Decentralized Oracle with Mechanism for Reputation and Incentive

Liao Chen[1,a], Haomin Zhuang[1,a] and Yuxiang Feng[1*]

[1]School of Software Engineering, South China University of Technology, Guangzhou, Guangdong, 510000, China

[a]These authors contributed to the work equally and should be regarded as co-first authors

[*]Corresponding author's e-mail: yxfeng@scut.edu.cn

Abstract. Blockchain was originally designed as a peer-to-peer network to handle cryptocurrency transactions. And with the advent of smart contracts, more and more applications are using Blockchain as an infrastructure. The smart contract provides a mechanism to automatically execute code according to programmed conditions without requiring other reliable third-party intervention. However, smart contracts have a nonnegligible limitation in that they can only operate on data that is on the Blockchain. This is why we need trusted entities called oracle which is in charge of writing external data into Blockchain. To avoid security issues such as malicious oracles writing incorrect data on Blockchain. This paper introduced a decentralized oracle based on reputation and incentive mechanism. To evaluate the feasibility and reliability of the proposed scheme, a simulation-based experiment was conducted. The results showed that our solution can well defend against attacks and penalize those malicious oracles.

1. Introduction

Blockchain, the underlying support technology for Bitcoin, first appeared in Satoshi Nakamoto's "Bitcoin: A Peer-to-Peer Electronic Cash System"[1], a text that describes in detail how to create a new, decentralized, peer-to-peer transaction system that does not require a foundation of trust. Smart contract[2] is a special computer protocol designed to digitally facilitate, validate or enforce contract negotiation and performance, which is of great significance to Blockchain.

A primary limitation of Blockchain is its inability to interact with the "outside world"[3]. This is because Blockchain is deterministic and consistent while the outside world is uncertain which means applications on blockchain do not have direct access to data from the outside world.

To overcome this limitation of smart contracts, a mechanism for writing external data to the Blockchain is needed. And this mechanism is known as Oracle[4,5,6]. These oracles are represented by smart contracts on the Blockchain that respond to data requests for other smart contracts. The main function of oracles is to bring external data into Blockchain. External data can be a stock price at a certain time, game results, weather conditions, and so on. Oracle collects information and sends it into the Blockchain in the form of transactions so that the Blockchain can interact with the outside world. In summary, oracle is the middleware between the real world and Blockchain.

2. Related Methods

2.1. Centralized Oracle

Provable[7], also known as Oraclize, is an Amazon Web Services-based Oracle service designed to provide data feedback for smart contracts and Blockchain applications. Town Crier[8] is a centralized oracle that provides Ethereum-based data feedback. It focuses on providing reliable data feedback for smart contracts and Blockchain applications through Inter Software Guard Extension. However, Town Crier is limited by the types of APIs and data feeds it can provide. And due to its centralized structure, it is also susceptible to the single point of failure.

2.2. Decentralized Oracle

ASTRAEA[9] is a decentralized oracle that determines the trustworthiness of data submitted by voters mainly by using different roles (voter and verifier) for voting. By splitting users' roles and specific incentives, it can resist malicious voting attacks to a certain extent. Chainlink[10] is originally a decentralized oracle network on the Ethereum platform. Its main purpose is to provide reliable data to smart contracts by accessing key data resources through a specified API. In addition to this, Augur[11], is a decentralized blockchain-based prediction marketplace. Augur is strongly influenced by Truthcoin[12], which inhibits the fraudulent behavior of participants by introducing a reporting and dispute system. Witnet[13] is a reputation-based decentralized oracle network protocol that enables connection between the smart contract and external data providers. Witnet's reputation system rewards the successful majority consensus witnesses while penalizing the erroneous witnesses. Aeternity[14] is an open-source decentralized application platform that utilizes a decentralized oracle system, which can be used to request and access real-world data from various providers.

3. Our Proposal Oracle

In this paper, we propose a new scheme the reputation mechanism and incentive mechanism work together to deal with the robust performance design of a decentralized oracle network. In our proposal, the adaptive reputation algorithm can aggregate the result from more professional oracles, which prevents attackers' oracles from joining a network at a low cost. Meanwhile, the incentive mechanism introduces gambling and reputation to calculate their prize, which forces

them to give a guarantee of good behavior to receive more rewards. Our proposal is adaptive to most situations and reduces human participation.

Table 1. Explanation of symbols.

