LEDGER
ledgerjournal.org

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.166

RESEARCH ARTICLE

Decentralized Common Knowledge Oracles

Austin K. Williams,[*†] Jack Peterson[‡]

Abstract. We define and analyze three mechanisms for getting common knowledge, *a posteriori* truths about the world onto a blockchain in a decentralized setting. We show that, when a reasonable economic condition is met, these mechanisms are individually rational, incentive compatible, and decide the true outcome of valid oracle queries in both the non-cooperative and cooperative settings. These mechanisms are based upon repeated games with two classes of players: *queriers* who desire to get common knowledge truths onto the blockchain and a pool of *reporters* who possess such common knowledge. Presented with a new oracle query, reporters have an opportunity to report the truth in return for a fee provided by the querier. During subsequent oracle queries, the querier has an opportunity to punish any reporters who did not report truthfully during previous rounds. While the set of reporters has the power to cause the oracle to lie, they are incentivized not to do so.

1. Introduction

1.1. Background—In order for smart contracts to condition their execution on the state of the world, they need access to information about the world. While smart contracts can verify *a priori* claims with mathematical or cryptographic certainty, they cannot independently verify *a posteriori* claims about the world with the same assurances. As a matter of epistemological necessity, smart contracts which condition their behavior on *a posteriori* knowledge must rely on trusted oracles to provide that knowledge. As a result, we can trust these smart contracts only if we can trust their oracles.

With no possibility of mathematical or cryptographic verification of *a posteriori* claims about the world, we instead look to *economic incentives* when considering whether to trust an oracle. We require that the cost (to the oracle operators) of lying be greater than the benefit. More specifically, we require that truth-telling be incentive compatible. We also want the operation of the oracle to have a non-negative expected return for the operators. That is, we require that the operation of the oracle be individually rational. Finally, we want the oracle to be decentralized in order to avoid both censorship and a single point of failure.

A common approach to designing such an oracle is to create a coordination game in which individual human players are presented with an oracle query and are asked to report the correct outcome by staking some tokens.[1,2] The oracle outputs whichever outcome received the most stake as the "winning outcome," and then players are rewarded if and only if they staked in

[*] 3NubPpzpE3DQm9g6XxkueB7GEjuDZDewCF
[†] A. K. Williams (austin.williams@onewayfunction.com) is a researcher at OpenZeppelin.
[‡] J. Peterson (jack@tinybike.net) is a researcher at INDY Labs, USA.

agreement with the winning outcome. The hope in these "Schelling scheme" approaches is that the truth will act as the Schelling point of a coordination game, which would result in the oracle returning the true outcome to the oracle query. Although these approaches are appealing because they are easy to implement and have a high degree of social scalability, they have serious drawbacks that make their real-world success unlikely.

First, Schelling points themselves are an informal solution concept used in the context of coordination games in which pre-play communication is incomplete or impossible, and in bargaining games in which players cannot make binding agreements.[3] However, in the open blockchain setting, players can freely engage in pre-play communication (via Reddit, Twitter, email, *etc.*) and make binding agreements (for example, via smart contracts). So the Schelling point solution concept is not one that is known to be applicable in the types of strategic settings in which blockchain oracles operate.

Second, the coordination game approach to oracle design can be incentive compatible only in the non-cooperative model. As soon as players are able to make binding agreements, they can perform bribing attacks and form coalitions that are large enough to make the oracle lie without members of the coalition receiving any penalty.[4] If players can form coalitions, we cannot rely on truth-telling being incentive compatible for mechanisms following the coordination game paradigm. The root of the problem is that players are rewarded for agreeing with the majority *whether or not* the majority tells the truth.

For these reasons we think it is unlikely that the coordination-game approach to decentralized oracle design will work in practice. We desire an oracle that is incentive compatible in the cooperative model—where players can engage in pre-play communication and make binding agreements.

In addition to being incentive compatible in isolation, one must also consider the trustworthiness of oracles in the presence of "extraneous incentives": truth-telling must be incentive compatible when the output of the oracle is consumed to control the irreversible payout of large amounts of cryptocurrency. For example, every bet placed on a decentralized betting platform increases the gross incentive to make the platform's oracle lie. Many existing blockchain oracle designs do not explicitly consider the incentives introduced by the consumption of the oracle's output when analyzing the incentive compatibility of their mechanisms. (With the exception of Augur's oracle design,[5] we are unaware of *any* proposed blockchain oracles, centralized or decentralized, that explicitly quantify and address this risk.) However, explicit consideration of such extraneous incentives is crucial when considering the security of oracles to be deployed for real-world use. For a given oracle design, we can quantify this risk and state precisely how much extraneous incentive the design can handle before losing incentive compatibility.

1.2. Our approach—In this paper, we describe a new approach to oracle design that does not follow the coordination-game paradigm. For the mechanisms in this paper, truth-telling is—under certain reasonable economic conditions—incentive compatible in both the non-cooperative and cooperative game-theoretic models. A large coalition may form that makes the oracle lie, but members of such a coalition are severely penalized. Indeed, they are penalized more than they might gain from causing the oracle to lie. This stands in contrast to mechanisms based on coordination games, which *reward* such large coalitions for lying.

At a high level, we begin with a separation of concerns. We create a distinction between those who want to get common knowledge, *a posteriori* information on-chain and those who possess such common knowledge. We refer to the former as *queriers* and the latter as *reporters*. For example, a trustless, decentralized platform for betting on the outcomes of elections would be a querier because it wants to get the true outcomes of elections on-chain in order to settle bets. A collection of humans that tells the platform which candidates have won would be a set of reporters.

We then create mechanisms based upon repeated games—played by queriers and reporters—wherein each stage game corresponds to an oracle query and the outcome of each stage game determines how the oracle responds to the query. We show that under certain economic conditions (which explicitly include any extraneous incentives introduced by any consumption of the oracle's output) there exists equilibrium behavior in these stage games that results in the oracle returning the true outcome of real world events. These first incentive compatibility proofs take place in the non-cooperative model and assume that the set of queriers and the set of reporters is disjoint.

We present three such mechanisms. Each mechanism is more complex than the last, but also has better scalability properties. We more closely examine the conditions under which our results hold and discuss the weaknesses of our approach. We also consider the cooperative game-theoretic model, where the queriers and reporters can form coalitions (or may even be the same people). We show that, with some additional conditions, our results hold in the cooperative model as well.

2. Definitions

Definition (*Outcome space*). For all events $\mathtt{E}$, an *outcome space* of $\mathtt{E}$, denoted $\Omega_{\mathtt{E}}$, is a finite set of possible outcomes of $\mathtt{E}$. We require that every outcome space contain the special element `Invalid`, and that no outcome space contain the special element `Abstain`. When the event $\mathtt{E}$ is clear from context, we may drop the subscript and denote the outcome space Ω.

Definition (*Oracles and queries*). An *oracle* is any algorithm that accepts as input an event $\mathtt{E}$, a corresponding outcome space Ω, (and, optionally, some additional arguments) and outputs some $\omega \in \Omega$. A call to the oracle is referred to as a *query*.

Definition (*Common knowledge*). A proposition P is said to be *common knowledge* among a group of agents G if all agents in G know P, they all know that they all know P, they all know that they all know that they all know P, and so on, *ad infinitum*.[6]

The group of agents G is the collection of all users who interact with the oracle. Informally, one may consider a proposition to be common knowledge if it can be quickly verified by any user with access to the World Wide Web.

Definition (*True outcome*). For every oracle query with arguments $\mathtt{E}$ and Ω, we define a unique outcome in Ω to be the *true outcome* for the query. We denote such an outcome `True`, and it is defined as follows. If there exists a unique outcome $\omega \in \Omega \setminus \{\mathtt{Invalid}\}$ such that—at the time of the oracle query—it is common knowledge that the outcome of event $\mathtt{E}$ is ω, then ω is the true outcome for the query. Otherwise, `Invalid` is the true outcome for the query.

Definition (*False outcome*). For every oracle query, every outcome in Ω that is not `True` is a *false outcome*.

It is important to note that simply corresponding to objective reality is not a sufficient condition for an outcome to be the True outcome for a query. The fact that the outcome corresponds to objective reality must also be common knowledge at the time of the query.

Definition (*Valid query*). A query whose True outcome is not Invalid is referred to a *valid query*.

Using this terminology, our objective for this paper is to construct a decentralized, incentive compatible, individually rational mechanism that decides the True outcome of valid queries.

Definition (*Ω-partition*). If T is a finite set of tokens, E is an event, and Ω is an outcome space of E, then an *Ω-partition of T* is an indexed family of $|\Omega|+1$ mutually disjoint subsets of T (referred to as *cells*) indexed by $\Omega \cup \{\texttt{Abstain}\}$ where the union of the cells is T. An Ω-partition may be represented succinctly using the indexed-family notation $\mathbf{C} = (C_\omega)_{\omega \in \Omega \cup \{\texttt{Abstain}\}}$, or in expanded form via $\left\{C_{\texttt{Abstain}}, C_{\omega_1}, \ldots, C_{\omega_{|\Omega|}}\right\}$.

Colloquially, an Ω-partition of a set of tokens is simply the separation of the tokens into "piles" (cells) that are labeled by the outcomes in $\Omega \cup \{\texttt{Abstain}\}$. Such partitions arise naturally in the context of voting with tokens. For example, suppose that everyone who owns at least one token in a set of tokens, T, is asked to cast a vote in favor of some outcome in Ω. If we separate the tokens into cells according to how the owner of each token voted, the resulting partition would be an Ω-partition of T. (Any tokens owned by someone who refused to vote is put into the pile labelled "Abstain.")

Next, we develop notation for a simple algorithm that asks a player to report the outcome of some event. The player's response, along with the set of tokens controlled by the player, are returned.

Definition (*Report*). The algorithm *Report* takes as input a tuple $(j, \texttt{E}, \Omega, T)$, where T is a set of tokens, j is the owner of at least one token in T, E is an event, and Ω is an outcome space of E. The owner, j, is asked to report which element ω in $\Omega \cup \{\texttt{Abstain}\}$ is True. If j fails to respond then ω is understood to be Abstain. *Report* returns the tuple (ω, R), where R is the set of all tokens in T that are owned by j.

Next, we define an important algorithm referred to as the *fork*. The fork is not an oracle, but will be used as an important subroutine in the oracles we construct in this paper. In brief, the fork is the process whereby owners of tokens stake their tokens on some outcome as a response to a query.

Definition (*Fork*). The algorithm $\mathcal{F}$, referred to as the *fork*, accepts as input a tuple $(\texttt{E}, \Omega, T)$—where E is an event, Ω is an outcome space of E, and T is a finite set of tokens—and returns an Ω-partition of T.

At a high level, $\mathcal{F}$ works as follows. Each owner of tokens in T is queried to ask which outcome in Ω is True. The owner's tokens are assigned to the cell that corresponds to the outcome they reported. If the owner does not respond (or if their response is not in Ω), then their token are put in cell $C_{\texttt{Abstain}}$. Once all tokens in T have been assigned to a cell, $\mathcal{F}$ returns the collection of cells, which is an Ω-partition of T. In pseudocode:

```
def $\mathcal{F}(\texttt{E}, \Omega, T)$:

  //begin with all cells empty
```

```
    for each $\omega \in \Omega \cup \{\texttt{Abstain}\}$:
      $C_\omega \leftarrow \emptyset$
    endforeach

    //query all token owners for reports
    for each owner j of tokens in $T$:
      $(\omega, R) \leftarrow Report(j, \mathrm{E}, \Omega, T)$
      //put the owner's tokens in the cell corresponding to the
         reported outcome
      $C_\omega \leftarrow C_\omega \cup R$
    endforeach

    //return the $\Omega$-partition of $T$
    return $\left\{C_{\texttt{Abstain}}, C_{\omega_1}, \ldots, C_{\omega_{|\Omega|}}\right\}$

enddef
```

For our purposes, all calls to $\mathcal{F}$ are assumed to be public, as are the resulting outputs. (In practice, $\mathcal{F}$ would be implemented as a smart contract on a blockchain, and its entire history of calls and responses would be public.) The `for` loop in which token owners are queried may be run in parallel so that all token owners are queried simultaneously.

Finally, we define two simple subroutines: *Pay* and *PluralityWinner*.

Definition (*Pay*). The subroutine *Pay* accepts as input a tuple (T, ϕ), where T is a finite set of tokens and ϕ is some amount of currency. Each owner of tokens in T is given a *pro rata* share of ϕ. In particular, if $R \subseteq T$ is the set of tokens owned by some agent, then that agent will be paid a total of $\frac{\phi|R|}{|T|}$.

Definition (*PluralityWinner*). The subroutine *PluralityWinner* accepts as input an Ω-partition and returns an outcome in Ω whose corresponding cell in the Ω-partition is of the maximum size. Any ties are broken uniformly at random. (How ties are broken is unimportant for our purposes. Our results remain unchanged so long as the winner is chosen from among those outcomes whose corresponding cells have the maximum size.) In pseudocode:

```
def $PluralityWinner(\{C_{\texttt{Abstain}}, C_{\omega_1}, \ldots, C_{\omega_{|\Omega|}}\})$
    //get outcomes in $\Omega$ with largest corresponding cells
    $X \leftarrow \{\omega \mid \omega \in \Omega \wedge \forall \gamma \in \Omega : |C_\omega| \geq |C_\gamma|\}$

    //break any ties uniformly at random
    $\hat{\omega} \xleftarrow{\$} X$

    return $\hat{\omega}$

enddef
```

Colloquially, *PluralityWinner* simply interprets an Ω-partition as the outcome of a plurality vote and returns the winner. Note that *PluralityWinner* never returns `Abstain`, because $X \subset \Omega$ and `Abstain` $\notin \Omega$.

