On Public Crowdsource-based Mechanisms for a Decentralized Blockchain Oracle

Keerthi Nelaturu, John Adler, Marco Merlini, Ryan Berryhill, Neil Veira, Zissis Poulos and Andreas Veneris

***Abstract*—Blockchain technology has created an excitement that was last seen two decades ago when the internet was entering the mainstream. An appealing feature of blockchain technology is smart contracts. A smart contract is an executable code. It runs on top of the blockchain facilitating an agreement between untrusted parties. These smart contracts have a major limitation, namely they cannot operate on information external to the blockchain. The inability to query such information has paved the need for trusted entities called "oracles." These oracles attest to facts without the robust security guarantees that blockchains generally provide. This can potentially harm the integrity of the network and lead to centralized points-of-failure. To address this concern, this paper proposes a decentralized oracle which is based on a voting-based game that decides the truth or falsity of queries. In the context of this work, we are only interested in *binary markets*, *i.e.*, queries which can be `True`, `False`, or `Unknown`. When requesting facts from an oracle, a user submits binary queries. Reporters (or certifiers) respond to the queries by placing monetary stake. A formal analysis of the system parameters is presented, which shows that the proposed platform incentivizes a Nash equilibrium for truthful reporting. An extension to the base protocol is also described and profiled against the original framework. Lastly, we discuss a prototype architecture, along with additional features to be considered during implementation.**

Index Terms - blockchain, decentralized oracle, crowdsourcing, voting, permissioned, permissionless, Nash equilibrium

I. Introduction

Crowdsourcing, also known as the "wisdom of crowds," is a technique that can help break cost barriers by outsourcing tasks to market places. In general, the process consists of a set of requesters posting tasks that can be unrelated and require varied expertise. A set of submitters take up these tasks and submit solutions for a minimal reward. In this context, it is important to note that the stakeholders of the system do not need have mutual trust. A similar topology is observed in blockchain networks.

A blockchain is a distributed database that defines an order of its transactions. It is immutable, trustless and permissionless. Even though the initial implementations of blockchain were developed to create digital currency [1], the technology has revolutionized a wide range of industries. These include a wide range of industries, such as supply chains and logistics [2, 3], insurance [4], healthcare [5], and financial services [6], among others. Smart contracts—software code executed on a virtual machine—transform conventional contracts to digital agreements. One well-known limitation with smart contracts is that they can only access data that is stored in the virtual machine's memory. This makes the network disconnected from any real world data [7, 8]. Trusted entities called *oracles* are needed to attest to facts, in an effort to bring external data into the blockchain's state. These oracles obtain off-chain information, by allowing members of the public to provide answers to questions, using a concept similar to crowdsourcing. Reward mechanisms incentivize members to present truthful responses.

In this paper, we propose a novel decentralized oracle protocol leveraging crowdsourced voting mechanisms, which are agnostic to the core consensus in a blockchain network. At a high level, a user querying an oracle would submit questions along with a bond. These questions converge to Boolean responses that consist of either a `True` or `False` value. Once a question is posted, the oracle collects votes from the participants. If this converges to a result then it will distribute the posted bond to all the contributors who arrived at the winning outcome. We introduce a general mathematical model of decentralized oracles and present a base protocol along with an extension. Using this model, we provide a thorough theoretical analysis to prove that there exists an honest Nash equilibrium for these binary markets.

The remainder of this paper is organized as follows. In Section II, we identify a general mathematical model. Section III follows with a description of the novel decentralized oracle protocol and its analysis. Section IV provides a simplified version of the protocol. In section V, a prototype architecture is described with implementation details. Section VI examines existing blockchain oracles and contrasts them to the proposed mechanisms. In section VII we suggest additional features that can be added during implementation. Section VIII highlights practical applications for decentralized oracles, while section IX concludes this paper.

II. Preliminaries

A. A Decentralized Oracle model

In this paper, we propose a model for a decentralized oracle that decides the truth value of Boolean queries posted by submitters. The oracles presented in this work are permissionless, for instance, users do not undergo any Know Your Customer (KYC) process before joining the system. We assume each query p has a response truth value t that is either `True` (`T`) or `False` (`F`) or `Unknown` (ϕ). There are $N\,players$ —reporters and certifiers. For each query p and each randomly chosen player $i \in [1, N]$, let the *private opinion* of the player regarding the truth value of p be $PO_i(p) \in \{\texttt{T}, \texttt{F}\}$. We assume that $PO_i(p)$ is fixed, and in an honest scenario it is unknown to reporters other than i. Reporters who choose to collude and share their private opinions are considered adversarial, which is discussed in further detail in section IV-B. Each player i

has an accuracy $q_i \in [0,1]$ *i.e.,* the probability that player i is correct about a given query. Formally:

$$PO_i(p) = \begin{cases} t & \text{with probability } q_i \\ \neg t & \text{with probability } (1-q_i) \end{cases} \quad (1)$$

