory requirements, and ensuring the integrity and confidentiality of data.

Advantages:

- Physical Security. HSMs provide strong physical protection against tampering and unauthorised access. They are designed to withstand physical attacks and often feature tamper-evident seals and design.
- Performance. HSMs are optimised for cryptographic operations, allowing for high-speed processing for encryption, decryption, key management, and signature generation.
- Centralised Key Management. HSMs centralise key management, making it easier to implement security policies, generate secure keys, and manage key lifecycles.

Disadvantages:

- Cost. HSMs can be expensive to purchase, maintain, and operate, especially for small organisations or startups.
- Single Point of Failure. By centralising security functions, HSMs can become a target. If an HSM is compromised, the security of all systems relying on it could be jeopardised.
- Limited Flexibility. HSMs are specialised devices and changing configurations or adding functionalities may be limited compared to software-based solutions.
- Physical Access Requirement. Depending on its deployment, physical access to the HSM may be necessary for maintenance or monitoring, which could impact operational efficiency.

4.3.1.3. Trusted Execution Environment

Trusted Execution Environments (TEEs) are secure areas within a processor that ensure sensitive data and code are isolated and protected from the rest of the system, including the operating system, device drivers, and other applications. TEEs provide a trusted environment for executing code securely, enabling enhanced security and confidentiality for critical operations. Trusted Execution Environments provide a powerful mechanism to enhance security by isolating sensitive data and operations from the rest of the system using a

combination of hardware and software techniques. Their applications in secure payment, IoT, DRM, and cloud services illustrate their importance in modern secure computing environments. An example of this method is Secure Enclave, identified in a couple of wallets analysed in the first phase.

Advantages:

- Isolation. TEEs create a secure area within the main processor that runs sensitive code and processes data in isolation from the operating system and other applications, providing a high level of security against software attacks.
- Performance. Running within the processor, TEEs typically offer near-native performance for cryptographic operations and sensitive computations without the overhead associated with communication between multiple parties.
- Secure Data Processing. TEEs can handle sensitive data while ensuring that the data is not exposed to the host OS or potentially malicious software.

Disadvantages:

- Limited Memory. TEEs typically have restricted memory resources. This limitation can constrain the complexity of applications that can reliably run within a secure environment.
- Attack Surface. While TEEs are designed to be secure, they are still ultimately part of the device's hardware and cater to certain vulnerabilities. If an attacker can compromise the hardware, they could bypass the security of the TEE.
- Vendor Lock-in. TEEs are often tied to specific hardware vendors and architectures, potentially limiting portability and increasing reliance on a limited set of technology suppliers.
- Lack of Decentralisation. TEEs are tied to individual devices and don't provide the benefits of decentralisation, unlike some alternative technologies.

4.3.1.4. Multi-Party Computation

Using the Multi-Party Computation (MPC) method, instead of dealing with an entire private key, it will be divided into two or more parts, called key shares, and distributed among different devices. This means that when needed, this protocol must connect all the

devices that share the key to calculate the desired outcome. This way, none of the devices has full access to the private key and no point-of-failure.

Advantages:

- Data Privacy. MPC allows parties to jointly compute results while keeping their individual inputs private. This is crucial for scenarios where data confidentiality is paramount.
- Decentralisation. MPC does not require a trusted central entity, reducing the risk of a single point of failure and enhancing trust among participants.
- Flexibility. MPC can be applied in various distributed scenarios and can accommodate different types of computations, making it adaptable to various applications.
- No Need for Hardware. Unlike HSMs, MPC is typically software-based, which means it can run on standard computing hardware, reducing its initial investment costs.

Disadvantages:

- Complexity. Implementing MPC protocols can be complex in terms of both development and computation. Debugging and maintaining MPC systems can also be more challenging.
- Performance Overhead. MPC often incurs communication and computational overhead because it requires data to be split among multiple parties. This can lead to slower execution times compared to HSMs.
- Requires Trust Among Participants. While MPC mitigates the need for a central trusted entity, parties must still have some level of trust in each other's actions.

4.3.1.5. Smart Contract Based

Finally, there are smart contract-based wallets. As mentioned, a wallet is essentially an account with a pair of public and private keys where the public key corresponds to the wallet address and the private key to the password. However, there are also contract accounts, used to create a contract-based wallet, and are controlled using the code present in smart contracts and therefore have no private key.

Advantages:

- Programmability. Smart contracts can be programmed to execute specific actions automatically based on conditions defined by the user. This allows for customisable features like automated transactions, custom security rules, and multi-signature requirements.
- Transparency. Transactions conducted through smart contracts are recorded on the blockchain, offering transparency. This public ledger allows anyone to verify transactions, increasing trust and accountability.
- Enhanced Security Features. Smart contract wallets can include features like multi-signature setups, requiring multiple approvals for a transaction, which can mitigate risks related to a single point of failure or fraud.

Disadvantages:

- Complexity. The use of smart contracts can add complexity for average users, as they may need to understand programming concepts or the terms coded in the smart contracts.
- Security Risks. Smart contracts can have bugs or vulnerabilities in their code, which may lead to exploits and loss of funds. Auditing smart contracts is crucial, but not always foolproof.
- Cost of Transactions. Smart contract wallets operate on blockchain networks like Ethereum, which can be subject to high gas fees during periods of network congestion.
- User Experience. The interface and experience of using a smart contract wallet may not be as user-friendly as traditional wallets, especially for less technically inclined users.

4.3.2. Final Conclusions and Considerations

In this section, Table 6 shows a list of the advantages and disadvantages of each Key Management System (KMS) to make it clearer how they compare, and then conclusions are drawn based on its data.

Table 6 - KMS Comparison

KMSs	HSM	TEE	MPC	Smart Contract
Advantages	• Physical Security • Performance • Centralised Key Management	• Isolation • Performance • Secure Data Processing	• Data Privacy • Decentralisation • Flexibility • No Need for Hardware	• Programmability • Transparency • Enhanced Security Features
Disadvantages	• Cost • Single Point of Failure • Limited Flexibility • Physical Access Requirement	• Limited Memory • Attack Surface • Vendor Lock-in • Lack of Decentralisation	• Complexity • Performance Overhead • Requires Trust Among Participants	• Complexity • Security Risks • Cost of Transactions • User Experience

This second phase of research indicated that Multi-Party Computation was the most suitable option for the characteristics that WIT was looking for in this wallet. When choosing this method, there was concern about its complexity and the lack of existing paperwork, although it was decided to proceed with the exploration of its available open sources to try to implement a prototype wallet with the most efficient key management that had been identified, which was the internship's primary goal.

This lack of paperwork is due to this technology having only recently started to be used in crypto wallets and, with this being a key component that brings real value to the product, it is not surprising that companies don't want to fully disclose their developments. Zengo Wallet [25] is probably the biggest case of a crypto wallet that uses MPC for its private key management, however, as already mentioned, it provides minimal information on how they accomplish this procedure.

5. Implementation and Development Process

This chapter reports the implementation and development process. This chapter presents the architecture of the proposed solution for developing a digital wallet using Multi-Party Computation to manage its keys (section 5.1) and shows the development versions (section 5.2), the technologies used (section 5.3), and the functionalities implemented (section 5.4).

5.1. Final Solution Architecture

The solution consists of a digital wallet key management system prototype using the Multi-Party Computation method. When proposing this solution, there was an awareness that implementing all the components would be challenging due to the uncertainty of the time and effort that would be spent on the main component, the application server responsible for the key management system. Even so, it was decided to proceed with this proposal so that the components would be implemented step by step and those that were not completed would be referenced for future work, which turned out to be the case, as will be mentioned below.

In short, this solution consists of a REST API in Golang language that connects to the Ethereum blockchain to sign transactions, all executable through a React JS web app. Figure 19 shows the digital wallet proposed solution and the five integrated components:

- **Backend**

The backend is the main component and core of the wallet, responsible for the key management system. It was built in Golang because it is the language of the MPC open-source code integrated into this solution, as will be described later. Through the REST API developed, it is possible to have features such as user account management (including authentication and wallet management) and perform tasks on each wallet such as balance management, signing transactions, and visualizing transaction history.