3. Assumptions

We model all agents as being rational. In particular, all agents come equipped with a von Neumann-Morgenstern utility function and always prefer actions that maximize their expected utility. For simplicity, we model agents as being risk neutral and having utility functions that are quasilinear in money. We assume agents are not budget constrained.

We further model all agents as being able to engage in costless communication with one another before making decisions. For the majority of the paper, we will assume that the set of queriers and the set of reporters are disjoint, and we model the players as *not* being able to make binding agreements with one another—and so our first analyses will be done in the non-cooperative setting.

In section 11 we consider the effects of costless binding agreements and transferable utility by analyzing our approach in the cooperative model. Note that this also covers the case where the set of queriers and the set of reporters are not necessarily disjoint.[7]

Throughout the paper we assume that the center of each mechanism is a smart contract that has no *a posteriori* knowledge of the world, and that the platform on which the smart contracts are executed is censorship resistant.

4. The Simple Oracle, $\mathcal{A}_0$

4.1. Motivation—We construct a simple decentralized oracle that treats the fork $\mathcal{F}$ as a plurality vote and returns the winning outcome. The key is to wrap $\mathcal{F}$ with a mechanism that encourages token owners to report `True` when they are queried during $\mathcal{F}$. When a certain “economic soundness” condition is met (see section 5.2) we can expect the winning outcome of the fork to be the `True` outcome of the event. In its most basic form, the algorithm consists of:

- Beginning with a reporting pool of tokens of equal value
- Paying a reporting fee to the owners of tokens in the reporting pool before calling $\mathcal{F}$ and
- Permanently removing from the reporting pool any tokens that were not used to report `True` during the previous query

Tokens removed from the reporting pool no longer earn their owners a reporting fee, so they are expected to have strictly lower value than tokens that remain in the pool. The price difference serves as an incentive for agents to report `True`.

4.2. Construction—The simple oracle, denoted $\mathcal{A}_0$, works as follows. We create an initial finite set, $T_{genesis}$, of tokens that have no intended value outside of their use in this context. This set of tokens serves as the initial reporting pool. When querying the oracle, the caller pays the oracle a fee, denoted ϕ. The oracle distributes this fee to owners of tokens in the reporting pool.

The oracle passes the query to the fork, which asks the owners of tokens in the reporting pool to report the `True` outcome of the event. The response from the fork is interpreted as the outcome of a plurality vote, and the outcome with the most votes is returned by the oracle.

It is those that query the oracle—assisted by smart contracts—that execute the algorithm $\mathcal{A}_0$. In particular, it is those that query the oracle that determine which tokens were used to report truthfully during the previous call, and thus which token owners will be paid during the next call. While token owners decide the outcome that the oracle returns, it is those that query the oracle in the future that determine whether the previous response was true.

After the oracle returns an outcome, all tokens in the reporting pool that were not used to tell the truth are removed from the reporting pool for the next round. In pseudocode:

```
//initial state
$C^0_{\texttt{True}} \leftarrow T_{genesis}$
$i \leftarrow 0$

def $\mathcal{A}_0(\mathrm{E}, \Omega, \phi)$:
  //increment query counter
  $i \leftarrow i + 1$

  //update the reporting pool
  //only truth-tellers from previous query remain in pool
  $T_i \leftarrow C^{i-1}_{\texttt{True}}$

  //pay owners of tokens in $T_i$
  $Pay(T_i, \phi)$

  //call $\mathcal{F}$ with inputs $(\mathrm{E}, \Omega, T_i)$
  $\{C^i_{\texttt{Abstain}}, C^i_{\omega_1}, \ldots, C^i_{\omega_{|\Omega|}}\} \leftarrow \mathcal{F}(\mathrm{E}, \Omega, T_i)$

  //select winning outcome
  $\hat{\omega}_i \leftarrow PluralityWinner(\{C^i_{\texttt{Abstain}}, C^i_{\omega_1}, \ldots, C^i_{\omega_{|\Omega|}}\})$

  //return winning outcome
  return $\hat{\omega}_i$

enddef
```

5. Analysis of $\mathcal{A}_0$

5.1. Introduction—Our goal is to design an incentive compatible, individually rational mechanism that implements a decision function that outputs the True outcome for a valid oracle query. We will show that when a certain "economic soundness condition" is satisfied (see Section 5.2) the simple oracle $\mathcal{A}_0$ is such a mechanism.

The execution of the oracle $\mathcal{A}_0$ is modeled as a repeated game with two classes of players: reporters and a querier. Each stage game of the repeated game is associated with an oracle query, and is modeled as a sequential game that operates as follows:

(1) Each reporter chooses an outcome in response to the oracle query associated with the current stage game
(2) The querier chooses how to update the reporting pool

We will show that $\mathcal{A}_0$ is incentive compatible by showing that there exists a Pareto efficient, subgame-perfect Nash equilibrium in the stage game which results in $\mathcal{A}_0$ returning the True outcome for the oracle query. In particular, we will show that our desired player behavior—where every reporter always reports the True outcome and the querier always removes from the reporting pool all and only those tokens used to lie—is in equilibrium in the stage game. We will then show that $\mathcal{A}_0$ is individually rational by demonstrating that the payouts for all players at this equilibrium are positive and strictly greater than their minmax payouts.

5.2. The Economic Soundness Condition—Let $I_{i,j}$ denote the benefit to reporter j from the oracle returning a false outcome in response to the ith oracle query. It is important to note that

$I_{i,j}$ is intended to capture *all* benefit to reporter j—including any "extraneous" benefit—from the oracle returning a false outcome in response to the ith oracle query. These benefits may be paid out in any currency. (We do not assume, for instance, that these benefits are paid out with tokens in T.) For example, if the oracle query is being used to determine payouts on a prediction market for a national election, and reporter j has placed a large bet on the losing candidate, the value $I_{i,j}$ would include the value of reporter j's bet. Similarly, any losing secondary bets (*e.g.*, derivatives or other side-bets on the outcome, which may be denominated in entirely different currencies) placed by reporter j are also included in the value $I_{i,j}$.

Let $I_i = \sum_j I_{i,j}$ denote the total benefit received by all reporters from the oracle returning a false outcome in response to the ith oracle query. This represents the maximum total collective benefit (over all reporters) that could be gained from the oracle returning a false outcome to the ith query.

Let p_i denote the market price of a token in the ith reporting pool T_i, and let p'_i denote the market price of a token in $T_{genesis} \setminus T_i$. (That is, p'_i denotes the market price of a token that has been removed from the reporting pool for lying.)

Definition (*Economic soundness condition*). We say the *economic soundness condition* is satisfied for the ith oracle query if $I_i < \frac{1}{2}(p_{i+1} - p'_{i+1})|T_i|$.

When i can be inferred from context, we may omit the subscripts and simply say that the economic soundness condition is satisfied when $I < \frac{1}{2}(p - p')|T|$. If one assumes that tokens which have been removed from the reporting pool have zero value (that is, in the case where $\forall i, p'_i = 0$), the economic soundness condition can be expressed as $I < \frac{1}{2}p|T|$, and can be interpreted as saying that the total collective benefit of causing the oracle to lie is less than half of the market cap of the reporting pool.

The motivation behind this definition is as follows. In order for the oracle $\mathcal{A}_0$ to return a false outcome in response to the ith oracle query, at least half of all tokens in the ith reporting pool—that is, at least $\frac{1}{2}|T_i|$ tokens—must be used to lie or abstain (otherwise, the `True` outcome would necessarily receive the most votes and thus become the winner). Each token used to lie or abstain loses $p_{i+1} - p'_{i+1}$ in value. Thus the minimum total cost of causing $\mathcal{A}_0$ to return a false outcome in response to the ith oracle query is given by $\frac{1}{2}(p_{i+1} - p'_{i+1})|T_i|$.

Colloquially, then, the economic soundness condition is satisfied exactly when the total cost of forcing the oracle to lie exceeds the total collective benefit—including all "extraneous" benefit—from doing so. As we will show, this condition is necessary and sufficient for $\mathcal{A}_0$ to be an incentive compatible and individually rational implementation of our desired truth-telling decision function.

It is not surprising that our most important results are predicated on the economic soundness condition being satisfied. All incentive compatible oracles—even centralized ones—necessarily have an analogous soundness condition: if the cost of causing the oracle to lie is less than one could steal by doing so, it would be irrational not to cause the oracle to lie. Unsurprising as this may be, it is important not to take economic soundness for granted. We are not, in general, guaranteed to have the economic soundness condition be satisfied, even if the oracle has been reporting `True` outcomes since its genesis. We investigate the conditions under which we may expect the economic soundness condition to be satisfied in practice in section 5.5.

5.3. Incentive Compatibility—In this section we show that, when the economic soundness condition is satisfied, the simple oracle $\mathcal{A}_0$ is an incentive compatible implementation of our desired truth-telling decision function. In other words, when the economic soundness condition is satisfied and all players are behaving the way we want them to, no player can do better for themselves by unilaterally deviating from that behavior. We do this in the standard way, by showing that the resulting game contains an equilibrium strategy profile that results in the oracle deciding the `True` outcome.

The strategy spaces in the stage game are modeled as follows. Each stage game is associated with an oracle query which comes with an event `E` and an outcome space Ω. The strategy set for each reporter is $\{True, False\}$, modeling the choice reporters make during a fork. The strategy $True$ represents the reporter choosing to report the `True` outcome during the fork, while the strategy $False$ represents the reporter abstaining or reporting a false outcome during the fork. Afterwards, the querier chooses how to update the reporting pool by choosing from $\{PunishFalse, PunishTrue\}$, where $PunishFalse$ represents the querier removing from the reporting pool any tokens used to abstain or lie during the fork, and $PunishTrue$ represents the querier removing from the reporting pool any tokens used to report the `True` outcome during the fork.

Definition (*Honest play*). Let *honest play* refer to the strategy profile in which every reporter always chooses to report `True` and the querier always chooses the move $PunishFalse$.

The following two theorems establish that honest play is in equilibrium in the stage game.

Theorem 5.1. *If the economic soundness condition is satisfied, then always choosing the move PunishFalse is a best response by the querier to any strategy profile chosen by the reporters that results in the oracle returning `True`.*

Proof. See Appendix A.

Theorem 5.2. *If the economic soundness condition is satisfied and the querier always chooses the move PunishFalse, then reporting the `True` outcome is always the best response by every individual reporter.*

Proof. See Appendix A.

As an immediate result of Theorems 5.1 and 5.2, we can see that honest play is in equilibrium in the stage game. Moreover, the payouts (in the stage game) to all players during honest play are strictly greater than their minmax payouts: the minmax payout is $\phi - r_j|T|$ for the individual reporter j, and $-\phi - I$ for the querier. So, by Friedman's folk theorem,[8] honest play is also in equilibrium in the repeated game. (In fact, the folk theorems tell us that honest play is in equilibrium in the infinitely repeated game without discounting, the infinitely repeated game *with* discounting, and the finitely repeated game without discounting. This is nice, as it means our incentive compatibility result is robust against our choice of repeated-game model.) In other words, no individual player has any incentive to deviate from honest-play, and so our mechanism is incentive compatible.

Theorem 5.3. *If the economic soundness condition is satisfied, then $\mathcal{A}_0$ is an incentive compatible mechanism that decides the `True` outcomes of oracle queries.*

Proof. The result follows immediately from Theorems 5.1 and 5.2.

