On Public Decentralized Ledger Oracles via a Paired-Question Protocol

Marco Merlini*, Neil Veira*, Ryan Berryhill*, and Andreas Veneris*†
*Department of Electrical and Computer Engineering, University of Toronto
{mmerlini,nveira,ryan,veneris}@eecg.toronto.edu
†Department of Computer Science, University of Toronto

***Abstract*—Blockchain technology enables the operation of fully decentralized applications without the need for a central authority to manage the execution of the underlying process. However, a critical limitation in the technology today is the inability for such applications to query information external to the blockchain. Applications must make use of a decentralized oracle, *i.e.* a trusted source of external information. In this work we propose the *paired-question* decentralized oracle protocol, designed to extract true answers from the public. When querying the oracle, a user submits pairs of antithetic questions and voting users answer them for the chance to receive rewards. This new protocol lends itself to a simple formal analysis, and it is shown to strongly incentivize a Nash equilibrium of truthful reporting. This paper also discusses a number of extensions to the base protocol to improve its cost-effectiveness, security, and applicability.**

***Index Terms*—blockchain, decentralized, oracle, voting, Ethereum.**

I. Introduction

A key benefit enabled by blockchain or Distributed Ledger Technology (DLT) is decentralization, which removes the need for trusted third parties. Many applications for trustless services have been proposed, such as decentralized insurance [1], decentralized prediction markets [2], and management of financial instruments [3]. Each of these services requires information that is external to the blockchain. For example, decentralized insurance must access information about property damage and prediction markets must discover actual market outcomes. Contemporary blockchain platforms are unable to directly access real-world information, requiring any such data to be input by a user or some form of centralized mechanism. In many applications it can be difficult to find such a user who has no incentive to misrepresent or to withhold information, resulting in a loss of trustlessness that compromises the essence of DLT. This is known as the *gateway problem* [4].

A *decentralized oracle* can address the gateway problem for applications requiring decentralization. These oracles obtain off-chain information by allowing members of the public to provide answers to a question; these answers are then used by the oracle to compute a final output. In order to create trust in the oracle's output, a reward mechanism is used to incentivize users to truthfully report their actual beliefs.

A major challenge to these incentive schemes is the *lazy equilibrium* problem, a variant of the Verifier's Dilemma [5]. The problem, simply stated, is the existence of a degenerate Nash equilibrium where voters always report the same answer on all questions regardless of what they believe to be true, so as to secure economic incentives/profits. A simple mechanism behind such an oracle cannot guarantee that payoffs for truthful reporting are greater than payoffs in a lazy Nash equilibrium, and therefore cannot guarantee that voters will contribute correct information.

The decentralized oracle ASTRAEA [6] was the first to address this problem, using two groups of users voting on each question. Different reward schemes for these groups serve to reduce expected payoffs in a lazy equilibrium. However, ASTRAEA's relatively complex mechanism leads to significant challenges in its analysis. Ideally this analysis should be straightforward enough to convince voters that truthful reporting is their best course of action.

Following, the decentralized oracle SHINTAKU [7] attempts to reduce the complexity of ASTRAEA while still disincentivizing the lazy equilibrium. Instead of two voting groups, all voters answer two questions at once each time they participate. This enables a simple reward scheme which forces the lazy equilibrium to have an expected payoff of zero. SHINTAKU does not provide a formal analysis of its protocol, and although its design may seem to address the lazy equilibrium problem, its pair-voting protocol admits new types of strategies which are unaccounted for in the paper.

In this paper, we propose a novel *paired-question* protocol for decentralized oracles. When a user wishes to query the oracle, they submit two antithetic questions and post 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 (penalized) for agreement (disagreement) with the majority answer. If the questions converged to the *same* answer, the submitter *loses* their bond and voters receive neither rewards nor penalties. In contrast to ASTRAEA and SHINTAKU this protocol is readily analyzed and has many desirable properties: for example, the interface for voters is simplified and truthful voters receive larger expected payoffs. We also introduce a general mathematical model of decentralized oracles and apply it to show that reporting one's beliefs is a Nash equilibrium with better payoffs than in the lazy equilibrium.

The remainder of this paper is organized as follows. Sections II and III give a brief overview of blockchain oracles, set a general mathematical model for decentralized oracles, and present a summary of recent oracle projects. Section IV follows with a description of the new decentralized oracle protocol and its analysis from a game-theoretical point of view. Section V examines the protocol's resilience to adversaries and suggests a simple technique to improve security. Section VI discusses several extensions to the base protocol. Finally, section VII concludes the paper.

II. Preliminaries

A. Blockchain Oracles

In the context of blockchain applications (*e.g.* smart contracts), an *oracle* is an entity which provides trustworthy information. Several oracle solutions currently exist: `Oraclize.it` [8] fetches data from a specified web source and publishes it to a blockchain application. Outside the blockchain, they also maintain cryptographic proofs which show that the information originated from the correct source. `Town Crier` [9] works in a similar way, making use of Intel Software Guard Extensions [10] to protect against malicious operating systems. `Augur`'s market closing mechanism [11] instead allows token holders to report outcomes or to challenge reported outcomes in a multi-phase procedure.

A subclass of blockchain oracles are *decentralized* blockchain oracles. We will consider an oracle decentralized if it is both permissionless (any user may join the protocol without requiring other users' permission) and equi-privileged (all users have identical privileges). In this light, each of the above oracle solutions exhibits a strong form of *centralization*. `Oraclize.it` maintains a central server which handles all requests for off-chain information (giving it the privilege to deny requests, or to collude with website owners to produce false information). `Town Crier` also operates a centralized server with the notable difference that trusted hardware proofs are used to verify authenticity. Finally, though `Augur` includes a decentralized reporting/disputing mechanism, all markets must declare a *Designated Reporter*: a single Augur user with the privilege of reporting first on the market's outcome.