- **Ethereum Blockchain**

To represent Ethereum's blockchain, Ganache is used to create a private test environment that simulates the existence of the blockchain and allows transactions to be controlled. To establish the connection between Ganache and the API, it uses the Geth

library, a GO implementation of Ethereum that is a gateway into the decentralised web. Geth is an Ethereum client, meaning it executes and handles transactions, deployment, and execution of smart contracts and contains an embedded computer known as the Ethereum Virtual Machine. Running Geth alongside a consensus client turns a computer into an Ethereum node [26].

- **Relational Database (RDBMS)**

In this solution, it is important to use a relational database, such as SQL Server, to be able to store user information, the assigned wallets, and their movements. The main benefit of the relational database model is that it provides an intuitive way to represent data and allows easy access to related data points. As a result, relational databases are most commonly used by organisations that need to manage large amounts of structured data, from tracking inventory to processing transactional data to application logging.

- **Web Application**

It is necessary to build a Web App so users can interact with the wallet and take advantage of all its features. In this case, it was suggested to implement a Single-Page Application (SPA) in React as it is an environment where I feel comfortable programming. React [27] is a popular JavaScript library for building user interfaces and offers several advantages that make it a good choice for straightforward web applications. Examples of that include component-based architecture, declarative syntax, strong community and resources, code reusability and flexibility.

- **Virtual Machine**

By using only, the MPC method from the Taurus lib within the application server, it is possible to run it in two separate threads and simulate communication between two devices to manage the keys. However, to enhance the proof of concept (PoC), the ideal scenario is to actually have two separate environments, so it was proposed to use a virtual machine that also runs the MPC method and stores a key share simultaneously with the server.

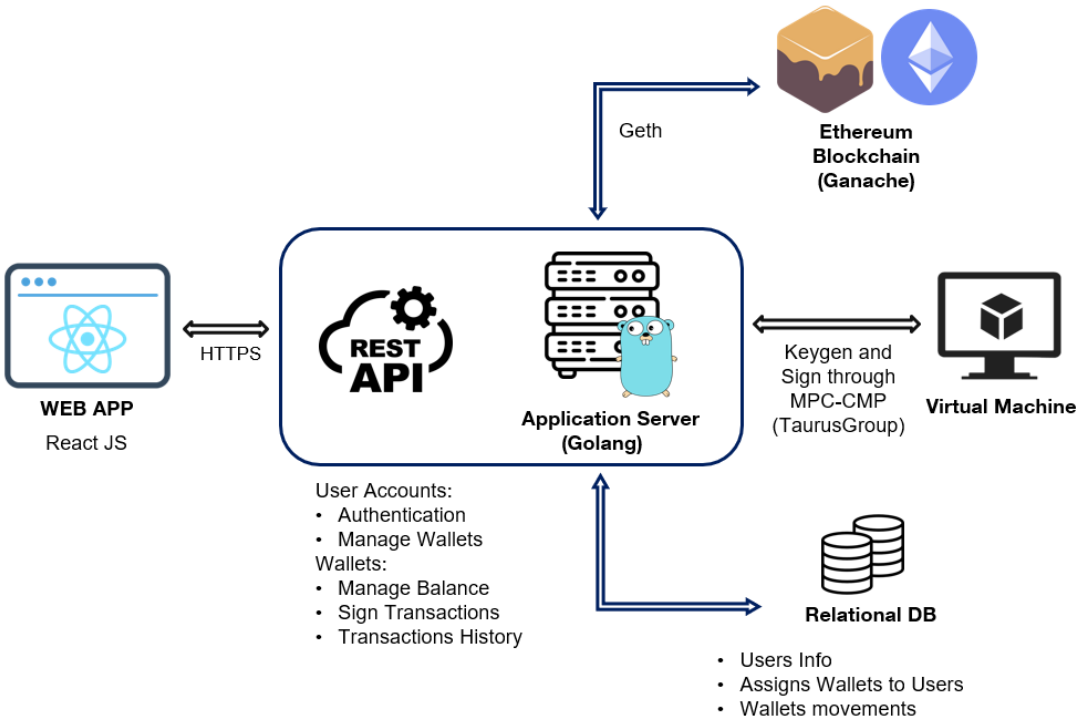

Figure 19 - Project Proposed Solution Diagram

5.2. Development Versions

As mentioned in section 5.1, the architecture above represents an ideal solution that incorporates multiple components and delivers maximum value to the wallet and the project. Given that implementing everything would not be possible, it was important to define different development milestones so the wallet could have five different versions. Figure 20 shows these five versions, with the green one being the implemented version in my internship and the blue ones being the versions that would add even more potential to the wallet and are assigned to future work.

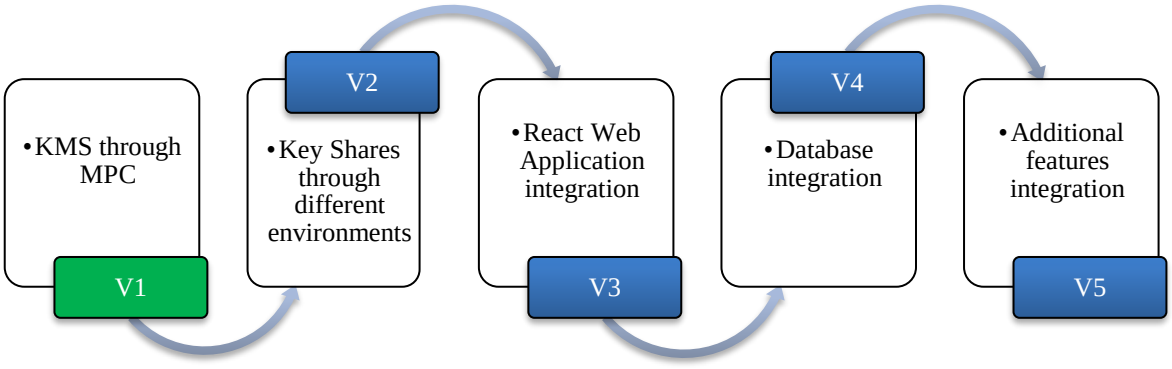

Figure 20 - Proposed Solution Development Versions

The first version (V1) is the core functionality of my entire project and is therefore the minimum version required for this to be a successful development. This version involves implementing an API to generate a digital wallet using an MPC method and sign transactions on an Ethereum network. In this first version, the two parties that store the key shares are simulated using two threads running in the same environment. The purpose is to be able to demonstrate all these processes taking place in the background through the command line of the IDE where it is being developed.

In the second version (V2), the idea was to reinforce the value of this PoC and MPC method. To achieve this, the prototype also had to be run on a virtual machine so it could run in different environments. In this way, both key shares generated when creating the wallet would be stored in separate environments, strengthening the wallet's security.

In the third version (V3), the frontend component would be added to the project so the wallet's capabilities could have a more appealing visibility. In this React web app, as the intention is to demonstrate the key management system, it would be possible to visualise all the processes that were previously only visible via the command line, from the key shares generated when the wallet was created to its signatures.

In the fourth version (V4), the database component would be added to make it possible to implement a basic authentication method, with email and password, which would allow users to be managed and associated with each wallet.

By implementing these four versions, the wallet would include all the components of the solution and would be ready to add other features such as transaction history, thus reaching version five (V5).

5.3. Technologies

This section describes all the technologies used during development and demonstrates how they were applied in the project.

5.3.1. Ganache (Ethereum)

Ganache [28] is an application used for Ethereum blockchain development. It is part of the Truffle Suite, which is a set of tools designed to simplify the development process for dApps on the Ethereum network. Ganache allows developers to create and manage a private Ethereum blockchain, giving them a controlled environment for testing and deploying smart contracts.

Key features of Ganache include:

- Local Blockchain: Ganache creates a personal Ethereum blockchain that runs on your local machine, allowing for quick testing without needing to deploy to the main Ethereum network.
- Instant Mining: Transactions are mined instantly, which speeds up the testing process compared to waiting for confirmations on the main network.
- Adjustable Gas Price and Balance: Users can customise gas prices and assign ether balances to various accounts to simulate different conditions and scenarios.
- User-Friendly Interface: Ganache comes with a graphical user interface (GUI), making it easier for developers to interact with the blockchain, view transactions, and test contracts.
- Integration with Truffle: Ganache integrates seamlessly with Truffle, allowing developers to compile, deploy, and test smart contracts using a familiar workflow.
- Advanced Debugging: Ganache supports advanced debugging features, making it easier to troubleshoot and rectify issues in smart contracts.