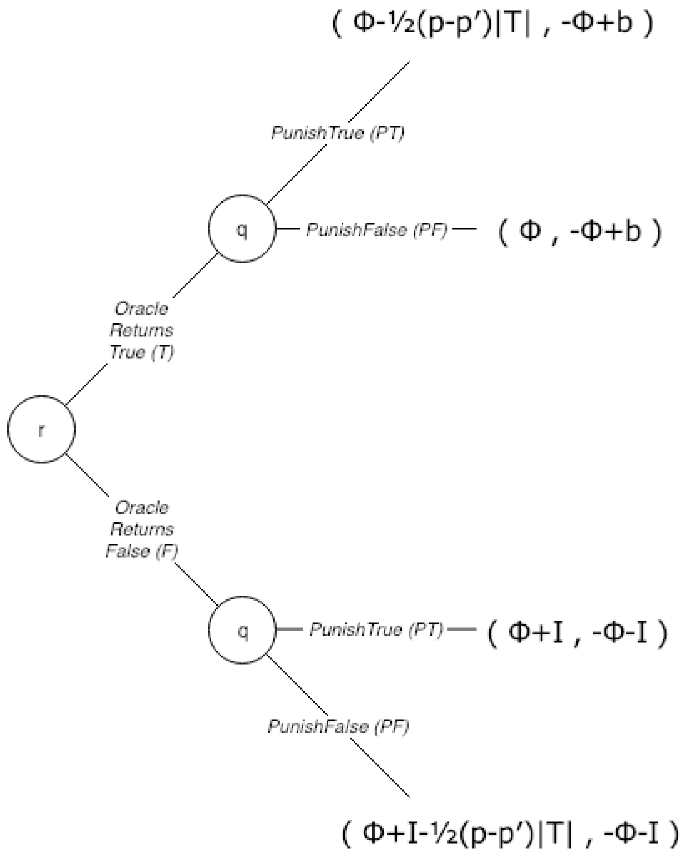
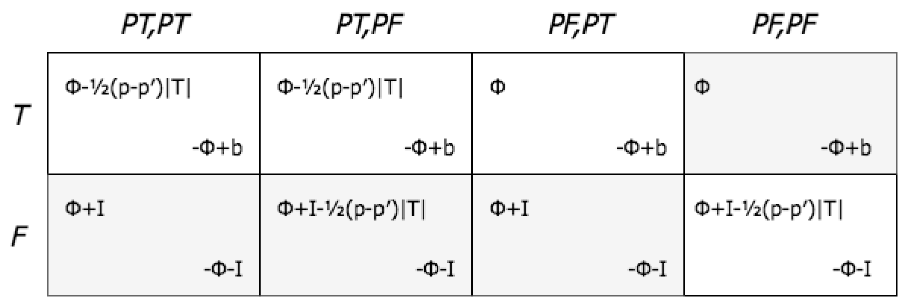


Fig. 1. The stage game shown in extensive form, as described in the proof of Theorem 5.1, with the set of reporters modeled as a single player (r) who may unilaterally decide the outcome of the oracle at the minimum possible cost. The querier (q) then decides which set of coins to remove from the reporting pool. Observe that, at each move, the querier is indifferent among her available moves, and therefore it is trivially the case that *every* response by the querier is a best response and *every* Nash equilibrium is a subgame-perfect Nash equilibrium. This game is shown again in normal form in Figure 2.

5.4. Individual rationality—We have shown that when the economic soundness condition is satisfied, honest-play is in equilibrium for all players engaging with the oracle. Of course, in the real world, we cannot *force* agents to engage with our mechanism. If they are to engage, they must do so willingly. Since players have the choice of not participating, we want our mechanism to satisfy an individual rationality constraint. That is, we want it to be the case that the outcomes for honest-play are better (or at least not worse) for all players than they would achieve by not playing at all. To show that our mechanism is individually rational, we must show that the equilibrium at honest-play results in a non-negative payout for all players.

As in the proof of Theorem 5.1, let b denote the benefit the querier receives when the oracle returns the `True` outcome, and recall that ϕ denotes the fee paid by the querier to the reporters.

Theorem 5.4. *If the economic soundness condition is satisfied and $b > \phi$, then the simple oracle $\mathcal{A}_0$ is individually rational.*



	PT,PT	*PT,PF*	*PF,PT*	*PF,PF*
T	Φ-½(p-p′)|T| ; -Φ+b	Φ-½(p-p′)|T| ; -Φ+b	Φ ; -Φ+b	Φ ; -Φ+b
F	Φ+I ; -Φ-I	Φ+I-½(p-p′)|T| ; -Φ-I	Φ+I ; -Φ-I	Φ+I-½(p-p′)|T| ; -Φ-I

Fig. 2. The stage game from Figure 1 shown in normal form. The set of reporters (modeled here as a single player as described in the proof of Theorem 5.1) is the row player and the querier is the column player. The pure strategy Nash equilibria—when the economic soundness condition is satisfied—are highlighted in gray.

Proof. Suppose the economic soundness condition is satisfied. Then, since honest play is in equilibrium, it will suffice to show that the outcomes for each player during honest play are non-negative. From Figure 2 we can observe that honest play results in individual reporters being paid a *pro rata* share of ϕ, which is always non-negative. Also from Figure 2 we can see that the querier receives a payout of $-\phi + b$, which is non-negative when $b > \phi$. Thus, when the economic soundness condition is satisfied and $b > \phi$, the payout to all players is non-negative during honest play.

5.5. *Tenability of Economic Soundness*—Since all of the results above depend upon the economic soundness condition being satisfied, we will now examine whether it is reasonable to expect the economic soundness condition to be satisfied in practice. Naturally, speculation on the future value of a token can result in arbitrarily high token prices, so it is certainly always *possible* for the economic soundness condition to be satisfied. However, we are interested in whether the reporting fees *alone* can justify a high enough token price for the economic soundness condition to be satisfied.

In particular, we want to know whether the reporting fee can, simultaneously, be small enough that the querier is willing to pay it and large enough to make the market price of tokens (and therefore the market cap of the reporting pool) high enough to satisfy the economic soundness condition. As we will show, this depends very much on the market's current appetite for risk, how the reporting fee is chosen (as a function of I and time), and how high a fee the queriers are willing to bear.

To aid the discussion, consider a betting platform that uses an instance of the oracle $\mathcal{A}_0$ to report the outcomes of national elections. At any given time, the total value of all open bets on the platform is referred to as the *open interest*, which is the maximum benefit that the set of reporters could gain by making the oracle lie: malicious reporters can steal open interest by betting on low-likelihood outcomes and then forcing the oracle to resolve to the outcome on which they bet. So, in this case, I is the open interest.

To consider the simplest case, suppose that the querier (which, in this case, is the betting platform) always chooses *PunishFalse*, and that any tokens used to lie have zero value (that is, $\forall i, p'_i = 0$). Thus the minimum cost of causing the oracle to lie is $\frac{1}{2}p|T|$ and the economic soundness condition is satisfied if and only if $I < \frac{1}{2}p|T|$.

Now suppose the betting platform charges the bettors a fee that is some percentage x of their bet size. Every bet incurs a fee, and all fees collected by the platform will be pooled together and used as the reporting fee when it queries the oracle to decide the outcome of an election. So $\phi = xI$ for every oracle query. For simplicity, assume that every election results in the same volume of bets, and so I remains constant over multiple oracle queries.

Finally, let the market's expected current yield for tokens in T be Y. In other words, we expect the market to behave in such a way that $Y = \frac{A}{p|T|}$, where A is the sum of all reporting fees collected over one year. Therefore, if the betting platform makes n oracle queries in one year, then $A = nxI$, and market behavior will result in a token price of $p = \frac{nxI}{Y|T|}$. Using this value for p in the definition of the economic soundness condition, we can see that we may expect the economic soundness condition to be satisfied when $x > \frac{2Y}{n}$.

As we can see from this simple example alone, the tenability of the economic soundness condition is dependent on factors outside of the implementer's control. While the creators of the betting platform may be able to exert control over the fees they charge (x) and the number of times that they query the oracle in a given year (n), they cannot control the market's expectation of current yield (Y) or whether not their users are *willing to pay* a high enough fee (x) to satisfy the condition $x > \frac{2Y}{n}$.

Example 1: Suppose the market expects a current yield of 30% for holding tokens in T, and the betting platform is making one oracle query per month (so that $Y = 0.3$ and $n = 12$). Then if the bettors are unwilling to pay a fee of 5% on their bets, we should not expect the economic soundness condition to be satisfied.

Example 2: Suppose the market expects a current yield of 25% and the betting platform is making one oracle query per week (so that $Y = 0.25$ and $n = 52$). Then if the bettors are willing to pay a fee of 1% on their bets, we should expect the economic soundness condition to be satisfied.

In conclusion, we are not guaranteed to have the economic soundness condition be satisfied *in general*. Under some conditions it is satisfied quite easily, and under other conditions it is not. Moreover, the tenability of the economic soundness condition depends on some factors outside of the oracle implementer's control, such as the market's appetite for current yield and user tolerance to the minimum required fees.

In plain terms, this is not a simple, drop-in "oracle solution" that is certain to work for any project that needs an oracle. Projects that are considering implementing the oracles presented in this paper should give special consideration to how they structure their fees, the market's appetite for current yield, and whether their users will be tolerant of the minimum required fees.

5.6. Weaknesses—This mechanism achieves only a weak version incentive compatibility (Bayesian-Nash incentive compatibility). While honest play is in equilibrium, and while its resulting equilibrium is Pareto efficient and subgame-perfect, the strategies played during honest play are not strictly dominant. Indeed, the game that arises from $\mathcal{A}_0$ has no strictly dominant strategies for any of the players at all. Reporting truthfully is a best response for individual reporters only if the querier always chooses *PunishFalse*. However, always choosing *PunishFalse* is only a *weakly* dominant strategy for the querier.

The truthfulness of $\mathcal{A}_0$ hinges upon the querier choosing one particular weakly dominant strategy from among several available to her. As we have shown above, it is rational for her to always choose *PunishFalse* during each stage game. However, it is also rational for her to choose one of her other weakly dominant strategies (in the stage game setting). After all, as shown in Figure 1, the querier is indifferent between her available moves during the stage game. It is only when the querier considers the larger repeated game setting that always choosing *PunishFalse* becomes more appealing than the alternatives.

Of course, queriers are unlikely to engage with the mechanism at all unless they intend to engage in honest play. This is because the mechanism is not individually rational outside of honest play, and so the querier would do better for herself by not engaging with the mechanism at all than to engage and play dishonestly. Nevertheless, the situation would be greatly improved if the querier's preference for honest play were strict.

An ideal mechanism would be *dominant strategy incentive compatible* (DSIC), so that players could choose to play honestly without having to give any consideration to the behavior of other players. It is an open question whether there exists such a mechanism that can be implemented in a setting where the players have common knowledge of the truth but the center of the mechanism (*i.e.*, the smart contract) does not.

A less ambitious goal would be to design a variation of $\mathscr{A}_0$ for which always choosing *PunishFalse* were a strictly dominant strategy for the querier in the stage game, even if there were still no dominant strategy available for individual reporters. While reporters would still have to reason about the future behavior of the querier before deciding whether to report truthfully, the resulting stage game would have just one equilibrium: honest play. This would be a clear improvement upon $\mathscr{A}_0$, as $\mathscr{A}_0$ results in multiple equilibria in the stage game—only one of which results in the oracle returning `True`. As with the decentralized DSIC mechanism, it is an open question whether there exists a mechanism with this property that can be implemented in the setting where the center of the mechanism (*i.e.*, the smart contract) does not possess common knowledge of truth.

Finally, this approach necessitates that the economic soundness condition be satisfied. As we saw in section 5.5, this condition is dependent upon things outside of the oracle implementer's control. It is impossible, for example, for the oracle implementer to prevent third-party derivatives being resolved by the oracle's outputs. These derivative bets can be made without the secondary bettors paying any fee to the reporters. As a result, third-party derivatives increase the value of I but may not increase the market cap of reporting tokens, and thereby jeopardize the incentive compatibility of the oracle. This is known as the "parasite problem" and we conjecture that it is unsolvable for all public oracles, both centralized and decentralized.

6. Scaling Strategy

Real life execution of the simple oracle $\mathscr{A}_0$ requires that participants agree on the state of the reporting pool T. The correct state of the reporting pool cannot be verified by a smart contract alone—in particular, determining which outcomes were `True` for each previous oracle query cannot be done by a smart contract. It requires *a posteriori* knowledge of what was common knowledge during each of the previous calls of $\mathcal{F}$ by $\mathscr{A}_0$. To verify the correctness of the current reporting pool, a new user must examine the output of *every* previous call of $\mathcal{F}$ by the oracle. For each Ω-partition returned by $\mathcal{F}$, the new user must decide which set of tokens corresponds to $C_{\texttt{True}}$. Only then can they determine the correct state of the current reporting pool. In brief, on-boarding a new user requires the new user to manually determine the `True` outcome for every previous oracle query. This does not scale.

In the following sections, we define and analyze oracles that do not need to call $\mathcal{F}$ on every oracle query. Instead, we allow queriers to submit a proposed outcome along with their oracle query, and we give the reporters an opportunity to dispute the proposed outcome if they think it is false. If the proposed outcome is not disputed, then the oracle returns the proposed outcome without having to call the fork. (Our assumption that the smart contract platform is censorship resistant is critical here. An attacker that can censor dispute transactions can cause these new oracles to return false outcomes by preventing reporters from disputing false proposed outcomes.) If the proposed outcome *is* disputed, then the oracle uses the fork to determine which outcome to return, just like $\mathcal{A}_0$.

This creates a subgame for each oracle query, where the querier chooses whether to submit their query with the `True` outcome or a false outcome as the proposed outcome, then reporters decide whether or not to dispute the proposed outcome. We leverage the credible threat of a fork along with some bonds to make honest play incentive compatible in the subgame. The result is that, when the economic soundness condition is satisfied, the oracle is expected to return the `True` outcome of a query without having to call $\mathcal{F}$, so the reporting pool does not get updated after every oracle query. On-boarding new users does not require them to manually verify the results of every previous oracle query, but only those which required a fork.

7. An Oracle with a Single Dispute Round, $\mathcal{A}_1$

7.1. Construction—For this new oracle we introduce a *dispute round* which leverages bonds and the credible threat of the fork. When the oracle is queried, the query must be accompanied by a *tentative outcome*, $\hat{\omega} \in \Omega$, and some initial *stake* on that outcome. Then begins a period of time, referred to as a *dispute round*, during which token owners have the opportunity to dispute the tentative outcome in favor of some other outcome in Ω by adding some specific amount of stake to their chosen outcome.