B. A Decentralized Oracle Model

1) Oracle Operator: In order to provide decentralization it is assumed that the main operation of the protocol is handled on a suitably decentralized platform, such as through smart contracts on Ethereum [12] or Hyperledger [13]. This entity is hereafter referred to as the *operator*.

The operator maintains a list of active *Boolean propositions* (*i.e.* statements which are either `True` or `False`). Any user may submit new propositions to the operator, and any user may vote on an active proposition. At certain times (as defined by the protocol) propositions are considered *closed*: the operator tallies votes, distributes rewards, and ceases to accept new votes on the closed proposition.

2) Voters: In general, a decentralized oracle requires input from users. We say that a user who inputs data into an oracle is a *voter*, and could be any member of the public. Let $V = \{v_1, v_2, v_3, ..., v_n\}$ be the set of all voters participating in the decentralized oracle protocol.

For a proposition p_j, each voter $v_i \in V$ has a *private opinion* $\mathrm{PO}_{ij} \in \{\texttt{True, False}\}$. This indicates what voter v_i honestly believes about the validity of p_j after reading its text. We assume the value of PO_{ij} is fixed, but in the honest scenario it is unknown to voters other than v_i. Voters who choose to collude and share their private opinions are considered adversarial, which is discussed in further detail in Section V.

Each voter v_i also has a *voting strategy* σ_{ij}, such that $\sigma_{ij}(\mathrm{PO}_{ij})$ is the answer that v_i actually reports to the oracle on proposition p_j. For example, an *honest* voter has $\sigma_i(\mathrm{PO}_{ij}) = \mathrm{PO}_{ij}$, while a *lazy* voter has either $\sigma_i(\mathrm{PO}_{ij}) = \texttt{True}$ for all j or $\sigma_i(\mathrm{PO}_{ij}) = \texttt{False}$ for all j. Finally, we define two random variables: let $\Gamma_j \in \{\texttt{True}, \texttt{False}\}$ denote the private opinion of a voter selected randomly from V on proposition p_j, and let $A_j \in \{\texttt{True}, \texttt{False}\}$ denote the *answer reported* by a voter selected randomly from V on proposition p_j.

3) Correctness of Oracle Outputs: No corporeal entity (let alone a blockchain oracle) is capable of determining the objectively true answer to a question [14, §1]. Even in the elusive scenario where this were possible, users may choose to submit subjective questions (which do not really have an objective truth in the first place). Therefore, to rigorously define the notion of *correctness* in this work, we make the following definitions:

Definition 1. *The* Most Probable Private Opinion (MPPO) *is a randomly selected voter's most likely private belief. On proposition p_j,*

$$\mathrm{MPPO}_j \doteq \begin{cases} \textit{True} & \mathrm{P}(\Gamma_j = \textit{True}) > 0.5 \\ \textit{False} & \mathrm{P}(\Gamma_j = \textit{True}) < 0.5 \\ \text{Undefined} & \mathrm{P}(\Gamma_j = \textit{True}) = 0.5 \end{cases} \tag{1}$$

Definition 2. *We say that the oracle is* correct *on proposition p_j when its output is equal to* MPPO_j.

Additionally, let c_{ij} denote voter v_i's perceived probability of reporting an answer to p_j equal to MPPO_j. Assuming that v_i does not know PO_j for other voters, this is the same as the probability that v_i answers MPPO_j on a PO_j value *selected randomly* from V:

$$c_{ij} \doteq P(\sigma_{ij}(\Gamma_j) = \mathrm{MPPO}_j) \tag{2}$$

We also let c_j denote the probability that a voter selected randomly from V reports an answer equal to MPPO_j:

$$c_j \doteq \mathrm{P}(A_j = \mathrm{MPPO}_j) \tag{3}$$

The key distinction between the definitions of c_{ij} and c_j is that c_{ij} pertains to a specific voter v_i with a fixed strategy σ_{ij}, and thus may be different for voters with different strategies, whereas in c_j the strategy also varies with the random variable A_j.

As an example, if all voters in V adopt the honest voting strategy, then $\mathrm{P}(A_j = \texttt{True}) = \mathrm{P}(\Gamma_j = \texttt{True})$ and we have

$$c_j = \begin{cases} \mathrm{P}(\Gamma_j = \texttt{True}) & \mathrm{MPPO}_j = \texttt{True} \\ \mathrm{P}(\Gamma_j = \texttt{False}) & \mathrm{MPPO}_j = \texttt{False} \end{cases} \tag{4}$$

Because much of this formulation applies to any arbitrary proposition p_j, we drop the subscript j from this point forward where no ambiguity arises from doing so.

III. Prior Art

A. Astraea

In Astraea [6], there are three groups of users: *voters*, *certifiers*, and *proposition submitters*. Voters post a relatively small bond and are given a random proposition to answer. Certifiers post a relatively large bond and *choose* which proposition to answer. A submitter sends a proposition along with a bounty which is used to fund rewards for the two voting groups.

Once voting ends, ASTRAEA always outputs the majority answer from the voters. After tallying votes on a proposition, the majority answer from the voters is compared to the majority answer from the certifiers, splitting the reward structure into two cases:

Voters and Certifiers Agree: Certifiers are rewarded for agreement with the majority and penalized for disagreement. In the event a certifier receives a reward, they are paid from a *certifier reward pool*. Specifically, there is one certifier reward pool for propositions decided as `True`, and one for propositions decided as `False`. Penalties exacted from the certifiers are used to fund the certifier reward pool of the *opposite* value (*e.g.* if a proposition was decided as `True`, certifier penalties would be used to fund the `False` reward pool). Similarly, voters are rewarded for agreement with the majority, and penalized for disagreement; the rewards are disbursed from the bounty paid by the proposition submitter, and penalties are used to fund the opposite reward pool.