Each player's beliefs are independent of all other players' private opinions *i.e.,* the value of $PO_i(p)$ is independent of $PO_j(p)$ for all $j \neq i$. Further, $PO_i(p)$ is independent of $PO_i(p')$ for all $p \neq p'$. That is, a player's private opinion in a proposition is independent of her beliefs in all other propositions. Additionally, each reporter i has a *voting strategy* $\sigma_i(p)$ where

$$r_{ip} = \sigma_i(PO_i(p)) \quad (2)$$

determines the response r_{ip} that reporter i actually submits to the oracle. Based on this definition we identify three kinds of reporters: an *honest reporter* has $\sigma_i(PO_i(p)) = PO_i(p)$, a *lazy reporter* who irrespective of $PO_i(p)$ reports $\sigma_i(PO_i(p)) = \texttt{T}$ or $\sigma_i(PO_i(p)) = \texttt{F}$ always (*i.e.,* they publish the same response all the time), and a *deceptive reporter* is one who submits against their $PO_i(p)$ always, *i.e.,* $\sigma_i(PO_i(p)) = \vdash PO_i(p)$.

Lazy and deceptive reporters are the two types of adversaries we consider in our system. Lazy reporting is a special case of *Verifier's Dilemma*[9]. The problem, simply stated, is the existence of a degenerate Nash equilibrium where voters always report the same answer on all questions regardless of what they believe to be true, so as to secure economic incentives/profits. A simple mechanism behind such an oracle cannot guarantee that payoffs for truthful reporting are greater than payoffs in a lazy Nash equilibrium, and therefore cannot guarantee that voters will contribute correct information. Specifically, in voting-based protocols, lazy voting can cause degenerate coordination strategies. Deceptive reporters would try to manipulate the oracle into generating an outcome of their choice. The protocol analysis establishes honest Nash equilibrium in the presence of these adversaries.

We define two random variables: $\Gamma(p) \in \{\texttt{T}, \texttt{F}\}$—the private opinion of a reporter selected randomly on a query p, and $A(p) \in \{\texttt{T}, \texttt{F}\}$—the *answer reported* by a reporter selected randomly on query p.

For the proposed protocol to work, we assume the oracle is deployed on an existing decentralized platform such as Ethereum [10] or Hyperledger [11]. We refer to the platform as *the executor*. The executor maintains a list of Boolean queries as part of its state, which can be submitted by any user of the system. Once added to the list, any reporter can communicate their response. The voting stake, which is the total amount deposited by all reporters, is denoted by D_r. As the protocol converges to a conclusion on a query, the executor will move it to a *completed* state, distribute rewards and cease to accept new reports for the same query. At this stage, the completed query can be replaced by another one.

Lastly, we define the *correctness* of an oracle. No oracle can determine the actual fact of a question by itself [12]. Also, users can submit subjective questions which do not have an objective answer. Hence, we formalize the correctness with following set of definitions:

Definition 1. Let us consider reporter $i \in N$. A randomly selected reporter's private opinion on query p is called *Estimated Private Opinion (EPO).*

$$EPO(p) = \begin{cases} \texttt{T} & P(\Gamma(p) = \texttt{T}) > 0.5 \\ \texttt{F} & P(\Gamma(p) = \texttt{T}) < 0.5 \\ \phi & P(\Gamma(p) = \texttt{T}) = 0.5 \end{cases} \quad (3)$$

We say oracle has concluded correctly with respect to the query p if it arrives at an output *equal to* $EPO(p)$.

Additionally, let β_{ip} denote the probability that a specific reporter $i \in N$ reports an answer to p which is equal to $EPO(p)$. Assuming that i does not know $PO(p)$ for other reporters, this is the same as the probability that i answers $EPO(p)$ on a $PO(p)$ value selected randomly from N:

$$\beta_{ip} = P(\sigma_i(\Gamma(p)) = EPO(p)) \quad (4)$$

We also let β_p denote the probability that a reporter selected randomly from *N* reports an answer equal to $EPO(p)$:

$$\beta_p = P(A(p) = EPO(p)) \quad (5)$$

The key distinction between the definitions of β_{ip} and β_p is that β_{ip} pertains to a specific reporter i with a fixed strategy $\sigma_i(p)$, and thus may be different for reporters with different strategies, whereas in β_p the strategy also varies with the random variable $A(p)$.

For example, if all reporters in *N* adopt the honest reporting strategy, then $P(A(p) = \texttt{T}) = P(\Gamma(p) = \texttt{T})$ and we have:

$$\beta_p = \begin{cases} P(\Gamma(p) = \texttt{T}) & EPO(p) > 0.5 \\ P(\Gamma(p) = \texttt{F}) & EPO(p) < 0.5 \\ P(\Gamma(p) = \phi) & EPO(p) = 0.5 \end{cases} \quad (6)$$

III. Base protocol

This section introduces our voting-based oracle protocol. We begin by expanding on the user roles and the underlying process of the voting game. We conclude with a detailed description of the gaming workflow.

