Decentralized Oracle

February 2021

Abstract

Decentralized Oracle (DEOR) is a fully decentralized oracle aggregation network. Unlike existing oracles, it's not centralized or semi-centralized and therefore is more secure. This paper details the connection of on-chain components for smart contracts and the (randomly and appropriately) selected off-chain world. We also describe a reputation scoring algorithm that incorporates randomness to improve the security and integrity of the oracle output and a security monitoring system for DEOR, helping people make informed data provider choices.

Contents

1 Introduction

Smart contracts provide an infrastructure for creating decentralized applications (Dapps) and DAOs (Decentralized Autonomous Organizations) to automate jobs on top of the blockchain. Such jobs are triggered algorithmically and cannot be modified after deployment. However, smart contracts cannot access external data.

Today, the solution to this problem is called an oracle. An oracle connects the off-chain world with smart contracts. This document details the design of DEOR, the problems with current oracles, and how DEOR improves on these solutions. Subsequently, we will delve into how on-chain smart contract components connect with the random selectable off-chain world. We also describe a reputation and security monitoring system implemented within DEOR, which helps people make informed provider selections. Finally, at the end of the paper, the roadmap of the project is put forward with specific objectives being declared, including adding various data sources, responses with various kinds of proofs and developing decentralized oracle services.

Compared to traditional contracts, smart contracts are more secure, as everyone has the same authority, including the author. Smart contracts are automatically executed when they meet the requirements; all parties of the contracts can reach an agreement without trust. This feature turns smart contracts into a superior tool for realizing and implementing digital agreements. It should be noted that smart contracts are trying to digitalize real-world agreements. Consequently, in order to carry out such a task, they need to access real-world data. However, due to the specific nature of the underlying consensus protocols, blockchains are unable to connect to external data sources; therefore, they cannot access data from outside the blockchain. Thus, smart contract developers encounter a connectivity issue according to which most smart contracts cannot function practically.

Decentralization reduces the need for trust among parties of the contracts. DEOR ensures the security of the entire smart contract execution procedure, including obtaining data from off-chain sources.

This is the prerequisite of connecting smart contracts to the real world and taking the place of traditional digital contracts. DEOR can output data securely to off-chain systems, thereby implementing the connection to the real world and ensuring that smart contracts remain tamper-proof.

In this document, Section 2 provides the technical overview how DEOR links the on-chain and off-chain worlds in terms of both off-chain and on-chain components. Section 3 describes how DEOR is secured, with Section 4 concluding the discussion with a wrap-up summary and an insight into the tokenomics.

2 Technical Overview

DEOR aims to link the on-chain and off-chain worlds in a secure manner. We describe the architecture and technical innovations of each DEOR component below. A pictographic overview can be found in Fig. 1

2.1 On-chain

- **Custom Oracle Selection with Randomizer:** Oracle services purchasers evaluate their specific requirements, then select nodes and services that can fulfill their requirements (node-related data is available in the list for consumers to choose appropriate nodes and services); however, manual matching is not possible in all cases.

  In this case, the oracle selects the voting node with a randomizer. Seventy percent of the oracles are randomly selected per transaction. By design, not all oracles participate in voting, so DEOR can respond effectively to Sybil attacks. In a Sybil attack, the attacker subverts the reputation system of a network service by creating a large number of pseudonymous identities and uses them to gain a disproportionately large influence. A reputation system's vulnerability to a Sybil attack depends on how cheaply identities can be generated, the degree to which the reputation system accepts inputs from entities that do not have a chain of trust linking them to a trusted entity, and whether the reputation system treats all entities identically.

An entity on a peer-to-peer network is a piece of software that has access to local resources. An entity advertises itself on the peer-to-peer network by presenting an identity. More than one identity can correspond to a single entity. In other words, the mapping of identities to entities is many to one. Entities in peer-to-peer networks use multiple identities for purposes of redundancy, resource sharing, reliability, and integrity.

In peer-to-peer networks, the identity is used as an abstraction so that a remote entity can be aware of identities without necessarily knowing the correspondence of identities to local entities. By default, each distinct identity is usually assumed to correspond to a distinct local entity. In reality, many identities may correspond to the same local entity. An adversary may present multiple identities to a peer-to-peer network in order to appear and function as multiple distinct nodes. The adversary may thus be able to acquire a disproportionate level of control over the network, such as by affecting voting outcomes.

DEOR is implemented with a randomizer, which grants excellent robustness against Sybil attacks.

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

- **Data Aggregation:** Once the oracles have revealed their results to the oracle contract, their results will be fed to the aggregating contract. There will be no universal aggregator contract. For every demand, the result can be different. DEOR will include a standard (i.e. a template) for users to customize their contracts. There are currently two types of data aggregation in DEOR:

price feed and data query. There two types of data aggregation in current DEOR: price feed and data query.

2.2 Off-chain

The off-chain component of the DEOR network is the oracle node. It is connected to the Ethereum network, and it will support all leading smart contract networks. The DEOR oracle nodes are powered by the standard open-source core implementation, which handles standard blockchain interactions, scheduling, and connecting with common external resources.

