Computer Science and Engineering
Software Engineering 2 Research Project - Prof. Elisabetta Di Nitto

DeepThought: a Reputation and Voting-based Blockchain Oracle

Marco Di Gennaro (10596841)
Thomas Golfetto (10745981)
Lorenzo Italiano (10800992)

Contents

1. Abstract

One of the core properties which a blockchain relies on is the *immutability* of a transaction, it means that nobody can alter it and its integrity is guaranteed. Problems arise when someone inserts fake data on chain, for example in a supply chain is not rare that data are compromised during the process, because people lie and sensors can be altered. The solution to the problem described is called **Oracle**: a layer between the blockchain and the outside world, which has the duty to verify, query, and authenticate data inbound or outbound. Different typologies exist, in this project we focused on decentralized oracles, but in particular voting-based oracles. After a detailed description of the main protocols, we proposed and developed a voting and reputation-based oracle, called **DeepThought**, based on two protocols: **ASTRAEA** [6] and a **peer prediction-based protocol** [13], with the aim to overcome the problem of *fake news*.

Each protocol described tries to go against the challenges given by the voting game used as *consensus mechanism*, in particular, the main problems are: *the verifier's dilemma*, a degenerated coordination strategy where voters vote with a constant value, maximizing their reward with less effort possible; *eavesdropping* (or herd behavior), which is a general voting problem, emphasized by the transparency property of the blockchain, where voters can read the other votes ending up with voting according to the majority; *sybil attack*, where an attacker tries to control the oracle outcome using multiple nodes.

In the end, we describe the idea DeepThought is based on, listing the main characteristics and showing various examples of its functionalities. The core idea of DeepThought is to go against Fake News, but it can fit well in various other applications, such as academic environments or in any other situations in which objective opinions on a topic are needed.

2. Introduction

This project aims to review and contribute to the scientific research surrounding decentralized oracles. In particular, we focused on two decentralized oracle methodologies (reputation-based and voting-based) and we mixed them in order to reduce the problems of one and the other method and to benefit from their combination. Specifically, we will start by presenting a summary of the state of the art in the domain of decentralized oracles, and then propose our method of doing so.

During the realization of this project, we were inspired by the scope of use of blockchain technology and oracles to counter Fake News.

In this section, we introduce the context of the project by talking about **Blockchain**, **Oracles** (focusing on Decentralized Oracles), and **Blockchain against Fake News**.

The rest of the document is structured as follows: Section 3 (*Related Works*) contains the work related to this project, which we have used or inspired, the section 4 (*Our Idea and Implementation*) describes how we structured the project and how the latter was implemented, the section 5 (*Results*) contains our research results and section 6 (*Conclusions*) contains our final considerations about this project.

2.1 Blockchain

Blockchain is a distributed, immutable and decentralized ledger, where transactions are secured and verified. Originally, the blockchain was conceived to process monetary transactions using a peer-to-peer network without relying on a central trusted entity. Transactions are grouped up inside a *block*, which are ordered lists of transactions. Each block contains a hash of the previous block, creating a hash-link chain of blocks. The process to save and generate the block on the chain is called *mining* and it consists of a complex mathematical problem.

The first *miner* solving the problem generates the so-called "Proof-of-Work" (PoW) and he is rewarded with a certain amount of cryptocurrencies. Rewriting the chain to undo a transaction requires repeating PoW, and the further back in history the transaction is, the more work is required [19].

A more advanced and secure mining procedure called "Proof-of-Stack" (PoS) is now gathering interest, because it doesn't depend on computational power, but is based on the amount of cryptocurrency the miner holds [12].

There are two types of blockchain:

- A **permissionless blockchain**, such as Ethereum [11] and Bitcoin, is an open and decentralized ledger, where anyone can join and leave every time and no entity manages the membership. In this case, the content is available for everyone, and privacy is not guaranteed. Some blockchain is using cryptographic primitives to hide privacy-relevant information (e.g. Zerocash).
- A **permissioned blockchain**, such as Hyperledger, has instead a central entity who decides the rights of any individual peers. Only a small set can join and not everyone is authorized to make transactions. A *public permission blockchain* is a variant with fewer restrictions, where everyone can read but only a few can write [26].

Nowadays, multiple blockchains are risen and their applications are countless: fintech, Internet of Things, NFTs, e-voting, and so on. All of this, is now possible thanks to Szabo's idea came up in the 1996 [22],[23]: the "smart-contract", a computer protocol that digitally facilitates, verifies, and enforces the contracts made between two or more parties on blockchain [25].

But still, there is a huge limitation: a blockchain cannot interact safely with the "outside world" [14].

2.2 Oracles

While using blockchain-based applications we also have to consider the possibility of exchanging data between the blockchain and the external world. However the blockchain itself is not able to communicate with external systems, blockchain oracles provide this service. They act as a trusted bridge for connecting the external world of applications with blockchain smart contracts providing them with validated information [17].

Blockchain oracles are a layer solution that verifies, queries and authenticates data from different type of sources both hardware and software and transmit them either into the blockchain (inbound) or outside the blockchain (outbound) [9].

There are different categories of blockchain oracles, such as:

- **Software oracles** are the most powerful type of oracle because of their inherent interconnection with the internet, this connection allows online sources of information such as websites and public databases to supply the most up-to-date information to the blockchain.
- **Hardware oracles** usually has the task to send into the blockchain data as a result of an occurrence in the physical world, for example in the supply chain management information about an object can be inserted into the blockchain whenever an object arrives in a warehouse, facilitating the tracking of good along the supply chain.
- **Inbound oracles** have the function of simply supplying external data to smart contracts upon receiving the information, vice versa **outbound oracles** communicate the blockchain data to an external source.
- In some cases, we can also have **Consensus based oracles**, where a group of oracle sources make a query to retrieve some data, and based on the consensus of them the oracle arrive at an outcome and transmit the data to the blockchain.

Blockchain oracles can also be categorized in several other ways, for example oracles can be centralized or decentralized depending on the number of nodes that communicate the information to the smart contracts or in **human oracles**, which relays on people with deep knowledge on the information to be transmit voting and verifying the truth of this data, and **unmanned oracles**, where this verification is made automatically through a set of rules and algorithms. One of the great advantages of unmanned oracles is the speed at which they provide answers, because human oracles need a certain latency for their answers to achieve consensus.

On our research, we will focus more on decentralized human oracles since to verify if a news is fake or not, we cannot rely on algorithms and we want this validation to be decentralized in order to have a more secure and reliable vote.

2.2.1 Centralized Oracle

When the decision to transmit a piece of information from the outside world to the blockchain is taken by a single node, we are talking about a centralized oracle. A single oracle is useful as *Outbound Oracle*, to export data from the blockchain to the world, and it has high efficiency, but it represents a *single Point-of-Failure*

of the system, because availability, accessibility, and in particular validity of the data relies only on one node. Moreover, the use of a centralized oracle brings back the *centralization problem* to the blockchain, relying on a single source is in contrast with the decentralization principle and risks to bring corruptness and incorrect data on the blockchain.