A. Overview

Users of our protocol participate in one (or more) of the following three roles: *submitters, reporters, and certifiers.* We further discuss the behavior of these roles, also depicted in Figure 1:

- **Submitters** post queries to the executor along with a bounty to fund (in part) the further process of evaluating the Boolean queries.
- **Reporters** play a low-risk/low-reward game. When a reporter wants to participate in the voting process, a certain amount of stake is to be deposited. A query is uniformly chosen at random by the executor and is assigned to a reporter. As depicted by steps $2, 4, 5$ & 9 in Figure 1. The outcome of the voting process is a function of the sum of the votes weighted by the deposits. The maximum voting deposit is a parameter of the system and is discussed later in the paper. A reporter does not know what query they will be assigned beforehand.
- **Certifiers** play a high-risk/high-reward game. Unlike reporters, certifiers get to choose which query they would

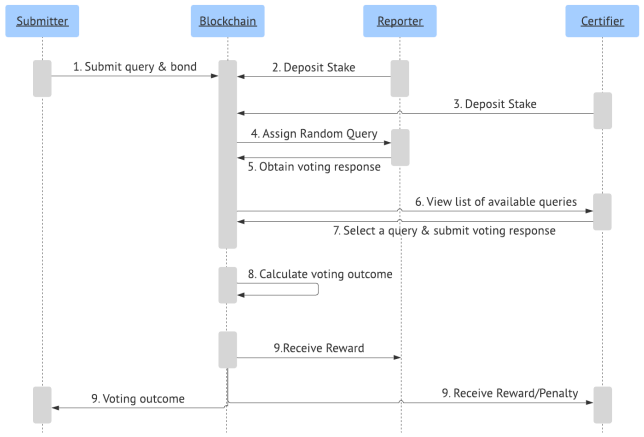


Fig. 1: Submission & Voting Process

like to place a deposit on. The certifier voting outcome is a function of the sum of certifications weighted by the deposits. This process is illustrated through steps $3, 6, 7$ & 9 in Figure 1. The intuition behind introducing ceritifiers is to encourage them to place bets on queries for which there is a high degree of confidence that they are `True` or `False`.

As the system is permissionless, anyone can chose to be a reporter, a certifier, or both. Not all queries will be certified as they are selected by choice and there is no mandate to do so. The minimum certification deposit size is a system parameter and should be large enough that certifying incurs a substantial cost, and therefore there are substantial penalties in case of malfeasance.

Both reporters and certifiers submit their voting response as a sealed vote. The major distinction between these two roles is that certifiers get to choose the query and have a restriction on the minimum stake. Contrarily, reporters get assigned a random query with a constraint on the maximum amount of stake. By splitting the roles and incentives between users we protect the system from malicious attacks as demonstrated through analysis in section III-F.

Choosing appropriate system parameters is highly important for the validity of the voting process. In our game, we have two such parameters: the maximum voting deposit and the minimum certification deposit. The maximum voting deposit size should be small relative to the total voting stake on each query. If a single vote can account for 100% of a query's total voting stake, an adversary can have total control over the outcome of a randomly drawn query. Conversely, if it is very small, an adversary would somehow need to draw the same query repeatedly to control its validity outcome.

On the other hand, the minimum certification deposit should be large enough that certifiers incur sufficient risk. A large deposit would avoid the possibility of certifiers abusing the system by manipulating their voting response. At first sight, it seems like individual certifiers have enormous influence on the process for individual queries, and indeed this can be the case. However, as described later in the paper, the certifiers alone cannot force the oracle to produce an incorrect value and they are encouraged to behave honestly by the incentive structure; otherwise they face large penalties.

For each player, let s_p and c_p denote the amount of stake that reporter and certifier have deposited to provide their $EPO(p)$, respectively. Let s_{max} and c_{min} denote the maximum voting stake and minimum certifying stake parameters, respectively.

B. The Query List

The query list is constructed by a smart contract in the executor based on the requests from submitters. Also denoted by $\mathbb{Q}$, the query list will have a fixed size of $|\mathbb{Q}|$. Each query $p \in \mathbb{Q}$ has an unknown truth value t and is associated with bounty value B. The voting game described later is executed by all players (reporters and certifiers) simultaneously on all the queries in $\mathbb{Q}$. We assume there are two reward pools for certifiers: $R_{\texttt{T}}$ and $R_{\texttt{F}}$ monetary units, to provide rewards for `True` and `False` outcomes, respectively. This is specifically targeted to avoid a *lazy equilibrium* scenario in which voters always have a constant response `True` or `False` so as to maximize their profits without putting any effort in considering a specific truth value. We discuss the rationale behind this later in the paper.

As the space is restricted to $|\mathbb{Q}|$, the list construction involves a separate process which is not described here as it is outside the scope of this work. As an example, one could run an auction for the $|\mathbb{Q}|$ spaces, with the auction price becoming the bounty for the query.

C. System Description

An overview of the protocol workflow is described in this subsection. It basically consists of interactions of reporters and certifiers with the queries list. Reporters must be engaged to vote on random queries, while certifiers submit their voting response for a query of their choosing.

1) Reporting: This process is initiated by a reporter $i \in N$ whenever they deposit a certain amount of stake $(s_{i,x})$ where x is the yet to be assigned query. Once a stake is registered, the reporter is prepared to vote for a query. The executor then validates if the amount satisfies condition that $s_{i,x} \leq s_{max}$ and assigns a randomly-chosen query from the query list $[1, |\mathbb{Q}|]$. So, a reporter may be voting on a query more than once. Random number generation within smart contracts has been a widely researched topic in recent years and several techniques exist to do it securely [13, 14]. The final step of this process is for the reporter to submit the $r_{i,x}$ which they derive from their voting strategy and private opinion. A sealed vote can be created using a *commit-reveal* scheme, which requires a reporter to commit a hash of their vote concatenated with a nonce, later revealing the vote and the nonce to unseal the vote.

