Slovak University of Technology in Bratislava

Faculty of informatics and information technologies

Reg. No.: FIIT-100241-102906

Igor Ďurica

Internet of Value and DeFi in Solana

Bachelor's thesis

Thesis supervisor: Ing. Kristián Košťál, PhD.

May 2023

Slovak University of Technology in Bratislava
Faculty of informatics and information technologies

Reg. No.: FIIT-100241-102906

Igor Ďurica

Internet of Value and DeFi in Solana

Bachelor's thesis

Study programme: Informatics

Study field: Computer Science

Training workplace: Institute of Computer Engineering and Applied Informatics

Thesis supervisor: Ing. Kristián Košťál, PhD.

May 2023

Slovak University of Technology in Bratislava	Faculty of Informatics and Information Technologies
Ústav počítačového inžinierstva a aplikovanej informatiky	Academic year: 2022/2023
	Reg. No.: FIIT-100241-102906

BACHELOR THESIS TOPIC

Student:	**Igor Ďurica**
Student's ID:	102906
Study programme:	Informatics
Study field:	Computer Science
Thesis supervisor:	Ing. Kristián Košťál, PhD.
Head of department:	Ing. Katarína Jelemenská, PhD.

Topic: **Internet of Value and DeFí in Solana**

Language of thesis: English

Specification of Assignment:

DeFi – Decentralized Finance – is the latest buzzword in FinTech, cryptocurrency, blockchain, and digital assets. Nevertheless, very little is known about the implications of this concept for technology as such and the future of financial innovation. Today, we have most of the content in the cloud, on the Internet, so we can call the current period the ”internet of content.” DeFi opens up new possibilities for expansion, so we are gradually moving towards a transformation to the ”internet of value.” Everything online will be a bearer of value, and we will exchange these values with each other, not the asset itself. Analyze DeFi projects (e.g., in Polkadot), their current trends, and applications on the Internet. Design a solution for how decentralized finance could change the technological direction of the Internet and accelerate its transformation into the ”Internet of Value.” Implement the solution in the Solana blockchain and verify the design in the context of similar solutions. Discuss the results obtained and evaluate the impact of your design on the DeFi ecosystem.

Length of thesis: 40

Deadline for submission of Bachelor thesis:	22. 05. 2023
Approval of assignment of Bachelor thesis:	18. 04. 2023
Assignment of Bachelor thesis approved by:	doc. Ing. Valentino Vranić, PhD. – study programme supervisor

I declare on my honor that I have prepared this work independently, on the basis of consultations and using the listed literature.

In Bratislava, 22.5.2023 . . . .

Igor Ďurica

Acknowledgements

I would hereby like to thank Ing. Kristián Košťál, PhD. who was always willing to provide guidance and valuable feedback for the many hours of his time he has dedicated to aiding me in the completion this thesis.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Informatika

Autor: Igor Ďurica
Bakalárska práca: Internet of Value and DeFi in Solana
Vedúci projektu: Ing. Kristián Košťál, PhD.
May 2023

Blokchainové aplikácie dokážu pristupovať k dátam, ktoré sú uložené v rámci blockchainu. Toto platí aj v prípadoch, kedy sa dáta nachádzajú na inom blockchaine ako decentralizovaná aplikácia, ktorá sa k nim pokúša dostať. V momente, kedy decentralizovanej aplikácii potrebujeme poskytnúť dáta, ktoré sú uložené mimo blockchainu však narazíme na problém. Tzv. „oracles“, ktoré slúžia ako dôveryhodné tretie strany, ktoré sprostredkovávajú prenos informácií z centralizovaného webu do blockchainu vznikli ako riešenie pre tento problém. Táto práca sa zaoberá oracle ekosystémom a skúma niektoré z jeho nedostatkov. Primárne nedostatkom rozmanitosti v typoch informácií, ktoré oracles momentálne poskytujú. Väčšina existujúcich oracle systémov totiž poskytuje len dáta o vývoji cien rôznych digitálnych aktív. V našom projekte sme sa teda rozhodli pre iné zameranie. Zámerom tejto bakalárskej práce je navrhnúť a implementovať univerzálne oracle, ktoré používateľom umožní dodať dáta z akejkoľvek centralizovanej API priamo do svojej decentralizovanej aplikácie. Blockchainový komponent tohto projektu bude realizovaný na sieti Solana, zatiaľ čo časti, ktoré budú bežať mimo blockchainu budú napísané v jazyku TypeScript.

Annotation

Slovak University of Technology in Bratislava

Faculty of informatics and information technologies

Degree Course: Informatics

Author: Igor Ďurica

Bachelor's thesis: Internet of Value and DeFi in Solana

Supervisor: Ing. Kristián Košťál, PhD.

May 2023

Applications that live on the blockchain do not have problems accessing data stored on-chain as such, even in cases where the data is stored on a different chain than the app itself. However. when a decentralized application needs data that is stored off-chain, issues arise. Blockchain oracles are trusted third party sources of information and a gateway to the centralized web, that exist as the solution to this problem. This paper delves into the ecosystem of blockchain oracles and explores some of its shortcomings. Primarily in regard to the variety of data they provide, or rather, the lack thereof. Most oracles currently only operate as price feeds for various digital assets. Our project deviates from this use case and instead focuses on giving users the ability to receive data from any off-chain API. The aim of this bachelor's thesis is to design and implement a universal oracle which will allow users to supply data from any external API directly to their decentralized application on-chain. We will be developing the on-chain component of our solution on the Solana blockchain in the Rust programming language, while our off-chain components will be written in TypeScript.

Table of contents

1 Introduction **1**

1.1 Decentralized Finance 2

1.2 Internet of Value 2

1.3 Document structure 3

2 Analysis **5**

2.1 Solana 5

2.2 Blockchain oracles 6

2.2.1 Oracles by data source 7

2.2.2 Oracles by design pattern 8

2.2.3 Oracles by trust model 8

2.3 An overview of existing oracle technology 10

2.3.1 Hardware based centralized oracles 10

2.3.2 ChainLink 11

2.3.3 Optimistic Oracle (UMA Protocol) 12

2.3.4 Band Protocol 13

2.3.5 Airnode Protocol (API3) 14

2.4 Oracle solutions in Solana 15

2.4.1 Pyth Network 15

2.4.2 Umbrella Network 16

2.5 The Oracle Problem 17

2.5.1 Sybil attacks 18
2.5.2 Front-running 19
2.5.3 Miscellaneous 20
2.6 Summary 20

3 Proposed Solution 22
3.1 Chosen consensus mechanism 23
3.2 Solution structure 23
3.2.1 Oracle smart contract 24
3.2.2 Oracle node 24
3.2.3 Web application 25
3.3 Functional requirements 26
3.4 Non-functional requirements 26

4 Implementation 29
4.1 Oracle smart contract implementation 29
4.1.1 Accounts 30
4.1.2 Instructions 30
4.1.3 Consensus 31
4.2 Web application implementation 32
4.2.1 Home page 33
4.2.2 New subscription page 33
4.2.3 All subscriptions page 34
4.3 Oracle node implementation 34

5 Evaluation 36
5.1 Testing and input validation 36
5.2 Cost analysis 37
5.3 Performance 39

5.4 Security concerns 39
5.5 Other limitations 40
5.6 Evaluation 40

6 Conclusion 42

6.1 Future Work 42

7 Resumé 44

7.1 Analýza 44
7.1.1 Blockchainové oracles 45
7.1.2 Rozdelenie oracles 45
7.2 Existujúce blockchainové oracles 46
7.2.1 ChainLink 46
7.2.2 Oracles na sieti Solana 46
7.3 Návrh 47
7.3.1 Oracle smart kontrakt 47
7.3.2 Oracle sieť 47
7.3.3 Webová aplikácia 48
7.4 Implementácia 48
7.4.1 Oracle smart kontrakt 48
7.4.2 Webová aplikácia 48
7.4.3 Oracle sieť 49
7.5 Evaluácia a záver 49

References 50

A Project task schedule

B Technical documentation

C Contents of the digital medium

1. Introduction

The blockchain is a relatively modern technology, which was first successfully implemented with the launch of the bitcoin cryptocurrency in 2009. It has experienced considerable growth in the past decade however and now makes up a vast ecosystem. Most blockchains effectively function as distributed ledgers consisting of a network of independent nodes, the purpose of which is to process transactions [1]. These transactions are then stored into chunks of data commonly referred to as blocks. Blocks are appended to the end of a chain in the order in which they were created, with each block typically containing a hash of the previous one. Every block is immutable, meaning that once it is appended to the chain, no further modifications to the data within it are possible [2], [3]. This chain contains the complete history of all transactions recorded on a given blockchain, with each transaction being public and verifiable. Nodes use various types of consensus algorithms to agree upon which sets of transactions are valid and should be included in a block. This in turn removes the need for third-party authorities that validate transactions such as banks in centralized financial systems [1].

1.1 Decentralized Finance

De-Fi or decentralized finance expands the original concept of simple transactions between two parties to more complex financial operations while still utilizing the blockchain technology. De-Fi offers permissionless access as well as anonymity to anyone while also being completely transparent with every transaction being publicly available. This absence of third-party regulation comes with a multitude of drawbacks and possibilities for new forms of financial crime unique to the ecosystem. One of the most pressing concerns being the lack of accountability on the side of the perpetrators who can operate anonymously without a fear of consequences unlike in centralized financial systems [4], [5].