The most well-known centralized oracle is **Provable** [2] (also known as *Oraclize*). Its main objective is to provide data from a web API specified by a user along with cryptographic proofs produced by mechanisms such as *Trusted Execution Environment* (TEE) and **TLSNotary** [4] in order to ensure that the retrieved information is genuine from the chosen source [7].

Another famous centralized oracle is **TownCrier**, which enables hardware-based TEEs where data authentication can be performed and provides secure execution environments. TownCrier's main core is given by Intel's *Software Guard Extensions* (SGX), to protect the attestations against malicious operating systems, providing integrity and confidentiality [27].

Evidently, both protocols rely on a centralized server to handle query requests. This exhibits a strong form of centralization that violates the equi-privilege property (all system users have identical priority). [13]

2.2.2 Decentralized Oracle

In case the decision to transmit data from an external systems to the blockchain smart contracts is taken by multiple nodes, we are talking about a decentralized oracle. This category of oracle is useful as an *Inbound Oracle* to import data into the blockchain from multiple and decentralized sources. This allows much application especially with hardware oracles, but it also introduces a consensus problem in case the multiple oracle nodes have to agree on the validation of data.

The idea of decentralized oracles is more compliant with the idea of blockchain since it's distributed and resolves the single Point-of-Failure problem of centralized oracles. The consensus between nodes can be solved either by introducing a reputation system or a voting system.

2.2.2.1 Reputation-based Oracle

Consensus between nodes in a decentralized oracle network is obtained using a reputation system where the mining power of the contributors to the validation of information is based on its reputation score, which increases or decreases if the solution provided by the node is complaint with the one provided by the majority of the nodes.

One of the most famous reputation-based oracles is **Witnet**, which runs its native customized blockchain and its own protocol token namely *Wit* [24].

2.2.2.2 Voting-based Oracle

In a voting-based oracle network, the consensus is obtained using a voting system game where the entities can act either as a submitter or a voter. The *submitters* are the nodes that feeds the data inside the oracle network, this data has to be validated and approved from the majority of the voter nodes before its insertion into the blockchain. The *voter* nodes will have to vote to approve or reject the data insertion, usually betting an amount of money or cripto-currency to incentive them to vote correctly. Other variations of this approach are possible.

A famous example of this type of oracles is **ASTRAEA** where entities can act as submitters, voters, or certifiers [6].

The trust model of the voting-based oracles is the Nash equilibrium of the betting game, with the assumption that all participants have the interest of behaving honestly and tend to reduce the impact of malicious indi-

viduals since with a large number of nodes it is proven that is more economically convenient for every node to behave correctly.

2.3 Blockchain against Fake News

In recent years, the increase in the use of social networks and online information platforms has led, on the one hand, to an increase in freedom of speech, however, on the other hand, it has led to the generation of a large number of fake news. Given the impossibility of stopping the publication of a news story in the bud, which would go against a democratic approach, the scientific community is constantly looking for methods to instruct the reader to recognize real news from a fake. A blockchain based solution has the potential to change the way information is produced and disseminated while playing the major role to tackle disinformation over the longer term. In fact, due to his traceability, transparency and decentralization nature, the problem of fake news can be handled effectively. The blockchain enabled platform can provide online readers with a reliable way of verifying the content and its source [21].

3. Related Works

3.1 Reputation-Based Oracle: Witnet

Witnet is a reputation-based decentralized oracle network that connects the on-chain smart contracts with the external world data providers [24]. The Witnet network runs on its native customized blockchain and its own token namely Wit that miners can earn by retrieving and validating the external information to be inserted into the smart contracts.

These miners, called *witnesses*, perform a series of activities named *Retrieve–Attest–Deliver (RAD)* work, on the other end the external data provider pays the witnesses for their work. Witnesses contribute with their mining power which is mainly determined by their reputation for efficient, honest, and trustworthy execution of RAD jobs. This creates a powerful incentive for witnesses to do their work honestly, protect their reputation, and not deceive the network.

In addition, Witnet enables a special kind of participating nodes on the network, called bridge nodes, which only perform the 'Deliver' part of the RAD work.

The protocol is designed to assign the RAD jobs to witnesses in a way that mitigates most attacks and at the same time, it includes features that guarantee efficiency and scalability of the network. Other features allow to keep the price of RAD tasks within reasonable bounds and give to the clients the possibility of adjusting how many witnesses work on their RAD tasks.

Witnet was designed considering the notion of decidability and verifiability, which are the basic requirements to attest the RAD requests on the decentralized network. Although the witnesses and bridges can easily process the decidable requests, it's hard to process the undecidable requests whose truth or the false result could not be determined within a limited time interval.

In this case, the network enables a special type of task in order to handle the undecidable request, which can remain unresolved for an indefinite period of time. Since the provability or verifiability of RAD requests could be determined by the client smart contracts hence Witnet does not deploy any special RAD requests to cater to these special needs [8].

3.2 Voting-Based Oracle: ASTRAEA

ASTRAEA[6] is a general-purpose decentralized oracle voting-based that runs on a public ledger and leverages human intelligence through a voting game. The entities in ASTRAEA are:

- *Submitters*, people who submit a Boolean proposition to the system paying a fee;
- *Voters* play a low-risk/low-reward game, where they can vote a random proposition by placing a stake on their answer. The outcome of the voting process is a function of the sum of the votes weighted by the deposits;

- *Certifiers* play a high-risk/high-reward game, where they can place an important stake on the outcome of the voting and certification processes of a chosen proposition. Since certifiers choose which propositions to certify, not all the propositions are guaranteed to have a certification.

Different from similar oracles, such as **Augur** and **Truthcoin**, where the voters must be online during the process and they are not allowed to leave and join whenever they want, ASTRAEA does not require anything of this. In particular, all roles may enter or exit the system anytime.

The *proposition list* has a fixed size and the voting game is played over all the propositions in the list simultaneously. For each proposition, there are a *bounty amount* associated, used to reward voters, and two different *certifier reward pools*, containing the reward for the certifiers with respect to the outcomes. The main objective is to avoid a degenerate coordination strategy in which users always vote and certify with a constant value (`true` or `false`), in order to maximize their profit without expending any effort. This problem is also known as *Verifier's Dilemma* [16].

The voting game is schematized in three steps:

1. *Voting*: players place a desired amount of money (less than a certain amount), in order to receive a proposition not yet known. Then, the system chose uniformly at random the proposition over all the propositions in the proposition list. In the end, the voters submit their vote, using a hash value of the vote concatenated with a nonce. Only later the votes are revealed.

2. *Certifying*: players place an amount of money (larger than a certain amount) and a certification for a proposition chosen over all the propositions in the proposition list.

3. *Termination*: Once a proposition has accumulated a sufficient voting stake, the voting is terminated and the result is decided. At this time, the voting and certifying outcomes with their respective stake are computed.

Depending on the system, a simple majority or a super-majority is used to decide the outcome. When the result of the voting or the certifying ended in a tie, the outcome will be `unknown`. The oracle outcome is `true` or `false` only when certifiers and voters agreed, otherwise will be `unknown`. Instead of a boolean statement, the output of the oracle can be expressed as the *confidence* in the truth or falsity of the proposition, using a continuous value in the range $[0, 1]$.