2) Certifying: A certifier participates in voting for a query of their choice by placing a large deposit. The certifier simply submits a monetary stake $c_{i,p} \geq c_{min}$ and a sealed certification in accordance with their voting strategy *i.e.,* $r_{i,p}$ for a query p.

3) Termination and Decision: Once a query p has accumulated a sufficient amount of funds, it is available for the final decision. The total amount of reporting stake accumulated is D_r. At this stage, the oracle computes four values: $s_{TOT,p,\texttt{T}}, s_{TOT,p,\texttt{F}}, c_{TOT,p,\texttt{T}}$ and $c_{TOT,p,\texttt{F}}$. The values represented by these variables are disclosed in Table I. For

TABLE I: Decision variables

Variable	Value represented by variable
$s_{TOT,p,\mathtt{T}}$	Total stake collected for query p from reporters for truth value `True`
$s_{TOT,p,\mathtt{F}}$	Total stake collected for query p from reporters for truth value `False`
$c_{TOT,p,\mathtt{T}}$	Total stake collected for query p from certifiers for truth value `True`
$c_{TOT,p,\mathtt{F}}$	Total stake collected for query p from certifiers for truth value `False`

TABLE II: Reporting outcomes

Outcome	Condition
T	$s_{TOT,p,\mathtt{T}} > s_{TOT,p,\mathtt{F}}$
F	$s_{TOT,p,\mathtt{F}} > s_{TOT,p,\mathtt{T}}$
ϕ	$s_{TOT,p,\mathtt{T}} = s_{TOT,p,\mathtt{F}}$

each $b \in \{\mathtt{T}, \mathtt{F}\}$, the above mentioned values are computed as follows:

$$s_{TOT,p,b} = \sum_{i=1}^{N} s_{i,p,b} \quad c_{TOT,p,b} = \sum_{i=1}^{N} c_{i,p,b}$$

The outcomes for both reporting and certifying are computed by a simple majority rule as shown in Tables II and III. In the case of a tie, we assign the outcome to be ϕ. For simplicity, we exclude a detailed description of this scenario as this does not affect the design and analysis of the game.

In Table IV, we illustrate the game and oracle outcomes for each of the nine possible combinations of certification and reporting results. The headings in the top row correspond to certification, while the labels in the first column correspond to reporting. The game has three possible outcomes (T, F and ϕ) each of which carries its own reward structure. Note that the game outcome is only used to determine rewards and penalties, and does not correspond to the oracle's output. Indeed, anyone observing the system is free to compute oracle outcomes as they wish depending on the context of the query. For the sake of this presentation a suggested mapping is presented in Table IV-(B). The suggested oracle output follows the reporting outcome if it matches the certification outcome or the certification outcome is ϕ. The oracle is not restricted to an output of $\{\mathtt{T}, \mathtt{F}, \phi\}$, but could instead have an output in the range $[0, 1]$ indicating confidence in the truth or falsity of the query.

TABLE III: Certifying outcomes

Outcome	Condition
T	$c_{TOT,p,\mathtt{T}} > c_{TOT,p,\mathtt{F}}$
F	$c_{TOT,p,\mathtt{F}} > c_{TOT,p,\mathtt{T}}$
ϕ	$c_{TOT,p,\mathtt{T}} = c_{TOT,p,\mathtt{F}}$

TABLE IV: Outcomes for (A) The Game and (B) The oracle

(A) Game			
R \ C	T	F	ϕ
T	T	ϕ	ϕ
F	ϕ	F	ϕ
ϕ	ϕ	ϕ	ϕ

(B) Oracle			
R \ C	T	F	ϕ
T	T	ϕ	T
F	ϕ	F	F
ϕ	ϕ	ϕ	ϕ

D. Rewards and Penalties

The principal rule is to reward players whose positions match with either T and F outcomes. Players who took opposing positions are penalized. In case of `Unknown` outcomes, certifiers are penalized and reporters are left with no rewards or penalties. As argued in this paper, this scheme incentivizes the participants to behave honestly.

For the rest of this subsection, we fix a player $i \in [1, N]$ and query $p \in \mathbb{Q}$ so as to enumerate the rewards and penalties for each of the three possible game outcomes. Reporter and certifier rewards are presented separately although, as noted earlier, nothing prohibits a player from being both a reporter and a certifier.

Rewards and penalties are distilled into a single value rew_r for reporting and rew_c for certification. A negative value indicates a penalty, while a positive value indicates a reward. The results are summarized in Table V.