If no dispute takes place during the dispute round, then the oracle returns $\hat{\omega}$ and the initial stake is returned to its original owner. Otherwise the oracle calls $\mathcal{F}$ to determine the winning outcome, just as before. Any stake—whether it was the initial stake that came with the oracle query or stake placed during the dispute round—that was placed on a losing outcome is transferred to those who staked on the winning outcome. In this way, token owners are incentivized to dispute any tentative outcomes that would not win a fork in favor of outcomes that would win a fork.

We will show that, at equilibrium, the oracle returns the `True` outcome without the reporting pool having to be updated.

Later in this section we will construct an oracle that uses a single dispute round and discuss its strengths and limitations. In the following section, we will define an oracle that uses multiple dispute rounds to address those limitations.

First, we define a few algorithms that will be used as subroutines in the following oracles.

Definition (*DisputeRound*). The algorithm *DisputeRound* accepts a tuple $(\mathtt{E}, \Omega, \hat{\omega}, \mathbf{D}, s)$, where $\mathtt{E}$ is an event, Ω is an outcomes space of $\mathtt{E}$, $\hat{\omega} \in \Omega$ is a tentative outcome, $\mathbf{D} = (D_\omega)_{\omega \in \Omega \cup \{\mathtt{Abstain}\}}$ is an Ω-partition of the set of tokens that have been staked on some outcome $\omega \in \Omega$ during the present oracle query, and s is the amount of dispute stake required to dispute the tentative outcome. We let $d = |D_{\hat{\omega}}|$, and we require that $D_{\hat{\omega}}$ not be empty. That is, we require that the dispute round begins with some positive amount, $d > 0$ of stake on the tentative outcome.

Let T be the set of tokens in the reporting pool. All owners of tokens in $T \setminus \bigcup \mathbf{D}$—that is, tokens that are in the reporting pool but have not already been used to stake on some outcome during the current oracle query[9]—have the opportunity (but not the obligation) to dispute the tentative outcome in favor of some other outcome in Ω.

A dispute consists of staking s reporting tokens (referred to as *dispute stake* in this context) on some outcome other than the current tentative outcome. In this paper, for simplicity, we say that disputes require double the stake on the current tentative outcome. Our results remain unchanged if disputes are be made to be α times the stake on the current tentative outcome, so long as $\alpha > 1$.

Dispute rounds have a fixed maximum time limit. If a dispute occurs in favor of outcome ω before the time limit, then *DisputeRound* updates D_ω to include the new dispute stake and returns $(\omega, \mathbf{D}, \texttt{TRUE})$. Otherwise, *DisputeRound* does not modify any cell of the Ω-partition $\mathbf{D}$ and returns $(\hat{\omega}, \mathbf{D}, \texttt{FALSE})$.

At most one token holder may actually dispute a tentative outcome during any given dispute round. That is, while all token holders have the opportunity to dispute a tentative outcome, at most one token holder can actually perform the dispute.

Definition (*Distribute*). The algorithm *Distribute* accepts as input a tuple $(\mathbf{D}, \hat{\omega})$ where $\mathbf{D}$ is an Ω-partition (of dispute stake) and $\hat{\omega} \in \Omega \cup \{\texttt{Abstain}\}$. The algorithm pays out all the tokens in $\bigcup \mathbf{D}$ to those reporters who own tokens in $D_{\hat{\omega}}$, in proportion to the number of tokens they own in $D_{\hat{\omega}}$. That is, if a token holder owns X tokens in $D_{\hat{\omega}}$ then *Distribute* will pay that token holder $\frac{X}{|D_{\hat{\omega}}|} |\bigcup \mathbf{D}|$ tokens.

Colloquially, *Distribute* simply distributes the tokens in $\mathbf{D}$, *pro rata*, to the reporters who staked on outcome $\hat{\omega}$.

Definition (*ChoiceByFork*). The use of the fork as a fallback for deciding the winning outcome is expressed in the subroutine *ChoiceByFork*, described in pseudocode here:

```
def $ChoiceByFork(E, \Omega, T, D)$:
  //call the fork but let only non-dispute stake participate
  $\{C_{\texttt{Abstain}}, C_{\omega_1}, \ldots, C_{\omega_{|\Omega|}}\} \leftarrow \mathcal{F}(E, \Omega, T \setminus \bigcup D)$

  //select winning outcome (remembering to include the dispute
     stake)
  $\hat{\omega} \leftarrow PluralityWinner(\{C_{\texttt{Abstain}} \cup D_{\texttt{Abstain}}, C_{\omega_1} \cup D_{\omega_1}, \ldots, C_{\omega_{|\Omega|}} \cup D_{\omega_{|\Omega|}}\})$

  //redistribute dispute stake
  $Distribute(D, \hat{\omega})$

  //put all dispute stake in the cell corresponding to the
     winning outcome
  $C_{\hat{\omega}} \leftarrow \bigcup D \cup C_{\hat{\omega}}$

  //return winning outcome and the $\Omega$-partition
  return $(\hat{\omega}, \{C_{\texttt{Abstain}}, C_{\omega_1}, \ldots, C_{\omega_{|\Omega|}}\})$
enddef
```

With these definitions in place, we are ready to describe the oracle $\mathcal{A}_1$—the oracle with a single dispute round. A query to $\mathcal{A}_1$ must come along with a tentative outcome $\hat{\omega} \in \Omega$ and some initial stake of $d > 0$ tokens, which we denote $\{t_1, \ldots, t_d\}$. When the query is received, all reporters have an opportunity to dispute the tentative outcome by staking $2d$ tokens on any outcome other than the tentative outcome. If no dispute occurs, the oracle returns the tentative outcome and returns the initial stake back to the querier. If some reporter *does* dispute the tentative outcome, then the oracle calls the fork to determine the winner.

In the event that the tentative outcome is disputed, the oracle will use the fork to determine the winning outcome, just as we did with $\mathcal{A}_0$. We require that any stake that was placed in favor of some outcome during a dispute round must be used to report the same outcome during the fork. That is, if a reporter stakes 10 tokens on outcome ω_1 during a dispute round and then the oracle calls the fork, then that player has no choice but to use those 10 tokens to report outcome ω_1 during the fork. In other words, if a player has staked on an outcome, then they remain committed to that outcome in the event that a fork is called.

In the event that the oracle calls the fork, the oracle returns whatever outcome wins the fork (just as with $\mathcal{A}_0$). Additionally, the initial stake and the dispute stake are redistributed to whichever players staked on the outcome that won the fork. In pseudocode:

```
//initial state
$C^0_{\texttt{True}} \leftarrow T_{genesis}$
$i \leftarrow 0$

def $\mathcal{A}_1(\texttt{E}, \Omega, \phi, \hat{\omega}, \{t_1, \ldots, t_d\})$:
  //increment query counter
  $i \leftarrow i + 1$

  //update the reporting pool
  //only truth-tellers from previous query remain in pool
  $T_i \leftarrow C^{i-1}_{\texttt{True}}$

  //pay owners of tokens in $T_i$
  $Pay(T_i, \phi)$

  //init $\Omega$-partition of dispute stake
  for $\omega \in \Omega \cup \{\texttt{Abstain}\}$:
    if $\omega == \hat{\omega}$:
      $D_\omega \leftarrow \{t_1, \ldots, t_d\}$
    else:
      $D_\omega \leftarrow \emptyset$
    endif
    $\texttt{D} \leftarrow \{D_{\texttt{Abstain}}, D_{\omega_1}, \ldots, D_{\omega_{|\Omega|}}\}$
  endfor

  //have a dispute round
  $(\hat{\omega}_i, \texttt{D}, \texttt{DISPUTED}) \leftarrow DisputeRound(\texttt{E}, \Omega, \hat{\omega}, \texttt{D}, 2d)$

  //if there was no dispute
  if $\texttt{DISPUTED} == \texttt{FALSE}$:
    //then give the initial stake back to the querier
    $Distribute(\texttt{D}, \hat{\omega}_i)$
```

```
        //no reporting tokens are removed from the reporting pool
        $C^i_{\texttt{True}} \leftarrow T_i$

        //return tentative outcome
        return $\hat{\omega}_i$

      //else if there was a dispute
      elseif $\texttt{DISPUTED} == \texttt{TRUE}$:

        //resort to fork
        $(\hat{\omega}_i, \{C^i_{\texttt{Abstain}}, C^i_{\omega_1}, \ldots, C^i_{\omega_{|\Omega|}}\}) \leftarrow \mathit{ChoiceByFork}(\texttt{E}, \Omega, T_i, \texttt{D})$

        //return outcome that won the fork
        return $\hat{\omega}_i$

    endif
  enddef
```

8. Analysis of $\mathcal{A}_1$

8.1. Introduction—We want to show that if the economic soundness condition is satisfied then, at equilibrium, the oracle $\mathcal{A}_1$ returns the `True` outcome of an oracle query *without having to invoke a fork*. In section 5.3 we showed that honest play during a fork is in equilibrium when the economic soundness condition is satisfied. So for the purposes of this section it will suffice to show that, if honest play is expected during a fork, then there exists a unique subgame-perfect equilibrium in the subgame induced by the dispute round for which:

- The querier submits their query with the `True` outcome as the tentative outcome
- When the tentative outcome is the `True` outcome, it is not disputed
- When a tentative outcome is false, it is disputed in favor of the `True` outcome

We will show that, when honest play is expected during forks, there exists a unique subgame-perfect equilibrium in the dispute round game that satisfies these three properties.

8.2. Unique Equilibrium—The dispute round game is modeled as a sequential game in which the querier moves first by choosing whether to submit their query with the `True` outcome as the tentative outcome, or some false outcome as the tentative outcome. The querier necessarily stakes d tokens on this tentative outcome. Then, the reporters choose whether or not to dispute the tentative outcome. If a reporter chooses to dispute the tentative outcome, they must do so by staking $2d$ tokens in favor of some other outcome. We are interested in the case where honest play is expected during a fork, so any play that results in a fork pays out as if the `True` outcome were returned.

Theorem 8.1. *When honest play is expected during a fork, there exists a unique subgame-perfect equilibrium in the single dispute round game. The unique subgame-perfect equilibrium results in the following behavior:*

- *The querier submits their query with the `True` outcome as the tentative outcome*
- *If the querier submits the `True` outcome as the tentative outcome, then no reporter disputes*

- *If the querier submits a false outcome as the tentative outcome, then there exists a reporter who disputes in favor of the* `True` *outcome*

Proof. The strategic decisions facing the querier and an arbitrary reporter during the single dispute round game are illustrated in Figure 3. Reducing the chance moves to their expected values results in a simplified game tree shown in Figure 4, which is solved by backward induction to see the unique subgame-perfect equilibrium (also shown in Figure 4), which results in our three desired behaviors.

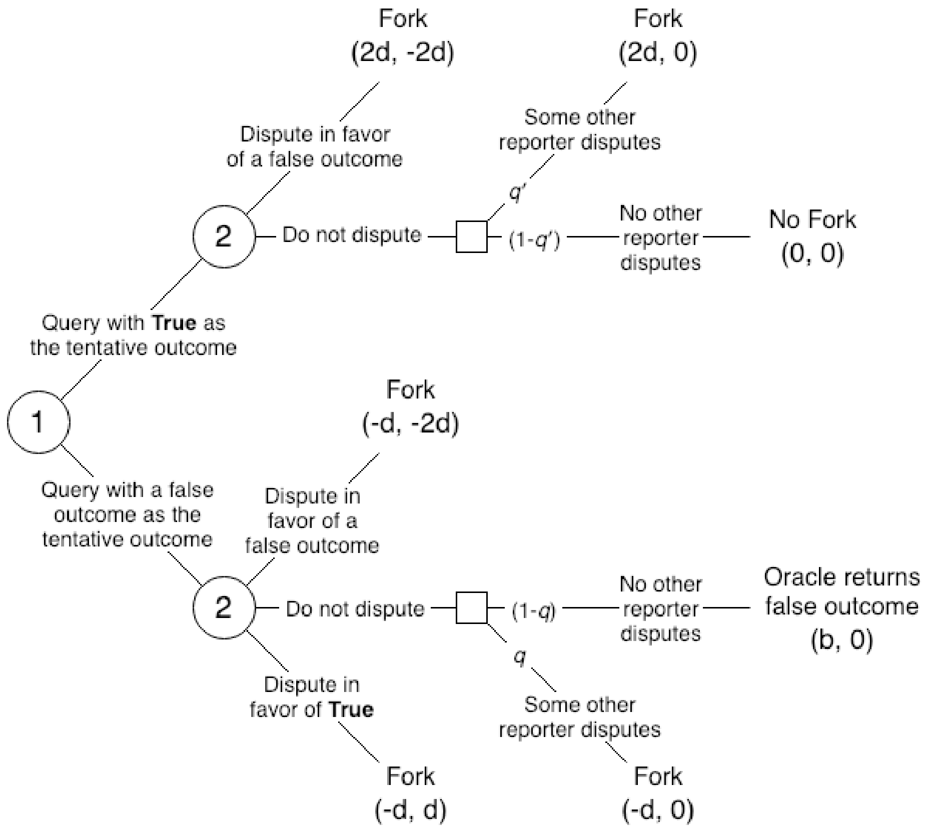

Fork
(2d, -2d)
Fork
(2d, 0)
Dispute in favor of a false outcome
Some other reporter disputes
q'
2
Do not dispute
(1-q')
No other reporter disputes
No Fork
(0, 0)
Query with True as the tentative outcome
1
Fork
(-d, -2d)
Query with a false outcome as the tentative outcome
Dispute in favor of a false outcome
2
Do not dispute
(1-q)
No other reporter disputes
Oracle returns false outcome
(b, 0)
q
Dispute in favor of True
Some other reporter disputes
Fork
(-d, d)
Fork
(-d, 0)

Fig. 3. The single dispute round game in extensive form. The value **d** is the amount of stake on the initial tentative outcome. The value **b** is the benefit to player 1 if the oracle returns a false outcome. Player 2 is an arbitrary single reporter, and the effects of the decisions made by the remaining reporters are modeled as chance moves.