The *game outcome*, which not always is equal to the *oracle outcome*, is used to decide the rewards and the penalties. Voters are rewarded only if their vote corresponds to the outcome of the proposition, instead, if their vote is in contrast with the outcome they are penalized. In the `unknown` case, only certifiers are penalized, while voters keep back their initial stakes. The motive is linked to the fact that certifiers choose the proposition, while voters receive it at random. The submitter fees are used to fund bounties, while the certifier reward pools are initially left empty. Those are funded by unclaimed bounties and penalties. Rewarding pools and bounties are balanced to provide a reasonable incentive for both voters and certifiers. In the relative paper, [6], is demonstrated the difficulty in manipulating the voting outcomes and moreover is proven the existence of the Nash Equilibrium where all players play honestly regarding their true beliefs. Furthermore, the reward structure is built to avoid Verifier's Dilemma problems. The main issues are related to the propositions submitted to the voting. In the case in which the proposition is undecidable, or the problem has not a clear answer, how do the voters assign their vote if the truth value doesn't exist? A possible solution is to add a third option besides `true` and `false`: `unknown`. For the certifier, there is no problem, because they can simply don't place any stake, since they can choose the proposition.

How John Andler, one of the authors of ASTRAEA, reported on this article [5], originally, the project was called *Athena* and the main objective was to developed a fake-news detection platform. But, discouraged by the big challenge and by all the issues involved in the task, his research group and him had redirected all their studies to a new topic: a decentralized network of individuals with the duty to decide on the truthness or falseness of a premise, in other words, an *oracle*.

3.2.1 ASTRAEA-based oracles

Since the advent of ASTRAEA, multiple derivations of its mechanism are rapidly born. The most known and criticized ones are those listed below.

The first attempt more successful is probably **Shintaku** [15], which has also developed a prototype available in the relative GitHub link. The main difference with respect to ASTRAEA protocol is the absence of certifiers. With this absence, the certification of the voting result is given by the voters their-self. In order to avoid the *Verifier's Dilemma*, a voter has to answer a pair of the proposition, chosen at random in the proposition list, instead of a single one. Voters are eligible for rewards only if their choices for the two propositions differ; their stake is simply returned to them otherwise. The reward for each voter can be calculated as the proportion of each voter's eligible stake and the proposition's bounty. Voting toward an exclusive way becomes impossible, and hence the verifier's dilemma is overcome.

In many articles, such as [18] and [13], the authors argued about the method used by Shintaku to reduce complexity with a proposition pair, because in theory the lazy voting is avoided, but in a practical view, this is true only if the penalties for disagreement are at least twice as large as rewards for agreement. Moreover, the honest voters are not incentive because the payoffs are low.

Then, both of the previously mentioned articles proposed a different approach to solving the problem.

The former proposes a **paired-question protocol** for decentralized oracles [18], where a submitter has to submit two antithetic propositions, posting a bond. The oracle collects votes and checks whether the two questions converged to different answers; if so, the submitter regains their bond and voters are rewarded for agreement with the majority answer. Otherwise, voters are penalized for disagreement. If the questions converged to the same answer, the submitter loses their bond and voters receive neither rewards nor penalties. With respect to ASTRAEA and Shintaku, this method is simpler for voters and the honest voters are incentivized because of the higher payoffs given by the protocol. On the other hand, the mechanism became more expensive for submitters, because antithetic propositions are not always sustainable.

The latter is a further evolution based on ASTRAEA, called "Truth-inducing Sybil resistant decentralized blockchain oracle" (for the rest of the document we will refer to this evolution under the name of `TISRDBO`), coming up with a **peer prediction-based protocol** with non-linear scaling of stake for decentralized oracles [13].
Compared to the paired-question protocol described earlier, there are two main enhancements:

1. Rewards for voters are determined by a score, which is calculated with a light-weight scoring rule by referencing the voting behavior of other voters with respect to the submitted answer;
2. Voting weight is scaled sub-linearly while award portion is scaled super-linearly with respect to the submitted stake.

The oracle assigns questions to voters and collects reports consisting of two components: a binary information answer and a popularity prediction. The oracle answer is determined by the majority of the information answer, weighted by the associated stakes and adjusted by a sub-linear function. Then, the oracle assigns a score to each report based on the accuracy and the degree of agreement with peers. Only the top-scored voters are awarded, while the share of award is determined by their stake adjusted by a super-linear function.

This ASTRAEA evolution is focused on overcoming in every way possible the *Sybil Attack*, where an attacker tries to control the oracle outcome using multiple voting nodes. In addition, it incentives an honest voting mechanism.

3.3 ChainLink

ChainLink is a permissionless framework of decentralized oracles that aims to connect off-chain and on-chain environments by ensuring data availability and the creation of a network of data providers. [10] The oracle node, denoted as CHAINLINK-SC, returns replies to data requests or queries made by a smart contract, denoted as USER-SC. Behind every CHAINLINK-SC the network has an on-chain component that consists of three parts: the reputation contract, the order-matching contract, and the aggregation contract.

The reputation contract keeps track of the oracle network performance, the order-matching contract takes a proposed *service level agreement (SLA)*, which is the request of off-chain data from a contract, logs the SLA parameters, and gets the results from the oracle providers. The aggregating contract then collects the oracle providers' responses and calculates the final result of the query, then feeds some metrics back to the reputation contract.

The on-chain workflow has three steps:

1. **Oracle Selection**: The service purchaser specifies some requirements that make up an SLA proposal. The SLA proposal includes details such as query parameters and the minimum number of oracles to use to retrieve the data. Additionally, the requester can specify the reputation and the aggregating contracts to be used. Using a set of data gathered from logs of past contracts the purchasers can also manually sort and filter the oracles to choose via off-chain services. Purchasers will submit SLA proposals to oracles off-chain, and come to an agreement before finalizing the SLA on-chain.

   ChainLink nodes then choose whether to accept the SLA or not based on their capabilities to satisfy the requirements. When the oracle accepts the SLA it bids for it so it is fully committed to it. Once the SLA has received enough bids the requested number of oracle nodes is chosen from the bidding pool and the nodes that weren't chosen get their bid back. When the SLA is finalized the chosen oracle nodes execute the agreement and report the result on-chain.

2. **Data Reporting**: Once the new oracle record has been created, the off-chain oracles execute the agreement and report back on-chain the result of the query.

3. **Result Aggregation**: Once the oracles have revealed their results to the oracle contract, their results will be sent to the aggregating contract. The aggregating contract collects the results from the pool of selected nodes and calculates a weighted answer based on their reputation. Finally, the answer is sent back to the USER-SC and the reputation contract will be reported with the outcome since it needs to update the reputation of the nodes involved in the computation. Detecting outlying or incorrect values is a problem that is specific to each type of data feed and application.

Off-chain, the ChainLink network consists of multiple independent nodes that support the connection with the Ethereum network and other smart contract networks. Their individual responses to the queries are validated with several possible consensus mechanisms before the response is returned to the USER-SC.

The network nodes have an open-source core so node operators can implement additional software extensions to provide and feed the network with their external data and services, that's one of the reasons behind ChainLink's increasing popularity and its market adoption.