Below is a demonstration of how the Ganache GUI operates and how I used it in development. As shown in Figure 21, Ganache GUI lists all the wallets that are active in the user's private environment, showing their address, balance, number of transactions and index. Initially, each wallet is generated with a 100 ETH balance, but the first one on the list already has 190 transactions and a current balance of 74.20 ETH.

Figure 22 shows an example of a wallet which initially had a balance of 100 ETH and which, after three transactions, only had 99.93 ETH due to the amount of gas fee spent on it. The transactions section shows the hash, from address, to address or contract address, gas

used, and value transferred. By offering these capabilities, Ganache was a key tool in making it possible to create private wallets and perform transactions between them and the wallet I was developing.

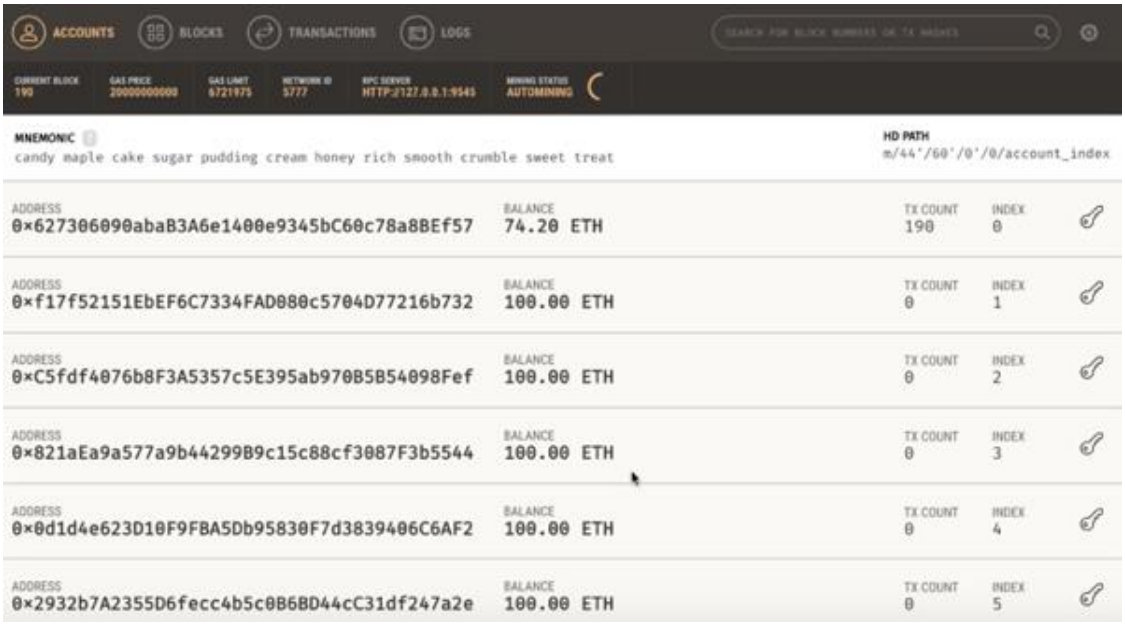

Figure 21 - Ganache Accounts List

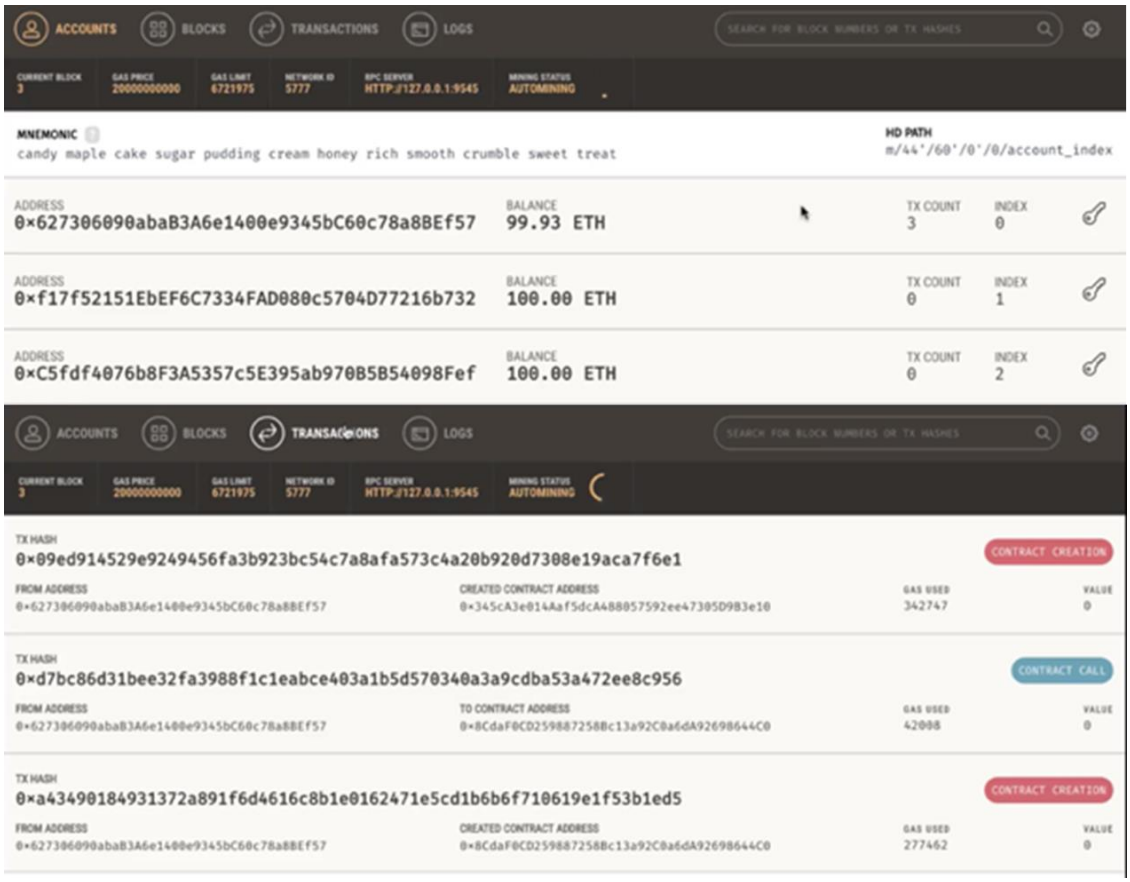

Figure 22 - Ganache Transactions List

5.3.2. Multi-Party Computation

A brief description of this method has already been written above, but in this section, in addition to being covered in more detail, the different algorithms of MPC will also be explained and which one was chosen for implementation.

5.3.2.1. MPC Concept

MPC stands for secure Multi-Party Computation and is a field of cryptography born 30 years ago. Generally, MPC allows two or more parties to jointly compute a function output without revealing their inputs. For example: Using MPC, a group of friends can securely calculate their average salary, without revealing how much each of them gets.

Specifically for cryptocurrency wallets, MPC allows the creation of a secure key management system without a single point of failure in which multiple parties can jointly perform all needed cryptographic functions (like key generation, transaction signatures and transaction verification) while neither of the parties reveal their respective secrets. It is important to highlight that in MPC a single private key is never generated, split, or reconstructed in the process: This makes it superior to traditional models based on a single private key.

By implementing this type of MPC technology, consumer-focused wallets can securely design an on-chain asset management system that removes the single point of failure of a private key. This offers a more secure self-custodial option by protecting against both private key theft (as there is no single private key to steal) and key loss, as each party can back up their secret input individually in a way that does not expose and compromise the entire system.

Figure 23 taken from Mercuryo.io [29] shows how multiple entities share the different key shares, which represent a partial signature, and ultimately result in just one signature that will allow the transaction to proceed.