Whether one chooses to accept De-Fi as a worthy counterpart to traditional financial markets, its growth and overall impact has been undeniable. In just 2 years, the total value locked inside of De-Fi projects across different blockchain platforms has risen from $700 million to a staggering $150 billion as of April 2022 [4]. Common examples of De-Fi projects include decentralized exchanges that allow users to trade tokens or earn additional tokens by providing liquidity to one of many pools. Lending protocols which allow users to borrow large amounts of assets, typically by putting up collateral or guaranteeing that they can pay the loan back within the span of one transaction - see flash-loans. De-Fi is also supports equivalents of more traditional securities commonly traded in centralized financial markets such as derivatives [6], [7].

1.2 Internet of Value

Decentralized finance and blockchain as a whole both play a key part in a concept known as the Internet of Value. Currently, when a user purchases a digital asset, they are typically receiving a copy of the asset. The buyer however does not receive the asset itself which potentially allows the seller to sell the same asset multiple times e.g., purchasing piece of digital art. To receive ownership of the asset, a third party is often required to verify and transfer the proof of ownership [8]. Another common concern is that geographic location can often be a constraint when exchanging assets between two parties. For reference, the cost of a cross-border payment can be up to 10000 times higher than that of a transaction of the same value in the blockchain [9], [10].

Internet of value specifically refers to a system where this transfer can happen seamlessly between as little as two peers located anywhere in the world, without any requirement for the involvement of another party. In such a system, value is transferred as fast as information is on the internet today, all thanks to the capabilities of the blockchain. These exchanges do not have to be limited to financial assets, any asset with an underlying value attached can be exchanged, hence the name Internet of Value [8], [11], [12].

1.3 Document structure

In the following chapter of this document, we will be focusing on a single subset of the De-Fi ecosystem - blockchain oracles, and conducting a detailed analysis. We will be covering the basics of what oracles are, why they are needed as well as their different types, variations and how they can be categorized. We will then delve further into an overview of current oracle technology with a focus on exploring as many varying approaches as possible. The subsequent section will be focused on the shortcomings and vulnerabilities of oracles in addition to the common challenges inherent to building oracle networks. The following portion of this thesis will then be focusing on the design, implementation and testing of a solution with the end goal of providing a meaningful contributing to the blockchain oracle ecosystem. The remaining sections will contain the evaluation of our implementation and a conclusion based on our findings followed by a short resumé of the thesis written in the Slovak language.

2. Analysis

In its current state, the existing blockchain infrastructure is not enough to support worldwide use of the IoV and will require the support of additional technology to overcome existing challenges. One such challenge is the blockchain scalability problem. As things stand, most major blockchains could hardly replace established payment systems, let alone facilitate seamless exchange of assets in IoV, due to their low transaction per second rate and relatively high transaction fees. This comes as a result There are however promising alternatives.

2.1 Solana

Launched in 2020, Solana is a new blockchain platform, promising a TPS rate of up to 50000 in addition to minimal transaction fees, which would potentially enable it to rival conventional payment systems such as VISA or MasterCard. For reference, the TPS of the Ethereum blockchain is more than 3000 times lower at 16. Much like other established blockchains, Solana also supports on-chain smart contracts. These contracts are written in Rust, C or C++, all of which can be compiled to Low Level Virtual Machine (LLVM) bytecode, resulting in fast execution times. This alone however would not be enough to push Solana's performance to its impressive heights. A key component of Solana's success is its

Proof of History (PoH) consensus mechanism. In PoH a sequence of transactions is formed by using a new transaction and a hash of the previous transaction as inputs. These inputs are used to generate a new hash which will then be used as the input for the next transaction. The chain keeps track of the given hash values as well as their corresponding transaction counts. This creates a verifiable order of transactions in the passage of time. All that is necessary to verify this sequence is to check the individual hashes and to check whether accounts have sufficient balance to complete transactions. This process is significantly faster than a Proof of Work mechanism. It is not hard to see why Solana has a lot of potential, however scalability is not the only challenge IoV currently faces [2] [13].

2.2 Blockchain oracles

As the De-Fi ecosystem grows and expands, so does the need for external sources of data. One of the major downsides of decentralized applications is that they can generally only access data stored on their respective blockchains. This poses a problem because many services require outside sources of data to function, e.g. digital currency exchanges which rely on external APIs for exchange rates. This is where blockchain oracles come into play. An oracle acts as a trusted third party which collects, verifies, and aggregates data before providing it to the smart contract. A major part of this process happens off-chain, however most oracles tend to be linked to an on-chain oracle smart contract. This contract serves as an interface for other contracts on the chain to interact with and use to query data from off-chain sources. It is also important to understand that oracles themselves are never the source of the information they relay to smart contracts, rather just a method for its transmission into the blockchain. This transmission doesn't necessarily have to be inbound (data being supplied to the blockchain), it can also be outbound (data being supplied to the oracle) [14].

There are multiple factors by which we can categorize different oracle projects. The three that we will primarily focus on are the source of the oracle's information, the design pattern of the oracle and finally the model of trust they use.

2.2.1 Oracles by data source

Software oracles make up the majority of blockchain oracles and often operate on simple request/response or data feed models. They query information from websites, servers, or databases available on the web. An example use of a software oracle is the aforementioned checking of currency exchange rates via a web API.

Hardware oracles use sensors and trackers to gather information based on physical changes or interactions in the real world. Hardware oracles could be utilized in supply chain tracking by informing a smart contract when an item is picked up or reaches its destination. Any decentralized application designed around Internet of Things will likely rely on hardware oracles as well. Computation oracles exist specifically to perform computationally intensive tasks that would be too expensive to perform on the blockchain. These tasks are instead performed by the off-chain oracle and then relayed to an on-chain smart contract in order to avoid paying high gas fees [15].

Human oracles are experts tasked with observing and evaluation information in the real world. Human oracles could for example be tasked with verifying a piece of art that is represented by a digital token in the blockchain or giving the verdict on whether an even did or did not occur in the real world [16].

2.2.2 Oracles by design pattern

Request-response oracles are used when the set of data is too large to feasibly be stored on-chain and users also do not ever need to retrieve more than small subsets of said data at a time. For example, data can be stored in an off-chain database which can then be queried by the smart contract through an oracle. This is done as an alternative to storing the contents of the database in the contract and accessing it directly, which greatly improves scalability by allowing significantly more data to be stored.

Publish-subscribe oracles provide an on-chain contract with a stable feed of data that is subject to frequent change. Every time the oracles records a change in this data, it publishes the updated version to its subscribers. This type of oracles is commonly used by decentralized exchanges which need accurate and up to date information on prices of digital assets.

Immediate-read oracles provide data that is required to execute an operation, such as an authenticity check on a legal document or an artwork [16].

2.2.3 Oracles by trust model

Centralized oracles obtain data from a single source and transmit it directly into the on-chain contract. These solutions tend to be faster and more efficient as they only require a single node but are also more susceptible. A potential attacker only needs to compromise a single oracle to provide faulty data to the smart contract [16].

Decentralized oracles collect data from multiple sources and typically rely on some type of a consensus mechanism to determine what data should be sent to an on-chain oracle contract. Examples include a K-out-of-N system, where at least K oracles out of the total N oracles need to agree on the data for a consensus to be reached or systems where individual nodes can stake tokens to challenge the agreed upon result. While decentralized networks of oracles tend to be more secure, they are also slower due to the time it takes for a consensus to be reached. They are also more hardware intensive given the fact that they need to operate multiple nodes [17]. Decentralized oracles can further be split into two defining categories based on what trust mechanism they utilize to ensure the security of the network: voting-based and reputation based [18].

Voting-based oracle systems are designed to combat discrepancies in reported data or potential malicious misreporting by letting users certify data or vote on its validity. To participate in this process, users have to put up a stake. If the agreed upon outcome matches the one the user proposed, they are rewarded. If the opposite is true, they are penalized, typically by losing a portion of their stake. The more oracles participate in such system, the more secure it becomes [18].

In **reputation-based** systems, each oracle is assigned a reputation score which is a reflection of its performance. When oracles truthfully report data, they are rewarded and gain reputation. When they report incorrect data or do not report when expected to, they lose reputation. The rate at which reputation is gained is typically much lower than the rate at which it is and losing enough reputation leads to complete removal from the network. The primary deterrent is the opportunity cost of not being able to earn rewards via participating in the network because oracles with a higher score are more likely to be selected than those with a lower one [19].

2.3 An overview of existing oracle technology

Given the necessity for outside data in the blockchain, it should come as no surprise that there are many commercial oracle providers on the market as well as decentralized applications operating oracles tailored to their own needs. While there are too many different oracle services to cover, the aim of this section is to showcase a variety of existing mechanisms that can be used to ensure the accuracy of data provided by oracles.

2.3.1 Hardware based centralized oracles

It should be noted that while only decentralized oracles will be discussed, centralized alternatives do exist as well, albeit with certain downsides in regard to ease of implementation, data availability and hardware requirements. Centralized oracles also introduce a single point of failure as a result of only using one node. Examples of centralized oracles include Town Crier or Provable. Provable uses TLS-Notary to generate a proof of authenticity, ensuring that the data provided to a smart contract is the same data that was received from an external data source [17]. Town Crier on the other hand utilises the Intel Software Guard Extensions or SGX to ensure the integrity and confidentiality of data using trusted hardware. Intel CPU's with SGX capabilities use a special set of instructions to run programs in an isolated part of the CPU known as the enclave. Other programs on by the CPU cannot access the enclave's memory effectively creating a secure execution environment [20].