- **DEOR Core:** The DEOR oracle off-chain node's core is responsible for interacting with the blockchain, assignment scheduling, and balancing work across its various external services. Each assignment is a set of smaller job specifications, known as subtasks, processed as a pipeline. Each subtask passes its result to the next subtask; they run in tandem to get the final result.

- **External Adapters:** Users can customize subtasks within an external adapter. Beyond the built-in subtask types, custom subtasks can be defined by creating adapters. Adapters are external services with a minimal REST API.

- **Subtask Schemas:** With the applications of DEOR becoming wider, there can be more open-source adapters, all available for public review. DEOR currently operates with a schema system, based on JSON, to specify what inputs each adapter needs and how they should be formatted.

3 Oracle Security

In order to explain DEOR's security architecture, we must first explain why security is important and what it means. If a smart contract gets a false data feed, it will likely behave in an undesired manner.

For instance, insurance fraud could occur if smart contract insurance data feeds could be tampered with by the insured party. More generally, a well-functioning blockchain, with its ledger or bulletin-board abstraction, offers very strong security properties. An oracle must therefore serve users as an effective trusted third party, providing correct and timely responses with very high probability. There are four ways through which DEOR offers security; a decentralized reputation system, secure data aggregation, secure oracle nodes and voluntary contract upgrades.

3.1 Reputation System

The reputation system proposed for DEOR would record and publish user ratings of oracle providers and nodes, offering a means for users to evaluate oracle performance holistically. Reputation metrics should be easily accessible off-chain where larger amounts of data can be efficiently processed and more flexibly weighted. For a given oracle operator, the reputation system is initially proposed as supporting the following metrics, both at the granularity of specific assignment types and also in general for all types supported by a node:

- Total number of assigned requests: The total number of past requests that an oracle has agreed to, both fulfilled and unfulfilled.
- Total number of completed requests: The total number of past requests that an oracle has fulfilled. This can be averaged over the number of requests assigned to calculate the completion rate.
- Total number of accepted requests: The total number of requests deemed acceptable by calculating contracts compared with peer responses. This can be averaged over total assigned or total completed requests to get insight into accuracy rates.

The reputation score affects not only the voting but also the rewards for true voting oracles. The weighting system uses the idea of levels, as can be seen in Table 1.

Level	Reputation Score
5	1.0$\leq x < 5.09$
4	5.09$\leq x < 7.7$
3	7.7$\leq x < 9.18$
2	9.8$\leq x < 9.83$
1	9.83$\leq x \leq 10.0$

Table 1: Reputation Table

In this, oracle's level can be easily upgraded from Level 5 to Level 4. But it is very difficult to be upgraded from Level 2 to Level 1. We define 5 Levels and set 10.0 as the max reputation score:

$$\beta \sum_{i=1}^{i \leq MaxLevel} (\frac{i}{2})^2 = Score_{max} - Score_{min}, Score_{max} = 10.0, Score_{min} = 1.0 \tag{1}$$

The reputation score line is divided into 5 levels according to:

$$Low_{level} = Score_{max} - \beta \sum_{i=1}^{i \leq MaxLevel} (\frac{i}{2})^2, \sigma = 0.01 \tag{2}$$

The additional score point σ is the point which is added to the score when an oracle does a true voting, σ increases as the score increases. When an oracle does a false voting, its reputation score is set to the lower score limit of the level underneath its level prior to the voting. An example of re-weighting is shown in Fig. 2.

3.2 Secure Data Aggregation

We can obtain data from several different data sources to mitigate the impact of an abnormal data source. An aggregating function can aggregate the results into a single output. There can be many ways to do aggregation, such as calculating the weighted average after removing abnormal data.

3.3 Secure Oracle Node

Several nodes form the oracle network, and each node has its data source set, which may overlap with the others'. An oracle aggregates data from its data sources and sends the aggregated result to the request. A request may choose several nodes to ensure accuracy. As faulty nodes may exist, there should be a plan to mitigate the influence. In our case, voting oracles are selected by a randomizer, and their votes have their own weight-reputation score. If the sum of upvoting weights is over 70 percent of the total weight, it is verified as true, and the answer is verified data by oracles. There is a security monitoring system for oracle nodes, which calculates average response time and last active time. Also, each oracle must have to pay penalty payments. If penalty payments were locked in to assure a node operator's performance, the result would be a financial metric of an oracle provider's commitment not to engage in an "exit scam" attack, where the provider takes users' money and doesn't provide services. This metric would involve both a temporal and a financial dimension.

3.4 Contract-upgrade service

As recent smart contract hacks have shown, coding bulletproof smart contracts is an extremely challenging exercise. No one can control the actions of smart contracts once deployed; this makes the security of the oracle important. For instance, a decentralized exchange can suffer a massive loss if it receives incorrect data from an oracle.

DEOR proposes a contract-upgrade service for security reasons.