Voters and Certifiers Disagree: Certifiers receive the maximum possible penalty (*i.e.* any previously posted bond is *not* returned). This penalty funds both certifier reward pools equally. Voters are not penalized, but they are not rewarded either (*i.e.* all of their posted bonds are returned). The bounty provided by the submitter is instead used to equally fund the certifier reward pools.

Over time, if propositions often converge to the same answer (*e.g.* `True`) then the corresponding reward pool will be drained much faster, causing the opposite answer to look more attractive. In order for voters and certifiers to receive the best payoffs for honesty, this mechanism depends on the assumption that proposition submitters submit a roughly equal amount of `True` and `False` propositions. Furthermore, if one reward pool could deliver significantly higher rewards, this may incentivize voters/certifiers to provide a dishonest answer. Nonetheless, this temporary dishonesty would still serve to stabilize the system against the lazy equilibrium.

B. SHINTAKU

Instead of having two types of voters, SHINTAKU has only one [7]. To participate, each voter posts a bond and receives *two* random propositions from the active proposition list. The voter returns both their answers (their *vote-pair*) to the oracle, and is rewarded for each answer independently. Similarly to ASTRAEA, a vote is rewarded for agreement with the majority, and penalized otherwise. However, SHINTAKU adds an extra condition: in order for the votes in a vote-pair to be eligible for rewards (or penalties) the two votes must have different answers. In the lazy equilibrium, all voters are reporting the same answer on all propositions. Thus, the vote-pair eligibility condition forces lazy voting to have lower expected rewards than honest voting.

Although SHINTAKU disincentivizes lazy voting, its ballot mechanism admits new types of strategies. A rigorous analysis of the SHINTAKU protocol can show that this voting strategy would have positive payoffs unless penalties for disagreement are at least twice as large as rewards for agreement. Although beyond the context of this paper, a sketch of the proof for this is found in Section IV-C. Not only does SHINTAKU require relatively harsh penalties, similarly to ASTRAEA its rewards to honest voters are lowered even further if submitters add an unbalanced number of `True` and `False` propositions.

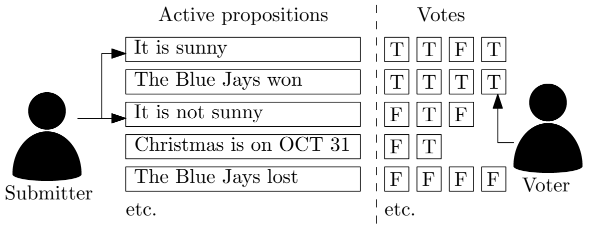

Figure 1: Overview of proposed oracle system

IV. A PAIRED-QUESTION PROTOCOL

This section describes our proposed decentralized oracle protocol. It begins with a general overview, then presents a detailed analysis. The oracle's correctness and expected payoffs are examined in the scenario where all voters are honest; we find that honest voters receive higher payoffs than lazy voters, while not sacrificing the oracle's correctness. Additionally, we demonstrate that honest voting is a Nash equilibrium. The next section analyzes the protocol's properties in an adversarial scenario.

A. Description

An overview of the new protocol is depicted in Figure 1. At any time a *submitter* can add propositions to the active proposition list, and at any time a *voter* can place a vote on an active proposition.

Submitting propositions. To add to the list of active propositions, in a single transaction a submitter provides:

- A bond
- Two propositions, called p and p'
- A bounty
- A duration

The submitter's bond is returned if and only if the answers to p and p' converge to different `True/False` outcomes after voting. Thus, the propositions should be designed to have opposite answers: this is easily done if the submitter constructs p' to be the converse of p (*e.g.* "Team X won the sports game" and "Team X did not win the sports game"). Constructing the converse proposition would be an expensive and error-prone operation if done automatically by the oracle itself, as the propositions are assumed to be written in natural language; it is a fair assumption that it will be done by a submitter who would otherwise risk penalization. The bounty is used to pay voters, and the duration determines the amount of time available for voting on p and p'.

Submitting votes. To place a vote, a voter must engage in the following dialogue with the oracle:

1) The voter posts a bond
2) The oracle selects a proposition and passes it to the voter
3) The voter returns a sealed vote to the oracle
4) Once the proposition closes, the voter reveals their vote

At step 3, the voter computes their vote based on their private opinion PO_i and their voting strategy σ_i. Sealed voting is necessary to prevent against certain types of undesirable strategies or attack vectors and is further discussed in section VI-B. Additionally, at step 2, it is advantageous for the oracle to choose a random proposition (discussed in section VI-C).

Once a proposition's duration expires, it is closed. At this point the oracle tallies votes and outputs the majority answer. It then rewards voters as follows: for each pair of propositions, it checks if the majorities converged to different answers. If so, voters are rewarded for agreement, penalized for disagreement, and the submitter's bond is returned. If the propositions converged to the same answer, the submitter loses their entire bond, and voters receive neither a reward nor a penalty (*i.e.* their bond is returned to them in full).

B. Analysis

1) Correctness: Of primary importance to applications querying the oracle is its correctness (in the context of Definition 2). Consider a proposition p with n honest voters and where c is the probability that a randomly selected voter agrees with MPPO. P_{Corr}, the probability that the oracle produces a correct output, is simply the probability that a majority of voters agree with MPPO:

$$P_{\mathrm{Corr}} = \mathcal{M}(n, \mathrm{MPPO}) \tag{5}$$

Here, we use the *majority function* $\mathcal{M}(n, x)$ to denote the probability that a majority out of n voters report x to the oracle. In the case of honest voters,

$$\mathcal{M}(n, \mathrm{MPPO}) = 1 - B\left(\left\lfloor \frac{n}{2} \right\rfloor, n, c\right)$$

where $B(k, n, p)$ denotes the cumulative binomial density function. $\mathcal{M}(n, \mathrm{MPPO})$ is calculated differently depending on the configuration of voters and voting strategies, such as the adversarial scenario in section V-B.