2.3.2 ChainLink

ChainLink is an oracle provider with full support for decentralized oracle networks on both the data source and oracle level, meaning it aggregates data from multiple oracles querying multiple data sources. All aggregation of data is performed by the off-chain, components of the system, meaning that only a single transaction is required to publish this data onto the blockchain. Delegating most computations to the off-chain network of oracles results in a significant decrease in overall gas fees. ChainLink utilizes the Off-Chain Reporting protocol or OCR for generation of reports based on data from third party sources. OCR is composed of 3 sub-protocols [21].

The **pacemaker protocol** serves to divide the process into time intervals referred to as epochs. Each epoch has a leader whose task is to generate a report based on data from the other oracles. All oracles have an internal timer which is reset every time the leader emits a progress event on the report. If enough oracles declare that the timer has run out, a new epoch is started, and a new leader is elected [21], [22], [23].

The report **generation protocol** divides epochs into rounds of gathering observations from oracles. The frequency of these rounds is ultimately decided by the leader of a given epoch. Once a set number of observations with valid signatures is reached, the leader may generate and sign a report and submit it to the transmission protocol [22], [23].

Once a report is handed off to the **transmission protocol**, it is distributed to each oracle in the network. Oracles then verify the data and sign it if they approve of the reported values. To prevent unnecessary reporting, data can only be submitted to the smart contract if the median value of observations in the report has changed by a large enough margin since the last reported value or enough time has passed. This is done to prevent identical reports from successive rounds from being reported. If this condition is met and enough nodes have signed the report, a pseudo-random function is used to select a set of nodes. These oracles will attempt to transmit the report to the on-chain smart contract. Each transmitting oracle is assigned a delay to ensure a scheduled sending process. If an oracle manages to publish the report to the blockchain, the process ends, otherwise the next oracle in the schedule is selected [22], [23].

As far as security is concerned, ChainLink utilizes standard public key encryption and digital signatures for authenticity. Oracles themselves operate on a reputation-based model of trust, where oracles are rewarded and or penalized based on their reputation. Reputation is calculated based on multiple factors such as the total number of accepted assignments, completed assignments, the average response time as well as how many times the oracle has been penalized for misreporting in the past. If an oracle behaves maliciously, its reputation will drop resulting in an eventual removal from the network as well as the potential removal of other oracles owned by the operator of said oracle. Instances of malicious behaviour also undermine the trustworthiness of the entire network. This can potentially decrease the value of the native LINK token which is the primary method of compensation for oracle operators [16].

2.3.3 Optimistic Oracle (UMA Protocol)

Universal Market Access, also known as UMA is an Ethereum based protocol which utilises the "optimistic oracle" mechanism which serves as a price feed for De-Fi platforms in the following manner. First a contract requests the price for a digital asset and sets a dispute period. A proposer then proposes a price for this asset while also putting up a proposal bond. If the dispute period expires without a dispute being raised the proposed price is submitted to the requester contract. The oracle is optimistic because the proposed information is expected to be correct in the majority of scenarios. In the event that a user wishes to, they can raise a dispute by proposing a differing price and putting up a bond of their own. This dispute is then handled by UMA's DVM or Data Verification Mechanism [24].

When a dispute occurs, the DVM holds a vote that UMA token holders can participate in by deciding whether the proposer's or the disputer's price is correct. Votes are then automatically aggregated, and a result is reached. If the original proposed price is voted to be correct, the disputer loses their bond, while the proposer receives their own bond as well as half of the disputer's bond. In the event that the disputer wins the vote, the opposite happens. Regardless of the outcome, the requester of the data is compensated for the dispute [25].

To potentially manipulate the outcome of a vote, a malicious party would have to own a majority of the UMA tokens in circulation, also known as a 51% attack. The system is designed in a way where the cost of attacking an oracle would outweigh the profit that can be gained from it [25].

2.3.4 Band Protocol

Band protocol or Bandchain is a decentralized oracle service that allows users to issue requests to web API's. It is also designed to be fully permissionless and has cross-chain capabilities. Band operates on a mechanism called delegated proof of stake. This system divides consists of 3 roles on the network: providers, delegators, and validators [19].

Providers simply create the means of fetching data from an external data source.

To be selected as a validator, a user first needs to stake their Band tokens. The selection of a **validator** is pseudo-random with the chance of being selected directly proportional to the number of tokens staked. The role of the validator is to perform a request for data from a provider on behalf of a client using the oracle service. The validator then has to verify this data, publish it to the chain and generate a cryptographic proof. Successful validations are rewarded with tokens while malicious actions by validators are penalized by subtracting a fine from their stake. Delegators stake their Band tokens to vouch for the validity of data in a given block. **Delegators** receive a commission from the validation reward once the block is accepted [19], [23].

2.3.5 Airnode Protocol (API3)

Previously listed examples all had mechanisms in place to ensure that oracles were truthfully reporting information from external data sources. API3 takes a different approach and proposes the idea of "first-party" oracles. API3 suggests that we can eliminate the need for third-party oracles services if API owners operate their own oracle in addition to their standard API. API3 hope to accomplish this using their very own Airnode Protocol [26].

The aim of Airnode is to serve as a lightweight cloud-based, API gateway that wraps around an existing API. It is meant to give API providers the means to host their own oracle while abstracting away the technical challenges of designing and operating a traditional oracle node in addition to reducing associated overhead. Airnode is designed to function without requiring additional maintenance after deployment in the cloud. The protocol functions on a pay-per-request basis with the requester covering all associated gas fees on-chain. Despite still being in development and not being available for commercial use, API3 and Airnode provide a non-traditional approach to blockchain oracles as well as a new perspective [27].

2.4 Oracle solutions in Solana

Due to its high throughput, fast transaction speeds and minimal transaction fees, Solana would seem as the ideal platform for oracle services. However as of November 2022, there are only a handful of commercial oracle providers on this blockchain. The aforementioned Chainlink protocol operates an oracle service on Solana, providing live price feeds for select digital assets. According to the company's listing on Solana's official community ecosystem page, data feeds can deliver updates as often as every 400 milliseconds, thanks to Solana's high-performance infrastructure. Aside from Chainlink, there are 2 other major oracle providers on Solana. First being the Umbrella Network and second, the Pyth Network.

2.4.1 Pyth Network

The Pyth network provides price feeds, using a form of a delegated proof of stake algorithm. The Pyth protocol runs in epochs, with each epoch lasting for the duration of a single week. Users participating in the protocol are divided into 3 groups [18], [28].

Consumers simply read data from price feeds but unlike in other protocols, they have the additional option to pay a data fee. Fees are one-time payments which serve as insurance for the consumers and are tied to a period spanning 4 epochs. If any of the data provided to consumers during these epochs turns out to be inaccurate, consumers receive a pay-out from delegators [18], [28].

Delegators can stake their PYTH tokens to choose to back an aggregate price.

Additionally, they also have to choose to back a publisher who contributed to creating this price. The weight of a publisher's stake is then determined as the sum of their own stake and the delegators' stakes. Delegators aren't betting on a given publisher however, rather on the aggregate price. This incentivizes different delegators to distribute their stakes across multiple publishers [18], [28].

Publishers publish price data as well as their confidence interval for this data, which determines how much weight is attributed to it during the aggregation process. This system is also designed to be resilient against malicious reporting by publishers and assigns smaller weights to reported values depending on how far they deviate from the median. Publishers are also required to stake their PYTH tokens which will in turn be slashed if they report inaccurate data [28].

Much like in other delegated proof of stake mechanisms, users can stake tokens to challenger the accuracy of reported data, at which point a vote take place. If the vote determines that the data was inaccurate, delegators who backed it as well as publishers who reported incorrect price data will be penalized by losing a portion of their stakes. These tokens are then used to compensate consumers who paid data fees. If the reported data is accurate, these data feeds are distributed among delegators and publishers in proportion to their stakes instead. All tokens are distributed at the end of an epoch [28].

2.4.2 Umbrella Network

There is only a limited amount of information available on the inner workings of the Umbrella Network. According to the official whitepaper, the Umbrella protocol relies on a standard delegated proof of stake algorithm for consensus, similar to that used by the Band protocol, described in section **3.4.**. The Umbrella Network hopes to achieve high scalability through aggregating data off-chain and submitting it in a single transaction. The network also promises higher security with data submitted on-chain being verifiable using Merkle Trees, in which each leaf represents a single value reported by an oracle node [29].

2.5 The Oracle Problem

Blockchains are designed as isolated, trust-less environments with a strict set of validation rules and a consensus mechanism, which ensures that all data on the chain can be verified. However as soon as we introduce any data that did not originate on the chain, we also re-introduce the need for a trust model, seeing how there is no reliable way for the blockchain to verify outside data. Another concern is that if a smart contract depends on a single, centralized source of information, it can no longer be considered decentralized.

This is referred to as the oracle problem. It effectively introduces a major vulnerability in an otherwise secure system and affects any decentralized application reliant on data from a third party [23]. Oracles act as a single point of failure, they can be hacked or otherwise manipulated and cannot operate on a secure and trustless basis. This means that additional mechanisms such as authenticity proofs or economic security need to be put in place to disincentivize attacks and avoid easily compromising the applications oracles supply information to [18]. The oracle problem can be divided into 2 distinct dimensions, technical and social.

The **technical dimension** refers to inherent flaws in the design of an oracle or technical issues that may arise during runtime. Examples include unintentional misreporting of data as a result of computation errors or a lack of stability or availability on the side of the oracle provider. A partial solution to the technical portion of the problem already exists in the form of decentralized networks of oracles. They are significantly more resilient and can continue to operate even if one or multiple nodes temporarily cease to function [30].