1) `True` *and* `False` *Outcomes:* In the case of a `True` outcome, the reporting reward is as follows.

$$rew_r = \left(\frac{s_{i,p,\mathtt{T}}}{s_{TOT,p,\mathtt{T}}}\right) \times B - s_{i,j,\mathtt{F}} \tag{7}$$

The player's reporting reward is their share of the *T-reporting* stake times the query's bounty amount. Their penalty is equal to their *F-reporting* stake. The certifier reward is shown in Eq. 8 below.

$$rew_c = \left(\frac{c_{i,p,\mathtt{T}}}{c_{TOT,p,\mathtt{T}}}\right) \times \left(R_{\mathtt{T}} \times \frac{1}{\tau}\right) - c_{i,p,\mathtt{F}} \tag{8}$$

A certifier's reward is equal to her share of the *T-certifying* stake times the *true certifier reward pool* amount $R_{\mathtt{T}}$ times the reciprocal of the *certification target* τ. The `True` reward pool is a reward pool used to reward certifiers who correctly certify articles as `True`. The certification target can be seen as the number of certifications that the pool should have enough funds to pay for. For instance, if $R_{\mathtt{T}} = 1000$ and $\tau = 10$, then 100 monetary units will be distributed to the certifiers. The next proposition will have $R_{\mathtt{T}} = 900$, and therefore 90 units will be distributed.

In the case of a `False` outcome, the rewards and penalties are similar.

2) `Unknown` *Outcome:* For an `Unknown` outcome, reporters are neither rewarded nor penalized ($rew_r = 0$), while certifiers are penalized as follows.

$$rew_c = -\left(c_{i,p,\mathtt{F}} + c_{i,p,\mathtt{T}}\right) \tag{9}$$

That is, certifiers forfeit all of their stake, regardless of agreement with reporters. The rationale for penalizing certifiers and not reporters is that certifiers chose to vote for a query p. Reporters received a query at random.

E. Monetary Flows

Note that reporters are not rewarded for `Unknown` outcomes. Therefore sometimes bounties are not claimed. Thus far, the distribution of penalties and unclaimed bounties has not been discussed. The funding of reward pools and bounty amounts has only been briefly mentioned.

TABLE V: Summary of rewards and penalties

	Reward		Penalty	
Outcome	Reporters	Certifiers	Reporters	Certifiers
T	$\left(\frac{s_{i,p,\mathtt{T}}}{s_{TOT,p,\mathtt{T}}}\right) \times B$	$\left(\frac{c_{i,p,\mathtt{T}}}{c_{TOT,p,\mathtt{T}}}\right) \times \left(R_{\mathtt{T}} \times \frac{1}{\tau}\right)$	$s_{i,p,\mathtt{F}}$	$c_{i,p,\mathtt{F}}$
F	$\left(\frac{s_{i,p,\mathtt{F}}}{s_{TOT,p,\mathtt{F}}}\right) \times B$	$\left(\frac{c_{i,p,\mathtt{T}}}{c_{TOT,p,\mathtt{T}}}\right) \times \left(R_{\mathtt{F}} \times \frac{1}{\tau}\right)$	$s_{i,p,\mathtt{T}}$	$c_{i,p,\mathtt{T}}$
ϕ	0	0	0	$c_{i,p,\mathtt{F}} + c_{i,p,\mathtt{T}}$

Submitter fees are used to fund bounties, while the certifier reward pools are initially left empty. Certifier reward pools are funded by unclaimed bounties and penalties. In the absence of reward pools, certification will be rare, which will lead to most bounties being unclaimed. This method will approach equilibrium where reward pools and bounties are balanced to provide reasonable incentives.

An additional consideration is the draining of reward pools. Each time a query is decided with voting outcome `T`, an amount $\frac{R_{\mathtt{T}}}{\tau}$ is deducted from $R_{\mathtt{T}}$. If the certification outcome is also `T`, the funds are used to pay out certifier rewards. Otherwise, the funds are added to $R_{\mathtt{F}}$. A symmetric case applies for `False` outcomes. Intuition lies in the fact that it ensures that the reward pools encourage certifiers to certify equal number of `True` and `False` queries. Thereby, discouraging reporters from voting with constant `T` or `F` values in order to maximize profit without considering the actual queries.

F. Base Protocol Analysis

To analyze the base protocol, we first analyze the probability of the voting procedure producing incorrect results (*i.e.*, where the outcome for query p with truth value t is $\neg t$). Next, we examine the minimum accuracy needed by reporters so that they remain profitable. Subsequently, we prove that a Nash equilibrium exists under a honest majority assumption. Finally, we argue that the certifier reward structure avoids a situation where players profitably report and certify everything with a constant `T` or `F`.

1) Voting Outcomes and Manipulation: This subsection determines the probability of a correct reporting outcome as a function of accuracy with the analysis of an adversary's probability to force incorrect outcomes. For simplicity, we assume all non-adversarial players are honest, have the same accuracy q, and always vote with s_{max} monetary units of stake. We can therefore treat the reporting process on a single query as a sequence of $\frac{D_r}{s_{max}}$ Bernoulli trials with probability q of success. The probability that the reporting outcome is correct is therefore:

$$P\left[B\left(\frac{D_r}{s_{max}}, q\right) > \frac{D_r}{2 \cdot s_{max}}\right]$$

where $B(n, p)$ denotes a binomial random variable. For example, if $D_r = 20$, $s_{max} = 1$, and $q = 0.8$, the probability of obtaining a correct reporting outcome is roughly 99.7%.