By Theorem 8.1, our desired player behavior is incentive compatible when honest play occurs during a fork. By Theorem 5.3, honest play during a fork is incentive compatible when the economic soundness condition is satisfied. It follows that, when the economic soundness condition is satisfied, the oracle $\mathcal{A}_1$ is an incentive compatible implementation of our desired truth-telling function that does not need to call the fork on every oracle query.

Moreover, one can quickly verify that $\mathcal{A}_1$ is individually rational by observing that the payouts for all players are non-negative when playing according to the unique subgame-perfect equilibrium shown in Figure 4.

8.3. Weakness—While $\mathcal{A}_1$ does satisfy all of our stated design goals, it has the unfortunate property that an attacker can grief honest participants by intentionally causing many forks.

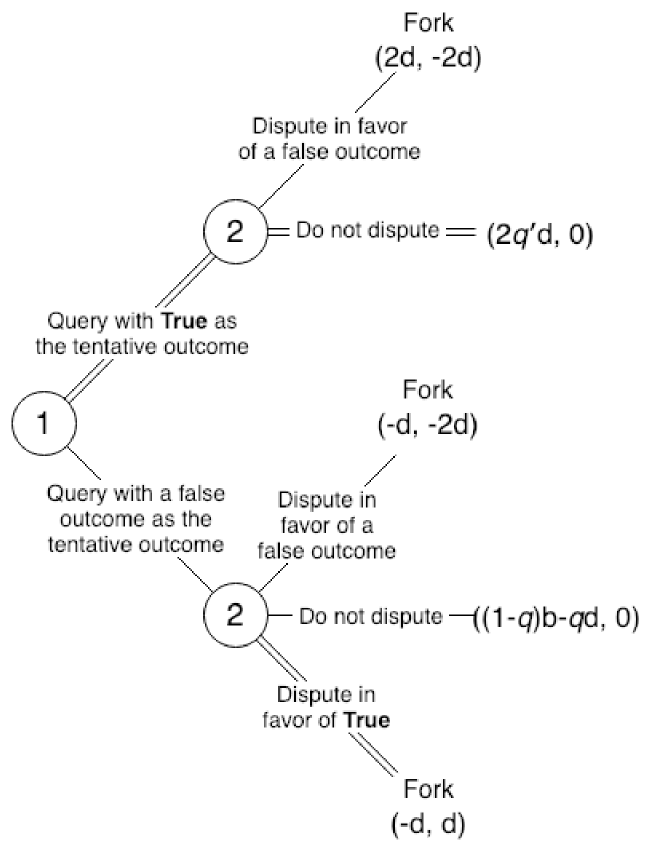

Fork
(2d, -2d)
Dispute in favor of a false outcome
2
Do not dispute
(2q′d, 0)
Query with True as the tentative outcome
1
Fork
(-d, -2d)
Query with a false outcome as the tentative outcome
Dispute in favor of a false outcome
2
Do not dispute
((1-q)b-qd, 0)
Dispute in favor of True
Fork
(-d, d)

Fig. 4. The single dispute round game from Figure 3 with chance moves replaced with expected payouts. The unique subgame-perfect equilibrium is derived via backward induction and is shown with doubled edges.

While such behavior would be irrational in the context of our simple model, it may well be incentivized by extraneous circumstances when interacting with $\mathcal{A}_1$ in the real world. (For example, a competing oracle platform may be financially motivated to damage the scalability properties of $\mathcal{A}_1$ by intentionally causing many forks—perhaps in the hopes of steering new users to their own platform.) We address this weakness in the following section.

9. An Oracle with Multiple Dispute Rounds, $\mathcal{A}_2$

9.1. Motivation—Note that the minimum cost of causing a fork during a call of $\mathcal{A}_1$ is exactly the same as the cost of querying $\mathcal{A}_1$ with a false outcome. An attacker who desires to cause a fork (so as to negatively impact scalability) can simply query the oracle with a false tentative outcome. This outcome will be disputed, causing a fork, and costing the attacker d tokens per query.

If d is small, then it is cheap for honest users to query the oracle, but it is also cheap for an attacker to cause many forks. If d is large, then causing many forks is expensive, but the capital needed to query the oracle in the first place may be prohibitive for honest users. Oracle implementers may want to keep the capital required to query the oracle small, while also increasing the cost of causing a fork.

There may be many ways to achieve this property.[10] Here we will focus on just one: using multiple consecutive dispute rounds.

9.2. Construction—For this new oracle we introduce a *dispute sequence*, which is simply a finite sequence of dispute rounds. As before, the query must be accompanied by a tentative outcome, $\hat{\omega}_1 \in \Omega$, and some initial stake of d tokens. Then the dispute sequence begins.

During the first dispute round in the dispute sequence, token holders have an opportunity to dispute the current tentative outcome by staking $2d$ tokens on any outcome *other than* the current tentative outcome. If nobody disputes $\hat{\omega}_1$, then the d tokens are returned to the querier, the dispute sequence ends, and the oracle outputs $\hat{\omega}_1$.

However, if $\hat{\omega}_1$ *is* disputed, then a new dispute round begins, with the newly championed outcome, $\hat{\omega}_2$, as the tentative outcome for the new dispute round. Once again, all token holders have the opportunity to dispute the new tentative outcome, $\hat{\omega}_2$, in favor of any other outcome—but this time they are required to stake $4d$ tokens on the newly championed outcome.

This continues either until some tentative outcome survives a dispute round without being disputed, or until the dispute stake posted by a disputer reaches some threshold M (a constant chosen by the implementers of the oracle). If the former occurs, the oracle returns the tentative outcome without calling the fork. If the latter, then the oracle calls the fork, just as was done for $\mathcal{A}_1$.

In general, the nth dispute round has tentative outcome $\hat{\omega}_n$, and token holders can dispute $\hat{\omega}_n$ in favor of any other outcome by staking $2^{n-1}d$ tokens on the newly championed outcome. If no such dispute occurs then the dispute sequence ends, the oracle will output $\hat{\omega}_n$, and all token holders who have staked on $\hat{\omega}_n$ (during any dispute round) will receive their stake back, in addition to receiving a *pro rata* share of all stake that was staked on outcomes other than $\hat{\omega}_n$ during any of the dispute rounds. If a dispute *does* occur, and the dispute stake was below the threshold (that is, if $2^n d < M$), then another dispute round begins. And finally, if a dispute does occur, and if the dispute stake has met the threshold (that is, if $2^n d \geq M$), then the oracle resolves via the fork (just as with $\mathcal{A}_1$), and those who staked on the winning outcome receive a *pro rata* share of all the stake that was staked on losing outcomes. The dispute sequence is formalized with the following definition.

Definition (*DisputeSequence*). The algorithm *DisputeSequence* accepts as input $(\mathtt{E}, \Omega, \hat{\omega}, \mathbf{D})$, where $\mathtt{E}$ is an event, Ω is an outcomes space of $\mathtt{E}$, $\hat{\omega} \in \Omega$ is a tentative outcome, and $\mathbf{D} = (D_\omega)_{\omega \in \Omega \cup \{\mathtt{Abstain}\}}$ is an Ω-partition of the set of tokens that have been staked on some outcome $\omega \in \Omega$ during the present oracle query.

The algorithm runs a sequence of dispute rounds, terminating either when a dispute round completes without the tentative outcome being disputed, or until a dispute occurs for which the dispute stake is at least M. The algorithm returns $(\omega, \mathbf{D}, \mathtt{FALSE}, \mathtt{FALSE})$ if no dispute occurred in any dispute round. It returns $(\omega, \mathbf{D}, \mathtt{TRUE}, \mathtt{FALSE})$ if at least one dispute occurred, but no dispute occurred with dispute stake at least M. Finally, it returns $(\omega, \mathbf{D}, \mathtt{TRUE}, \mathtt{TRUE})$ if there was a dispute with dispute stake at least M. In pseudocode:

```
def $DisputeSequence(\mathtt{E}, \Omega, \hat{\omega}_1, \mathtt{D})$:
  $n \leftarrow 1$
  $d \leftarrow |D_{\hat{\omega}}|$
  $\mathtt{EVERDISPUTED} \leftarrow \mathtt{FALSE}$
  while $2^{n-1}d < M$
    //have a dispute round
```

```
      $(\hat{\omega}_{n+1}, \mathtt{D}, \mathtt{DISPUTED}) \leftarrow DisputeRound(\mathtt{E}, \Omega, \hat{\omega}_n, \mathtt{D}, 2^n d)$

      //if there was no dispute
      if $\mathtt{DISPUTED} == \mathtt{FALSE}$:
        return $(\hat{\omega}_{n+1}, \mathtt{D}, \mathtt{EVERDISPUTED}, \mathtt{FALSE})$

      else: // there was a dispute
        $\mathtt{EVERDISPUTED} \leftarrow \mathtt{TRUE}$
      endif

      $n \leftarrow n+1$
    endwhile

    return $(\hat{\omega}_n, \mathtt{D}, \mathtt{TRUE}, \mathtt{TRUE})$
  enddef
```

Next we define the algorithm *BurnAndDistribute*, which behaves the same as *Distribute* with the exception that it burns some of the dispute stake before distributing the rest, *pro rata*, to those that disputed in favor of the chosen outcome.

Definition (*BurnAndDistribute*). The algorithm *BurnAndDistribute* is called in the context of a dispute sequence. It accepts as input a tuple, $(\delta, \mathbf{D}, \hat{\omega})$, where δ is a positive number (chosen by the oracle designer), $\mathbf{D}$ is an Ω-partition of dispute stake where $0 < \delta < |\bigcup \mathbf{D}|$, and $\hat{\omega}$ is an outcome. The algorithm burns (by sending to a provably unspendable address) δ tokens from $\bigcup \mathbf{D}$ and distributes the remaining tokens to the reporters, proportional to the number of tokens they staked in favor of $\hat{\omega}$.

Next, we define a small variation of *ChoiceByFork* that burns some amount of tokens before distributing the rest to token owners. Without such a burn, an attacker with access to a large amount of capital could cause forks without cost, by iteratively disputing every tentative outcome until a fork is initiated. (As long as the final tentative outcome is `True` the attacker would recoup—as a reward for staking on `True`—all the stake they lost by staking on false outcomes.)

Definition (*ChoiceByFork*$'$). The algorithm *ChoiceByFork*$'$ is a variation of *ChoiceByFork* which simply calls $BurnAndDistribute(|\bigcup \mathbf{D}| - \frac{7}{5}|D_{\hat{\omega}}|, \mathbf{D}, \hat{\omega})$ instead of $Distribute(\mathbf{D}, \hat{\omega})$. Its purpose is to guarantee an ROI of 40% to all token holders who disputed an outcome in favor of $\hat{\omega}$. (Note that, while we use a 40% ROI in this paper, our results hold for any ROI strictly greater than 0% and strictly less than 50%.)