Figure 2 shows the probability that the oracle produces a correct output for different values of c (if all voters are honest). If a low number of voters is expected, then only propositions with widely accepted answers are likely to come out correctly. However, even if a proposition is highly contentious (with c near 0.5), the oracle will eventually converge on the MPPO with high probability provided there are enough voters.

2) Expected Rewards for Honest Voting: In our new protocol a voter is rewarded if they are in the majority and penalized if they are in the minority. In the case of a tie, neither rewards nor penalties are applied.

Suppose that voter v_i with strategy σ_i is one of n voters on p, and that p and p' are paired questions (*i.e.* they were submitted together). P_{Maj}, the probability that voter v_i is in the majority on p, is equal to the probability that at least $\lfloor \frac{n-1}{2} \rfloor$ other voters agree with them:

$$\begin{aligned} P_{\mathrm{Maj}} = \quad & (c_i)\mathcal{M}(n-1, \mathrm{MPPO}) \\ & +(1-c_i)\mathcal{M}(n-1, \neg\mathrm{MPPO}) \end{aligned} \tag{6}$$

P_{Tie}, the probability that a tie occurred, is the probability that exactly $\frac{n}{2}$ voters voted according to the MPPO and is calculated differently depending on the configuration of voters. Finally, P_{Min}, the probability that voter v_i is in the minority, is simply the probability that they were not in the majority and that there was no tie:

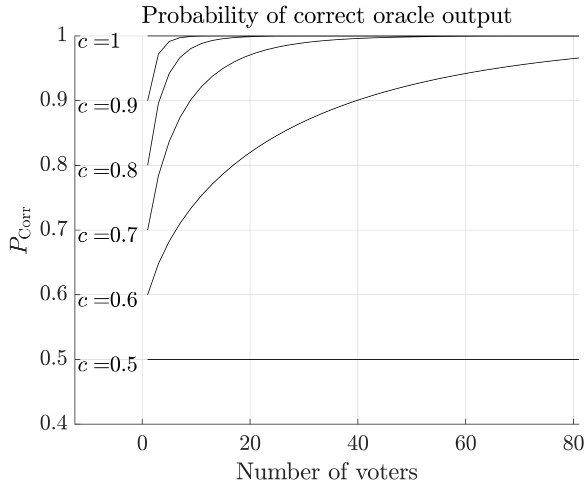

Figure 2: Probability of correctness as a function of n and c

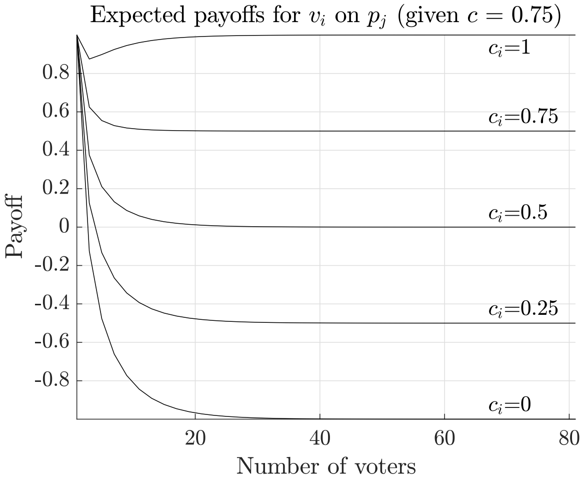

Figure 3: Expected payoffs as a function of n and c_i assuming $r = p = 1$ and $P(o \neq o') \approx 1$

$$P_{\mathrm{Min}} = 1 - P_{\mathrm{Maj}} - P_{\mathrm{Tie}} \tag{7}$$

For a voter to be paid or penalized, p must converge to a different answer from p'. Letting o and o' denote the oracle's outputs on p and p',

$$\begin{aligned} P(o \neq o') = \; & \mathcal{M}(n, \texttt{True})\mathcal{M}(n', \texttt{False}) \\ & +\mathcal{M}(n, \texttt{False})\mathcal{M}(n', \texttt{True}) \end{aligned} \tag{8}$$

Combining Eqs. (6) and (8), we can compute the probability that v_i receives a reward/penalty:

$$P_{\text{Rew}} = P_{\mathrm{Maj}} \cdot P(o \neq o')$$

Similarly,

$$P_{\text{Pen}} = P_{\mathrm{Min}} \cdot P(o \neq o')$$

Finally, if the reward size is r and the penalty size is p, then a voter's expected payoffs are $rP_{\text{Rew}} - pP_{\text{Pen}}$.

Figure 3 shows the expected rewards for a voter v_i depending on their own value of c_i assuming $r = p = 1$ and that all other voters are honest with $c = 0.75$. Note that rewards increase monotonically with an increasing c_i.

3) Expected Rewards for Lazy Voting: The main challenge for any oracle protocol is to disincentivize lazy voting. The previous subsection shows that honest voting enjoys positive expected payoffs in the honest scenario (and section V-B shows that payoffs are positive provided an adversary is not too powerful). In the lazy case, it is clear to see $P(o \neq o') \approx 0$. This forces expected payoffs to also be zero, which causes lazy voting to be less efficient than honest voting.