Symbol	Meaning
$\gamma_{i,j}$	The reputation of i-th oracle after j-th task
β	The coefficient of attenuation
$\alpha_{i,j}$	The reputation value i-th oracle acquires in the j-th task
$score_{i,j}$	The score of i-th oracle in the j-th task
$stake_{i,j}$	The amount of stake from i-th oracle in the j-th task
$prize_{i,j}$	The amount of prize from i-th oracle gets in the j-th task
$pool_j$	The amount of prize pool in the j-th task
K	There are K oracles in the network
K_W	There are K correct oracles in the j-th task
$result_{i,j}$	The result from i-th oracle in the j-th task
$result_j$	The result of j-th task after aggregating
$P_{x,j}$	The sum of the proportion of oracles in $D_{x,j}$
$D_{x,j}$	The class with oracles' $result_{i,j}$ are x in the j-th task

3.1. Reputation Mechanism

The reputation mechanism is related to the aggregation of the result and the calculation of the reward. So it will encourage oracles to provide reliable service consistently with less downtime.

Every oracle in the decentralized oracle network owns a value of reputation, which is changing according to the performance of the oracle in previous tasks. The reputation takes oracles' recent performance into consideration because we tend to trust the most reliable oracles that provide the correct result in the past. But even the most honest oracles will experience downtime. So in this paper, the calculation considers time as a dimension of reputation and assigns high reputation values to prophetic machines that have performed well recently.

When a new oracle first joins the decentralized oracle network, the reputation of the oracle is set to 0, which means that the oracle should accumulate its reputation through the correct answers in the later tasks. The reputation of one reputation is changed after a task. Oracles providing the correct answer in the task can get 100 reputation score in this task, while oracles providing wrong answers get 0 reputation score. The reputation in this task has multiplied by a coefficient and then added the historical reputation is multiplied by another coefficient. The historical reputation declines by multiplying a coefficient and adds to the reputation given in a new task to form a new reputation using in the next task.

$$\gamma_{i,j} = \beta * \gamma_{i,j-1} + (1-\beta) * \alpha_{i,j} \tag{1}$$

$$\alpha_{i,j} = \begin{cases} 100 & True \\ 0 & False \end{cases} \tag{2}$$

One of the advantages of using this model is that there is no need to keep reputation scores for each past task. And a simple linear equation is used as opposed to a complex non-linear equation, in order to accommodate the smart contract computation capabilities in a practical manner.

Here the $\alpha_{i,j}$ is 100 if the i-th oracle provides the correct result in the j-th task. But if the result is incorrect, it should be 0. From the formula, a high reputation score oracle will drop down quickly if it has downtime and could not provide a correct result.

3.2. Incentive Mechanism

The more oracles the network has, the more robustness it has. In order to attract more oracles, our decentralized oracle network would have an adaptive incentive mechanism to organize the prize, which is an economic reward like bitcoin or the token in the blockchain, or money in real. The incentive mechanism is a supplement to the reputation mechanism.

At first, gambling is a fascinating method not only for the oracles giving their guarantee to their result, but also the showing their confidence toward their result, which makes sense to trust to whom give the higher stake. But considering the situation that those oracles owning high reputation scores and daring to trust their result would give very high stake and then give the most prize. We limit the amount of the stake, the maximum of the stake should not overtake the amount of the payment from users. The prize pool $pool_j$ consists of users' payments and all oracles' stakes. When the answer to the task is generated, the money in the prize pool is distributed to the winners who provide the correct answer to this task. The amount of $prize_{i,j}$ for i-th oracle depends on the proportion of $score_{i,j}$ among all winners.

In this paper, our proposal believes the honest oracles who have high reputation scores should have the extra rewards, which encourages oracles to reduce their downtime and work consistently.

We define a score related to oracles' reputation and the stake. And the scores affect their importance in the result aggregation and prize distribution. Considering the attackers can own a large number of oracles and join the network before a specific task, then all those oracles give a wrong answer to mislead the answer aggregation. We set a threshold, which is 50 in this paper.