The **social dimension** focuses on the intentional manipulation of data supplied by an otherwise functioning oracle. This manipulation can happen as a result of an outside attack as well as a deliberate decision by the oracle operator. This can be especially harmful to the oracle network if the provider controls a large number of oracles within it [30]. Given the prevalence of decentralized oracle systems and their relative stability, we will primarily be focusing on the social dimension of the oracle problem in the remainder of this section.

2.5.1 Sybil attacks

Similar to other decentralized platforms, oracles too are susceptible to Sybil attacks. There are multiple types of Sybil attacks but they are all share a common characteristic. A malicious party within the network also referred to as the Sybil, who either duplicates their vote across multiple nodes or manipulates other participants to vote the same as them in order to increase the weight of their vote [30].

Mirroring and freeloading attacks are similar in nature, and both constitute a minor threat to a decentralized oracle network. Both attacks involve “lazy” oracle operators, who try to bypass the process of acquiring or validating data in order to minimize operation costs. Additionally maximizing the chance of receiving a reward for providing data by sending data that is more likely to be selected as the agreed upon truth by the consensus algorithm. A mirroring attack requires the oracle operator to control multiple oracles. After querying and validating data on a single attack, they then mirror the value across all other oracles. A freeloading attack on the other hand can be performed with any number of oracles. It involves copying of the data provided by another oracle in the network. Both practices decrease the diversity of reported data inside the network, effectively decreasing the accuracy of reports in the long term [23]. Malicious oracle operators will inevitably end up controlling a larger share of the network in the long term. This comes as a result of honest oracles not being able to compete with significantly lower operation costs of the dishonest ones [30], [31].

The last common type of Sybil attacks are so called majority attacks, sometimes referred to as 51% attacks. They occur when a malicious party gains control of the majority of nodes inside of the network. If successful, this type of attack completely subverts the voting mechanism present in the network, giving the attacker full control and allowing them to pass any vote of their choosing. While the most destructive, these types of attacks are also the least common and easiest to prevent against. There are a multitude of existing mechanisms that can reliably ensure that the cost of a majority attack would outweigh the overall benefit gained from it [19], [30], [31].

2.5.2 Front-running

Given the nature of how oracles operate and the fact that they are often used to supply price data to De-Fi applications it is immediately apparent how they could be utilized in a front-running attack. An oracle provider has access to price data before it is registered and updated on a De-Fi platform. They can then use this data to estimate the new price with varying levels of accuracy, depending on how large of a share of the network they control. Having access to both the current and the future price of a digital asset allows them to make profitable transactions at the expense of another party. A notable attack of this nature took place in 2019 on the Terra platform. An attacker took advantage of only a 2% difference between the price of Terra coins on the platform itself and the price reported by its oracle that was pending confirmation [30].

2.5.3 Miscellaneous

While some issues are intrinsic to blockchain oracles, extrinsic issues do also exist. Instead of attacking an oracle network itself, attackers can divert their attention towards manipulating the sources of data for oracles themselves. This type of tampering can often go unnoticed until it is too late and can only be reliably prevented by the diversification of data sources instead of relying on a single one [23]. Issues don't even necessarily have to be caused by the off-chain component of the oracle network, but rather by the technical limitations of the chain they provide data for. Scalability of oracle systems in regard to the frequency of reporting as well as amount of data being reported can be severely throttled by high transaction costs of a given blockchain. Transaction speed is also a factor and data could potentially be outdated by the time it reaches the on-chain oracle smart contract. It is worth noting however, that another factor contributing to this could also be the time it takes to validate data off-chain [19].

2.6 Summary

As the De-Fi ecosystem grows, so does the need for reliable sources of outside information. There are countless ways to build blockchain oracles and it is impossible to cover every existing solution within the scope of this thesis. That being said, we are confident that the analysis provided in this paper contains all the foundational knowledge required to understand the main approaches to building oracles and the numerous challenges that it entails. The aim was to cover the most relevant or promising projects as well as to highlight what their strengths and weaknesses are. No two solutions are the same and each one brings something different and unique to the table. However, it is abundantly clear that the vast majority of existing oracle projects were made with the same specific focus in mind. Every oracle covered in this paper thus far had one thing in common, the type of information that it supplies to the network.

3. Proposed Solution

A shared characteristic among most oracle projects analyzed in the previous sections of this paper is their primary purpose as providers of price data feeds. While this use case certainly makes the most sense in the world of De-Fi, it is not the only meaningful one. Hence, we have decided to explore other means through which blockchain oracles could provide outside information to the network. The next logical choice would be oracles that report on events that happen in the real world, such as the weather or the outcomes of sports matches. The main issue with such use cases however, is that they might be too specific and might not accommodate the needs of enough potential customers to warrant their existence. This is why we have decided to instead focus on a universal oracle, that gives users freedom to decide exactly what information they need for their decentralized application and how often they need it. Users will simply input a URL for their API as well as any necessary query parameters and HTTP headers, in addition to the length of the period they wish to make API calls for and the frequency at which they want these calls to happen.

It is worth noting that the use of blockchain oracles for supply chain tracking is also a very relevant subject and a great candidate for further research, however it is a topic that exceeds the scope of this thesis. The primary issue being the lack of access to hardware required for a functioning implementation as well as numerous logistical difficulties in addition to the overall complexity of such a system.

3.1 Chosen consensus mechanism

As previously discussed, oracles are a major vulnerability because they introduce a single point of failure to the network. To mitigate this, a mechanism needs to be put in place to help ensure the integrity and correctness of data. As a part of this paper, we have decided to develop a decentralized network of multiple oracle nodes, that will rely on a Proof of Stake (PoS) mechanism for consensus.

This type of solution was chosen as a result of multiple factors. First, the much higher prevalence of existing decentralized software-based solutions. Second, the overall amount of available academic literature and research documenting various approaches to these types of solutions as opposed to their centralized counterparts which generally rely on hardware for proof. Lastly, we do not have access to a CPU with support for the Intel Software Guard Extensions, which is the most common hardware-based authenticity proof mechanism used in oracles.

3.2 Solution structure

Our solution will be comprised of multiple components, each with their own distinct function. The on-chain portion will be a single decentralized application whose purpose will be to process requests from clients as well as to receive data from oracle nodes. The off-chain portion includes the implementation for oracle nodes themselves and the web application that clients will interact with directly.

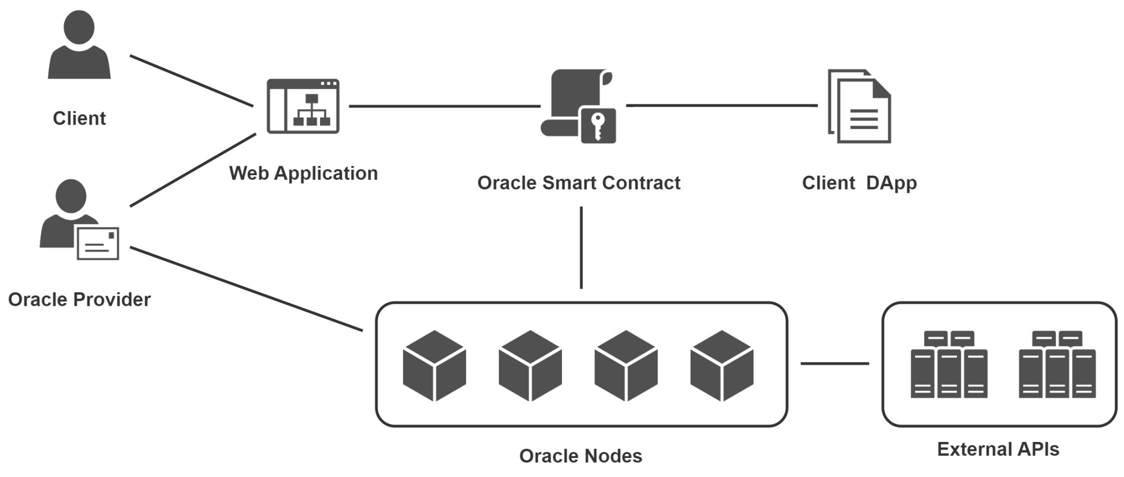

Figure 1: Solution structure visualization

3.2.1 Oracle smart contract

The oracle smart contract represents the on-chain part of our solution. It keeps track of accounts for both the customers of the oracle service as well as the oracle nodes themselves. Customer accounts contain information about specific API's they wish to receive data from. Oracle node accounts store their respective stakes. The purpose of the smart contract is to oversee the operation of the oracle network and the data collection process. It relays information about what data to fetch to the oracle nodes and selects leaders whenever necessary. After data is successfully received from the oracle network, it can be accessed in its respective account by the customer.

3.2.2 Oracle node

Oracle nodes are responsible for the collection and aggregation of data. They communicate with the oracle smart contract as well as amongst themselves. The smart contract provides them with all necessary information in regard to what API calls to make, as well as notifying oracles when a new cycle starts, and a new leader is elected. Whenever oracles query an API, they relay the information to the leader of the current cycle and await the leader's response containing the version of collected data chosen to be sent to the on-chain contract. At this point oracles can either accept or reject this data before it is sent to the oracle smart contract.

3.2.3 Web application

The part of the solution that a customer will interact with to manage their subscriptions (a subscription refers to a series of rounds of data collection from an API by the oracle network on behalf of the customer). A simple web application containing a form page and an overview page. The overview page will contain a list of all ongoing and expired subscriptions for a given user. Each entry will also contain some basic information such as the name of the API and the date of expiration for each subscription. The form page will allow users to purchase new subscriptions. First users will fill out a form containing fields for all necessary information. This page will also contain a display with the price of the subscription, calculated based on the total number of calls. Upon entering all the necessary information users will be able to finalize their subscription by paying. The subscription becomes active immediately after payment is received by the oracle smart contract.