3.4 Blockchain Technology against Fake News: Safe.press and ANSAcheck

Safe.press is a decentralized and consortium-based certification platform that uses IBM Blockchain technology to track and authenticate the source of a published news story with one click. Each news item is recorded on the platform's blockchain ledger and all members of the consortium have a copy. The details that the ledger provides — where, when, and by whom a story is published — are safely hosted by all consortium members. Protocols are made through consensus and agreed to by all participating members — building trust and transparency not only for consortium members but for journalists and the public at large. [3]

ANSAcheck is the blockchain news certification system chosen by ANSA "to control", explains the agency, "at best the flow of his news, so that they cannot be used or disclosed in an untruthful and inappropriate manner, guaranteeing the reader the highest quality and reliability of the source". Thanks to the 'ANSAcheck' news tracking sticker, it will be possible to trace the notarized history of each news, guaranteeing the reader the traceability of the data and the transparency of the information, thus allowing to verify the authenticity of the news sources [1]. The new ANSA solution is based on EY OpsChain Traceability technology, characterized by public transactions registered on the Ethereum blockchain. In particular, the core of the platform is the smart contract.

4. Our Idea and Implementation

After the outcome of the ASTRAEA idea, many oracles tried to improve their mechanism to create an effective and practical oracle. ASTRAEA, based on our studies, has never been developed. The only real application was provided by Shintaku, the other protocols have never been effectively used, also because they are quite new. For example, TISRDBO was presented and then published only at the end of 2020.

After carrying out a massive state-of-the-art study, focusing on all related works presented in the previous chapter, we have chosen as starting point of this research project TISRDBO [13]. In particular, we were inspired by the *scoring-based system* proposed by this latest research. We believe that this system is the most suitable to include the reputation factor. In the following sections, we will explain our idea specifically, justifying the various choices made.

4.1 The Idea

Starting from TIRSDBO, we chose not to take advantage of the *paired-question* used by the *peer prediction-based protocol*, keeping back the ASTRAEA **certifiers** figure — to avoid the lazy voting described in the *Verifier's Dilemma*. This choice is due to the difficulty of applying the *paired-question protocol* in certain applications. Taking the practical example proposed in our project (Fake News), it is not very convenient for the scope, because a submitter cannot have the complementary counterpart of news, and create a new credible one based on the first is a difficult task. Moreover, one of the challenges of TISRDBO was to add the **reputation factor** to the protocol, in order to take into consideration also the history of the voters. Someone who is always correct had to be compensated for his work, on the other hand, a person giving always the wrong answer cannot be paid as the first one. For this motive, the voters and the certifiers eligible for the reward will be compensated based on their reputation. This is adding another motivation to give the most truthful answer, if you submit the wrong one you will lose your reputation.

To better explain the operation of our oracle **DeepThought**, we will describe below all the operations that a user can perform through DeepThought and, how the oracle acts according to the requested operation.

4.1.1 Introduction

The main tasks a user can do with DeepThought, are:

1. **Subscribe**: to simulate the money flow, a fake currency is used inside the contract, which is based on the real amount a user have. This amount is stored from the user's real balance once he subscribed;
2. **Submit a Proposition**: a user can submit a proposition (i.e. a news) to the oracle with a bounty used to partially reward the voters and certifiers;
3. **Vote a Proposition**: when the voter places a stake, a random proposition taken from those available is provided, using a salt, the vote is hashed and registered;

4. **Certify a Proposition**: when the certifier places a stake, all the available proposition are showed and he/she can chose one to certify, using a salt, his/her certification is hashed and registered;

5. **Check a Proposition**: after an interaction with one or more proposition, the user can check the state, the outcome and the eventual reward earned.

When the proposition request are satisfied, the voting is closed and all the voters and certifiers have to reveal their vote, giving the salt used during the voting, to get any reward.

In the next subsections let's see the above mentioned tasks in more detail.

4.1.2 Submit a Proposition

After the subscription, a user can propose a news to the oracle using the bounty as a payment for the task done by the voters and certifiers. The proposition, once submitted is available for the *voting phase* and it is added to the available propositions list. After `n` votes, the proposition is automatically closed and it will begin the *reveal phase*. In this case, the proposition is popped by the available ones and it is pushed into the closed propositions list. The certifiers act in the same phase as the voters, but are not necessary to the scope of the vote. Once all the voters and all the eventual certifiers have revealed their vote, the proposition is closed, the scoreboard is created and the rewards are distributed across the voters.

4.1.3 Vote a Proposition

The voters have the main role in this application. The outcome of every single proposition is based on their work. Each vote is weighted proportionally of the stake submit requesting the proposition and the user's reputation, so more money you place, more your vote will count. TISRDBO protocol is based on the fact that also if you place the max you cannot control the vote, so there are no problem, moreover, we set a limit of the stake to avoid problems. After placing the stake, a random proposition is given and the voter has to vote `True` or `False`, set a salt to hash the vote and then he/she has to given a percentage of how much the voter believe that the outcome will be `True`, the vote is given by this tuple, called *Response Tuple*. For example, if I am very sure that a news is fake because I found some kind of proof, I will vote `False` and I will set to 0 this percentage. This value will decide your score based on two computation we will see later on.

4.1.4 Certify a Proposition

While the voters are needed, the certifiers are not, but to guarantee the truthfulness of a propostition outcome, they are very important. Concerning the voters, the certifiers are those people who can lose a lot more, but in exchange, if their certifications are correct a higher reward is guaranteed. After the stake is placed, the certifier receive the list of the available proposition and he/she can chose one. If no one are suitable, the certifier can wait and check each time, the stake will be always there. After the choice, the vote can be placed setting the salt to hash it.

4.1.5 Check a Proposition

Each user, after a transaction is made, can check the proposition which he/she has interacted with. If the proposition has been submitted, it can be found in the same menu where it has been; if the proposition was that one for which a vote has been submitted, it can be found on the menu of the voting system; the same for that one has been certified. In each section, the status of the propositions are available, along with the outcome and the eventual reward, if it has already been closed.

4.1.6 Bounty and Stake

One of the condition of the ASTRAEA protocol was about the differences between the stakes placed by voters and those placed by certifiers. The certifiers' stakes have to be a lot bigger than voters', in order to guarantee the correctness of the protocol. A certifier has to play a high-risk/high-reward game in order to be the most honest possible avoiding voters coordinated strategies, while voters play a low-risk/low-reward game. This brings us to create a mechanism that guaranteed the division described above.

The **stake function**, a function represents a parabola, which increments exponentially w.r.t to the reputation `r`, is used to compute the stake to pay depending on the reputation of the user:

$$s(r) = 10^3 * (r^2 - 2 * (r - 1))$$

So, the range that includes the stake that a voter or certifier has to pay to vote or certify a proposition is determined as described below:

- **The minimum stake a voter has to pay** is given by the function using his/her reputation.
- **The maximum stake a voter can pay** is given by the maximum reputation set on the oracle (in our prototype: 100)
- **The minimum stake a certifier has to pay** cannot be lower than the maximum stake of a voter, so the reputation used to make the minimum certifier's stake computation is: $10^3 * (maxReputation + certReputation)$
- **The maximum stake a certifier can pay** is $10^4 * (maxReputation)$

This guarantees that a more respected voter has to risk more in order to vote, because the voter is earning more and his/her vote has an higher weight thanks to his reputation. Obviously, the certifiers range is completely on another level and the risk for certifiers will be very high.