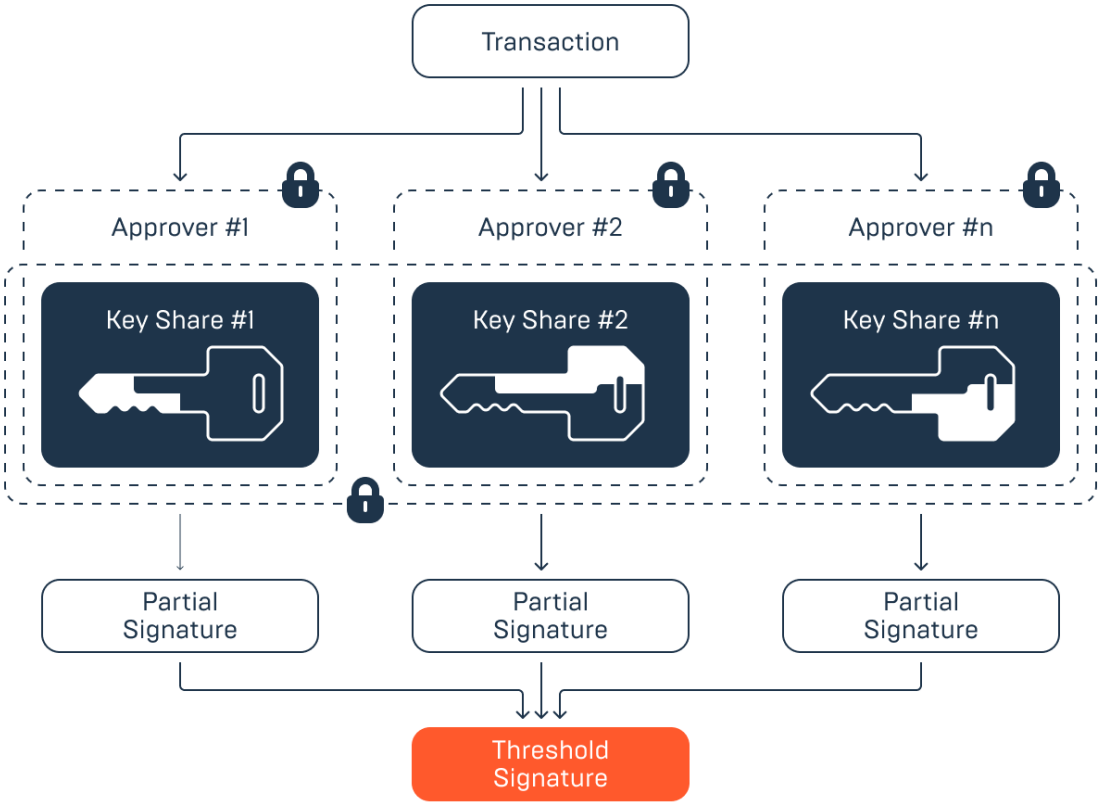

Figure 23 - MPC Key Shares [29]

5.3.2.2. MPC Algorithms

Given its inherent properties, multi-party computation, in and of itself, is a powerful tool for securing digital assets. However, not all MPC algorithms are created equal. There are:

- **Gennaro and Goldfeder Algorithm**

Gennaro and Goldfeder's algorithm [30] is currently one of the top MPC algorithms available, and many institutions that protect their private data using MPC utilise this algorithm. However, with Gennaro and Goldfeder's algorithm, the communication latency between the MPC shares (the devices that hold the key shares) does not reach the highest level of efficiency – as it requires users to wait for transactions to undergo up to 9 signature rounds. In addition, Gennaro and Goldfeder's algorithm does not offer any flexibility for institutions that need to use cold storage.

- **Lindell Algorithm**

Lindell et al. algorithm [31] offers a slight decrease in the number of transactions that need to be signed from Gennaro and Goldfeder, at 8. However, this still doesn't reach the

level of operational efficiency necessary for today's markets. Like Gennaro and Goldfeder, Lindell et al. do not offer support for cold storage.

- **Doerner Algorithm**

Doerner MPC algorithm [32] accomplishes a threshold using just 6 signatures. Yet, again, the level of efficiency that's possible with today's technology is still higher than this. And like the previous two algorithms, Doerner et al. can't provide solutions for institutions using cold storage in tandem with MPC.

- **MPC-CMP Algorithm**

MPC-CMP [33] enables digital asset transactions to be signed in just 1 round, offering the fastest transaction signing speeds of any MPC algorithm by 800%. MPC-CMP also solves the challenges faced by businesses looking to use cold storage in tandem with multi-party computation by allowing hot and cold key signing mechanisms – with at least one key share stored offline in an air-gapped device. In MPC algorithms, the leading factor that slows down the signing is the communication latency between the MPC-share (the devices that hold the key shares). Every communication round introduces additional latency. Using non-interactive signing and pre-processing, MPC-CMP requires only 1 round to sign a transaction, improving on the "round complexity," or time it takes to complete a round, of the previous algorithms.

Table 7 compares the different algorithms classifying their transaction rounds, if they are universally composable, cold storage compatible, and if they have open-source implementations. It can be concluded from this table that the ideal solution is to integrate an open-source implementation of the MPC-CMP algorithm, so it was decided that this was the algorithm to be implemented in this project.

Table 7 - MPC Algorithms

Algorithm	Transaction Rounds	Universally Composable	Cold Storage Compatible	Open-Source
Gennaro and Goldfeder	9	No	No	Yes
Lindell	8	No	No	No
Doerner MPC	6	No	No	No
MPC-CMP	1	Yes	Yes	Yes

5.3.2.3. Taurus MPC-CMP

The open-source of MPC-CMP available was developed by Taurus and according to them, it is the world's first implementation of this state-of-the-art cryptographic protocol that allows several distinct parties to collectively compute a digital signature with an Elliptic Curve Digital Signature Algorithm (ECDSA), targeting application to cryptocurrency wallets for Bitcoin, Ethereum, and any asset using this ECDSA signature scheme. The implementation uses the Go language and is released under the Apache 2.0 license, a permissive open-source license allowing the reuse of the code by other organisations [34].

This implementation supports multiple threshold signature schemes and protocols. Among them, those that stand out and are most relevant to this solution are generating a new ECDSA private key shared among all the given participants, refreshing all shares of an existing ECDSA private key, generating an ECDSA signature for a message Hash, and generating a pre-processed ECDSA signature which does not depend on the message being signed.

Keygen generates a new shared ECDSA key over the curve defined by ‘group’. After successful execution, all participants possess a unique share of this key, as well as auxiliary parameters required during signing. Figure 24 shows how keygen is implemented in the Taurus protocol used in the implementation.

```go
func Keygen(group curve.Curve, selfID party.ID, participants []party.ID, threshold int, pl *pool.Pool) protocol.StartFunc {
    info := round.Info{
        ProtocolID:       "cmp/keygen-threshold",
        FinalRoundNumber: keygen.Rounds,
        SelfID:           selfID,
        PartyIDs:         participants,
        Threshold:        threshold,
        Group:            group,
    }
    return keygen.Start(info, pl, nil)
}
```

Figure 24 - Taurus MPC-CMP Keygen

Refresh allows the parties to refresh all existing cryptographic keys from a previously generated Config. The group's ECDSA public key remains the same, but any previous shares are rendered useless. Figure 25 shows how key refresh is implemented in the Taurus protocol used in the implementation.

```go
func Refresh(config *Config, pl *pool.Pool) protocol.StartFunc {
    info := round.Info{
        ProtocolID:       "cmp/refresh-threshold",
        FinalRoundNumber: keygen.Rounds,
        SelfID:           config.ID,
        PartyIDs:         config.PartyIDs(),
        Threshold:        config.Threshold,
        Group:            config.Group,
    }
    return keygen.Start(info, pl, config)
}
```

Figure 25 - Taurus MPC-CMP Refresh

Sign generates an ECDSA signature for ‘messageHash’ among the given ‘signers’. Figure 26 shows how a sign is implemented in the Taurus protocol used in the implementation.

```go
func Sign(config *Config, signers []party.ID, messageHash []byte, pl *pool.Pool) protocol.StartFunc {
    return sign.StartSign(config, signers, messageHash, pl)
}
```

Figure 26 - Taurus MPC-CMP Sign

Presign generates a pre-processed signature that does not depend on the message being signed. When the message becomes available, the same participants can efficiently combine their shares to produce a full signature with the PresignOnline protocol. Figure 27 shows how pre-sign is implemented in the Taurus protocol used in the implementation.

```go
func Presign(config *Config, signers []party.ID, pl *pool.Pool) protocol.StartFunc {
    return presign.StartPresign(config, signers, nil, pl)
}
```

Figure 27 - Taurus MPC-CMP Presign

5.3.2.4. ECDSA

ECDSA stands for Elliptic Curve Digital Signature Algorithm and is a cryptographic algorithm used for digital signatures that is based on elliptic curve cryptography (ECC). ECDSA is widely used for securing communications and verifying the authenticity of messages, especially in contexts like blockchain technology, secure communications (TLS/SSL), cryptocurrencies, and various authentication protocols.