The following input fields will be present on the form page:

Subscription duration - single numeric input field for the total number of API calls a customer wishes to make, calls will be made approximately once per minute

API URL - single text input filed for the URL of the API that will be queried by the oracle nodes

Parameters - single text area input that accepts any valid JSON object containing various parameters required for querying the API specified above.

3.3 Functional requirements

1. Users should be able to create a subscription for any amount of time, provided they can cover the cost.
2. Users should be able to cancel existing subscriptions.
3. The application should be able to query any API accessible via the web, assuming all necessary information has been provided by the user.
4. The application should be able to display all existing API subscriptions associated with a given account if any exist.
5. The application should be able to relay information from outside API's to other smart contracts on-chain.
6. The oracle smart contract should verify the correctness of reported data.
7. The oracle smart contract should penalize the current leader and elect a new leader in the event of incorrect data being reported.

3.4 Non-functional requirements

1. The UI of the web application should be simple and intuitive.
2. The web application should display accurate and up-to-date information about existing subscriptions.
3. The application should enable the access to outside data on the Solana blockchain.

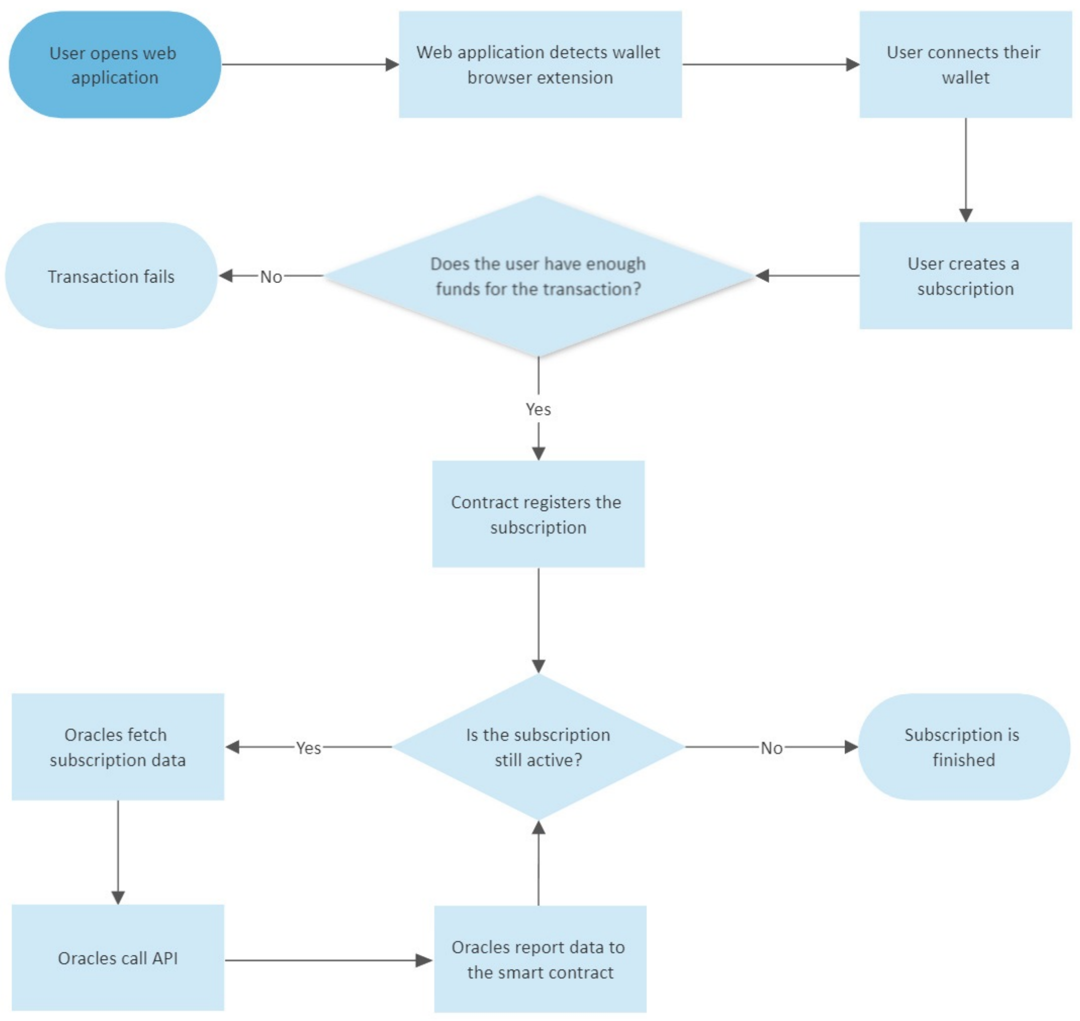
User opens web application
Web application detects wallet browser extension
User connects their wallet
User creates a subscription
Does the user have enough funds for the transaction?
No
Transaction fails
Yes
Contract registers the subscription
Is the subscription still active?
No
Subscription is finished
Yes
Oracles fetch subscription data
Oracles call API
Oracles report data to the smart contract

Figure 2: Solution control flow visualization

4. The application should create means for smart contract developers to access any type of information they need.
5. The oracle network should be resilient towards misreporting by oracles thanks to its consensus mechanism.

4. Implementation

Our implementation consists of 3 primary parts - the client web application and the oracle network, both of which operate off-chain and lastly the oracle smart contract which resides on-chain. The primary focus of this implementation was to utilize cutting edge technology at every step of the way while making sure that each individual component is fully compatible with the others. It is worth noting that that the on-chain part of the implementation can only run natively on UNIX-based operating systems. Running the oracle smart contract on a Windows operating system will require the use of a virtual machine or the Windows Subsystem for Linux.

4.1 Oracle smart contract implementation

When deciding on which platform to develop our solution for, we looked for a blockchain that best aligns with the core principles of simplicity and performance that we envisioned for the project. Its high throughput, fast transaction speeds and minimal transaction fees make Solana the ideal candidate. Additionally, using Solana also gives us the ability to write our contract in the Rust programming language, known for its high performance and memory-safety.

We have also opted to use the **Anchor framework** to streamline the development process of our smart contract. Anchor provides a higher level of abstraction and a more intuitive API, which lets us bypass a lot of the more tedious aspects of writing code for Solana. It is also generally more robust and secure, decreasing the likelihood of errors and vulnerabilities occurring in our project. Lastly, Anchor automatically handles more mundane tasks such as serialization and deserialization under the hood, making the code far more clear and readable.

4.1.1 Accounts

Solana smart contracts, more commonly referred to as programs are technically stateless. That being said, we can still store state inside of accounts. Each account serves as a self-contained pocket of data with its own internal state and can be accessed via its public key. Each account is owned by a specific program and only that program is allowed to execute write operations on it. Other programs are only allowed to read the contents of the account.

Our contract implements two types of accounts. First, a central state account, that will hold all the relevant information about the application, The state account will handle all the necessary information about oracles and the status of the oracle network as a whole. All off-chain calls to our application will access the same state account. Second, we implement a subscription account which holds all of the details of a given subscription. Each subscription is allocated its separate account upon being created.

4.1.2 Instructions

All write operations called by off-chain APIs are handled via program instructions. We can think of each instruction as a method inside the program that can be called by an outside API to perform an action. Instructions can be used to create and modify accounts, as well as to transfer funds.

Our program will contain the following instructions:

Initialize - only called once at the very start to initialize the smart contract and create a state account

AddSubscription - called whenever a user registers a new subscription, takes in the subscription parameters as well as the recipient's address and stores them in a new subscription account

AddOracle - registers a new oracle inside of the state account

ReportData - called by the oracle server to report data to the program, data is subsequently parsed and stored in the account of the subscription that it belongs to

EndRound - called by the oracle server once a round has ended, thereby starting a new round

4.1.3 Consensus

As mentioned before, our solution will utilize a decentralized network of oracles with the proof of stake mechanism for consensus. Whenever a new oracle enters the network, it must also deposit a stake. The network will operate in cycles and at the start of each one a leader is selected out of the available oracles. The chance of being selected is proportional to the size of their stake. Each cycle will contain a single series of API calls made by each oracle - (each oracle will make one call per active subscription). Oracles will then report their data to the leader, who will aggregate them into a single report. This report is subsequently distributed to the other oracles who can choose whether to sign off on it or not.

If the report receives at least two thirds of all total votes, it is submitted to the contract where it can be accessed by customers. If the report is accepted and the leader is rewarded, otherwise the leader's stake is slashed and the report is rejected. Too many rejections will result in the oracle being removed from the network. Regardless of the outcome, the next cycle begins and a new leader is selected.

4.2 Web application implementation

Our web application is implemented in the JavaScript framework Vue 3, also utilizing TypeScript for its numerous benefits such better readability as well as the ease of development and debugging it provides. Much like with our oracle nodes, TypeScript was the obvious choice thanks to its compatibility with the development kits for Solana and the Anchor framework. This is vital because all interaction with the smart contract happens via Solana RPC (Remote procedural call) which is provided by the Anchor SDK. Additionally, there is also a Solana wallet adapter available for the Vue framework, which we have decided to take full advantage of as it makes the overall experience more straight-forward and intuitive for our users.

The following conditions must be met for the web application to be able to function:

1. The oracle smart contract must already be deployed on-chain.
2. The IDL of the contract containing all of its methods and metadata must be provided to the web application in the form of JSON.
3. The IDL must contain a valid Solana address belonging to the currently

deployed version of the contract.