With these definitions in place, we are ready to describe $\mathcal{A}_2$, which behaves exactly the same as $\mathcal{A}_1$ with the exception that $\mathcal{A}_2$ uses *DisputeSequence*, *ChoiceByFork*$'$, and *BurnAndDistribute* in lieu of *DisputeRound*, *ChoiceByFork*, and *Distribute*. In pseudocode:

```
//initial state
$C^0_{\mathtt{True}} \leftarrow T_{genesis}$
$i \leftarrow 0$

def $\mathcal{A}_2(\mathtt{E}, \Omega, \phi, \hat{\omega}, \{t_1, \ldots, t_d\})$:
  //increment query counter
  $i \leftarrow i+1$

  //update the reporting pool
```

```
    //only truth-tellers from previous query remain in pool
    $T_i \leftarrow C^{i-1}_{\texttt{True}}$

    //pay owners of tokens in $T_i$
    $Pay(T_i, \phi)$

    //init $\Omega$-partition of dispute stake
    for $\omega \in \Omega \cup \{\texttt{Abstain}\}$:
      if $\omega == \hat{\omega}$:
        $D_\omega \leftarrow \{t_1, \ldots, t_d\}$
      else:
        $D_\omega \leftarrow \emptyset$
      endif
      $\texttt{D} \leftarrow \{D_{\texttt{Abstain}}, D_{\omega_1}, \ldots, D_{\omega_{|\Omega|}}\}$
    endfor

    //run the dispute sequence
    $(\hat{\omega}_i, \texttt{D}, \texttt{EVERDISPUTED}, \texttt{BIGDISPUTE}) \leftarrow DisputeSequence(\texttt{E}, \Omega, \hat{\omega}, \texttt{D})$

    //if there was a large enough dispute
    if $\texttt{BIGDISPUTE} == \texttt{TRUE}$:
      //resort to fork
      $(\hat{\omega}_i, \{C^i_{\texttt{Abstain}}, C^i_{\omega_1}, \ldots, C^i_{\omega_{|\Omega|}}\}) \leftarrow ChoiceByFork'(\texttt{E}, \Omega, T_i, \texttt{D})$

      //return outcome that won the fork
      return $\hat{\omega}_i$
    endif

    //if there was no dispute
    if $\texttt{EVERDISPUTED} == \texttt{FALSE}$:
      // Give the initial stake back to the querier
      $Distribute(\texttt{D}, \hat{\omega})$

    else: //there was a dispute but not with dispute stake greater
        than $M$
      //burn some of the dispute stake[11] and give the rest to the
         winners.
      $BurnAndDistribute(|\bigcup \texttt{D}| - \frac{7}{5}|D_{\hat{\omega}_i}|, \texttt{D}, \hat{\omega}_i)$
    endif

    //no reporting tokens are removed from the reporting pool
    $C^i_{\texttt{True}} \leftarrow T_i$

    //return tentative outcome
    return $\hat{\omega}_i$
  enddef
```

10. Analysis of $\mathcal{A}_2$

Observe that the only difference between $\mathcal{A}_1$ and $\mathcal{A}_2$ is that the latter runs *DisputeSequence*, *ChoiceByFork*$'$, and *BurnAndDistribute* whereas the former runs *DisputeRound*, *ChoiceByFork*, and *Distribute*. As before, we want to show that if the economic soundness condition is satisfied then, at equilibrium, the oracle $\mathcal{A}_2$ is expected to return the `True` outcome of an oracle query *without having to invoking a fork*. In particular, we want to show that, if honest play is expected during a fork, then there exists a unique subgame-perfect equilibrium in the subgame induced by the dispute sequence for which:

- The querier submits their query with the `True` outcome as the tentative outcome
- When the tentative outcome for any dispute round is the True outcome, it is not disputed
- When a tentative outcome for any given dispute round is false, it is disputed in favor of the `True` outcome

As we will show in the following theorem, when honest play is expected during forks, and when it is common knowledge that there exists at least one token holder who does not dispute a true tentative outcome in favor a false outcome, then there exists a unique subgame-perfect equilibrium in the dispute sequence game that satisfies these three properties.

Theorem 10.1. *If honest play is expected during a fork, and if it is common knowledge among token holders that there exists at least one token holder who has not disputed in favor of a false outcome, and if the return for disputing false outcomes is chosen so that* $0 < a < \frac{1}{2}$*, then there exists a unique subgame-perfect equilibrium in the dispute sequence game. The unique subgame-perfect equilibrium results in the following behavior:*

- *The querier submits their query with the* `True` *outcome as the tentative outcome*
- *If, during any dispute round, the* `True` *outcome is the tentative outcome, then no reporter is expected to dispute*
- *If, during any dispute round, a false outcome is the tentative outcome, then there exists a reporter whom we expect to dispute in favor of the* `True` *outcome*

Proof. See Appendix A.

In addition to being incentive compatible and individually rational, this oracle can be made expensive to grief while remaining cheap to query. This is achieved via a judicious choice of the parameters d, M, and the proportion of tokens burned after each dispute. It is important to note, however, that as the quantity $M - d$ increases, so do the number of dispute rounds required to initiate a fork. Thus the maximum amount of time it can take for the oracle to respond to a query increases as the gap between d and M increases. Implementers of this oracle ought to keep this point in mind when choosing parameter values.

11. Incentive Compatibility in the Cooperative Model

Our main results show desirable properties in the non-cooperative model, but it is important to verify that these properties hold in the cooperative model, where players can make binding agreements—after all, what is a smart contract if not a binding agreement? That is, we must verify that our results hold when users are able to collude and form coalitions. (This covers the case where the querier *is* a reporter, because the utility of a "querier-reporter" is the sum of the utilities of the querier role and the reporter role, and so a single "querier-reporter" is equivalent, with respect to utility, to a coalition consisting of the querier and a separate reporter.) As we will see, the forking mechanism is trivially secure against collusion when the economic soundness condition is satisfied, and dispute rounds (and dispute sequences) are secure against collusion if the tokens in T are sufficiently distributed.

	PT,PT	*PT,PF*	*PF,PT*	*PF,PF*
T	$-\frac{1}{2}(p-p')\lvert T\rvert+b$	$-\frac{1}{2}(p-p')\lvert T\rvert+b$	b	b
F	0	$-\frac{1}{2}(p-p')\lvert T\rvert$	0	$-\frac{1}{2}(p-p')\lvert T\rvert$

Fig. 5. The maximum total payout to a coalition that is both able to choose the outcome of an ($\mathcal{A}_0$) oracle query and also includes the querier as a member. If the reporters cause the oracle to lie, their benefit is the querier's loss, so it is zero-sum: the benefits and losses to the coalition exactly cancel out.

11.1. Coalitions in $\mathcal{A}_0$—First, consider the simple oracle $\mathcal{A}_0$ for which all oracle outputs are determined via the fork. Recall from section 5 that the only strategy that is individually rational for the querier is to choose PF, PF, and when the querier chooses this strategy, any coalition of reporters large enough to unilaterally determine the output of the oracle has a unique best response: make the oracle return `True`. It follows that any coalition that is powerful enough to determine the outcome of the oracle query but which does not include the querier as a member does strictly better by making the oracle return the `True` outcome.

Next, consider a coalition that is both powerful enough to determine the outcome of the oracle and also includes the querier as a member. Such a coalition is able to unilaterally decide among the 8 outcomes in Figure 2, and the payout to such a coalition is at most the sum of the payouts for the reporters and the querier for their chosen outcome. The maximum payouts for such a coalition are shown in Figure 5. Note that the maximum payouts (assuming $b > 0$) occur when the oracle returns the `True` outcome. Thus, any coalition that is powerful enough to decide the outcome of the oracle query does best when the oracle returns the `True` outcome.

When the economic soundness condition is satisfied, the analysis is even more straightforward. The economic soundness condition is satisfied exactly when the minimum cost of making the oracle return a false outcome is greater than the maximum total collective benefit for doing so. (Indeed, this is the entire motivation behind its definition.) If the economic soundness condition is satisfied, then the cost to any coalition that makes the oracle lie is greater than the maximum benefit that coalition could receive, and so by the pigeonhole principle the payout would not be individually rational for at least one member of the coalition. Therefore, any imputation must necessarily result in the oracle returning `True`.

11.2. Coalitions in $\mathcal{A}_1$ and $\mathcal{A}_2$—Next, consider oracles $\mathcal{A}_1$ and $\mathcal{A}_2$. For these oracles, there are two ways to get a false response to a query: either via a fork or by having a false tentative outcome go undisputed during a dispute round. For the reasons outlined in the previous section, we expect any coalition deciding the outcome of the oracle *via a fork* to choose a behavior that results in the oracle returning the `True` outcome. So, for this section, we turn our attention to coalitions that may cause the oracle to return a false outcome by causing a false tentative outcome to go undisputed during a dispute round. We will first consider coalitions in $\mathcal{A}_1$, and then show that the strategic situation for coalitions in $\mathcal{A}_2$ can be reduced to those in $\mathcal{A}_1$. We will assume, for this entire section, that the economic soundness conditions is satisfied.

For $\mathcal{A}_1$, recall that, during the dispute round (in the non-cooperative model), every token holder that controls at least $2d$ tokens in T has the opportunity to dispute the tentative outcome. If they dispute the tentative outcome then they receive a benefit of $a2d$,[12] and if they do not dispute the outcome, they receive nothing. In order for the oracle to return a false outcome to a valid oracle query without a fork, the querier must submit their query with a false tentative outcome, and all token holders (either individuals or mutually disjoint coalitions) must choose *not* to dispute the false tentative outcome. In order for such behavior to be incentive compatible, each token holder would have to receive at least $a2d$ when choosing *not* to dispute the false tentative outcome.

Therefore, any coalition payout that would result in the oracle $\mathcal{A}_1$ returning a false outcome to a valid query without a fork would necessarily payout out at least $a2dn$, where n is the number of players (either individuals or mutually disjoint coalitions) that control at least $2d$ tokens in T. In other words, the minimum cost of making $\mathcal{A}_1$ return a false outcome without causing a fork is $a2dn$, because this is the total cost of bribing all other token holders to *not* dispute the false tentative outcome.

As before, let I denote the maximum (gross) benefit a coalition could receive by having the oracle return a false outcome to a valid oracle query. Then we can make the following simple observation relating the distribution of tokens in T to the incentive compatibility of $\mathcal{A}_1$ in the cooperative game-theoretic model: if $\frac{I}{a2d} < n$, then there does not exist any imputation for any coalition that results in the oracle returning a false outcome to a valid oracle query without calling the fork.

Colloquially, if the tokens in T are sufficiently distributed, then the cost of paying all necessary token holders to *not* exercise their opportunity to dispute the false tentative outcome is greater than the maximum benefit of doing so.

Finally, consider the oracle $\mathcal{A}_2$. In this oracle there are several dispute rounds, each of which provides an opportunity for the oracle to return a false outcome without resorting to a fork. Analogous to the analysis of $\mathcal{A}_1$, the kth dispute round requires a bond size of $2^k d$. Every player with at least $2^k d$ tokens in T would have the opportunity to dispute a false tentative outcome during the kth dispute round. They would stand to gain $a2^k d$ for doing so, and would receive no benefit if they chose not to dispute. Thus, any individually rational payout for any coalition that would result in the oracle returning a false outcome in the kth dispute round (without calling a fork) would cost the coalition at least $a2^k dn$. Since such a coalition would receive a (gross) benefit at most I for making the oracle return false, we make the following observation: if $\frac{I}{a2^k d} < n$ then there does not exist any imputation for any coalition that results in the oracle returning a false outcome to a valid oracle query without calling the fork.[13] This is maximized during the first dispute round, when the token distribution requirements for $\mathcal{A}_2$ are identical to those in $\mathcal{A}_1$.

In conclusion, the oracles in this paper can be expected to behave as intended in the cooperative model. For the oracle $\mathcal{A}_0$ we need no further assumptions than those we made in the non-cooperative model. For oracles $\mathcal{A}_1$ and $\mathcal{A}_2$, incentive compatibility in the cooperative model requires that the tokens in the T be sufficiently distributed.

12. Conclusion

We have introduced a new approach to decentralized oracle design—one that is not based upon coordination games. We have presented three specific mechanisms which, under certain reasonable economic conditions, have been shown to be incentive compatible and individually rational in the non-cooperative model. Furthermore, we have shown that if the tokens in the reporting pool are sufficiently distributed, the mechanisms are also incentive compatible in the cooperative model.

Acknowledgements

The authors thank Micah Zoltu for his pioneering contributions to Augur's oracle design, of which the oracles described in this paper are abstractions. We also thank the Augur community for helpful feedback and discussions. Financial support for this research was provided by Forecast Foundation OU.

Notes and References

[1] Buterin, V. "SchellingCoin: A Minimal-Trust Universal Data Feed." (2014) (accessed 21 October 2019) `https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/`.

[2] Sztorc, P. "Truthcoin." (2015) (accessed 21 October 2019) `https://www.truthcoin.info/papers/truthcoin-whitepaper.pdf`.

[3] Schelling, T. *The Strategy of Conflict*. Cambridge: Cambridge University Press (1960).

[4] Buterin, V. "The P + Epsilon Attack." (2015) (accessed 21 October 2019) `https://blog.ethereum.org/2015/01/28/p-epsilon-attack/`.

[5] Peterson, J., Krug, J., Zoltu, M., Williams, A. K., Alexander, S. "Augur: A Decentralized Oracle and Prediction Market Platform." *arXiv.org* (2018) (accessed 21 October 2019) `https://arxiv.org/abs/1501.01042`.

[6] Aumann, R. J. "Agreeing to Disagree." *Annals of Statistics* **4.6** 1236–1239 (1976).

[7] This is because the utility of a player that is both a querier and a reporter is the sum of the utilities of their "querier role" and their "reporter role." So for the purposes of this analysis, the utility of a single "querier-reporter" is indistinguishable from that of a coalition containing a querier and some positive number of reporters.

[8] Friedman, J. "A Non-Cooperative Equilibrium for Supergames." *Review of Economic Studies* **38.1** 1–12 (1971).

[9] Here we are using the notation $\bigcup\mathbf{D}$ to denote the union of sets in $\mathbf{D}$. That is, $\bigcup\mathbf{D} = \bigcup_{X\in\mathbf{D}} X = D_{\texttt{Abstain}} \cup D_{\omega_1} \cup \ldots \cup D_{\omega_{|\Omega|}}$.

[10] A naive approach is to require that the querier submit their query with some small amount d staked on the initial tentative outcome, but then require, say, $10000d$ stake in order to dispute the tentative outcome. While this may technically work in our simple, abstract model with no consideration of external investment opportunities, it is unlikely to work in practice. In particular, it is unlikely that a real-life honest participant would lock up capital to dispute a false tentative outcome for an ROI of just 0.01%. By contrast, our proposed approach guarantees disputers of false outcomes (in favor of `True` outcomes) an ROI of 40-50%. We think this approach is likely to induce our desired player behavior in practice.

[11] Here, we are burning just enough dispute stake so that the ROI for those who disputed in favor of the `True` outcome is 40%.

[12] In $\mathscr{A}_1$ the value a is 0.5, while in $\mathscr{A}_2$ the value a is 0.4.