The service will be run by the organizations that launch DEOR nodes and follow DEOR's decentralized design philosophy. The contract-upgrade service is non-mandatory; users decide whether to turn this feature on according to their demand. Migration of users to new oracle contracts functions as a kind of "escape hatch" - something long advocated for by blockchain researchers as a mechanism to fix bugs and remediate hacks, without resorting to cumbersome approaches such as white hat hacking or hard forks.

4 Conclusion

This document introduced the fully decentralized oracle network, DEOR, and described its on-chain and off-chain components. We interpreted our security and decentralization schemes, described them in the context of oracles' existing design flaws, and gave future developments.

Tokenomics

- DEOR Tokens: DEOR oracle has issued its dedicated tokens dubbed DEOR in the Ethereum blockchain. The purpose of this token is to create a more affordable payment method for clients to pay for DEOR oracle services with a significant discount in comparison to paying with ERC20 tokens.

- Token Distribution Policy: The basic principle which has been observed in distribution policy is implementing maximum dispersion and avoiding centralization.

- In this procedure, twenty percent of tokens will be distributed in a fair distribution event that lasts for seven days. Twenty percent of the tokens are for the continued development and updating of DEOR. Fourty percent will be used for vote mining and propagation of network effects from various oracles and aggregators. The final twenty percent is for incentivization and fair distribution by traditional pool methodology.

20%	Fair Distribution
20%	Pool Mining
20%	Development
40%	Vote Mining

Table 2: Token Distribution

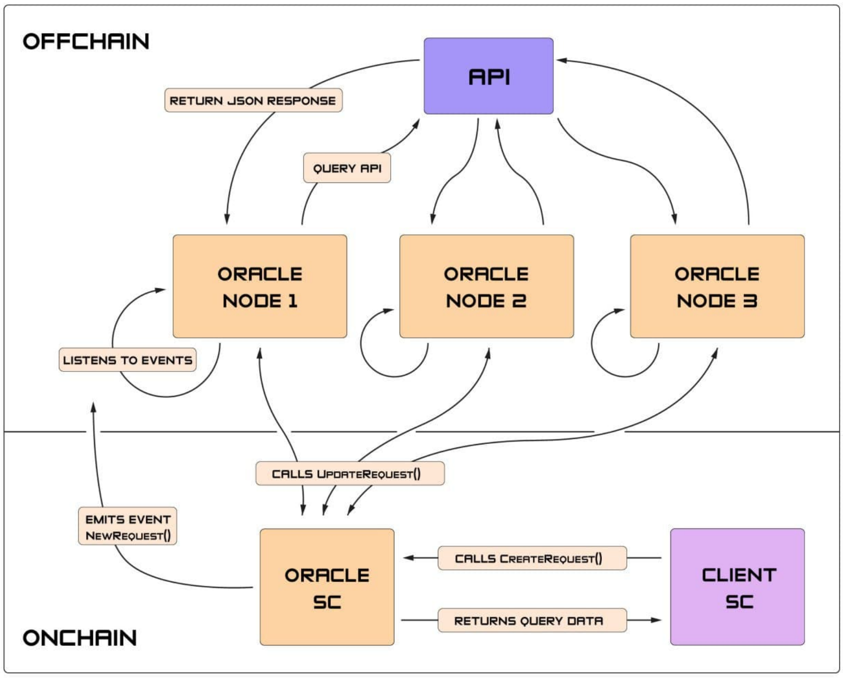

Figure 1: Overview

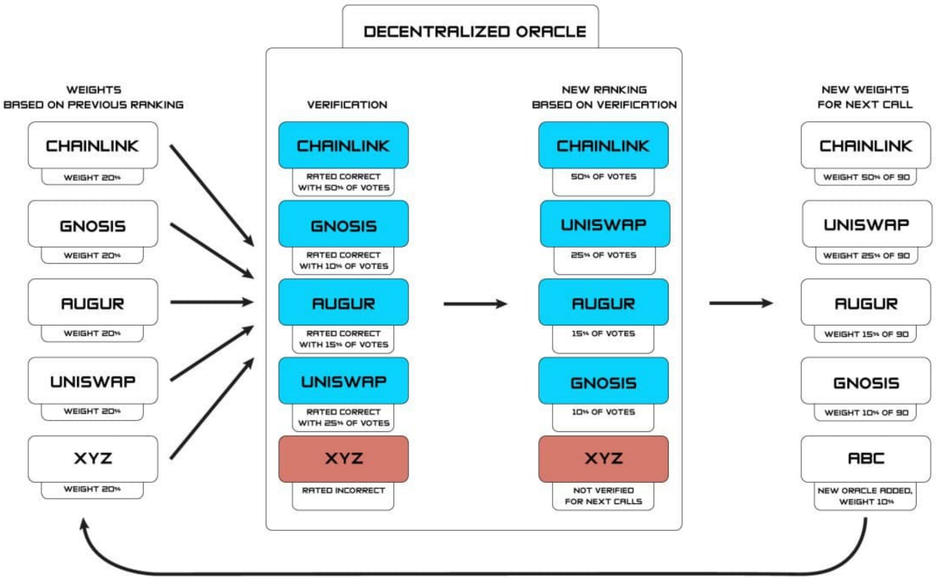

Figure 2: Weights example