Only those oracles whose reputations are greater than this threshold can play a defining role in the answer aggregation. However, in the situation that the network just start, none of the oracles have a reputation that is greater than the threshold. So we design the function $f(x)$ to tackle the cold boot process.

$$score_{i,j} = \frac{f(\gamma_{i,j-1})}{\sum_{i=1}^{K} f(\gamma_{i,j-1})} * stake_{i,j} \tag{3}$$

$$f(x) = \begin{cases} 51 + 10(x-50) & x > 50 \\ 1 + x & x \le 50 \end{cases} \tag{4}$$

$$prize_{i,j} = \frac{score_{i,j}}{\sum_{i=1}^{K_W} score_{i,j}} * pool_j \tag{5}$$

3.3. Result Aggregation

We propose that any value including continuities can be viewed as discontinuities when aggregating the result. When receiving different results from the oracles, the management contract can divide those oracles with the same result into one class, then calculates the proportion of the sum of the score of each oracle in each class. The highest score class's result will be the result in this request.

$$P_{x,j} = \sum \frac{score_{i,j}}{\sum_{i=1}^{K} score_{i,j}} \quad i \epsilon D_{x,j} \tag{6}$$

The result x of the class with the highest proportion $P_{x,j}$ is the final result. However, if there are two classes with the highest proportion means oracles do not reach a consensus. So the solution to this situation is that the management contract repeats the whole process again.

3.4. Whole Process

Step Ⅰ : Users' smart contracts can send requests to the management contract to get data and allow management contract get the payment from their accounts.

Algorithm 1: dataRequest

```
Data: User's data request
Result: Response to the request
initialization;
if Oracles are not available then
    Abandon the transaction;
else
    Log the current request;
    getAllOracleResult();// get results from all oracles
    Aggregate the results to get final result;
    if Not reach consensus then Abandon the transaction;
    else Return the final result;
end
```

Step Ⅱ : According to the request problem id, management contract will send message to the oracles, which subscribe to this problem to get result.

StepⅢ: Management contract gets the result and the stakes form oracles.

Algorithm 2: getAlloracleResult

```
Data: User's data request
Result: Oracles' data response
initialization;
for i ← 1 to n do
    Send request to Oracle_i;
    tempAddr ← Oracle_i.address;
    tempResult ← Oracle_i.result;
    tempStake ← Oracle_i.stake;
    if tempStake > 0 then
        oracles[tempAddr].result ← tempResult;
        oracles[tempAddr].stake ← tempSkate;
    else
        do nothing
    end
end
```

Step Ⅳ : Management contract calculates the $score_{i,j}$ of each oracles. Then it aggregates the result from all oracles and get a final result.

Algorithm 3: aggregateResult

```
Data: Oracles' results to data request
Result: Aggregated result
initialization;
flag ← false;
for i ← 1 to n do
    calcuOracleWeight(Oracle_i);
end
foreach result of results set do
    calcuResultSumWeight(result);
end
foreach result of results set do
    if result.sumWeight > 0.5 then
        request.result ← result;
        flag ← true;
    else
        do nothing
    end
end
```

Step Ⅴ : Management contract returns the final result to user's smart contract.

StepⅥ: Management contract distributes the $pool_j$ to oracles with correct result.

StepⅦ: Management contract updates the reputation of all oracles.

Algorithm 4: updateReputation

Data: Oracles' reputation value acquire in the j-th task
Result: Oracles' new reputation score

initialization;
for $i \leftarrow 1$ **to** n **do**
  $Oracle_i.reputation \leftarrow$
    $\beta \times Oracle_i.tempReputation + (1-\beta) \times Oracle_i.reputation$
end

The following figure 1 shows the flow chart in our design.