ECDSA involves two main processes: signing and verification. During signing, the signer selects a private key and computes the corresponding public key using the elliptic

curve equation. Then, the message to be signed is hashed using a cryptographic hash function (like SHA-256), resulting in a fixed-length message digest. In the next step, a random integer is selected from the range of the curve parameters to calculate the pair of the signature. During verification, the verifier receives the message, the signature, and the signer's public key. Then, the hash of the message is computed to determine if the signature is valid.

This Elliptic Curve Digital Signature Algorithm is a powerful and efficient method for generating digital signatures, leveraging the mathematical properties of elliptic curves to provide strong security with relatively small keys. This makes it a popular choice in modern cryptographic applications, particularly in scenarios where computational resources are limited or security needs are critical.

5.3.3. Golang

As mentioned, the open-source language of the MPC-CMP from Taurus is Golang (or Go Language) and therefore all the application server code is in this language.

Go is an open-source programming language developed by Google in 2007 and officially released in 2012. It was designed by Robert Griesemer, Rob Pike, and Ken Thompson to address shortcomings in existing programming languages, particularly in system programming and concurrent programming [35].

Go includes key features such as simplicity and clarity, static typing and efficiency, concurrency support, garbage collection, fast compilation, and cross-platform development. Highlight for concurrency support because it means that Go has built-in support for concurrent programming via goroutines and channels. Goroutines allow functions to run concurrently without the complexity traditionally associated with threads, while channels facilitate safe communication between goroutines. These features are crucial for simulating the existence of two or more devices, thus ensuring that the Multi-Party Computation method works properly.

It is particularly well-suited for system-level programming and building scalable applications, making it a popular choice for both startups and established tech companies. Whether developing web applications, cloud services, or network tools, Go offers an optimal blend of features that appeal to developers looking for efficiency and ease of use.

5.4. Functionalities

After studying and testing the Taurus MPC-CMP implementation, the prototype began to be developed.

5.4.1. Two-Party Key Generation

As the aim is to establish communication between two devices, it is important to understand which data is present in the communications between them. Figure 28 shows the key process of a two-party ECDSA protocol.

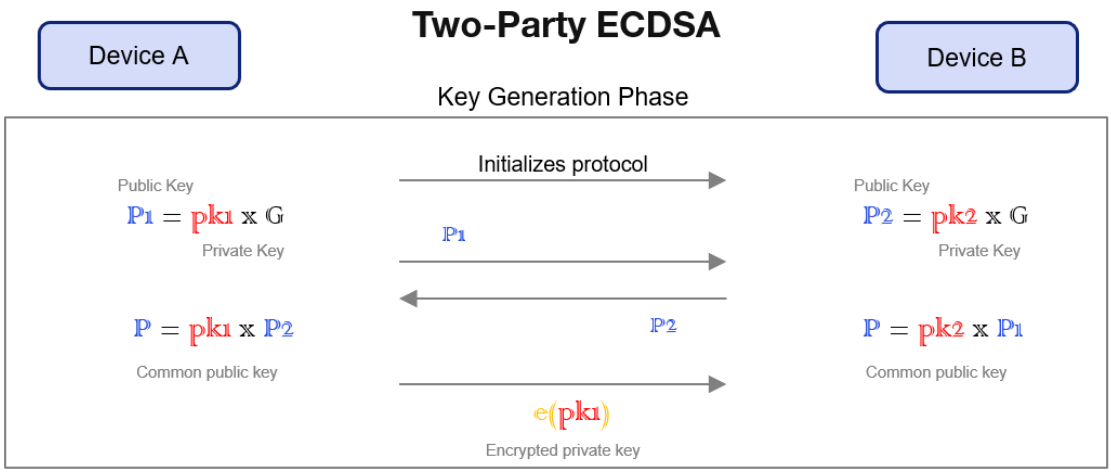

Figure 28 - Two-Party ECDSA Key Generation

In a key generation, device A initiates communication with device B so that they can share the public keys of their parts. Device A, using its private key (pk1) and B's public key (P2), manages to calculate a global public key (P) which will be the wallet's public key. As can be seen, there will never be a global private key.

With this in mind, the next step was to extract this data when running the Taurus protocol. This step turned out to require some effort because both the public key and the signature were being generated in different formats from what I was looking for, so it took some extra effort until I managed to generate values in the desired encoding.

Figure 29 shows the private key share being generated by the MPC-CMP algorithm and then converted into a hexadecimal String.

```
privKeyShare, err := r.(*cmp.Config).ECDSA.MarshalBinary()
if err != nil {
    return nil, err
}
fmt.Println("PrivateKeyShare ( ", id, " ): ", hexutil.Encode(privKeyShare))
```

Figure 29 - Private Key Share Generation

Figure 30 shows that this MPC-CMP protocol does not generate the public key directly. Instead, it generates a common "public point" from which it is possible to generate the public key, which is then converted into a Hexadecimal String.

```
publicPoint, err := r.(*cmp.Config).PublicPoint().MarshalBinary()
if err != nil {
    return nil, err
}

pubKeyCompressed, err := secp256k1.ParsePubKey(publicPoint)
if err != nil {
    return nil, err
}

// SerializeUncompressed serializes a public key in the 65-byte uncompressed format
pubKeyBytes := pubKeyCompressed.SerializeUncompressed()

pubKeyHex := hexutil.Encode(pubKeyBytes)
fmt.Println("PublicKey: ", pubKeyHex)
```

Figure 30 - Public Key Generation

Once the private key shares and the common public key are in place, it is also important to calculate the wallet's address to receive transactions from other wallets. To do this, with a public key of 64 bytes, it is necessary to apply the Keccak-256 algorithm and with the last 20 bytes of that result, the wallet address is obtained. Figure 31 shows the wallet address calculation.

```
//Address Calculation
address := common.BytesToAddress(crypto.Keccak256(pubKeyBytes[1:])[12:])
fmt.Println("Address: ", address)
```

Figure 31 - Wallet Address Calculation

To verify this process, the IDE terminal was used with the Ganache application. In this first phase of key generation, Ganache is irrelevant, but it is visible in the terminal that the

protocol was executed between two parties: A and B. From this execution, it was possible to extract the values of both private key shares, the global public key, and the wallet address, as can be seen in Figure 32.

Figure 32 - Keys Generation Demo

5.4.2. Ether Deposit

As the name suggests, this feature consists of receiving an Ether transaction. The demonstration uses a test wallet created in Ganache containing 12 ETH. To transfer 1 ETH from this wallet to ours, the first step is to enter the keys of the test wallet. Figure 33 shows that key setting.

```
//FUND NEW WALLET
fmt.Println("")
fmt.Println("-------- MY WALLET DEPOSIT --------")

senderPubKey := "0x79D59c31D04ea8b5BEcDB9792cAdd933FE830A76"
senderPrivKey := "289e2205e975e789658953f1458098d4eabec405e6eff47172b869ea41dd3d9d"

FundNewWallet(client, senderPubKey, myAddress, senderPrivKey)
```

Figure 33 - Sender Keys Setting

To proceed with the transaction, it is necessary to define all the values belonging to a transaction on the Ethereum blockchain. These are the nonce, receiver, value, gas limit, gas price, and data. Figure 34 shows how I obtained each of these values.

```
// Calling pending nounce - Preventing double spending attack
nonce, err := cl.PendingNonceAt(context.Background(), sender)
checkError(err)

value := new(big.Int)

// In wei: 100000000000000000, convert ether: 0.01
value.SetString("100000000000000000", 10)
gasLimit := uint64(21000)
gasPrice, err := cl.SuggestGasPrice(context.Background())
checkError(err)


receiver := common.HexToAddress(to)


var data []byte

// Executing Transaction
tx := types.NewTransaction(nonce, receiver, value, gasLimit, gasPrice, data)
```

Figure 34 - Definition of Transaction Values

As it is a deposit, our wallet is not responsible for signing the transaction. An external signer was used to simulate the signature by the other wallet, as shown in Figure 35.

```
signer := new(types.FrontierSigner)
signedTx, err := types.SignTx(tx, signer, privateKey)
checkError(err)
```

Figure 35 - Transaction Signing by External Signer

Once signed, Figure 36 shows how to send the signed transaction and register it on the network.

```
// Sending transaction amount, in case there is not enough checkError
err = cl.SendTransaction(context.Background(), signedTx)
checkError(err)
```

Figure 36 - Sending Transaction to Ethereum

Figure 37 shows a demonstration of this deposit process using a terminal and Ganache. On the terminal, it is possible to see that the initial balance of the test wallet is 12 ETH and ours is 0 ETH. After the transaction, its hash is printed and the balance of our wallet becomes 1 ETH. Below, the transaction is recorded in ganache, showing its hash, the sender's address, and the receiver's address.