4. A connection to the Solana network must be established.
5. A Solana wallet must be installed in the browser that is being used to access the web application.

4.2.1 Home page

The home page is displayed when the user opens the app. They are then prompted to connect their Solana wallet. Thanks to the aforementioned wallet adapter, the wallet functionality is seamlessly integrated into our web application. The only requirement is for the user to have a Solana wallet within the browser they are using to access our app. It should be noted that Phantom Wallet is currently the only wallet our app supports. Once connected, the user gains access to the other parts of the application.

4.2.2 New subscription page

The "New Subscription" page contains form fields for all the information necessary for creating a subscription. Once a user fills out the form and submits the form is properly validated to make sure that as much of the information as possible is correct. If contents of the form pass the validation a pop-up window is opened and the user is prompted to pay for the cost of the subscription. If the user has enough funds in his account and the transaction is successful the oracle smart contract registers the subscription. From this point onward, the never has to interact with the web application again and their dApp will continue receiving data from off-chain APIs for the remainder of the subscription's duration.

4.2.3 All subscriptions page

The "All Subscriptions" page contains a list of all active and finished subscriptions belonging to a certain user. As the page is loaded, the app makes a call to the oracle smart contract and fetches all subscription accounts linked to the public key of the currently selected wallet. All results, assuming there are any are then neatly displayed in a table. Each entry represents a single subscription and contains all relevant information such as the recipient's address as well as the remaining duration of the subscription. Finished subscriptions are still listed and can be identified by their duration being at 0.

4.3 Oracle node implementation

Oracle nodes will be implemented in TypeScript and will run on the node.js runtime. TypeScirpt was chosen for many of the same reasons that were outlined in the section about the implementation of the web application. Oracle nodes will be connected to an oracle server which they will communicate with via web sockets. This network will be powered using the socket.io library. The oracle server will merely help facilitate communication between the individual nodes. In addition, it will also control relay data to the smart contract at the end of each cycle using the Solana RPC in a manner similar to our web application. The oracle server will play no part in the aggregation or verification of reports which will be left entirely up to the oracles.

The role of the oracle nodes is relatively simple. At the start of each cycle they fetch all subscriptions from the smart contract and call endpoints for all active subscriptions. Afterwards they report their data to the leader. After the leader sends them the final version of the report they compare it to their own data and decide on whether to accept it and sign it with their public key or reject it. Regardless, they then send the data back to the leader afterwards. The leader aggregates all signatures and sends the report to the smart contract. Oracles will make API calls using the Axios HTTP client. We have opted to use Axios instead of the built-in fetch API because of its robustness and extra features which may be useful in the future. To compare data, we will use the MerkleJson library which can be used to generate hashes for JSON strings. MerkleJson will always produce an identical hash for two JSON strings with identical key-value pairs, irrespective of the order in which they are in.

5. Evaluation

The testing phase of development of this project was unfortunately plagued by some technical issues. Given the structure of the project and the fact that it is comprised of three separate off-chain applications (the client web application, the oracle server, the oracle nodes) in addition to the smart contract itself, we were not able to write dedicated unit tests that would sufficiently cover the control flow of the whole system. That being said, we have rigorously tested every part of the application, primarily via tailored inputs designed with specific edge cases in mind.

5.1 Testing and input validation

The web application and the oracle smart contract technically are not susceptible to invalid inputs, however we still want to prevent scenarios in which users unknowingly cause transactions to fail due to unexpected behaviour or create invalid subscriptions which will not receive any data. Because of this, users are barred from creating subscriptions unless they successfully managed to connect a wallet and the application was properly initialized. Users will also not be allowed to create a subscription if any of the form fields containing subscription information are invalid. The oracle network also verifies the validity of each API URL before making a call to ensure a crash does not occur.

Additionally we also tested for the following scenarios, to ensure that the oracle network behaves as intended:

- An oracle connects during network initialization.
- An oracle connects during an ongoing cycle.
- An oracle misreports data.
- A report is approved with sufficient votes.
- A report does not receive sufficient votes.
- The list of subscriptions contains an expired subscription.
- The list of subscriptions contains a subscription with an invalid URL field.
- Subscription data exceeds the maximum allowed transaction buffer size.

5.2 Cost analysis

There are two components which make up the total price of a subscription. The first is the total cost of transactions that will be made by oracles while reporting data back to the smart contract, the second is rent. Rent refers to the fee that a user pays for storing data on the Solana chain. This fee is directly proportional to the size of the account, which is set by the contract when the account is created. Solana currently requires all accounts to be rent-exempt and any account that fails to meet this condition will be deleted with immediate effect. To achieve a rent-exempt status, an account needs to contain enough of the native Solana token SOL to cover two years worth of rent. If the SOL balance ever drops below this threshold, the rent-exempt status is revoked.

Each of our subscription accounts has a total size of 9550 bytes. With this information we can find out the total the amount of SOL required for rent-exempt status using solana's built in CLI utility. Simply calling "solana rent 9550" provides us with the minimum amount of rent required, which in this case is 0.06735888 SOL. According to the official Solana documentation, this number is not expected to change and if it does, measures will be taken to ensure that existing accounts retain their rent-exempt status [32]. We can therefore consider the total rent sum to be a constant. As of the 22.5.2023, the cost of a Solana transaction is roughly 0.000005 SOL and the cost of 1 SOL is 19.88 USD. Using these numbers, we can estimate the cost of a single subscription using the following formula:

$$C_{USD} = (r + tx * t) * SOL_{USD} \tag{5.1}$$

where,
C_{USD} = total cost in USD
r = the cost of rent in SOL
tx = the cost of a single transaction in SOL
t = the duration of a subscription in rounds
SOL_{USD} = the price of 1 SOL

Using this formula and current price data, we can calculate the total cost of a 1000-round subscription as follows:

$$C_{USD} = (0.06735888 + 0.000005 * 1000) * 19.88 = 1.4384945344 \tag{5.2}$$

Rounded up to 2 decimal places, a 1000-round subscription comes out to 1.44 USD, a mostly negligible price made possible by Solana's low transaction costs.

5.3 Performance

All development and testing related activities were conducted on Solana localnet, using a locally run validator node. As a result, we were not able to emulate transaction times accurate to the actual Solana mainnet. In fact, transaction times were significantly higher than those stated on the official Solana website, averaging around 13 seconds. That being said, once deployed to the Solana network, our program should boast transaction speeds consistent with the rest of the network at roughly 400 milliseconds, according to official Solana documentation [33]. Such transaction times would be enough to ensure a seamless user experience, comparable to that of any centralized application.

5.4 Security concerns

Our smart contract currently has one major vulnerability. There are no checks in place to verify the data that is being reported to the contract is coming from the correct oracle node. While we were not able to patch this vulnerability during development, primarily because of a focus on other parts of the implementation, we are fully aware of its existence. We plan to address this issue in the future by requiring that oracles provide a public key during their registration. This key will be saved in the smart contract. When making reports, oracles will be required to sign its contents with their private key. The smart contract will then be able to select the public key of the current leader and use it to verify whether the data was signed by a corresponding private key or not. This same logic can also be applied to how ending rounds is currently handled.

5.5 Other limitations

One of the limitations we have discovered during development is that a serialized Solana transaction must not exceed 1232 bytes in size. This directly limits the volume of data that an oracle is able to relay to the contract inside of a single report. If we account for transactiopn metadata, the current version of our implementation only allows for up to 800 bytes of data to be submitted at one time. We do however have plans to address this in the future, using multiple transactions, first of which will initiate the reporting process, replacing the old data stored in the subscription account. Following transactions will then be made, each appending data to the account until all of the data has been stored.

5.6 Evaluation

Despite having a handful of shortcomings, our project was nonetheless able to deliver on the functionality we originally envisioned. We are able to offer a service that currently has very little representation in the blockchain oracle ecosystem while also maintaining affordability and ease-of-use. Our primary contribution to the ecosystem is the providing users the ability to set up subscriptions to any off-chain API of their choice. This feature currently has very limited availability across all blockchains with only two oracle platforms providing it in a form similar to ours. First being Chainlink which only offers API calls on Ethereum and second being Provable which does not offer decentralization. Neither solution offers the ability to set up a long-term subscription either, with both only giving users the ability to make a single API call at a time with a separate transaction being required to for each additional call [18].

Our solution on the other hand is decentralized and lets users create a subscription which will continue supplying off-chain data to their decentralized application without any further action on their part being needed. Additionally, our solution also requires little to no integration to set up with existing Solana dApps and requires very few changes to existing codebases to be made.

It is also worth noting that there are currently only 2 successful commercial oracle providers on the Solana blockchain, Chainlink and Pyth. Both of which only cover the standard price feed use case at the moment. Not only are we expanding the somewhat limited range of oracle options on Solana, we are also providing developers a tool that was not available prior to this point.

6. Conclusion

The need for trusted sources of outside information will only continue to grow as the adoption of blockchain technology and decentralized finance become more widespread. The goal of this thesis was to provide a comprehensive overview of oracle technology and to help fill in some of the gaps that are currently present in the ecosystem.

The analysis section covered the very basics of what oracles are, what purpose they serve and how we can categorize them, as well as provided some insight to why we think the Solana blockchain is well suited for facilitating their implementation. We documented a variety of existing oracle projects including the most established ones as well as those that provide a unique or innovative perspectives to creating oracle systems. We concluded our analysis with a section dedicated to "The Oracle Problem", containing an overview of the most significant challenges and vulnerabilities present in oracle systems.