The minimum **bounty** a submitter has to pay to submit a proposition, on the other hand, needs to be high in order to satisfy a minimum certifiers' reward and at least an half of the voter scoreboard reward, so this minimum has to be set.

Assuming that max_{cs} is the maximum stake a certifier can pay, max_{vs} is the maximum stake a voter can pay, min_{vs} is the minimum stake a voter has to pay, $potential_{cn} = num_{users} - max_{voters}$ is the potential certifier number in the oracle and max_{vr} is the maximum voters' reward considering the worst case (a voter with the current maximum reputation pays the maximum stake he can pay). To compute the minimum bounty, the oracle calculates two parameters:

$$min_{cb} = max_{cs} * (1 + potential_{cn})$$

$$min_{vb} = max_{vr} * (max_{voters}/2) - max_{vs} * (max_{voters}/2) - min_{vs} * (max_{voters}/2)$$

Where min_{cb} is the minimum bounty due to the certifiers' reward, and min_{vb} is the minimum bounty due to the voters' reward. Finally, the minimum bounty is calculated as below:

$$min_{bounty} = min_{cb} + min_{vb}$$

4.1.7 Main Phases

The protocol works in three main phases:

1. **Submission Phase**, when the submitter writes a proposition to submit placing a bounty; the proposition is added to the available proposition list and it became part of the voting-game.

2. **Voting Phase**, when the proposition is available, the random function can return its identifier to a voter who placed the stake, or a certifier can chose it freely; when a fixed number of voters pick the proposition and gives their votes the voting phase is concluded. The vote and the certification is sealed in order to avoid the *herd behaviour problem*, which is intrinsically linked to voting systems, in particular when a transparent environment is used (i.e. blockchain), malicious and lazy voters could read the past transactions in order to get others' votes and hence vote w.r.t the majority. To avoid this problem it is enough hash the vote with a salt given by the user, the hash function used is `keccak256`.

3. **Reveal Phase**, when the vote is ended, voters and certifiers are called to reveal the vote, this is possible only if they check the status of the propositions voted, where a message will tell them to insert their salt in order to reveal their vote. Adding the salt the hashed vote will be compared and the right vote will be stored.

4. **Closing Phase**, only when all the voters and certifiers have revealed their vote, the system will compute the scores of each user and the scoreboard will be filled.

4.1.8 Scoring System

After the time limit of the reveal phase is ended, the proposition outcome and the score of each voter are computed. Similarly as reported on TISRDBO, the total score of a user is based on two different scores:

- **Prediction Score:** Score based on the prediction of the result the user made during the voting phase. RT_i is the response tuple given by the voter i, while $RT_{i'}$ is the response tuble given by a *random* voter i', PR is the prediction (the percentage of how much the voter believes that the outcome will be `True`) and IR is the vote (`True/False`).

  $$u_{i,PR} = R_q(RT_i.PR, RT_{i'}.IR)$$

  R_q is a quadratic function, given q the prediction and w the outcome, the result will given by:

  $$R_q(q, w) = \begin{cases} 2q - q^2 & , \ w = 1 \\ 1 - q^2 & , \ w = 0 \end{cases}$$

- **Information Score:** Score based on the information given by the voter.

  $$u_{i,IR} = \begin{cases} 1 - (P_{-i,1} - RT_i.PR)^2 & , \ RT_i.IR = 1 \\ 1 - (P_{-i,0} - RT_i.PR)^2 & , \ RT_i.IR = 0 \end{cases}$$

  $P_{-i,q}$ is the geometric mean (G) of all the $RT.PR$ with $q = RT.IR$ excluding the voter i:

  $$P_{-i,q} = G(RT_q - \{RT_i\})$$

  The geometric mean is a complex operation to compute on chain, for this reason we decided to implement a simple arithmetic mean instead.

The score of each voter is computed $u_i = u_{i,PR} + u_{i,IR}$ and inserted into the scoreboard with an *in-order insertion* algorithm based on it.

4.1.9 Outcome and Rewards

The vote weight f, as highlight before, is based on the stake s and the reputation r, following this equation:

$$f(s) = \alpha\sqrt{s} + (1-\alpha)(s + \frac{sr}{r_{max}})$$

Moreover, also the reward g is based on the stake s and the reputation r, and it is computed directly after the vote in this way:

$$g(s) = \beta s^2 + (1-\beta)(s + \frac{sr}{r_{max}})$$

α and β are values in the range $[0, 1]$.
The vote is a sub-linear concerning the stake, while the reward is super-linear, this way the vote is balanced for the other and cannot overwhelm the others, while the reward allow to earn a lot more to the stake, but it is not guaranteed. Only the first voters in the scoreboard will earn something. The reputation in this configuration doesn't give a lot more, but is an additional feature to earn more and the users will not waste it.

Certifiers, on the other hand, works similar to ASTRAEA, the reward is always guaranteed if the certification is done correctly, otherwise is lost. All the winner certifiers will take in equal measure a portion of the bounty and a portion of the *lost reward pool*, where all the lost stakes are placed. In this way, the certifiers will have a more risky position, with an awesome reward at the end. The vote is weighted similar to the voters'.

4.2 Implementation

The application is mainly composed of two parts: the smart contract and the python front-end application. The smart contract is an Ethereum-based contract with all the functions useful to submit, vote and certify a proposition. To migrate the contract over the blockchain network we used the Truffle suite with Ganache as test network, which allow us to use a local blockchain network and give us the possibility to log and monitor every transaction in the nodes of the network.

```
2_deploy_contract.js
====================

   Replacing 'DeepThought'
   -----------------------
   > transaction hash:    0xb34744361eb1fc3acc80b3c94b1510d42ee9b043e3b0ae5519ad63b2e6beda73
   > Blocks: 0            Seconds: 0
   > contract address:    0x96E895F9B9deeE4EF58a0C5392228A2B2e2fd42e
   > block number:        3
   > block timestamp:     1624382580
   > account:             0x0bc7A159c35b86D7b50A84b06f55D31bA67bA3a8
   > balance:             99.93635744
   > gas used:            2947847 (0x2cfb07)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.05895694 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.05895694 ETH
```