Now let us assume an adversary has n monetary units and seeks to force an incorrect outcome on a specific query. For simplicity, we assume n is a multiple of s_{max} and that the query list does not change during the attack. Each query can once again be modeled as a sequence of $\frac{D_r}{s_{max}}$ Bernoulli trials. Each trial is successful with probability:

$$p + (1-p)(1-q) = 1 - q + p \cdot q$$

where p is the probability that the vote belongs to the adversary.

If the adversary uses all n tokens to report, then $\frac{n}{s_{max}}$ votes belong to the adversary across all $|\mathbb{Q}|$ queries. Once all queries are decided, the probability that an arbitrarily chosen report belongs to the adversary is:

$$\frac{n}{s_{max}} \times \frac{s_{max}}{|\mathbb{Q}|} = \frac{n}{|\mathbb{Q}| \cdot D_r}$$

So the probability that an arbitrary vote to be incorrect is:

$$1 - q + \frac{n \cdot q}{|\mathbb{Q}| \cdot D_r} \tag{10}$$

Note, both $|\ \mathbb{Q}\ |$ and D_r appear in the denominator demonstrating that increasing these parameters renders system manipulation more difficult. The probability of an adversary changing the outcome of the query is shown below:

$$P\left[B\left(\frac{D_r}{s_{max}}, 1 - q + \frac{n \cdot q}{|\ \mathbb{Q}\ | \cdot D_r}\right) > \frac{D_r}{2 \cdot s_{max}}\right]$$

It can be seen that if the quantity in Eq. 10 is less than 0.5, there are parameter values that make it arbitrarily difficult for the adversary to force an incorrect result. Further analysis showed that, if reporters are 95% accurate and $D_r = 100 \cdot s_{max}$, then manipulation is effectively impossible, even by a powerful adversary controlling 25% of the reports.

2) Minimum Voting Accuracy: Accuracy of honest voters is crucial to the security of the base protocol. This subsection quantifies the minimum accuracy (*i.e.*, the probability that the player's private opinion β_p matches the truth value t of the query p) players need to achieve profitability. Since difficulty to earn a profit is expected to lower participation, it is critical to set the parameters carefully. This will benefit in achieving a reasonable trade-off between accuracy and participation.

For simplicity, we assume the parameters are set such that the probability of incorrect decisions is negligible. Consider player i and accuracy q_i. A report with stake of s_{max} on query p yields a profit at decision time for:

$$\left[\frac{q_i \cdot s_{max}}{max(s_{TOT,p,T}, s_{TOT,p,F})}\right] B > (1 - q_i) \cdot s_{max} \tag{11}$$

In other words, the expected share of the reporting rewards is greater than the expected penalties. Note that, there is no need to account for `Unknown` outcomes since reporters receive neither rewards nor penalties. At decision time, the denominator is clearly at least half of D_r, and at most D_r. Therefore:

$$\frac{1}{2} \cdot D_r \leq max(s_{TOT,p,T}, s_{TOT,p,F}) \leq D_r \tag{12}$$

It is clear from Eq.11 that the reporter is profitable for sufficiently high values of B. Towards the goal of computing an upper bound, we over-approximate the range in which a reporter is profitable using Eq.12 as follows.

$$\left(\frac{q_i \cdot s_{max}}{D_r}\right) B > (1 - q_i) \cdot s_{max}$$

Re-arranging yields the following upper bound on B:

$$B \leq \frac{(1 - q_i) \cdot D_r}{q_i} \tag{13}$$

By capping the bounty according to Eq.13, it is possible to enforce a minimum accuracy such that, below that threshold reporting becomes unprofitable. For instance, if 80% accuracy is desired and $D_r = 1000$, the bounty must be capped at 250 monetary units. If instead, 50% accuracy is desired the bounty must be capped at 1000. Of course, this analysis only lower-bounds the threshold; it neglects the voter's costs in terms of time to evaluate queries, computing power, and blockchain transaction fees.

3) Desirable Nash Equilibrium: This subsection demonstrates the existence of a desirable Nash equilibrium in which all players are honest. Any situation in which reporters and certifiers are in concert is a Nash equilibrium, as a player who votes against all the others will only stand to lose. However, we seek to show that such an equilibrium exists under the assumption that the quantity in Eq. 10 is less than 0.5. This assumes that honest reporters are sufficiently accurate and in such plurality a majority of votes are correct. From the analysis in Section III-F2, this assumption is sufficient to show that there exists an assignment to the parameters such that all queries have the correct voting outcome with overwhelming probability.

Playing honestly is a Nash equilibrium when every voting outcome is correct with only feasible strategies being considered. Rewards are only paid to players who agree with the voting outcome. Since every query p has the voting outcome t, if player i reports or certifies β_p (*i.e.*, honestly), she agrees with the voting outcome with probability q_i. Naturally, an honest player could perform better by switching to a "perfect" strategy in which they always vote correctly rather than honestly. Such a strategy is not feasible, since players do not know the underlying truth values. All private opinions are independent by assumption. A player i cannot develop complex strategies to report correctly with probability better than q_i. Even an adversary with perfect accuracy, who controls 100% of the certifying stake, is incentivized to play honestly (or not play at all, *e.g.*, in the case where $R_{\mathtt{T}} = 0$ and every query is `True`). Indeed, any other strategy results in the adversary losing all of their stake.