Name	Strategy Function	c
Honesty	$\sigma_i(\mathrm{PO_i}) = \mathrm{PO_i}$	> 0.5, Lemma 1
Lying	$\sigma_i(\mathrm{PO_i}) = \neg\mathrm{PO_i}$	< 0.5, Lemma 1
Always `True`	$\sigma_i(\mathrm{PO_i}) = \texttt{True}$	0.5, Lemma 2
Always `False`	$\sigma_i(\mathrm{PO_i}) = \texttt{False}$	0.5, Lemma 2

Table I: Summary of c values for pure strategies in response to honest voting

4) Honest Nash Equilibrium: We conclude this subsection by showing that honest voting is a Nash equilibrium. First, we enumerate the pure strategies available to a voter, and consider the c values for each one (summarized in Table I). Assuming that PO_i is the only input signal to the strategy function σ_i, then all strategies can be expressed as a mixture of these pure strategies. We then argue that the pure honest strategy has a strictly higher c, which implies strictly higher expected rewards. Thus, since the pure honest strategy is a best response to pure honest strategies, honesty is a Nash equilibrium.

Lemma 1. *In a scenario where all other users are honest, honest voting has an expected $c_i > 0.5$ for an arbitrary voter v_i with no information about the PO of other voters. Additionally, lying has an expected $c_i < 0.5$ for such a voter.*

Proof: First, note that by definition of MPPO, it must always be the case that $P(\Gamma = \mathrm{MPPO}) \geq 0.5$ when all voters are honest. The only circumstance in which $P(\Gamma = \mathrm{MPPO}) = 0.5$ is an extreme case where $\mathrm{P}(\Gamma = \texttt{True}) = 0.5$ (*i.e.* the MPPO is undefined). Assuming most submitters ask questions which have an answer, it is reasonable to conclude that $P(\Gamma = \mathrm{MPPO}) > 0.5$. This means that from the perspective of a specific voter v_i with incomplete information about the PO of other voters, $P(\mathrm{PO}_i = MPPO) > 0.5$, and so $c_i > 0.5$ if v_i reports honestly. Additionally, if an honest voter has probability x of reporting the MPPO, then a lying voter has probability $1 - x$ of reporting the MPPO and thus $c_i < 0.5$. ■

Lemma 2. *In a scenario where all other users are honest, lazy voting has an expected c_i of 0.5 for an arbitrary voter v_i with incomplete information.*

Proof: If submitters act honestly, then we can conclude they are creating an equal number of `True` and `False` propositions, implying $P(\mathrm{MPPO} = \texttt{True}) = 0.5$. Thus, on any particular proposition, a voter who always votes `True` (or `False`) has a probability of 0.5 of reporting the MPPO, which implies $c_i = 0.5$. ■

Theorem 1. *Honest voting is a Nash equilibrium.*

Proof: By Lemmas 1 and 2, the pure strategy of honest voting has a strictly better expected c_i than the pure strategies of lying and lazy voting. This means that no mixed strategy can achieve a greater expected c_i than honest voting. Since expected payoffs increase monotonically with increasing c_i, honest voting is a strictly best response in a scenario where all other users are honest. ■

C. Advantages of Proposed Protocol

An important feature of our proposed protocol is a rebalancing of incentives. In ASTRAEA and SHINTAKU, proposition submitters can create unfavourable conditions for voters. Notably, if most propositions have $\mathrm{MPPO} = \texttt{True}$, expected rewards are significantly lowered in both protocols (and in the case of ASTRAEA, this would cause the `False` certifier reward pool to be disproportionately large, possibly incentivizing certifiers to vote dishonestly on future propositions). In the paired-question protocol, a submitter receives a penalty for creating an imbalance of $\mathrm{MPPO} = \texttt{True}$ and $\mathrm{MPPO} = \texttt{False}$ propositions.

Additionally, we are able to reduce the size of penalties without sacrificing incentive compatibility. In SHINTAKU a voter could vote honestly on the first proposition of their ballot, then place the *opposite* answer for their second vote (regardless of their private opinion). This voting strategy (σ_x) has $c_x > 0.5$ on the first proposition and $c_x = 0.5$ on the second proposition (see Lemmas 1 and 2). Using r to denote the size of rewards and p to denote the size of penalties, if $r = p$ then in expectation the first and second votes would receive positive and zero payoffs, respectively (see section IV-B). While we omit the details due to space limitations, in can be shown that SHINTAKU requires $p \geq 2r$ to disincentivize this particular strategy. In contrast, our proposed protocol is able to correctly balance incentives with $p = r$, thus improving payoffs for honest voters.

The simplicity of the new protocol is also advantageous. It is not enough that a protocol guarantees optimal rewards for honesty; its users must be *convinced* of this fact. Otherwise, they may act according to an incorrect belief that a dishonest strategy is optimal. Furthermore, a simpler protocol exhibits a simpler formal analysis. While making fewer assumptions, stronger guarantees are proven, and extensions or critical adjustments to system parameters are easily evaluated.

D. Discussion

The proposed protocol is most effective for scenarios in which voters are able to answer any proposition, and sufficiently many voters are available to answer. As indicated by Figures 2 and 3, the ideal scenario would have at least 10-20 voters per proposition in order to achieve strong incentives for voter honesty and a high probability of correctness, depending on the level of agreement among the voters.

It is also important to note that our oracle model involves several idealizations in order to lend itself to a more tractable analysis. In particular, by assuming that a voter's strategy σ_i depends only on $\mathrm{PO_i}$, we implicitly disregard strategies that make use of information contained in the proposition itself. For instance, a voter may try to guess which of the propositions p and p' is the positive statement and which is the negation, and always vote `True` on the positive and `False` on the negation. If voters are able to guess correctly with high probability then this strategy can be a Nash equilibrium with large payoffs. However, in many scenarios it is possible to construct proposition statements in such a way that guessing accurately is extremely difficult.