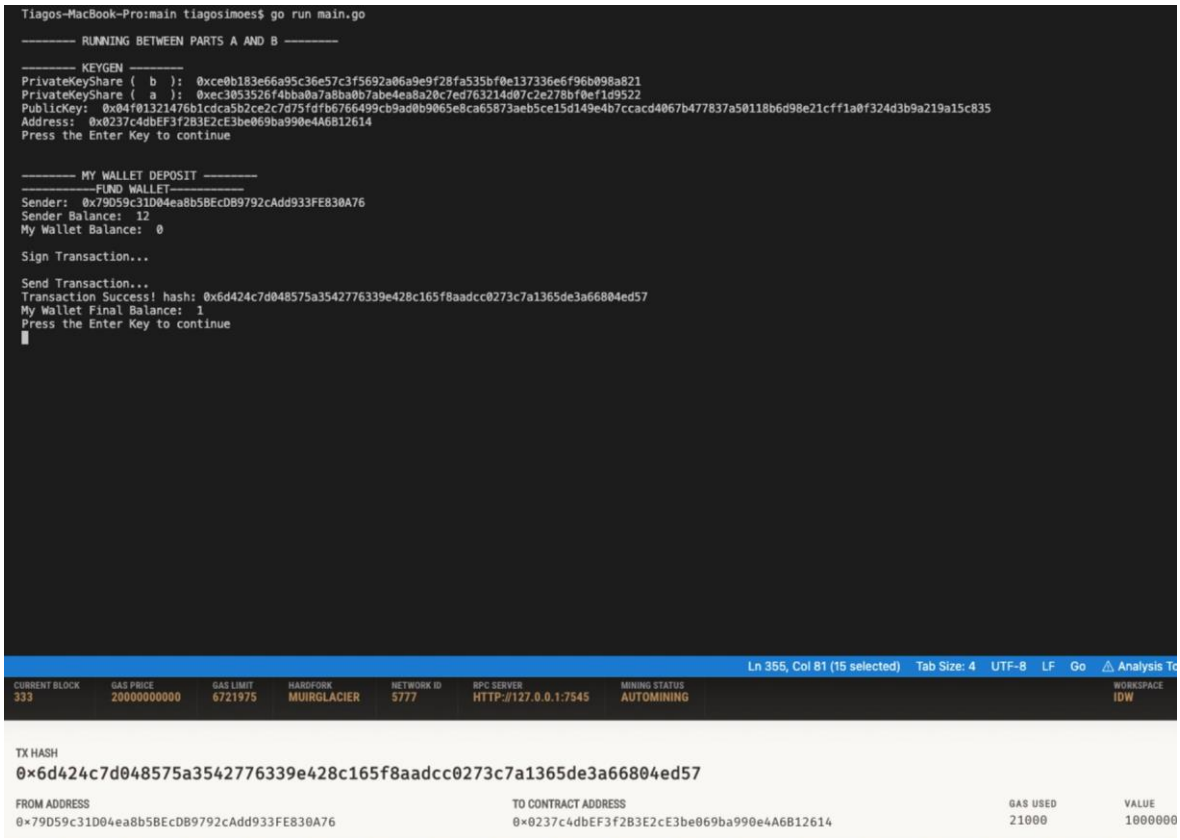

Figure 37 - Ether Deposit Demo

5.4.3. Send Ether Transaction

This feature consists of sending an Ether transaction. Following is a demonstration of how to perform a transaction of 1 ETH to the test wallet created in Ganache.

Similar to a deposit, it is necessary to define all the values belonging to a transaction on the Ethereum blockchain, as is shown in Figure 38.

After generating the transaction data, it is necessary to generate a signature for the hash of that data. Signing a message is a slightly more complex process involving several parties, but essentially device B will use device A's encrypted private key (e(pk1)) to calculate a partial signature (s'). Using this signature, device A calculates the full signature (r, s), as shown in Figure 39.

To achieve this, the “cmp.Sign” protocol is used to generate an ECDSA signature for `messageHash` among the given signers, and then the signature is verified. Figure 40 shows this process.

```
// Calling pending nounce - Preventing double spending attack
nonce, err := cl.PendingNonceAt(context.Background(), sender)
checkError(err)

value := new(big.Int)

// In wei: 10000000000000000, convert ether: 0.01
value.SetString("20000000000000000", 10)
gasLimit := uint64(21000)
gasPrice, err := cl.SuggestGasPrice(context.Background())
checkError(err)

// Painter's account address
receiver := common.HexToAddress(to)

var data []byte

// Executing Transaction
tx := types.NewTransaction(nonce, receiver, value, gasLimit, gasPrice, data)
```

Figure 38 - Definition of Transaction Values

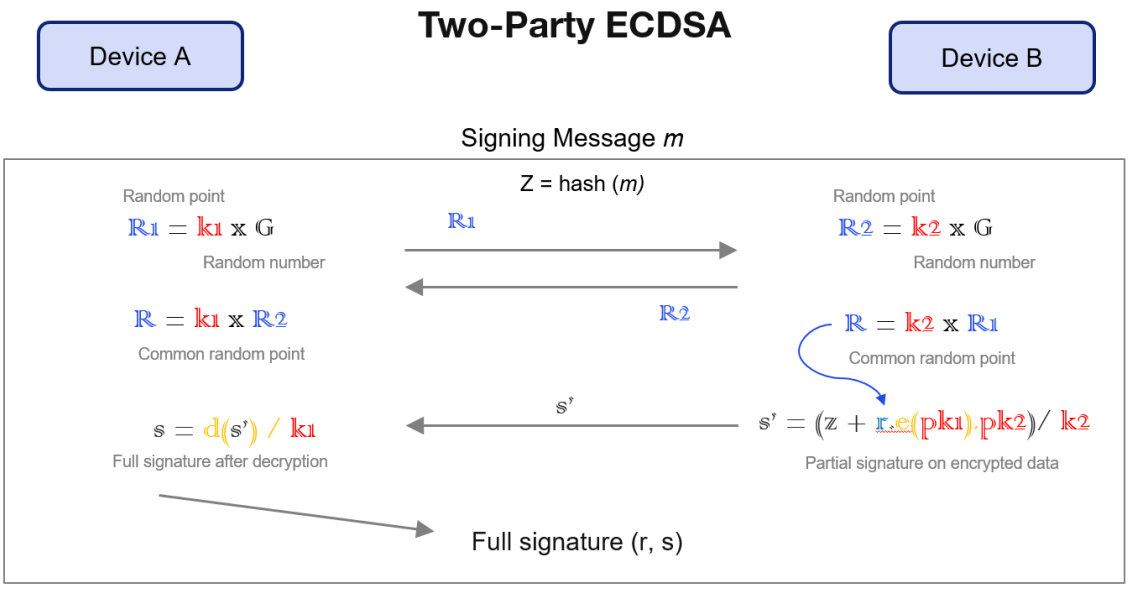

Figure 39 - Two-Party ECDSA Signing

```
h, err := protocol.NewMultiHandler(cmp.Sign(c, signers, m, pl), nil)
checkError(err)
test.HandlerLoop(c.ID, h, n)

signResult, err := h.Result()
checkError(err)

signature := signResult.(*ecdsa.Signature)

if !signature.Verify(c.PublicPoint(), m) {
    return nil, errors.New("failed to verify cmp signature")
}
```

Figure 40 - Generate Signature for Transaction

Once the transaction data and its signature are known, it is necessary to convert this data into a Recursive Length Prefix (RLP) encoding transaction. RLP serialisation is used extensively in Ethereum's execution clients and standardises the transfer of data between nodes in a space-efficient format. The purpose of RLP is to encode arbitrarily nested arrays of binary data.