4) Query Bias and Reward Pools: The previous subsection demonstrates that playing honestly is an equilibrium under the assumption that an adversary does not control reporting. This may not hold if a dishonest strategy is easier and still profitable. In this subsection, we identify a candidate for such a strategy and argue that the reward structure combats it.

Imagine `True` queries are more common than `False` ones. Consider a *lazy reporter*—one who always reports `T`. Let $p = P(t = \mathtt{T})$ denote the probability that a random query is `True`. Assume that all reporting outcomes are correct. So, the lazy reporter agrees with the reporting outcome on every `True` query and disagrees on every `False` query. Intuitively, this lazy strategy seems viable. But, it is also necessary to consider certifier behavior since without certification rewards are not paid out. Certifier incentives are tied to the certifier reward pools, and their values fluctuate over time.

When a `True` query is decided, the $R_{\mathtt{T}}$ pool will shrink by $\frac{R_{\mathtt{T}}}{\tau}$. Over time the $R_{\mathtt{T}}$ pool will drain much faster than the $R_{\mathtt{F}}$ pool when $p > 0.5$. Indeed, the $R_{\mathtt{F}}$ pool may actually grow in this case. Informally, this process incentivizes certifiers to vote for queries that they believe are `False`, since the potential rewards are greater. At equilibrium, roughly equal amounts of `True` and `False` queries should carry certifications. Hence the lazy strategy will not be profitable. Note that certifiers are never incentivized to certify a `True` proposition as `False` in an attempt to acquire the $R_{\mathtt{F}}$ reward. In that case the certifier will disagree with reporters and be penalized.

IV. Simplified Reporter Oriented Protocol

Section III introduced one version of our decentralized oracle protocol. A major aspect is the role of certifiers. Including certifiers ensures a high confidence in the voting response for a submitter. It imposes truthful outcomes from reporters as well. As a high risk seeker, a certifier benefits from the amplified rewards. Although the protocol provides the required security guarantees for blockchain, it comes with inherent complexities during implementation. It is vital to ensure certifiers are incentivized to participate in the system. The reward pools for certifiers are dependent on the unclaimed bounties and penalties, lacking which may make certifiers lose interest in staking. Further, stringent penalties on certifiers might cause hesitation to join the voting process. In other words, from a submitter's point of view, they need to ensure a balance in the reward pools by posting equal number of `True` and `False` queries in order for the system to operate efficiently.

In this section, we suggest a simplified version of the original protocol with modifications in the gaming structure. This protocol convinces voters that truthful reporting is their best course of action without relying on certifiers.

A. Description

We present an abstract flow of the protocol in the Figure 2. This protocol is similar to the base protocol, wherein submitters post queries with binary responses along with a bond to the oracle. The reporters will respond based on their private opinion. For each submission, a submitter needs to post two queries that are *antithetic* (*i.e.,* queries that are assured to arrive at contradicting responses). For example consider the submission: `It is sunny` and `It is not sunny` as in Figure 3. The submitter can evaluate the correctness of the oracle response by observing opposing outcomes for these two queries. In this way, we induce high confidence in the voting process by avoiding the role of certifiers. Another consequence is the reduction in the complexity of the user interface for reporters and truthful reporters receive larger expected payoffs.

1) Submitting queries: To add to the list of active queries, in a single transaction a submitter provides:

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

The bond is returned if the answers to p and p' converge to different `True`/`False` outcomes after reporting process is done. Thus, the queries have to be designed to have opposite answers: this is easily done if the submitter constructs p' to be the converse of p (*e.g.,* "X won the elections this term" and "X did not win the elections this term"). Leaving the construction of the questions to the oracle will be an expensive and possibly error-prone operation, as we assume queries to be in natural language; it is a fair assumption that it will be done by the