V. Resilience to Adversaries

In this section we treat the issue of adversaries participating in the paired-question protocol. To model the effects of multiple voting strategies on a proposition, we will first generalize $\mathcal{M}(n, \text{MPPO})$ to accept a vector representing the number of voters for each strategy. Then, we will apply this generalized expression to calculate the probability that an adversary can manipulate the oracle's outcome as well as the costs of doing so. Finally, we propose a simple method to increase the oracle's security against such attacks.

A. Generalized Majority Function

Let $\vec{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_m)$ denote a vector of voting strategies for proposition p, and let the vector $\vec{n} = (n_1, n_2, ..., n_m)$ represent the number of voters following each strategy in $\vec{\sigma}$. Using $N = \sum \vec{n}$ and c_i to denote the probability that a voter using strategy σ_i reports MPPO to the oracle,

$$\mathcal{M}(\vec{n}, \text{MPPO}) = \sum_{\vec{k} \in K_{0.5}} \prod_{i=0}^{m} b(k_i, n_i, c_i) \tag{9}$$

where $b(k, n, p)$ is the binomial probability density function and $K_{0.5} = \{(k_1, ..., k_m) : 0 \le k_i \le n_i,\ \sum_i k_i > \frac{N}{2}\}$. In other words, equation (9) sums over the probabilities that (for $1 \le i \le m$) there are k_i voters with strategy σ_i who report MPPO to the oracle, and such that $\sum_i k_i$ constitutes a majority out of N.

B. Analysis of Adversarial Scenario

Suppose that on proposition p there are n_h honest voters and n_a voters controlled by the adversary (*i.e.* $N = n_h + n_a$). As usual the MPPO is determined in the sense of Definition 1, and each honest voter has probability $c_h \ge 0.5$ of reporting the MPPO to the oracle. The adversary is free to select a value of c_a for its voters. It is a straightforward matter to apply the generalized majority function in Eq. (9) to the expressions for correctness and payoffs in section IV-B.

Figure 4 shows the effect of an adversary on the oracle's correctness, assuming $c_a = 0$ (the case where the adversary is trying to force an incorrect answer). If $n_a + (1 - c_h) n_h < c_h n_h$ (*i.e.* the adversary with the incorrect honest voters do not outnumber the correct honest voters) then the oracle still maintains correctness with relatively high probability. This is easily seen in Figure 4 when correctness starts to decrease sharply near $n_a = 13$ and $n_h = 17$. However, once the adversary-controlled voters outnumber all honest voters, the oracle is guaranteed to be incorrect, as when $n_a \ge 15$.

Figure 5 shows the effect of an adversary on expected payoffs for each voting strategy. Similarly to the case for correctness, if $n_a + (1 - c_h) n_h < c_h n_h$ the oracle maintains relatively good payoffs for honesty and relatively strong penalties for dishonesty. In Figure 5, we note that the adversary's expected payoffs outpace an honest voter's expected payoffs past $n_a = 13$. Of course, once the adversary outnumbers the honest voters, their expected payoffs are guaranteed to be a maximum, as when $n_a \ge 15$.

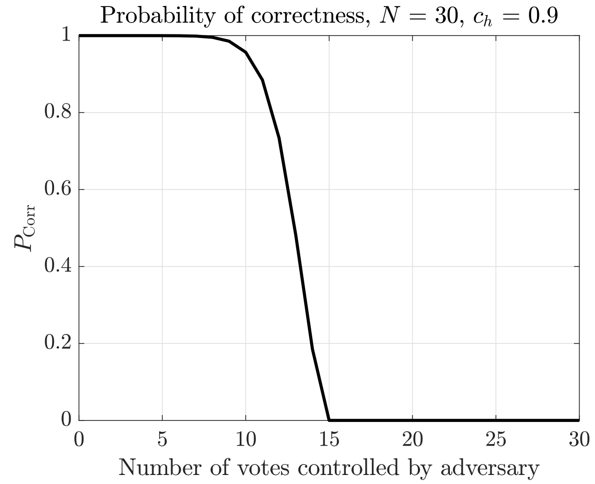

Figure 4: Oracle correctness as a function of proportion of adversary-controlled votes ($c_a = 0$).

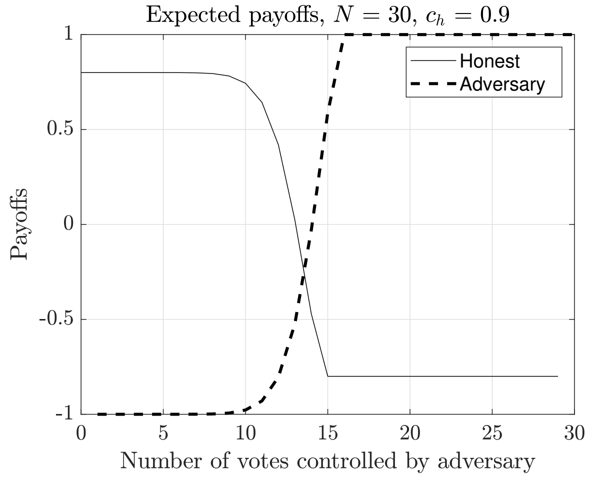

Figure 5: Payoffs to each voting strategy as a function of proportion of adversary-controlled votes ($c_a = 0$).

C. Quorum Payoff Schedule

Figures 4 and 5 demonstrate that if an adversary controls over 50% of votes, they can successfully manipulate an output at no cost (indeed, they will receive a profit). However, the payoffs to a successful adversary can be reduced if the condition for payment is to belong in a supermajority, *i.e.* a majority larger than 50%.