Without going into too much detail, Figure 41 shows the composition of all the elements of an RLP signature, and Figure 42 shows the assembly of this signed transaction through the code.

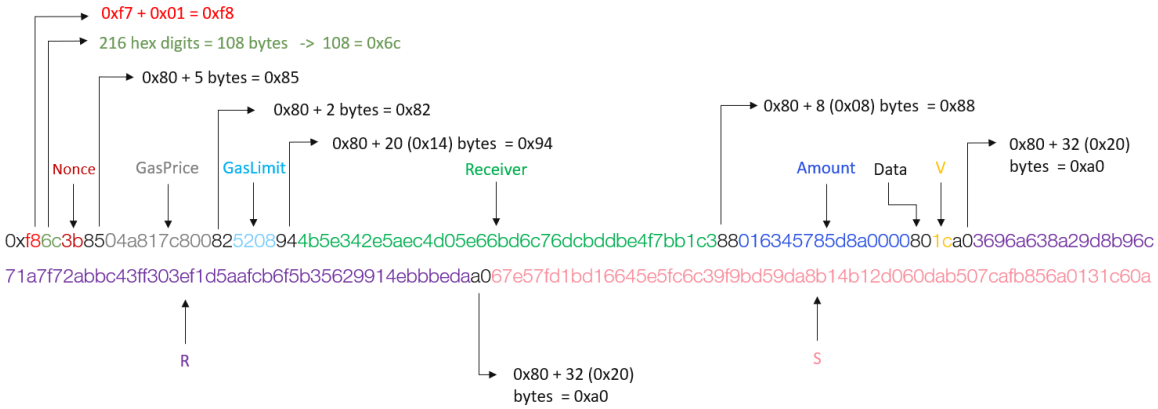

Figure 41 - Signed Transaction RLP Encoding

```
//Add signature to transaction
sigRandV, err := signature.R.MarshalBinary()
checkError(err)

sigR := sigRandV[1:]
sigV := sigRandV[:1]
sigV[0] -= 2

sigS, err := signature.S.MarshalBinary()
checkError(err)

fmt.Println("Signature (V,R,S): ", hexutil.Encode(sigV), " - ", hexutil.Encode(sigR), " - ", hexutil.Encode(sigS))

chainID, err := cl.ChainID(context.Background())

signatureRSV := sigR
signatureRSV = append(signatureRSV, sigS...)
signatureRSV = append(signatureRSV, sigV...)
signatureRS := append(sigR, sigS...)

//SIGNER
fmt.Println("")
fmt.Println("Add Signature to Transaction...")
signer155 := types.NewEIP155Signer(chainID)
signedTx, err := tx.WithSignature(signer155, signatureRSV)
checkError(err)
bw := new(bytes.Buffer)
signedTx.EncodeRLP(bw)
fmt.Println("signedtxRLP: ", hexutil.Encode(bw.Bytes()))
```

Figure 42 - Adding Signature to a Transaction

Figure 43 shows the code in charge of sending this RLP-encoded signed transaction to the Ethereum network.

```
// Sending transaction amount in case there is not enough checkError
err = cl.SendTransaction(context.Background(), signedTx)
if err != nil {
    return err
}

fmt.Println("")
fmt.Println("Send Transaction...")
// Print out the transaction hash if success
fmt.Printf("Transaction Success! hash: %s\n", signedTx.Hash().Hex())
//fmt.Printf("Transaction Success! sender: %s\n", signerSender.Hex())
```

Figure 43 - Send Transaction to the Network

Figure 44 shows a demonstration of this sending process using a terminal and Ganache. On the terminal, it is possible to observe that, before the "Send ETH from my wallet" feature, the balance of our wallet is 1 ETH, and the balance of the test wallet is 11 ETH. As the processes occur, the command line registers the unsigned transaction RLP, the signature for that transaction, and then, the signed transaction RLP which is sent to the network. Ganache shows that the transaction has been recorded and its values match the terminal data.

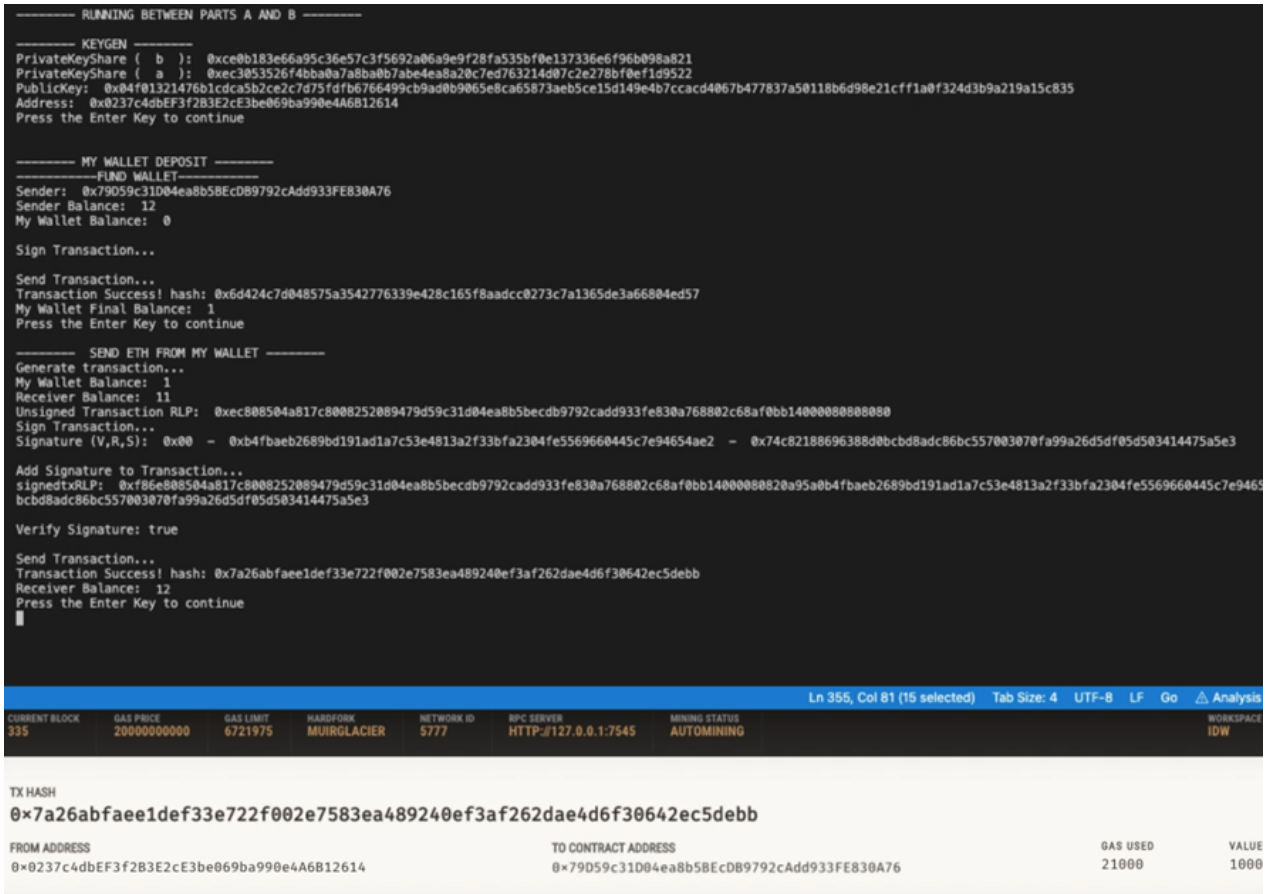

Figure 44 - Send Transaction Demo

5.4.4. Key Refresh

After performing a transaction, both parties communicate with each other to refresh their key shares, thus adding a degree of security to this KMS. As can be seen in Figure 45, the public key remains unchanged.