6.1 Future Work

The implementation detailed in the previous sections is fully functional, however there are a handful of shortcomings which still need to be addressed. Some parts of the proof of stake mechanism are also currently missing and will need to be added in the future. There is also room for additional features and improvements that exceed the scope of this thesis. The next logical step would be to build upon the existing web application to add a page which would allow oracle providers to register their own deployed oracle nodes into the network. Additional functionality within the oracle smart contract would of course be required to fully support this new feature. This could then be expanded on even further in the future with the release of a fully-fledged software development kit that would improve developer experience as well as provide a degree of standardization for oracle node implementation. Finally, further efforts could be made towards improving performance and overall scalability of the oracle network, once all features are fully implemented and thoroughly tested.

7. Resumé

Blockchain sa v priebehu posledného desaťročia rozrástol z pomerne novej technológie na rozsiahly ekosystém s miliónmi používateľov. Typická blockchainová sieť efektívne funguje ako masívna, distribuovaná účtovná kniha uložená na nezávislých uzloch, ktorej úlohou je overovať a spracovávať transakcie. Tieto transakcie sú následne ukladané do po sebe idúcich blokov nemenných blokov. Tieto transakcie sú typicky verejne dostupné a overiteľné. Nadstavbou blockchainu je aj ekosystém „De-Fi“ alebo decentralizovaných financií, ktorý sa zaoberá využitím blockchainovej technológie na vykonávanie komplexnejších finančných operácií [2], [3], [4].

7.1 Analýza

De-Fi ekosystém je stále pomerne novým konceptom a nie je teda veľkým prekvapením, že sa v ňom zatiaľ nepodarilo dosiahnuť niektoré metriky, ktoré považujeme v štandardnom centralizovanom bankovníctve za samozrejmosť. Jednou z týchto metrík je škálovateľnosť. Konkrétne sa často môžeme stretnúť s pomalým spracovávaním transakcií spojeným s vysokými nákladmi na každú transakciu. Pri hľadaní platformy pre náš projekt sme si práve preto vybrali sieť Solana, ktorá dokáže spracovať viac ako 50000 transakcií za sekundu, čím dokáže konkurovať aj platobným systémom ako je VISA. Pre porovnanie, sieť Ethereum dokáže spracovať približne 17 transakcií za sekundu [2], [13].

7.1.1 Blockchainové oracles

Škálovateľnosť však nie je jedinou prekážkou, ktorej De-Fi ekosystém čelí. Jednou z hlavných nevýhod decentralizovaných aplikácií, že nemajú priamy prístup ku zdrojom dát na bežnom centralizovanom internete. Aj napriek tomu však niektoré aplikácie na blockchaine bez externých dát nedokážu fungovať, ako napríklad decentralizované zmenárne kryptomien, ktoré potrebujú aktuálne výmenné kurzy. Ako riešenie pre tento problém vznikli tzv. „oracles“, ktorých úlohou je zozbierať, overiť a agregovať dáta a následne sprostredkovať ich prenos do blockchainu [14].

7.1.2 Rozdelenie oracles

Existuje niekoľko kritérií na základe ktorých môžeme dané oracle kategorizovať. Základným rozdelením je či ide o softvérové alebo hardvérové oracle. Softvérové oracle typicky získavajú informácie z nejakej API dostupnej na internete, zatiaľ čo tie hardvérové získavajú informácie z rôznych senzorov a zariadení, ktoré snímajú svoje okolie. Zriedka sa môžeme stretnúť aj s ľudskými oracles – expertami na danú problematiku, ktorí môžu ručiť za správnosť relevantných informácií. Ďalej môžeme oracles kategorizovať podľa toho, ako poskytujú informácie. Môže ísť o princíp podania žiadosti o dáta, nasledovanej odpoveďou alebo o dátový kanál, ktorý nepretržite vysiela informácie bez ohľadu na dopyty. Tretím typom sú navyše oracles, ktoré vykonávajú odborné rozhodnutia, napríklad o validite dokumentov. Na záver ešte oracles môžeme deliť na centralizované a decentralizované. Centralizované pozostávajú z jedného uzla, ktorý je často rýchlejší, ale jednoduchšie napadnuteľný. Decentralizované oracles sú siete uzlov, ktoré pred odoslaním informácií musia dosiahnuť konsenzus o tom, aké dáta vlastne odošlú. Decentralizované oracles často využívajú systémy hlasovania alebo reputácie na kontrolu siete [15], [16].

7.2 Existujúce blockchainové oracles

Existuje celá škála rôznych komerčných oracles no pre účely tejto práce sme sa rozhodli zamerať na tie, najúspešnejšie decentralizované softvérové oracles, keďže táto kategória je nielen najbežnejšia, ale aj najviac relevantná pre svet De-Fi.

7.2.1 ChainLink

ChainLink je najpopulárnejším aj najúspešnejším poskytovateľom oracle služieb na trhu. ChainLink prevádzkuje decentralizovanú sieť oracle uzlov, ktoré agregujú všetky dáta mimo siete, čím efektívne znižujú náklady na prevádzku, keďže na odoslanie dát stačí jedna transakcia. Funkčný cyklus siete je rozdelený na kolá. V každom je zvolený líder, ktorého úlohou je zozbierať dáta od ostatných uzlov a agregovať ich do jednej správy. Ostatné uzly sa následne rozhodnú či so správou súhlasia alebo nie. Ak so správou nesúhlasí dostatočný počet uzlov, reputácia lídra klesne a je zvolený nový líder. Keď reputácia niektorého z uzlov klesne na príliš nízku hodnotu, uzol je odstránený zo siete [22].

7.2.2 Oracles na sieti Solana

Sieť Solana sa na prvý pohľad javí ako ideálny kandidát pre oracle systémy, vzhľadom na jej krátku dobu spracovania transakcií a nízke poplatky. Napriek tomu však na Solane existuje len zopár komerčných poskytovateľov oracle služieb. Okrem už zmieneného ChainLinku, tu môžeme nájsť ešte siete Pyth a Umbrella. Obe siete poskytujú dátové kanály s údajmi o cenách digitálnych aktív a využívajú delegovaný Proof-of-Stake mechanizmus na dosiahnutie konsenzu [28], [29].

7.3 Návrh

Hlavná charakteristika, ktorú môžeme pozorovať u väčšiny úspešných oracle systémov je, že majú to isté špecifické zameranie. Ide samozrejme o poskytovanie údajov o cenách digitálnych aktív, primárne kryptomien. V kontexte sveta De-Fi toto využitie dáva zmysel, no myslíme si, že existujú aj iné zmysluplné zamerania. Z tohto dôvodu sme sa v rámci tejto práce rozhodli navrhnúť a implementovať univerzálne oracle, ktoré používateľom umožní získavať dáta z akejkoľvek API dostupnej na internete. Naše riešenie bude pozostávať z troch hlavných častí: oracle smart kontraktu, oracle siete a webovej aplikácie. Na konsenzus v sieti budeme využívať mechanizmus Proof of Stake.

7.3.1 Oracle smart kontrakt

Úlohou smart kontraktu je regulovať postupnosť cyklov, vyberať lídrov, dohliadať na činnosť jednotlivých oracles, ako aj celej siete. Kontrakt obsahuje aj účty zákazníkov, do ktorých sa po prijatí budú ukladať informácie z externých zdrojov.

7.3.2 Oracle sieť

Oracle sieť obsahuje individuálne oracle uzly, ktorých úlohou je získavať a agregovať dáta z externých API. Uzly budú komunikovať medzi sebou ale aj s kontraktom, z ktorého budú získavať informácie o danom cykle a o tom, z akých zdrojov získavať dáta. Po zozbieraní dát ich oracle uzly pošlú lídrovi, ktorých ich agreguje a následne pošle späť. Jednotlivé oracles sa potom môžu rozhodnúť či z finálnou verziou dát súhlasia alebo nie.

7.3.3 Webová aplikácia

Zákazníci budú s kontraktom interagovať hlavne prostredníctvom webovej aplikácie, cez ktorú si budú môcť vytvoriť „predplatné“ na danú API, z ktorej následne oracle sieť extrahuje dáta a sprístupní ich na sieti.

7.4 Implementácia

Implementácia pozostáva z troch hlavných častí opísaných v návrhu. Implementácia funguje natívne len na operačných systémoch na báze UNIX, no jej spustenie je možné aj na operačnom systéme Windows s použitím Windows Subsystem for Linux.

7.4.1 Oracle smart kontrakt

Smart kontrakt je písaný v jazyku Rust pre sieť Solana s použitím rámca Anchor určeného na zjednodušenie a sprehľadnenie vývoja smart kontraktov. Prvým kľúčovým konceptom pre pochopenie Solana programov sú inštrukcie, ktoré efektívne fungujú ako metódy a kontraktu povedia čo má robiť. Druhým konceptom sú účty. Solana program ako taký v sebe neukladá žiadne údaje. Všetky údaje sú uložené na účtoch, ktoré patria danému programu. Ak chceme k týmto údajom pristúpiť, musíme adresy na dané účty vložiť do danej inštrukcie.

7.4.2 Webová aplikácia

Webová aplikácia využíva rámec Vue pre jazyk JavaScript. Logika v rámci aplikácie je písaná v jazyku TypeScript, ktorý je nadstavbou JavaScriptu. Aplikácia s kontraktom komunikuje cez Solana RPC. Do aplikácie je taktiež integrovaná krypto peňaženka Phantom, ktorá poskytuje používateľské rozhranie pre vykonávanie transakcií používateľom.