[13] Notice that, as k grows, the token distribution burden is lessened. This is because the cost of bribing other players to *not* exercise their ability to dispute a false tentative outcome grows exponentially with k.

[14] Observe that this assumption is extremely conservative. A coalition of reporters would have to know the querier's chosen strategy in advance in order to guarantee such low costs. For example suppose the reporter wanted to make the oracle return `True` while minimizing their cost of doing so. If the querier where going to choose *PunishTrue*, then the reporter would minimize costs by voting for `True` with only *half* of the tokens in the reporting pool. If the querier where going to choose *PunishFalse*, then the reporter would minimize costs by voting for `True` with *all* of the tokens in the reporting pool. Here we are being extremely conservative and are assuming that the reporter will always be able to minimize their costs no matter how the querier behaves.

[15] Despite first appearances, we are not resting our fate on the hopes that just a *single* reporter will be motivated to dispute a false tentative outcome. The situation is not so dire. Observe that, since the existence of this reporter is common knowledge, all reporters expect that $q = 1$. Hence, in this case, disputing the false tentative outcome is also the dominant choice—by quite a large margin—for *all* token holders, even if they hold the maximum possible amount of false dispute stake.

[16] As in the base case, we are not resting our fate on the hopes that just a *single* reporter will be motivated to dispute a false tentative outcome. Since the existence of this reporter is common knowledge, all reporters expect that $q = 1$.

Appendix A: Calculations

Theorem 5.1. *If the economic soundness condition is satisfied, then always choosing the move PunishFalse is a best response by the querier to any strategy profile chosen by the reporters that results in the oracle returning* `True`.

Proof. For the purposes of this proof, we will model the set of all reporters as a single player, referred to here as "the reporter," attempting to maximize its total payout. The reporter chooses whether to make the oracle return the `True` outcome or some false outcome. (In this way, we capture the set of all possible strategy profiles of *actual* reporters and separate them into those that would cause the oracle to return the `True` outcome and those that would cause the oracle to return some false outcome; this simplifies our analysis significantly.) We assume that the reporter gets some benefit $I > 0$ if the oracle returns a false outcome, and that this benefit comes at the expense of the querier. We further assume that the reporter will minimize their costs wherever possible, so that the cost of causing the oracle to lie (when the querier chooses to *PunishFalse*) is the minimum possible: $\frac{1}{2}(p-p')|T|$. We make a similar conservative assumption for the cost of causing the oracle to return the `True` outcome when the querier chooses to *PunishTrue*.[14] We assume that the querier receives some benefit $b > 0$ if and only if the oracle returns the `True` outcome. And finally, we note that the querier pays an oracle fee ϕ to the reporter no matter the final outcome.

Given these payouts, the resulting sequential game is show in extensive form in Figure 1 and in normal form in Figure 2.

When the economic soundness condition is satisfied, $I - \frac{1}{2}(p-p')|T| < 0$, and so (as can be seen in Figures 1 and 2) the strategy profile $(True, (PunishFalse, PunishFalse))$ is a Pareto efficient, subgame-perfect Nash equilibrium.

In other words, when the economic soundness condition is satisfied, always choosing the move *PunishFalse* is a best response by the querier to any strategy profile of reporters that causes the oracle to return `True`.

Theorem 5.2. *If the economic soundness condition is satisfied and the querier always chooses the move PunishFalse, then reporting the* `True` *outcome is always the best response by every individual reporter.*

Proof. Suppose the economic soundness condition is satisfied and that the querier always chooses *PunishFalse*. Let j be an arbitrary reporter. We will show that reporting `True` is the best response by j no matter what choices are made by the remaining reporters.

Let r_j denote the proportion of tokens in the reporting pool T that are owned by j. Recall that each reporter always receives a *pro rata* share of the reporting fee ϕ. In particular, reporter j always receives $r_j\phi$.

Since the querier is always choosing the move *PunishFalse*, the reporter j suffers a loss of $\frac{1}{2}(p-p')r_j|T|$ if they lie or abstain during a fork.

We assume that if the oracle returns a false outcome then all reporters who lied or abstained will receive a *pro rata* share of I (the total benefit of causing the oracle to lie). We make the conservative assumption that if the oracle is made to lie, then it was done at the minimum possible total cost to the set of lying reporters. (This is conservative because it maximizes the benefit to a lying reporter in the event that the oracle returns a false outcome.) That is, if the oracle returns a false outcome, then just $\frac{1}{2}|T|$ tokens were used to lie or abstain, and reporter j will receive $2r_jI$ if and only if j reported false or abstained during the fork.

The decision faced by reporter j is modeled with the decision tree shown in Figure 6 where the outcome of the oracle is modeled as a chance move. Replacing the chance moves with their expected values, we get the simplified decision tree in Figure 7, from which we can observe that the payoff to j for reporting `True` is always $r_j\phi$ and the expected payoff for lying or abstaining is $q(r_j\phi-(p-p')r_j|T|)+(1-q)(r_j\phi+2r_jI-(p-p')r_j|T|)$. We need only show that the expected payoff for lying or abstaining is always strictly less than $r_j\phi$.

Because the economic soundness condition is satisfied, the quantities $-(p-p')r_j|T|$ and $2r_jI-(p-p')r_j|T|$ are both negative. It follows that the quantities $r_j\phi-(p-p')r_j|T|$ and $r_j\phi+2r_jI-(p-p')r_j|T|$ are both strictly less than $r_j\phi$. Thus the expected payoff for lying or abstaining is a convex combination of two values that are both strictly less than $r_j\phi$. Hence the expected payout for lying or abstaining is strictly less than $r_j\phi$.

Therefore, if the economic soundness condition is satisfied and the querier always chooses the move *PunishFalse*, then reporting the `True` outcome is always the best response by every individual reporter.

Theorem 10.1. *If honest play is expected during a fork, and if it is common knowledge among token holders that there exists at least one token holder who has not disputed in favor of a false outcome, and if the return for disputing false outcomes is chosen so that* $0<a<\frac{1}{2}$*, then there exists a unique subgame-perfect equilibrium in the dispute sequence game. The unique subgame-perfect equilibrium results in the following behavior:*

- *The querier submits their query with the* `True` *outcome as the tentative outcome*
- *If, during any dispute round, the* `True` *outcome is the tentative outcome, then no reporter is expected to dispute*
- *If, during any dispute round, a false outcome is the tentative outcome, then there exists a reporter whom we expect to dispute in favor of the True outcome*

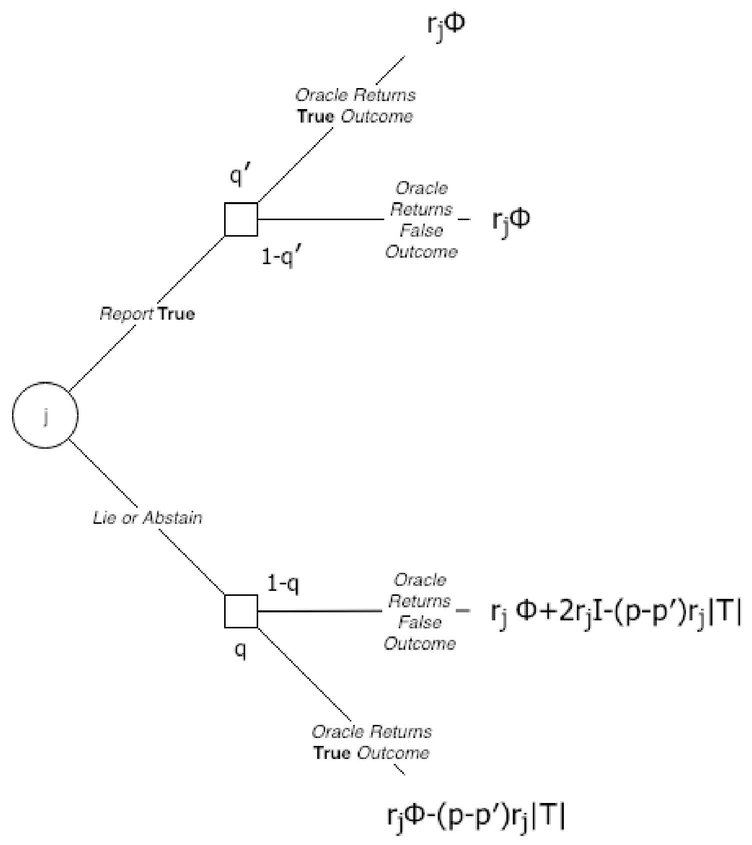

$r_j\Phi$
Oracle Returns True Outcome
q′
Oracle Returns False Outcome
$r_j\Phi$
1-q′
Report True
j
Lie or Abstain
1-q
Oracle Returns False Outcome
$r_j\,\Phi+2r_jI-(p-p')r_j|T|$
q
Oracle Returns True Outcome
$r_j\Phi-(p-p')r_j|T|$

Fig. 6. A decision tree modeling the decision faced by an individual reporter in the stage game when the economic soundness condition is satisfied and the querier always chooses the move *PunishFalse*. The effects of the choices of the remaining reporters are modeled as chance moves. Figure 7 shows a simplified version of this decision tree with the chance moves replaced by their expected values.

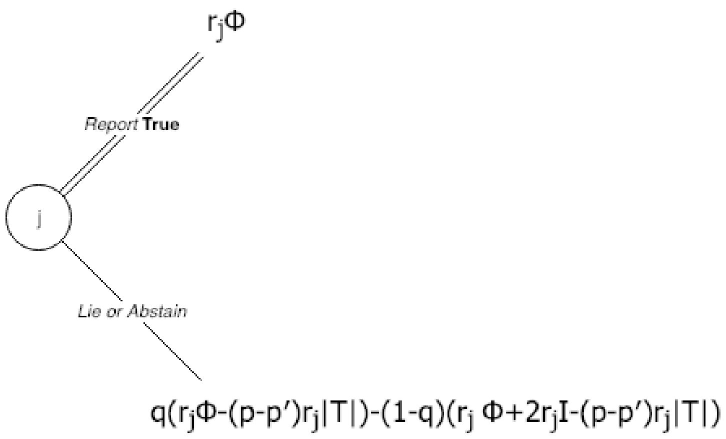


Fig. 7. A simplified version of the decision tree from Figure 6, with the chance moves replaced by their expected values. If the economic soundness condition is satisfied and the querier always chooses the move *PunishFalse*, then reporting the `True` outcome has a strictly better expected value than lying or abstaining.

Proof. We will argue by backward induction, beginning with the final dispute round that would occur before a fork. We will show that in such a final dispute round, a `True` tentative outcome is expected to go undisputed, while a false tentative outcome is expected to be disputed in favor of the `True` outcome. This will form our base case.

Next, we will assume our induction hypothesis, which is that all dispute rounds after (and including) the *k*th dispute round are expected to have their tentative outcomes go undisputed if they are the `True` outcomes, and be disputed in favor of the `True` outcomes if they are false.

Then we will show that the induction hypothesis implies that we can expect the $(k-1)$th dispute round to have its tentative outcome go undisputed if it is the `True` outcome, and be disputed in favor of the `True` outcome if it is false. The conclusion is that we can always expect—in every dispute round—that the tentative outcome will go undisputed if it is the `True` outcome, and will be disputed in favor of the `True` outcome if it is false. An immediate consequence is that the querier is expected to submit their query with the `True` outcome as the initial tentative outcome.

Base Case: Suppose the final dispute round in a dispute sequence occurs at round m. That is, if the tentative outcome of round m is disputed, then $\mathcal{A}_2$ will call the fork. There are two cases: either the tentative outcome at the beginning of the mth dispute round is the `True` outcome, or it is the false outcome.