Figure 4.1: The smart contract deployed on the test network

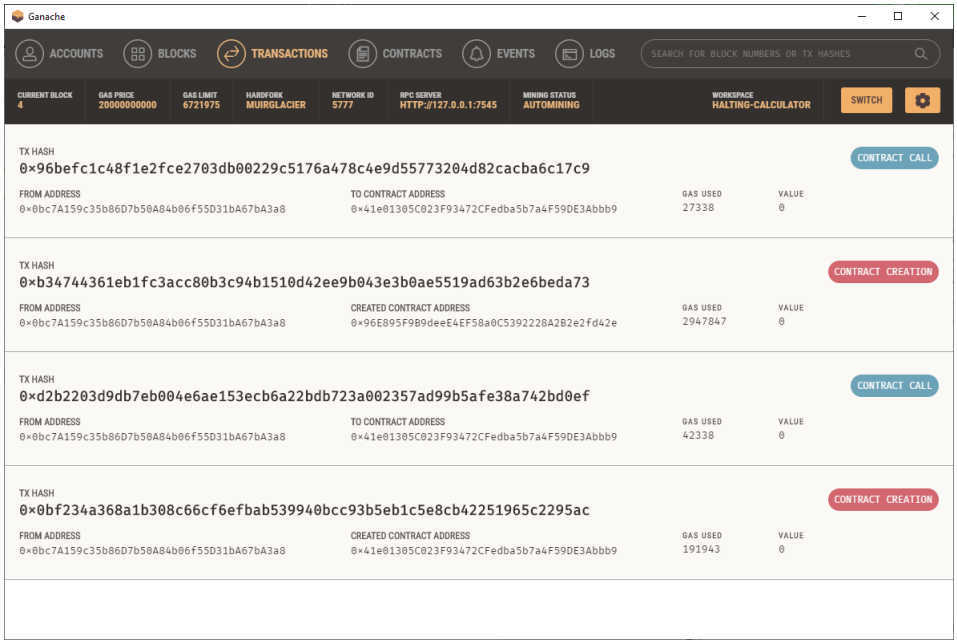

Figure 4.2: Transaction of the deploy

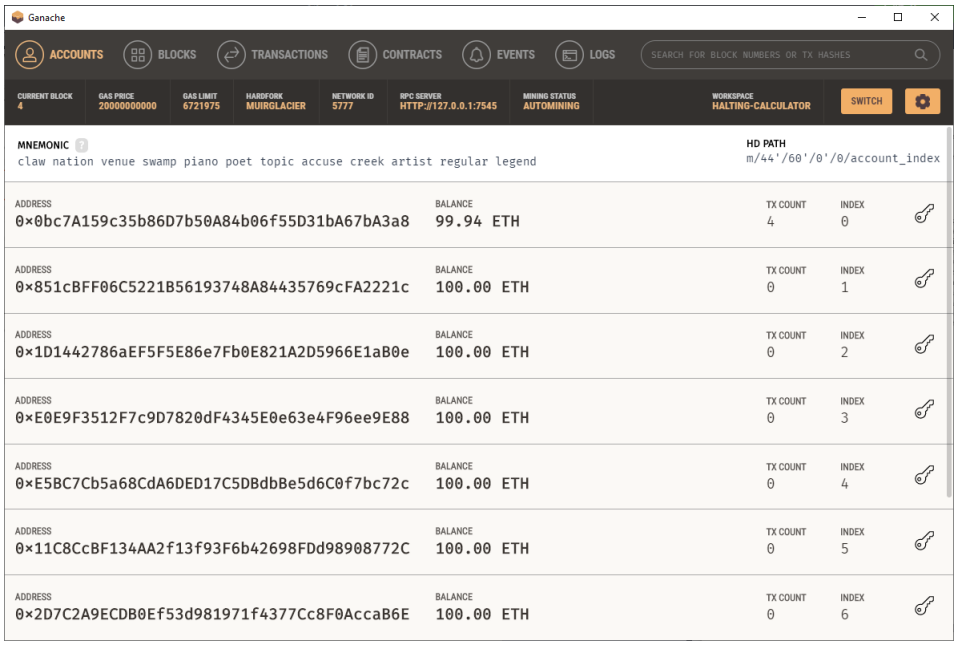

Figure 4.3: Blockchain nodes

The front-end application is coded in Python and allows us to easily interact with the smart contract, in the simple menu we can select every node of the chain and use it to either submit, vote or certify a proposition as long as the node has subscribed to the service and has some ETH tokens in its wallet.

4.2.1 Smart Contract

The smart contract is coded in Solidity version 0.8.0. Solidity is an object-oriented language for implementing contracts that runs on the Ethereum network. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. We used one of the latest version because it included the SafeMath library, a useful library to prevent the overflow during mathematical operations with integers.
We based our code structure on the only available resource, which is the Shintaku implementation, that is available on a public repository.

4.2.2 Script

To facilitate the testing of the prototype and increase its ease of use, it was decided to implement a script in Python that allows you to interact with the functions implemented in the Smart Contract Solidity and acts as a real interface between the user and the oracle. The script uses the Web3.py library, a library that allows to interact with Ethereum calling functions of the smart contract or make transactions on the latter. The python implementation part does not contain any logic of the smart contract.

5. Results

In this chapter we will present the results of our research project by commenting on some parts of the code and how the smart contract works.
There will be also showed some vote examples and their results with the respectively rewards assigned to the users of the system.

5.1 The Code

The repository of the code is at https://github.com/marcoDige/RP3project.
We tried to keep the code as clean as possible since one of the main issues with smart contracts is the size of the variables and information stored, so we used the minimum number of variables possible.
In our implementation of the voting oracle we fixed some variables that are worth mentioning because they can be tuned to try to have the most truthful result of the vote of a proposition:

- **max voters** is the maximum number of voters for a proposition, when the maximum is reached the proposition status change from "Open", which is when users can vote and certify, to "Reveal", which is when all the users can send their salt to unseal their secret vote.
- **max reputation** is the maximum reputation that a user in the network can reach.
- **alfa and beta** are the two parameters that are used in the normalization of a user's vote weight and the calculation of his reward amount.

Since all the variables saved in a smart contract are visible and we want to prevent any type of fraud or change in the user's vote behaviour we implemented a vote sealing mechanism that is based on the commit-reveal scheme.
The user's vote is sealed in the Python application using an hashing function that takes as input the user vote and a *salt*, a secret keyword that the user has to remember. The sealed vote is then sent to the smart contract and stored. When the voting phase for a proposition ends all the users have to reveal their vote to obtain their reward, so the user sends to the contract his salt. The smart contract then try to hash both of the voting options with the user's salt and if the result matches the information stored in the memory it means that the option was the user's vote.

5.2 Examples of working schemes of DeepThought

5.2.1 Vote Example 1

For the first proposition to vote we used a maximum number of voter of 2, in this case the submitter sends a proposition with a bounty of 0.0010288217582604 ETH.

```
MAIN MENU
HOW DO YOU WANT TO CONTINUE?
1 - Subscribe
2 - Sign in
3 - Your account info
4 - Quit
Enter here: 1
Now you are subscribed!

MAIN MENU
HOW DO YOU WANT TO CONTINUE?
1 - Subscribe
2 - Sign in
3 - Your account info
4 - Quit
Enter here: 2

ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 3

SUBMITTER MENU
1 - Submit a proposition
2 - Show submitted propositions state
1

SUBMIT A PROPOSITION
proposition id: 1
proposition content: is this news fake?
bounty (ETH) [0.0010288217582604 ETH, your balance]: 0.0010288217582604
```

Figure 5.1: Submitter user sending a proposition

The users can now start to vote, the first voter ask for a random proposition to vote and stake 1000 WEI for receiving it. After he receive the proposition proceed to vote True with a 80% prediction on the proposition truthfulness.

```
ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 2

VOTER MENU
1 - Make a vote request
2 - Show voted propositions state
3 - Go back to the MAIN MENU
1

VOTING REQUEST
stake (wei) [1000 wei, 9802000 wei]: 1000

You can vote the proposition 1 (content: "is this news fake?")

VOTE THE PROPOSITION
Vote (True/False): False
Prediction [0 %,100 %] (your prediction on the proposition truthfulness): 30
Insert your salt (REMEMBER IT!): salt
Nice! your vote has been recorded
```