7.4.3 Oracle sieť

Oracle sieť je písaná v jazyku TypeScript a skladá sa z dvoch častí. Prvou je server, ktorý sprostredkováva komunikáciu medzi jednotlivými uzlami. Druhou sú individuálne uzly, ktoré cez web sockety komunikujú so serverom a získavajú dáta z API pomocou HTTP knižnice Axios. Komunikácia siete so smart kontraktom prebieha pomocou Solana RPC.

7.5 Evaluácia a záver

Naša implementácia je zameraná na prípad použitia, ktorému sa momentálne nevenuje veľa oracle systémov. Navyše existuje na sieti, ktorej oracle ekosystém je veľmi obmedzený. Finálna aplikácia je taktiež relatívne lacná a rýchla. Vytvorenie účtu pre ukladanie dát spolu s 1000 opakovaniami získania informácií z externej API klienta bude stáť menej než 1.50 USD. Spracovanie transakcií prebieha rýchlo, keďže priemerný čas na vykonanie Solana transakcie je len menej ako pol sekundy [33], [32].

Dopyt po dôveryhodných zdrojoch informácií pre decentralizované aplikácie bude v budúcnosti spolu so svetom De-Fi len rásť. Zámerom tejto práce bolo nielen zostaviť prehľad existujúcich oracle systémov, ale aj prispieť do tohto ekosystému vlastnou implementáciou, zameranou na niektoré z jeho nedostatkov.

References

1. MONRAT, Ahmed Afif; SCHELÉN, Olov; ANDERSSON, Karl. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access*. 2019, vol. 7, pp. 117134–117151. Available from DOI: 10.1109/ACCESS.2019.2936094.
2. PIERRO, Giuseppe Antonio; TONELLI, Roberto. Can Solana be the Solution to the Blockchain Scalability Problem? In: *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 1219–1226. Available from DOI: 10.1109/SANER53432.2022.00144.
3. NOFER, Michael; GOMBER, Peter; HINZ, Oliver; SCHIERECK, Dirk. Blockchain. *Business & Information Systems Engineering*. 2017, vol. 59. Available from DOI: 10.1007/s12599-017-0467-3.
4. WERNER, Sam M.; PEREZ, Daniel; GUDGEON, Lewis; KLAGES-MUNDT, Ariah; HARZ, Dominik; KNOTTENBELT, William J. SoK: Decentralized Finance (DeFi). *CoRR*. 2021, vol. 101.08778. Available from DOI: 10.48550/ARXIV.2101.08778.
5. ZETZSCHE, Dirk; ARNER, Douglas; BUCKLEY, Ross. Decentralized Finance. *Journal of Financial Regulation*. 2020, vol. 6, pp. 172–203. Available from DOI: 10.1093/jfr/fjaa010.

6. QIN, Kaihua; ZHOU, Liyi; AFONIN, Yaroslav; LAZZARETTI, Ludovico; GERVAIS, Arthur. CeFi vs. DeFi – Comparing Centralized to Decentralized Finance. *CoRR*. 2021, vol. 2106.08157. Available from DOI: 10.48550/ARXIV.2106.08157.

7. JENSEN, Johannes Rude; WACHTER, Victor von; ROSS, Omri. DeFi: Decentralized Finance - An Introductionand Overview. *Complex Systems Informatics and Modeling Quarterly*. 2021, no. 26. Available from DOI: https://doi.org/10.24840/2183-0606_009.003_0001.

8. TRUONG, Nguyen B.; UM, Tai-Won; ZHOU, Bo; LEE, Gyu Myoung. Strengthening the Blockchain-Based Internet of Value with Trust. In: *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7. Available from DOI: 10.1109/ICC.2018.8423014.

9. AKBABA, Jakov; CHILLINGWORTH, Julian. Blockchain: The internet of value. *Rathbones*. 2017. Available also from: https://www.rathbones.com/knowledge-and-insight/blockchain-internet-value. [online] cited 8.10.2022.

10. VADGAMA, Nikhil; XU, Jiahua; TASCA, Paolo. *Enabling the Internet of Value How Blockchain Connects Global Businesses: How Blockchain Connects Global Businesses*. SpringerLink, 2022. ISBN 978-3-030-78183-5. Available from DOI: 10.1007/978-3-030-78184-2.

11. TASCA, Paolo. Internet of Value: A Risky Necessity. *Frontiers in Blockchain*. 2020, vol. 3. ISSN 2624-7852. Available from DOI: 10.3389/fbloc.2020.00039.

12. VISCONTI, Roberto Moro. *The Valuation of Digital Intangibles*. Palgrave Macmillan Cham, 2020. ISBN 978-3-030-36918-7. Available from DOI: https://doi.org/10.1007/978-3-030-36918-7.

13. DUFFY, Fintan; BENDECHACHE, Malika; TAL, Irina. Can Solana's high throughput be an enabler for IoT? In: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2021, pp. 615–621. Available from DOI: 10.1109/QRS-C55045.2021.00094.
14. PASDAR, Amirmohammad; DONG, Zhongli; LEE, Young Choon. Blockchain Oracle Design Patterns. *CoRR*. 2021, vol. 2106.09349. Available from DOI: 10.48550/ARXIV.2106.09349.
15. BENIICHE, Abdeljalil. A Study of Blockchain Oracles. *CoRR*. 2020, vol. 2004.07140. Available from DOI: 10.48550/ARXIV.2004.07140.
16. AL-BREIKI, Hamda; REHMAN, Muhammad Habib Ur; SALAH, Khaled; SVETINOVIC, Davor. Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges. *IEEE Access*. 2020, vol. 8, pp. 85675–85685. Available from DOI: 10.1109/ACCESS.2020.2992698.
17. LO, Sin Kuang; XU, Xiwei; STAPLES, Mark; YAO, Lina. Reliability analysis for blockchain oracles. *Computers & Electrical Engineering*. 2020, vol. 83, p. 106582. ISSN 0045-7906. Available from DOI: https://doi.org/10.1016/j.compeleceng.2020.106582.
18. PASDAR, Amirmohammad; LEE, Young Choon; DONG, Zhongli. Connect API with Blockchain: A Survey on Blockchain Oracle Implementation. *ACM Comput. Surv.* 2022. ISSN 0360-0300. Available from DOI: 10.1145/3567582.
19. ZHAO, Yinjie; KANG, Xin; LI, Tieyan; CHU, Cheng-Kang; WANG, Haiguang. Towards Trustworthy DeFi Oracles: Past,Present and Future. *CoRR*. 2022, vol. 2201.02358. Available from DOI: 10.48550/ARXIV.2201.02358.
20. WOO, Sangyeon; SONG, Jeho; PARK, Sungyong. A Distributed Oracle Using Intel SGX for Blockchain-Based IoT Applications. *Sensors*. 2020, vol. 20, no. 9. ISSN 1424-8220. Available from DOI: 10.3390/s20092725.

21. FERRULLI. *On demand decentralized oracles for blockchain: a new Chainlink based architecture*. 2022. Available also from: https://etd.adm.unipi.it/t/etd-02062022-124127/. [online] cited 5.11.2022.
22. BREIDENBACH. Chainlink Off-chain Reporting Protocol. 2021. Available also from: https://research.chain.link/ocr.pdf. [online] cited 5.11.2022.
23. LYS, Léonard; POTOP-BUTUCARU, Maria. *Distributed Blockchain Price Oracle*. 2022. Available also from: https://hal.archives-ouvertes.fr/hal-03620931. working paper or preprint.
24. LAMBUR. *Introducing UMA's Optimistic Oracle*. 2021. Available also from: https://medium.com/uma-project/introducing-umas-optimistic-oracle-d92ce5d1a4bc. [online] cited 6.11.2022.
25. *How does UMA's Oracle work?* [N.d.]. Available also from: https://docs.umaproject.org/protocol-overview/how-does-umas-oracle-work. [online] cited 6.11.2022.
26. BENLINGIRAY. *Airnode: The API gateway for blockchains*. 2020. Available also from: https://medium.com/api3/airnode-the-api-gateway-for-blockchains-8b07ff136840. [online] cited 6.11.2022.
27. BENLINGIRAY. *Decentralized APIs for Web 3.0*. 2020. Available also from: https://www.securities.io/wp-content/uploads/2022/05/API3-whitepaper.pdf. [online] cited 6.11.2022.
28. *Pyth Network, A First-Party Financial Oracle*. 2022. Available also from: https://pyth.network/whitepaper.pdf. [online] cited 25.11.2022.
29. *Umbrella network litepaper*. 2021. Available also from: https://www.umb.network/umb_litepaper_design_v3.1.pdf. [online] cited 25.11.2022.
30. CALDARELLI, Giulio; ELLUL, Joshua. The Blockchain Oracle Problem in Decentralized Finance—A Multivocal Approach. *Applied Sciences*. 2021, vol. 11, no. 16. ISSN 2076-3417. Available from DOI: 10.3390/app11167572.

31. KUMAR, Manoj; NIKHIL, Nikhil; SINGH, Riya. Decentralising Finance using Decentralised Blockchain Oracles. In: *2020 International Conference for Emerging Technology (INCET)*. IEEE, 2020, pp. 1–4. Available from DOI: `10.1109/INCET49848.2020.9154123`.
32. *Solana Transaction Confirmation*. [N.d.]. Available also from: `https://docs.solana.com/developing/programming-model/accounts`. [online] cited 19.5.2023.
33. *Solana Transaction Confirmation*. [N.d.]. Available also from: `https://docs.solana.com/developing/transaction_confirmation`. [online] cited 18.5.2023.