This change can be effected with a simple extension to the generalized majority function:

$$\mathcal{M}_q(\vec{n}, \text{MPPO}) = \sum_{\vec{k} \in K_q} \prod_{i=0}^{m} b(k_i, n_i, c_i) \tag{10}$$

where $K_q = \{(k_1, ..., k_m) : 0 \le k_i \le n_i \text{ and } \sum_i k_i > qN\}$. $\mathcal{M}_q$ requires an additional parameter $0.5 \le q \le 1$ which specifies the minimum quorum size for payment. Note that $\mathcal{M}_{0.5}(n, x) = \mathcal{M}(n, x)$.

This extended majority function can be used in Eq. (**??**), and an example is shown in Figure 6 for $q = 0.7$. Although expected payoffs have the same asymptotic values, for small values of n a minimum quorum size greater than 0.5 will reduce expected payoffs (as well as penalties).

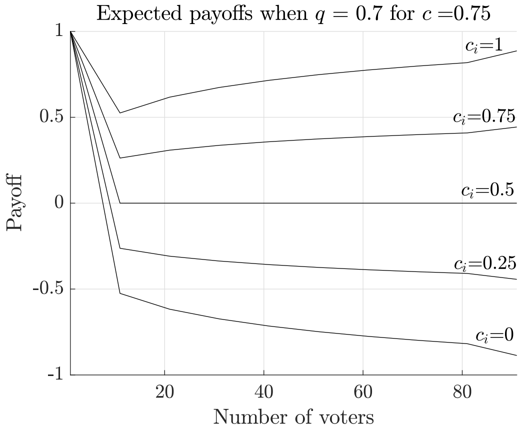

Figure 6: Effect of $q > 0.5$ in the honest scenario, for $r = p = 1$ and $P(o \neq o') \approx 1$. Compare with Figure **??**.

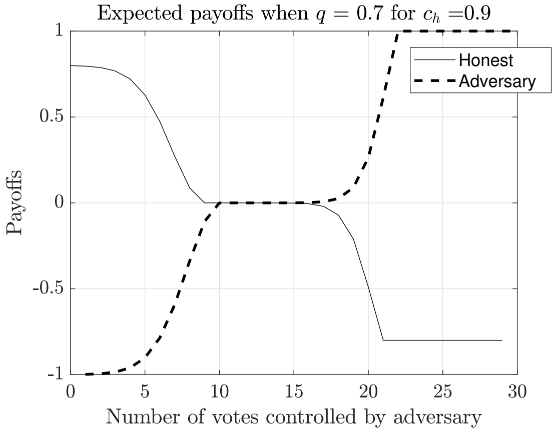

Figure 7: Effect of $q > 0.5$ in the adversarial scenario. Compare with Figure 5.

On the other hand, in the adversarial scenario, a larger minimum quorum size acts as an additional deterrent to adversaries. Figure 7 shows that an adversary must control qN votes before their expected payoffs are maximized. Although the adversary can force an incorrect response by controlling only $\frac{N}{2}$ votes, the quorum-based payment schedule makes this a more economically unattractive prospect.

VI. Extensions and Implementation Details

To complete the discussion of the new protocol, we discuss several important implementation details.

A. Voting Pools

The proposed oracle protocol exhibits its best performance when a large number of voters participate. As demonstrated in Figure 2, if $c = 0.6$ then at least 80 voters are required for P_{Corr} to exceed 95%. Of course, this would require many voters to submit voting transactions; however, on contemporary blockchain platforms, transactions can carry significant fees [15]. This indirectly raises the cost to proposition submitters who must provide a large enough bounty to offset the fees a voter must pay. Here, we describe a method to reduce costly transactions while not significantly lowering the oracle's correctness.

A voting pool consists of a *pool operator* and a number of *pool members*. Instead of interacting with the decentralized oracle, pool members answer propositions and receive rewards directly from the pool operator. The pool operator is free to use the answers collected from the pool members and to decide on individual rewards.

1) Advantages for a Pool Operator: In exchange for compensating pool members directly, a pool operator can use a variety of techniques which (due to its inability to hide information) are impossible for a decentralized oracle. For example, the pool operator could create their own propositions with known answers in order to test a pool member's knowledge, or a proposition could be sent multiple times to the same pool member in order to test for consistency. Furthermore, just as the oracle's correctness improves as it collects more votes, a pool operator's correctness improves as it collects more votes from its pool members. As a result, the pool operator has a higher c_i value and receives higher expected rewards from the oracle.

2) Advantages for Pool Members: In exchange for potentially smaller rewards, working for a pool operator greatly simplifies participation in the decentralized oracle protocol. For example, a pool operator can provide members with a simplified interface which does not require maintaining an address on a public blockchain, does not require the pool member to post a bond, and removes the need to pay transaction fees. Furthermore, voting in a pool (similarly to mining in a blockchain mining pool [16]) can lower the variance on rewards; even though direct participation with the oracle has positive expected rewards, it is possible for an honest voter to lose several times in a row. Should a voter run out of funds, they would no longer be able to post the necessary bond in order to continue participating.

3) Advantages for the Decentralized Oracle: First, the number of transactions is significantly reduced, which in turn lowers the cost to submitters. Second, though fewer participants are interacting directly with the oracle, each of them potentially has a much higher c_i parameter. As shown in Figure 2 the oracle needs only a small number of voters provided c is large enough.

B. Sealed Votes

To ensure the security of the oracle it is necessary for votes to remain secret until propositions are closed. One reason is to prevent voters from simply tallying existing votes and choosing to agree with the current majority (instead of honestly reporting their private beliefs).

One popular method for sealed voting is a cryptographic commitment scheme [17, §4.4.1]. When placing a vote, a voter sends $\text{Hash}(v, r)$ where v is their vote and r is a privately chosen random number. Once a proposition is closed, the voter reveals v and r, allowing the oracle (and any other participants) to verify that the voter committed to this vote.