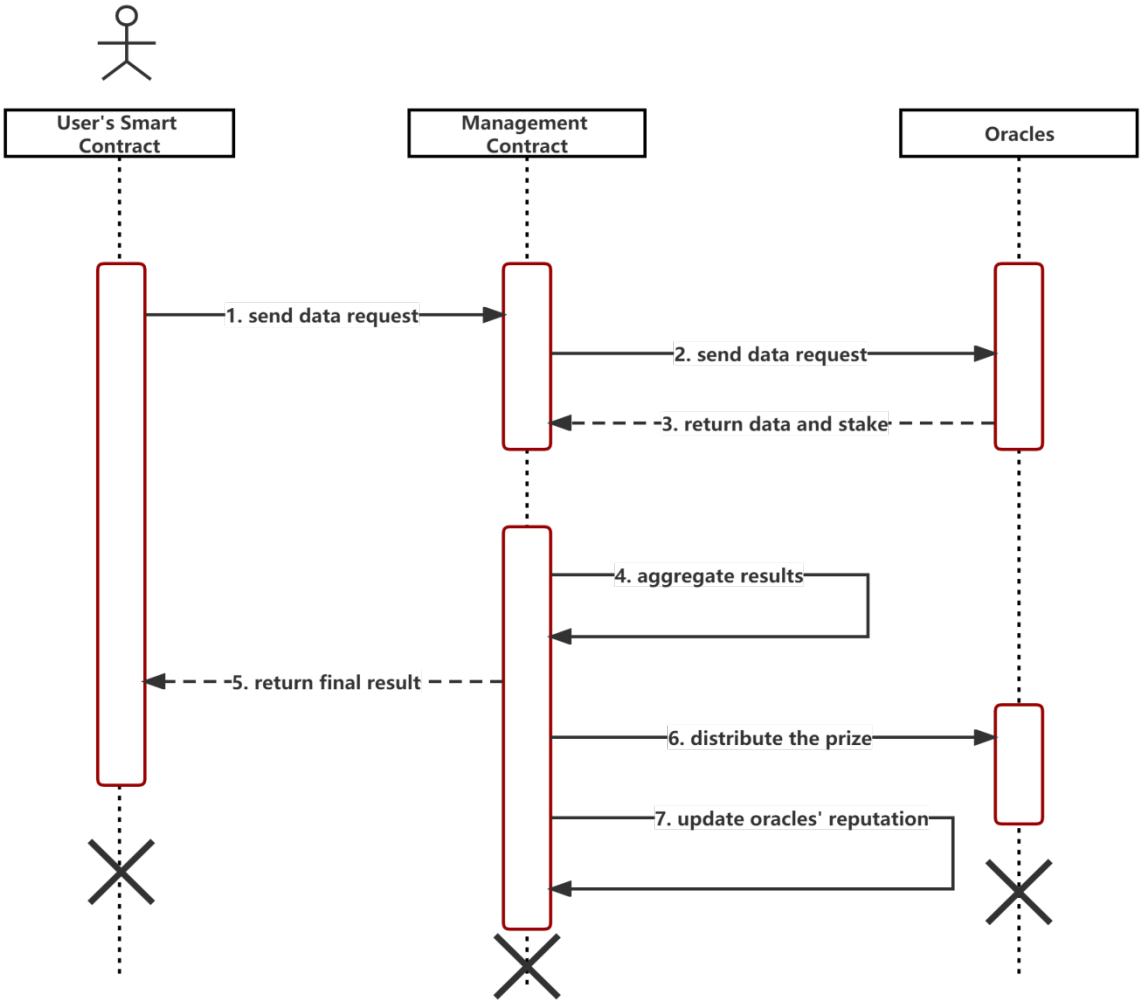

Figure 1. Data Request Service Process.

4. System Analysis

4.1. Reliability Analysis

For an oracle, its reputation is enhanced if its response data is consistent with the final aggregated data. The improvement in reputation score will lead to a greater weighting of the oracle in future

data aggregation and a larger share of the bonus. And for malicious oracles, false data reports deduct all its deposits and reduce its reputation score. Such a mechanism creates an effective closed-loop that motivates the oracles to perform correct data reporting.

Figure 2 shows our design can resist the oracles experiencing downtime well. And the situation in that attackers use oracles to send incorrect answers to the network is the same as figure 2. For an oracle that does not experience errors and downtime, its long-term infallible performance will continuously enhance its reputation score as shown in asterisk in figure 2. And for those oracles that have the probability of making incorrect data responses, their reputation score will be diminished by their mistakes. From the graph, we can learn that the higher the probability of a prophecy machine making a mistake, the smaller the mean of his reputation value and the larger the variance. For a reputable oracle, a sudden large number of errors can cause his reputation to drop significantly, which can seriously affect his future earnings.