Figure 5.2: User receive and vote a proposition

The second user also act as a voter but in this case stake 1000 WEI and vote that the proposition is False with a 30% prediction.
We also used another user to certify the proposition, in this case the certifier can chose the proposition to vote and stake 0.000010200798002 ETH to certify it as True.

```
ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 1

CERTIFIER MENU
1 - Make a certification request
2 - Show certified propositions state
3 - Go back to the MAIN MENU
1

CERTIFICATION REQUEST
stake (wei) [0.000010200798002 ETH, 0.000999998000002 ETH]: 0.000010200798002

PROPOSITION YOU CAN CERTIFY
Proposition id : content
1 : is this news fake?

Which proposition do you want to certify?
Proposition id: 1
Vote (True/False): True
Insert your salt (REMEMBER IT!): salt
Nice! your certification has been recorded
```

Figure 5.3: User choose and certify a proposition

Since the maximum number of voter as been reached the proposition's status become "Reveal" and both the voters and the certifiers have to send the salt they used to hash their vote to reveal it. The proposition result will then be computed.

```
ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 1

CERTIFIER MENU
1 - Make a certification request
2 - Show certified propositions state
3 - Go back to the MAIN MENU
2

PROPOSITION ID -> STATUS
1 -> Reveal
Do you want to reveal your vote about the "Reveal" proposition?(y/n): y
Proposition id: 1
Insert your salt to reveal your vote (YOU HAD TO REMEMBER IT!): salt
```

Figure 5.4: The certifier reveal his vote

In our case the result is Unknown since the voters gave a discordant vote and their weight was equal. In this case the bounty will be gave back to the submitter and the voters will get back their stake, the certifiers are the only one to lose the money they staked. This money will be put in a lost reward pool and used to reward the next propositions voting.

```
ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 3

SUBMITTER MENU
1 - Submit a proposition
2 - Show submitted propositions state
2

PROPOSITION ID -> STATUS
1 -> Close (result: Unknown)
```

Figure 5.5: The proposition result is Unknown

5.2.2 Vote Example 2

In the second vote example that we provide we set the maximum voters number to 3. In this case the submitter set a bounty of 0.0010288217582604 ETH to vote for his proposition.

The user acting as a certifier choose the proposition and stake 0.000012200798002 ETH to certify it as True like in the previous example.

```
ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 1

CERTIFIER MENU
1 - Make a certification request
2 - Show certified propositions state
3 - Go back to the MAIN MENU
1

CERTIFICATION REQUEST
stake (wei) [0.000010200798002 ETH, 0.000999998000002 ETH]: 0.000012200798002

PROPOSITION YOU CAN CERTIFY
Proposition id : content
1 : This is not a fake news

Which proposition do you want to certify?
Proposition id: 1
Vote (True/False): True
Insert your salt (REMEMBER IT!): salt
Nice! your certification has been recorded
```

Figure 5.6: User choose and certify a proposition

5.2.3 Vote Example 3

Then the three users acting as voters start to vote a random proposition, since in our test there is only one they will vote for the same proposition that the certifier chose. The first voter stakes 1000 WEI and vote that the proposition is True with a 85% prediction of truthfulness.

```
ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 2

VOTER MENU
1 - Make a vote request
2 - Show voted propositions state
3 - Go back to the MAIN MENU
1

VOTING REQUEST
stake (wei) [1000 wei, 9802000 wei]: 1000

You can vote the proposition 1 (content: "This is not a fake news")

VOTE THE PROPOSITION
Vote (True/False): True
Prediction [0 %,100 %] (your prediction on the proposition truthfulness): 85
Insert your salt (REMEMBER IT!): salt
Nice! your vote has been recorded
```

Figure 5.7: User vote the proposition as True

The second user also stakes 1000 WEI and vote the proposition as True with a 70% prediction.
The last user stakes 1200 WEI but vote that the proposition is False with a 80% prediction of truthfulness, this means that the user voted False because he believes it, even if he thinks that the majority of users will more likely vote for a True outcome.

```
MAIN MENU
HOW DO YOU WANT TO CONTINUE?
1 - Subscribe
2 - Sign in
3 - Your account info
4 - Quit
Enter here: 2

ROLE MENU
WHAT ROLE DO YOU WANT TO TAKE?
1 - Certifier
2 - Voter
3 - Submitter
4 - Your account info
5 - Go back to MAIN MENU
Enter here: 2

VOTER MENU
1 - Make a vote request
2 - Show voted propositions state
3 - Go back to the MAIN MENU
1

VOTING REQUEST
stake (wei) [1000 wei, 9802000 wei]: 1200

You can vote the proposition 1 (content: "This is not a fake news")

VOTE THE PROPOSITION
Vote (True/False): False
Prediction [0 %,100 %] (your prediction on the proposition truthfulness): 80
Insert your salt (REMEMBER IT!): salt
Nice! your vote has been recorded
```

Figure 5.8: User vote the proposition as False

The proposition status becomes "Reveal" and all the users involved will have to reveal the salt that the used to hash their vote. After that the proposition is resolved and the outcome results to be True.

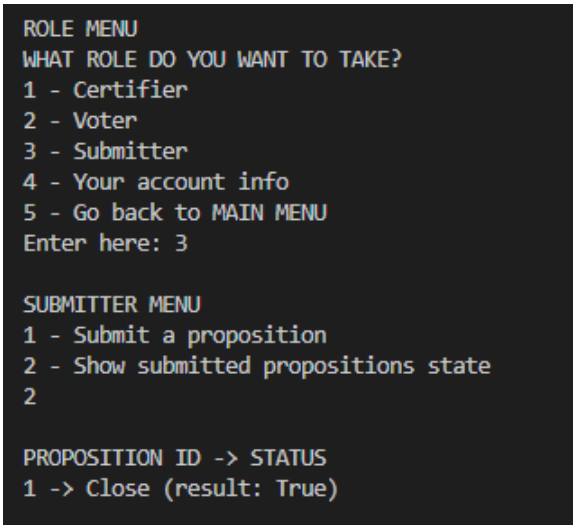

Figure 5.9: The submitter checks the result of his proposition

The users can also check the result of the propositions they voted or certified and see how much they earned as a reward. In this case the certifier got rewarded with 0.001012198798004 ETH. The two users that voted a True outcome were rewarded with 300700 WEI each and the user that voted False got rewarded with 432840 WEI. This proves that the scoreboard that the smart contract use to create the ranking of users that have to receive the reward incentives honest voting even when a user thinks that the result of the proposition will be different from his beliefs.