Unmodified, this scheme is not enough for the set purposes. An attacker could replay a vote, a voter could choose to never reveal their vote, or a voter could publicly announce their vote before the proposition closes. We propose the following techniques for dealing with these issues:

1) Replay Attacks: This is easily prevented by extending the committed information. When a voter places a vote, they should send $\mathrm{Hash}(v, j, r)$ to show that they answered v on proposition p_j. Additionally, when tallying votes on a particular proposition, commitments with identical r values should be ignored. This removes the possibility of a sealed vote being valid in more than one context.

2) Voter Never Reveals: This scenario could take place when a voter is trying to avoid receiving penalties. For example, they could place a number of both `True` and `False` votes on a proposition and selectively reveal only the votes which will earn rewards. In order to prevent this, we should require voters to post a bond which is larger than the maximum penalty. If a voter disagrees with the majority, they will still regain some fraction of their original bond; if they do not reveal their vote, they forfeit their *entire* bond. This ensures that revealing votes is incentivized.

3) Premature Revealing: Finally, a voter may reveal their vote before a proposition is closed. Recall that sealed votes were desired in order to prevent new voters from simply copying the majority of existing votes. One way to disincentivize this behavior is to allow users to report one another for doing so. For instance, if a user can prove that they know a vote placed by voter v_i (by producing the correct v, j, and r parameters) then they can be rewarded with a large fraction of the original bond posted by v_i. A portion of the bond should also be discarded in order to disincentivize users from reporting themselves at zero cost (effectively cancelling their vote).

C. Random Numbers

Though it is possible to allow voters to select which proposition to answer, randomly selecting a proposition from the active list has a number of advantages; it has the effect of evenly distributing votes over all propositions, and makes it more costly for a voter (or group of voters) to force the output of a single proposition.

Random numbers present a significant challenge on decentralized blockchain platforms, as all users must agree on the exact same random number (implying that its selection must be deterministic) and yet they must not be able to predict or manipulate it. A popular method for accomplishing this uses a `ranDAO` [18]. This scheme occurs in two non-overlapping phases: the committing phase and the revealing phase. In the committing phase, users send hashes of privately-generated random numbers. In the revealing phase, the users reveal their private numbers, which are combined into a final result. This ensures that users cannot predict the output during the commit phase (though they can influence it) and that they cannot influence the output during the reveal phase (though they can predict it).

Although this technique guarantees that no user can predict or predictably influence the output, it does not guarantee liveness (*e.g.* users could choose to never reveal their commitments). Pragmatically, a `ranDAO` can choose to end the reveal phase once a certain quorum is met, at the cost of guaranteed security.

Note that an oracle could efficiently use both sealed votes and a `ranDAO` by combining the two techniques. The r values used for sealed votes can be repurposed as the basis for random number generation.

VII. Conclusion

This work proposes a novel paired-question decentralized oracle protocol. Submitters enter pairs of antithetic propositions into the system, while voters answer propositions for the chance to receive rewards. We analyze oracle correctness and voter payoffs in an honest voting scenario and show the existence of an honest Nash equilibrium. We also analyze the protocols performance in a scenario involving adversaries, and show that only a powerful adversary can manipulate outcomes. Additionally, we provide a number of extensions which can increase the cost to force an outcome, reduce transaction costs, provide pseudo-random number generation, and allow for secret voting on a public decentralized blockchain platform. As future work, implementation and deployment on the blockchain would allow for a more detailed empirical analysis of performance and costs so to further justify the theoretical analysis present here.

References

[1] C. Mussenbrock and S. Karpischek. (2017) Etherisc whitepaper. [Online]. Available: https://etherisc.com/files/etherisc_whitepaper_1.01_en.pdf

[2] J. Clark, J. Bonneau, E. W. Felten, J. A. Kroll, A. Miller, and A. Narayanan, "On decentralizing prediction markets and order books," in *Workshop on the Economics of Information Security, State College, Pennsylvania*, 2014.

[3] A. Nordrum, "Wall street firms to move trillions to blockchain in 2018," *IEEE Spectrum: Technology, Engineering, and Science News*, 2017.

[4] H. Halaburda, "Blockchain revolution without the blockchain?" *Commun. ACM*, vol. 61, no. 7, pp. 27–29, Jun. 2018. [Online]. Available: http://doi.acm.org/10.1145/3225619

[5] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 706–719.

[6] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "ASTRAEA: A Decentralized Blockchain Oracle," in *Proceedings of the 2018 IEEE Conference on Blockchain*, 2018, pp. 1145–1152.

[7] R. Kamiya. (2018) Shintaku: An end-to-end-decentralized general-purpose blockchain oracle system. [Online]. Available: https://gitlab.com/shintaku-group/paper/raw/master/shintaku.pdf

[8] Oraclize.it. [Online]. Available: http://www.oraclize.it/

[9] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*. ACM, 2016, pp. 270–282.

[10] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[11] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform."

[12] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[13] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, vol. 310, 2016.

[14] R. Kraut, "Plato," in *The Stanford Encyclopedia of Philosophy*, fall 2017 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2017.

[15] A. Hertig. (2017) Bought your first bitcoin or ether? brace for the fees. [Online]. Available: https://www.coindesk.com/bought-first-bitcoin-ether-now-brace-fees

[16] B. Wiki. (2018) Pooled mining. [Online]. Available: https://en.bitcoin.it/wiki/Pooled_mining

[17] O. Goldreich, *Foundations of Cryptography: Volume 1*. New York, NY, USA: Cambridge University Press, 2006.

[18] randao.org. (2017) Randao: Verifiable random number generation. [Online]. Available: https://randao.org/whitepaper/Randao_v0.85_en.pdf