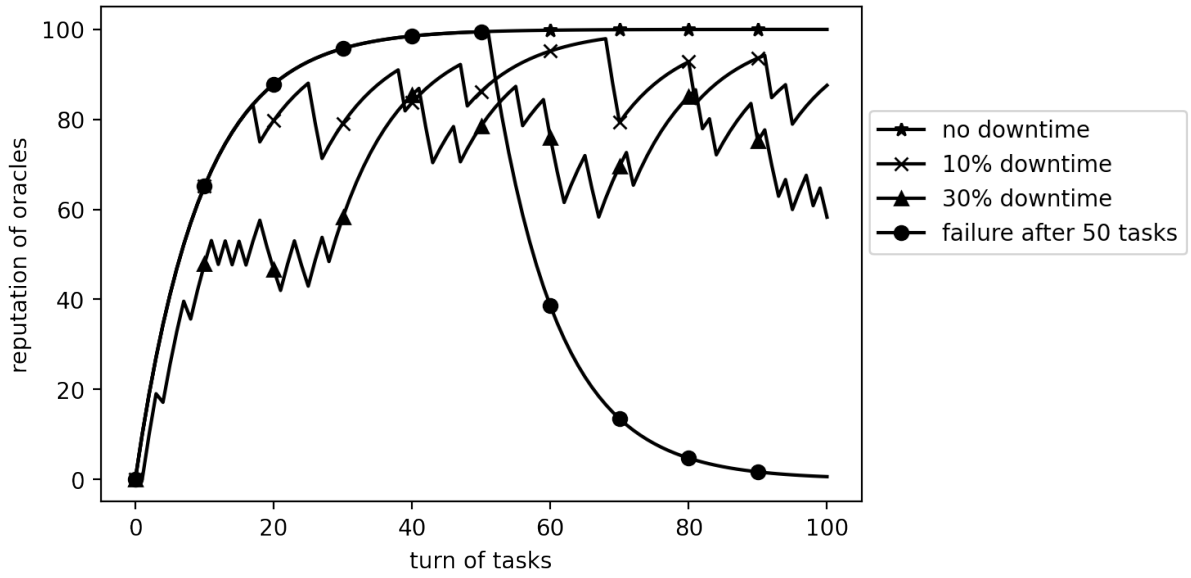

Figure 2. Reputation of Oracle Experiencing Downtime

4.2. Resistance to Attacks

If a malicious participant wants to control the final consensus outcome, then it needs to control more than fifty percent of the oracles, assuming that they all have the same reputation score. In this paper, since the deposit of incorrect oracles will be paid to correct oracles, when the benefits that participants can obtain from reporting the correct data are much more than bribes, it is more likely that the oracles will fend off attackers and report data correctly. And for the malicious participant, it would be extremely expensive to control the oracle network, which can be far more than the benefits it can obtain from its malicious behavior.

5. Conclusion

In this paper, we propose a reliable decentralized oracle system that is based on reputation and incentive mechanism. A simulation-based experiment is conducted to evaluate our reputation

mechanism. It shows that the malicious behavior of the oracle will bring down its reputation value significantly and cause it to take much less weight in aggregating results in future tasks. In future work, we will try to extend our scheme to accommodate all types of data requests, not just boolean types.

References

[1] Nakamoto, S. (2008) Bitcoin: A peer-to-peer electronic cash system. https://www.researchgate.net/publication/228640975_Bitcoin_A_Peer-to-Peer_Electronic_Cash_System

[2] BASHIR, I. (2018) Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained, 2nd Edition. https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5340530.

[3] Greenspan, G. (2016) Why many smart contract use cases are simply impossible. https://www. coindesk. com/three-smart-contract-misconceptions,.

[4] Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S. (2016) The blockchain as a software connector. In: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). Venice. pp. 182–191.

[5] Moudoud, H., Cherkaoui, S., Khoukhi, L. (2019) An iot blockchain architecture using oracles and smart contracts: The use-case of a food supply chain. In: 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications. Istanbul. pp. 1–6.

[6] Al Breiki, H., Al Qassem, L., Salah, K., Rehman, M.H.U., Sevtinovic, D. (2019) Decentralized access control for iot data using blockchain and trusted oracles. In: 2019 IEEE International Conference on Industrial Internet (ICII). Orlando. pp. 248–257.

[7] Provable Dev Community. (2019) Provable Documentation. https://docs.provable.xyz

[8] Zhang, F., Cecchetti, E., Croman, K., Juels, A. (2016) Town crier: An authenticated data feed for smart contracts. In: Proceedings of the 2016 ACM CIGSAC conference on computer and communications security. Vienna. pp. 270–282.

[9] Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N., Kastania, A. (2018) Astraea: A decentralized blockchain oracle. In: 2018 IEEE international conference on internet of things (IThings). Halifax. pp. 1145–1152.

[10] Ellis, S., Juels, A., Nazarov, S. (2017) Chainlink: A decentralized oracle network. https://www.academia.edu/41100629/ChainLink_A_Decentralized_Oracle_Network

[11] Peterson, J., Krug, J. (2015) Augur: a decentralized, open-source platform for prediction markets. https://arxiv.org/abs/1501.01042

[12] Sztorc, P. (2015) Truthcoin, peer-to-peer oracle system and prediction marketplace. https://github.com/psztorc/Truthcoin

[13] De Pedro, A.S., Levi, D., Cuende, L.I. (2017) Witnet: A Decentralized Oracle Network Protocol. In: CoRR. http://arxiv.org/abs/1711.09756

[14] Hess, Z., Malahov, Y., J. Pettersson. (2017) Aeternity blockchain: the trustless, decentralized and purely functional oracle machine. https://aeternity. com/aeternity-blockchain whitepaper. pdf.