Case 1: *The tentative outcome of the* mth *dispute round is the* `True` *outcome.*

The strategic decision facing an arbitrary reporter, j, during the mth dispute round is shown in Figure 8, with T_m denoting the total amount of dispute stake reporter j has placed on `True` by the beginning of the mth dispute round, F_m denoting the total amount of dispute stake reporter j has placed on the false outcome by the beginning of the mth dispute round, d denoting the amount of the querier's dispute stake, and a denoting the ROI received by reporters for holding dispute stake on the outcome to which the oracle ultimately resolves. In our case, where each successive dispute round requires 2 times the dispute stake of the previous round and we burn $|\bigcup \mathbf{D}| - \frac{7}{5}|D_{\hat{\omega}_i}|$

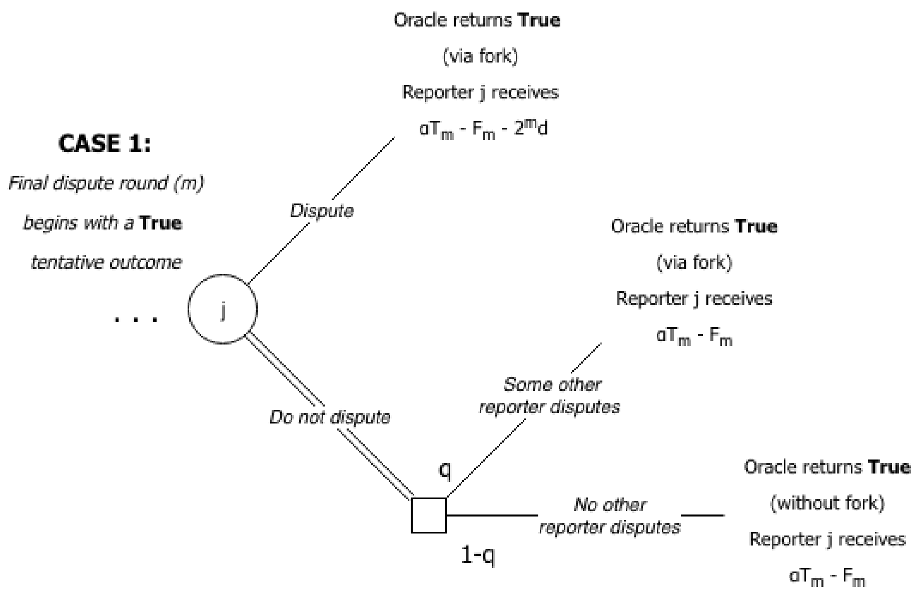

CASE 1:
Final dispute round (m) begins with a True tentative outcome
. . .
j
Dispute
Oracle returns True (via fork) Reporter j receives $aT_m - F_m - 2^m d$
Do not dispute
q
Some other reporter disputes
Oracle returns True (via fork) Reporter j receives $aT_m - F_m$
1-q
No other reporter disputes
Oracle returns True (without fork) Reporter j receives $aT_m - F_m$

Fig. 8. A decision tree modeling the decision faced by an arbitrary reporter during a final dispute round in a dispute sequence in the case where the tentative outcome is `True`. The choice with the highest expected payouts is indicated by doubled lines.

before distributing rewards, $a = 40\%$. (That is, if a reporter stakes X tokens disputing in favor of `True`, and the oracle ultimately returns `True`, then the reporter will receive their original X tokens back, in addition to $0.4X$ more.)

With this notation, disputing the tentative outcome requires a bond of size $2^m d$. Thus, if reporter j disputes the `True` tentative outcome, the oracle will call the fork, which is expected to return `True`, which would result in reporter j receiving a net payout of $aT_m - F_m - 2^m d$. Similarly, if the reporter does not dispute the tentative outcome, but someone else does, then the reporter can expect to receive a net payout of $aT_m - F_m$. Finally, if the reporter does not dispute the tentative outcome and nobody else does either, then the reporter will receive $aTm - Fm$.

Thus it is always best for a reporter to *not* dispute the tentative outcome of the mth round if it is `True`—no matter what other reporters choose to do.

Case 2: *The tentative outcome of the* mth *dispute round is a false outcome.*

The strategic decision facing an arbitrary reporter, j, during the mth dispute round is shown in Figure 9, using the same notation as in case 1. In this case, if reporter j decides to dispute, she can expect a payout of $aT_m - F_m + a2^m d$. Her expected payout if she doesn't dispute is $q(aT_m - F_m) + (1-q)(aF_m - T_m)$ where $0 \leq q \leq 1$.

Recall that, by assumption, there exists at least one token holder for whom $F_m = 0$. For such a token holder, the expected payout for disputing is $aT_m + a2^m d$, while the expected payout for not disputing is $qaT_m - (1-q)(T_m)$. Thus, for this disputer, choosing to dispute is always a dominant strategy, no matter what other reporters do, because for all q where $0 \leq q \leq 1$, $aT_m + a2^m d$ is greater than $qaT_m - (1-q)(T_m)$. So, we can always expect the false tentative outcome to be disputed.[15]

This concludes our base case. We have shown that in a final dispute round, a `True` tentative

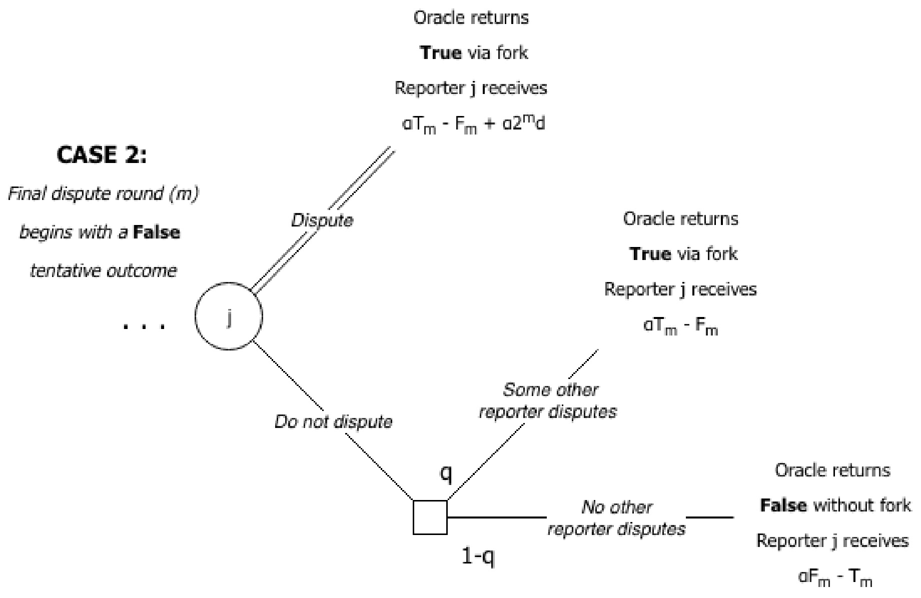

CASE 2:
Final dispute round (m) begins with a **False** tentative outcome
. . .
j
Dispute
Oracle returns **True** via fork
Reporter j receives
$aT_m - F_m + a2^md$
Do not dispute
q
Some other reporter disputes
Oracle returns **True** via fork
Reporter j receives
$aT_m - F_m$
1-q
No other reporter disputes
Oracle returns **False** without fork
Reporter j receives
$aF_m - T_m$

Fig. 9. A decision tree modeling the decision faced by an arbitrary reporter during a final dispute round in a dispute sequence in the case where the tentative outcome is false. By assumption, there exists at least one reporter for whom $F_m = 0$. For such a reporters, disputing is *always* the dominant choice, regardless of the value of q. Indeed, since this fact is common knowledge among all reporters, all reporters know that $q = 1$. Hence, in this case, disputing the false tentative outcome is the dominant choice for *all* reporters, even if they hold large amounts of dispute stake on the false outcome.

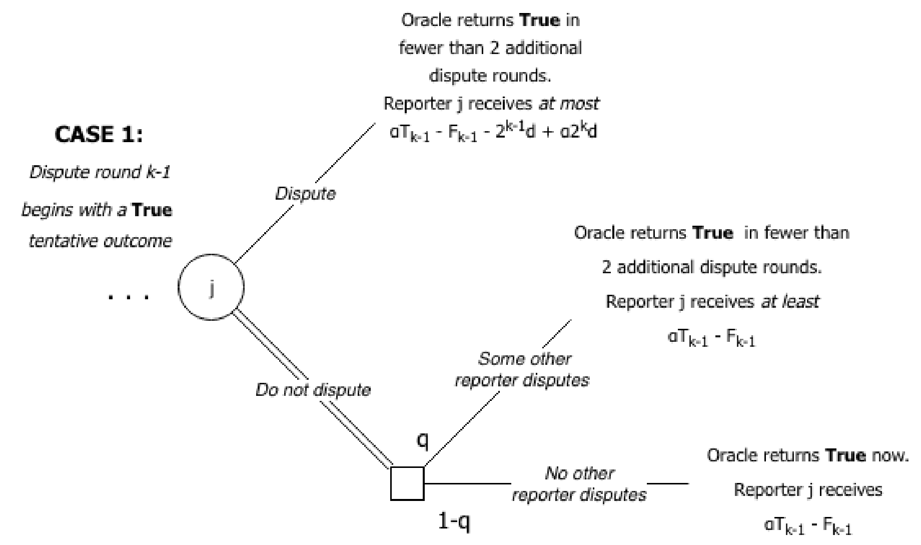

CASE 1:
Dispute round k-1 begins with a **True** tentative outcome
. . .
j
Dispute
Oracle returns **True** in fewer than 2 additional dispute rounds. Reporter j receives *at most* $aT_{k-1} - F_{k-1} - 2^{k-1}d + a2^kd$
Do not dispute
q
Some other reporter disputes
Oracle returns **True** in fewer than 2 additional dispute rounds. Reporter j receives *at least* $aT_{k-1} - F_{k-1}$
1-q
No other reporter disputes
Oracle returns **True** now. Reporter j receives $aT_{k-1} - F_{k-1}$

Fig. 10. A decision tree modeling the decision faced by an arbitrary reporter during the $(k-1)$th dispute round in the case where the tentative outcome is True and the induction hypothesis is assumed. The choice with the highest expected payouts is indicated by doubled lines

outcome is expected to go undisputed, while a false tentative outcome is expected to be disputed in favor of the True outcome.

Induction Step: Now suppose, as our induction hypothesis, that for all k where $2 \leq k \leq m$, we can expect that in the kth dispute round a True tentative outcome will go undisputed, while a false tentative outcome will be be disputed in favor of the True outcome. We need to show that we can expect the same behavior in the $(k-1)$th dispute round. As before, we have two cases: either the tentative outcome in the $(k-1)$th round is the True outcome, or it is a false outcome.

Case 1: *The tentative outcome of the $(k-1)$th dispute round is the True outcome.* The strategic decision facing an arbitrary reporter, j, during the $(k-1)$th dispute round in the case where the tentative outcome is True is shown in Figure 10. Choosing *not* to dispute the True tentative outcome results in the reporter receiving a payout of at least $aT_{k-1} - F_{k-1}$. Choosing to dispute the *True* tentative outcome gives the reporter a payout of at most $aT_{k-1} - F_{k-1} - 2^{k-1}d + a2^kd$. When $0 < a < \frac{1}{2}$, it is the case that $-2^{k-1}d + a2^kd < 0$, so choosing *not* to dispute the True outcome is the strictly dominant decision, no matter what other reporters choose to do.

Case 2: *The tentative outcome of $(k-1)$th dispute round is a false outcome.* The strategic decision facing an arbitrary reporter, j, during the $(k-1)$th dispute round in the case where the tentative outcome is false is shown in Figure 11. Recall that, by assumption, there exists at least one token holder for whom $F_m = 0$. For such a token holder, the expected payout for disputing the false tentative outcome in favor of the True outcome is at least $aT_{k-1} + a2^{k-1}d$, while the expected payout for not disputing is at most aT_{k-1}. Thus, for this disputer, choosing to dispute the false tentative outcome in favor of the True outcome is always a strictly dominant strategy, no matter what other reporters do. Hence, we can expect that the false tentative outcome will be

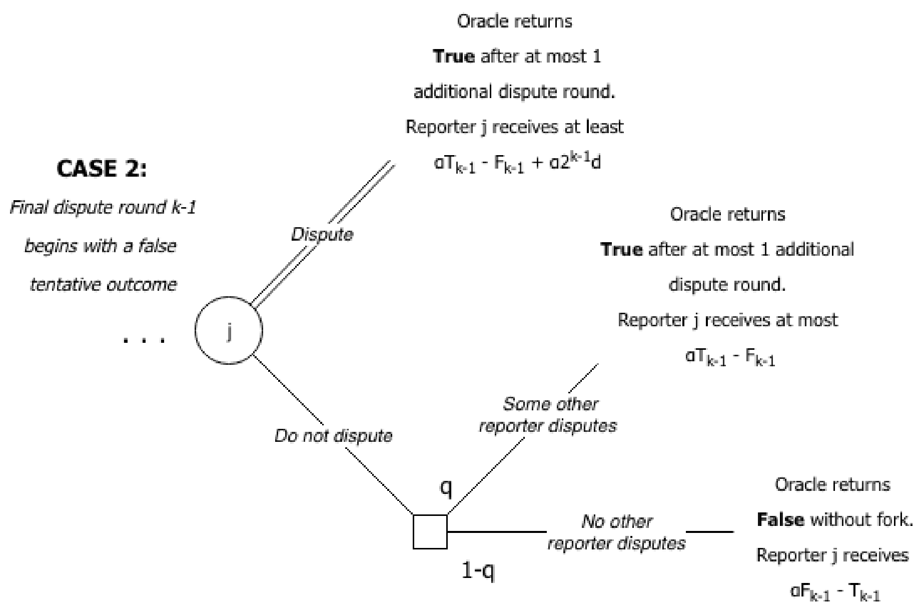

CASE 2:
Final dispute round k-1 begins with a false tentative outcome
. . .
j
Dispute
Oracle returns True after at most 1 additional dispute round. Reporter j receives at least $\alpha T_{k-1} - F_{k-1} + \alpha 2^{k-1}d$
Do not dispute
q
Some other reporter disputes
Oracle returns True after at most 1 additional dispute round. Reporter j receives at most $\alpha T_{k-1} - F_{k-1}$
1-q
No other reporter disputes
Oracle returns False without fork. Reporter j receives $\alpha F_{k-1} - T_{k-1}$

Fig. 11. A decision tree modeling the decision faced by an arbitrary reporter during the $(k-1)$th dispute round in the case where the tentative outcome is false and the induction hypothesis is assumed. The choice with the highest expected payout is indicated by doubled lines.

disputed in favor of the `True` outcome.[16]

This completes the induction step and shows that in every dispute round of the dispute sequence we can expect the tentative outcome to be disputed if and only if it is a false outcome. It follows immediately that we can expect the querier to submit their query with the `True` outcome as the initial tentative outcome.