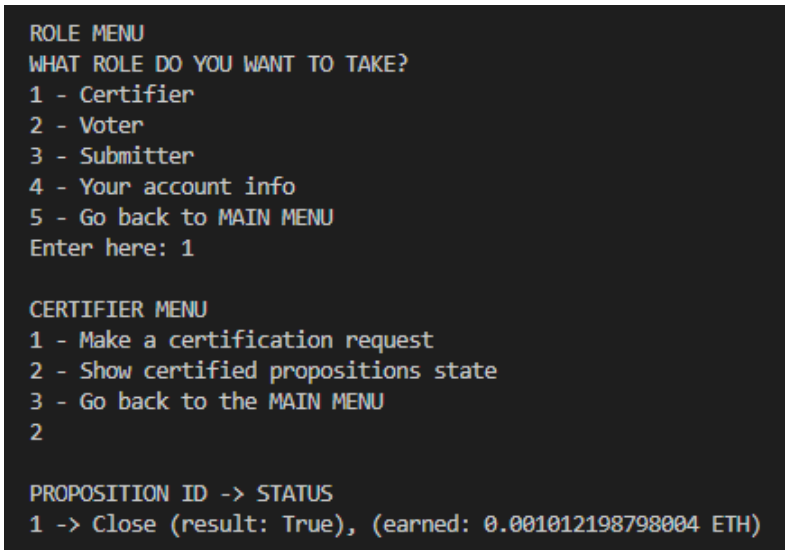

Figure 5.10: A certifier checks the proposition result

6. Conclusions

With this research work, we performed a massive study of the state of the art in the domain of decentralized oracles. After identifying the weaknesses of such domain, we both suggested a new approach on distributed oracles, with a combination between voting based oracles and reputation based, but we also provided an implementation of a smart contract that realizes it while most of the papers and the state of the art's works are not implemented yet.

ASTRAEA starts introducing the problems and proposes a way to resolve them, in particular the lazy voting, **TISRDBO** found a way to reduce the Sybil-attack incrementing the rewards, while **Shintaku** was the first to develop a working oracle.

The prototype proposed tries to develop a working oracle based on the *peer prediction-based protocol* (TISRDBO), introducing the reputation factor. The choice made about the reintroduction of certifiers, avoiding the *proposition pair*, is given by the fact in some circumstances could not be useful.

When information cannot be automatically checked, the only way to confirm its veracity is through a net of people. Doing so, many problems arise, but using rewards and game theory notions, this problem can be easily solved, at least theoretically speaking. The oracles are an important feature of the blockchain world, which can certify the truthfulness of a piece of information going on-chain from the outside world, or vice-versa. The studies on this field are currently going on, and each day more research is published with an interesting result. We believe, finally, that this work can give way to a new strand of oracles that mixes the strengths of voting-based and reputation-based.

6.1 Future Improvements

Certainly, for the purposes of further validating this work, it would be useful to make a comparison with other existing decentralized oracles (such as *ASTRAEA* and *TISRDBO*), implementing a version of them (have not yet been implemented) or waiting for third-party implementation. This comparison would help identify which exploited protocols actually make the oracle we structured more reliable or less reliable. An interesting comparison could also be done with *Shintaku*, the only other prototype ever developed.

Moreover, many challenges are still open:

- **Randomization** in decentralized systems, a pseudo-random function is proposed in this research, but it is based on timestamps and block numbers, it is not very easy, but it could be exploited in order to get the number wanted. Many functions have been developed (see **Randao** [20]), and many more have to arrive, but still no one works correctly.
- **Temporal Slot**, in this project a max number of voter is fixed in order close the voting phase, and all the voters needs to reveal for ending the reveal phase, a huge improvement could be done adding a timer function in order to set each phase in a certain temporal slot and closing it at the end. Many problematic behaviour could arise adding this feature.

A lot more can be done to find a perfect voting-based oracle, but after each new research, the goal becomes closer.

Bibliography

[1] "ANSAcheck". Available: https://www.ansa.it/sito/static/ansa_check.html.

[2] Provable documentation. [online]. Available: https://docs.provable.xyz/.

[3] "Safe.press | IBM". Available: https://www.ibm.com/case-studies/safe-press-blockchain.

[4] "TLSNotary - a mechanism for independently audited https sessions". Available: https://tlsnotary.org/TLSNotary.pdf.

[5] J. Adler. "The state of decentralized oracles", 2018. Available: https://media.consensys.net/the-state-of-decentralized-oracles-df45bf0dc51d.

[6] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania. "Astraea: A decentralized blockchain oracle". In *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)*, pages 1145–1152, 2018.

[7] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic. "Trustworthy Blockchain Oracles: Review, comparison, and open research challenges". *IEEE Access*, 8:85675–85685, 2020.

[8] Abdeljalil Beniiche. "A study of blockchain oracles", 2020.

[9] P. Bisola. "Blockchain oracles explained, 2018. Available: https://www.mycryptopedia.com/blockchain-oracles-explained/.

[10] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, and et al. "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks", 4 2021. Available: https://chain.link/whitepaper.

[11] V. Buterin. "Ethereum: A next-generation smart contract and decentralized application platform", 2014. Available: https://ethereum.org/en/whitepaper/.

[12] V. Buterin. "Slasher: A punitive proof-of-stake algorithm", 2014. Available: https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/.

[13] Y. Cai, G. Fragkos, E. E. Tsiropoulou, and A. Veneris. "A truth-inducing sybil resistant decentralized blockchain oracle". In *2020 2nd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, pages 128–135, 2020.

[14] Giulio Caldarelli. "Understanding the blockchain oracle problem: A call for action". *Information (Switzerland)*, 11:509, 10 2020.

[15] R. Kamiya. "Shintaku: An end-to-end-decentralized general purpose blockchain oracle system", 2018. https://gitlab.com/shintaku-group/paper/raw/master/shintaku.pdf.

[16] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. "Demystifying incentives in the consensus computer". pages 706–719, 10 2015.

[17] Kamran Mammadzada, Mubashar Iqbal, Fredrik Milani, Luciano García-Bañuelos, and Raimundas Matulevičius. Blockchain oracles: A framework for blockchain-based applications. In Aleksandre Asatiani, José María García, Nina Helander, Andrés Jiménez-Ramírez, Agnes Koschmider, Jan Mendling, Giovanni Meroni, and Hajo A. Reijers, editors, *Business Process Management: Blockchain and Robotic Process Automation Forum*, pages 19–34, Cham, 2020. Springer International Publishing.

[18] M. Merlini, N. Veira, R. Berryhill, and A. Veneris. "On public decentralized ledger oracles via a paired-question protocol". In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 337–344, 2019.

[19] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". 2008.

[20] randao.org. "randao: Verified random number generation, v1.10", 2019. Available: `https://github.com/randao/randao`.

[21] W. Shang, M. Liu, W. Lin, and M. Jia. Tracing the source of news based on blockchain. In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pages 377–381, 2018.

[22] N. Szabo. "Smart Contracts: Building blocks for digital markets"., 1996. Available: `http://www.fon.hum.uva.nl/rob/Courses/Information-InSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html`.

[23] N. Szabo. "The idea of Smart Contract". [online]., 1997. Available: `http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html`.

[24] Adán Sánchez de Pedro Crespo, Daniele Levi, and Luis Cuende. "Witnet: A decentralized oracle network protocol", 11 2017.

[25] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Wang. "Blockchain-Enabled smart contracts: Architecture, applications, and future trends". *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2266–2277, 2019.

[26] K. Wüst and A. Gervais. "Do you need a Blockchain?". In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.

[27] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. "Town Crier: An authenticated data feed for smart contracts". 10 2016.