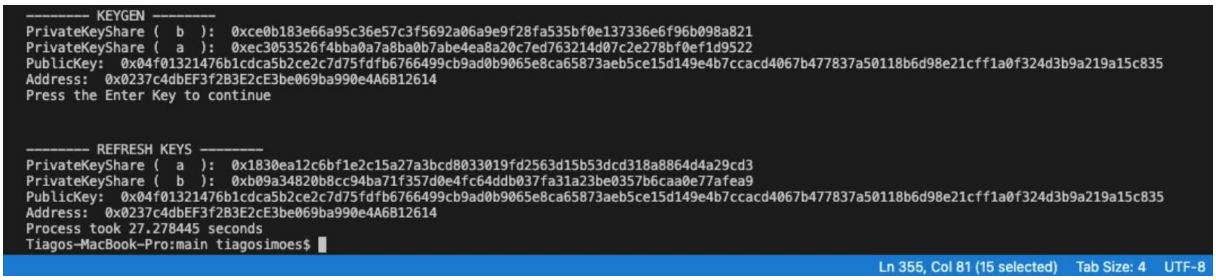

Figure 45 - Key Shares Refresh

6. Conclusion

The internship's initial goal was prototyping a digital wallet that allows operations to be performed on a blockchain while keeping the user's portfolio of digital assets. This implementation was supposed to be guided by two fundamental objectives: to guarantee maximum security and to offer the best user experience so that the wallet can be used by mainstream users. This wallet could later be integrated into any specific solution implemented under a blockchain.

Following an extensive research phase, it was concluded that the project's focus should shift to a more in-depth study of a wallet's private key management systems. This change occurred because I understood that this was the key component of a custodial wallet, which would bring greater value to my solution and possible solutions for future WIT projects in the blockchain market.

This decision led to a new phase of research where some difficulties were encountered in obtaining conclusive information. As the key component of custodial wallets, companies are not very keen to fully share how they manage them. Even so, it was concluded that the most effective method is Multi-Party Computation and proceeded with its integration.

A prototype solution and its development plan were proposed. The solution consists of an API that integrates the Taurus MPC-CMP prototype for key management along with a virtual machine, capable of communicating with the Ethereum blockchain, storing user data in a database, and having a Web App to enable the use of all the wallet features.

The minimum requirement was to implement the Key Management System in a way that allowed key generation and transaction signing to be performed by two different threads using the MPC-CMP protocol. This goal was achieved, and I was able to demonstrate all these processes using the IDE terminal.

Other planned versions remain for future work: integration of a virtual machine to communicate between two separate environments, integration of a relational database, and integration of the frontend component via a Web App.

In conclusion, I believe that my internship at WIT was undoubtedly an enriching experience, allowing me to develop and improve my skills and knowledge at a professional level, especially in this blockchain area.

Several course units in my Master's degree in Mobile Computing were crucial to completing this internship. In particular, the ‘Cybersecurity’ course provided essential knowledge to better understand the many particularities of ensuring security in digital wallet development. I also want to highlight the ‘IT Project Management’ course, which gave me my first experience of managing and coordinating the various tasks involved in a project of this type, which resulted in better time and effort organisation despite the change in focus halfway through the internship. The ‘Software Quality’ course helped to ensure that I was always aware of the importance of implementing well-organised and documented code.

Over the months, I realised that WIT is a company that is highly committed to the blockchain market and, I hope that, by doing this internship, I have contributed to these future solutions, not only due to all the research work that was involved, but also due to the Proof-of-Concept of using the MPC-CMP protocol to manage the keys of a digital wallet.

Also, by the end of the internship, I received a proposal to keep working on WIT Software as a Software Developer in the Field Services project, which was proudly accepted.

Bibliography

[1] B. Wu and T. Duan, “The Application of Blockchain Technology in Financial markets,” *Journal of Physics: Conference Series,* pp. 1-2, 2019.

[2] S. Suratkar, M. Shirole and S. Bhirud, “Cryptocurrency Wallet: A Review,” *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India,* p. 1, 2020.

[3] “WIT Software,” [Online]. Available: https://www.wit-software.com/. [Accessed October 2021].

[4] “Blockchain Facts,” Adam Hayes, [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp. [Accessed October 2021].

[5] S. Al-Megren, “Blockchain Use Cases in Digital Sectors: A Review of the Literature,” *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018,* pp. 1-2, 2018.

[6] S. Saksonova, “Cryptocurrency as an investment instrument in a modern financial market,” *St Petersburg University Journal of Economic Studies,* vol. 35, pp. 269-282, 2019.

[7] G. Habib and S. Sharma, “Blockchain Technology: Benefits, Challenges, Applications, and Integration of Blockchain Technology with Cloud Computing,” *Future Internet,* vol. 14, no. 341, 2022.

[8] “Blockchain Technology Market Size, Share & Trends Analysis Report,” [Online]. Available: https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market. [Accessed October 2021].

[9] S. Jokic, “Analysis and security of crypto currency wallets,” *Zbornik Radova Univerziteta Sinergija v19,* December 2019.

[10] “What is Agile,” [Online]. Available: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/. [Accessed 2022].

[11] “MetaMask,” [Online]. Available: https://metamask.io/. [Accessed October 2021].

[12] “Enjin Wallet,” [Online]. Available: https://enjin.io/technology/wallet. [Accessed November 2021].

[13] “Trust Wallet,” [Online]. Available: https://trustwallet.com/pt-PT. [Accessed October 2021].

[14] “Exodus Wallet,” [Online]. Available: https://www.exodus.com/. [Accessed November 2021].

[15] “MyEtherWallet,” [Online]. Available: https://www.myetherwallet.com/. [Accessed November 2021].

[16] “Coinbase Wallet,” [Online]. Available: https://www.coinbase.com/en-pt/wallet. [Accessed November 2021].

[17] “Crypto.com DeFi Wallet,” [Online]. Available: https://crypto.com/eea. [Accessed November 2021].

[18] “Guarda Wallet,” [Online]. Available: https://guarda.com/. [Accessed November 2021].

[19] “Binance Exchange,” [Online]. Available: https://www.binance.com/en. [Accessed October 2021].

[20] “Coinbase Exchange,” [Online]. Available: https://www.coinbase.com/en-pt/. [Accessed November 2021].

[21] “Meet Dapper,” [Online]. Available: https://meetdapper.com/. [Accessed November 2021].

[22] “Sorare Wallet,” [Online]. Available: https://sorare.com/help/c/360003882958/wallet. [Accessed November 2021].

[23] “Ledger Wallet,” [Online]. Available: https://www.ledger.com/pt-br. [Accessed November 2021].

[24] “Trezor Wallet,” [Online]. Available: https://trezor.io/. [Accessed November 2021].

[25] “Zengo Wallet,” [Online]. Available: https://zengo.com/security/. [Accessed March 2022].

[26] “Geth,” [Online]. Available: https://geth.ethereum.org/. [Accessed December 2022].

[27] “ReactJS,” [Online]. Available: https://react.dev/. [Accessed January 2022].

[28] “Truffle Ganache,” [Online]. Available: https://archive.trufflesuite.com/ganache/. [Accessed February 2022].

[29] “MPC Key Shares Scheme,” Mercuryo Comms, [Online]. Available: https://mercuryo.io/explore/article/mpc-wallets. [Accessed February 2022].

[30] R. Goldfeder and S. Gennaro, “Fast Multiparty Threshold ECDSA with Fast Trustless,” *2018 ACM SIGSAC Conference on Computer and Communications Security, New York, USA,* 2018.

[31] Y. Lindell, “Secure Multiparty Computation,” *Cryptology {ePrint} Archive, Paper 2020/300,* 2020.

[32] J. Doerner, "Secure Two-party Threshold ECDSA from ECDSA Assumptions," *Cryptology {ePrint} Archive, Paper 2018/499,* 2018.

[33] “MPC-CMP Wallet,” [Online]. Available: https://www.fireblocks.com/blog/pushing-mpc-wallet-signing-speeds-8x-with-mpc-cmp-9/. [Accessed February 2022].

[34] “First Open-Source Implementation of MPC-CMP,” [Online]. Available: https://www.taurushq.com/blog/first-open-source-implementation-of-mpc-cmp/. [Accessed March 2022].

[35] A. Donovan and B. Kernighan, The Go Programming Language, Addison-Wesley Professional Computing Series, 2015.