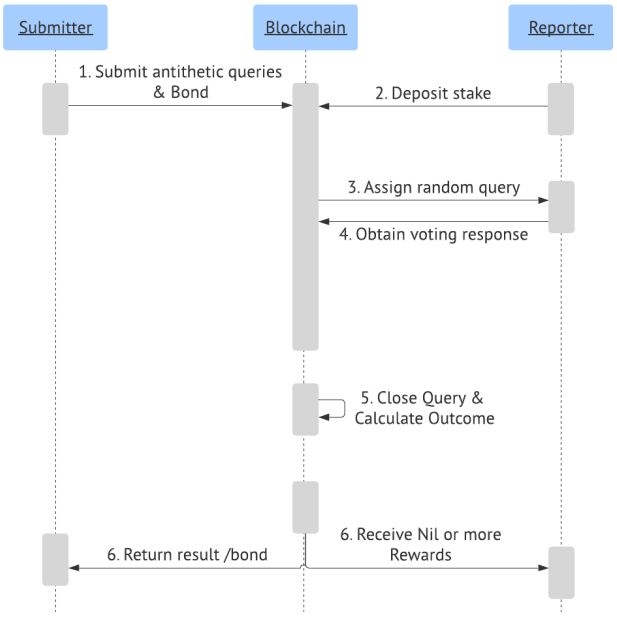


Fig. 2: Extension protocol submission & reporting process

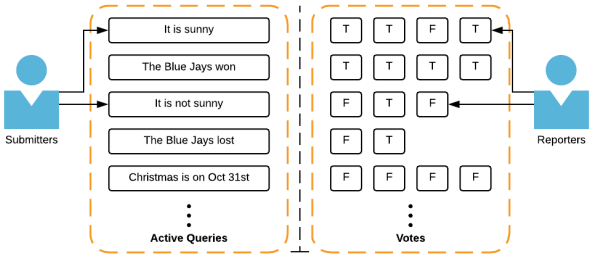


Fig. 3: Example of Extension protocol

submitter who would otherwise risk losing their bond. The bounty posted is used to reward the reporters, and the duration is to restrict the amount of time p and p' are available on the query list.

2) Submitting reports: A player $i \in N$ can report $PO_i(p)$ by engaging in a dialogue with the oracle:

1) The reporter posts a bond
2) The oracle selects a query uniformly at random and passes it to the reporter
3) The reporter returns a sealed vote to the oracle
4) Once the query closes after the duration time to submit votes has elapsed as set by the submitter, the reporter reveals their vote

At step 3, the reporter computes their $r_{i,p}$ based on their voting strategy $\sigma_i(p)$. Sealed voting is to prevent undesirable strategies or attack vectors.

After a predefined duration, a query is said to be closed. At this point the oracle tallies all the votes posted for that specific query. It will also compare and check if both the questions have converged to different answers. If the majority of reports converged to the same answers for both queries, then the submitter loses their entire bond and reporters would

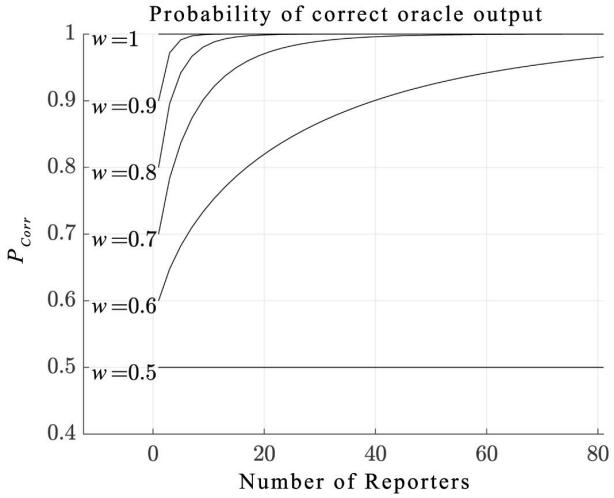


Fig. 4: Probability of correctness as a function of n and w

neither get a reward nor will they get penalized (*i.e.*, their bond is returned to them in full). If the reports converge to different responses, reporters are rewarded when they are in agreement or penalized for disagreement, and the submitter's bond is returned.

B. Modified Protocol Analysis

1) Correctness: Applications expect strict correctness as to the responses delivered by the oracle (in the context of Definition 1). Consider a query p with n honest reporters and where w is the probability that a randomly selected reporter agrees with $EPO(p)$. P_{Corr}, the probability that the oracle produces a correct output, is simply the probability that a majority of reporters agree with $EPO(p)$:

$$P_{Corr} = \mathcal{M}(n, EPO(p)) \tag{14}$$

We use $\mathcal{M}(n, z)$ to denote the probability that a majority out of n reporters vote z to the oracle. In the case of honest reporters,

$$\mathcal{M}(n, EPO(p)) = 1 - B(\lfloor \frac{n}{2} \rfloor, n, w)$$

where $B(k, n, p)$ denotes the cumulative binomial density function. $\mathcal{M}(n, EPO(p))$ is calculated differently depending on the configuration of reporters and voting strategies.

We evaluate the correctness of the oracle by comparing the probability of correctness with the number of reporters depicted in Figure 4. Assuming all reporters are honest, if a low number of reporters is expected then only queries with widely accepted answers are likely to be decided correctly. However, even if a query is highly contentious (with w near but not equal to 0.5), the oracle will eventually converge on the $EPO(p)$ with high probability provided there are enough reporters.

2) Expected Rewards for Honest Voting: As mentioned earlier, a reporter is rewarded only when they are part of the majority. They are penalized if they are in the minority. In cases of a tie, neither rewards nor penalties are applied.

TABLE VI: Summary of β values for pure strategies in response to honest reporting

Name	Strategy Function	β
Honesty	$\sigma_i(PO_i) = PO_i$	> 0.5, Lemma 2
Lying	$\sigma_i(PO_i) = \neg PO_i$	< 0.5, Lemma 2
Always `True`	$\sigma_i(PO_i) =$ `True`	0.5, Lemma 3
Always `False`	$\sigma_i(PO_i) =$ `False`	0.5, Lemma 3

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

$$P_{Maj} = (\beta_{ip})\mathcal{M}(n-1, EPO(p)) + (1-\beta_{ip})\mathcal{M}(n-1, \neg EPO(p)) \quad (15)$$

P_{Tie}, the probability that a tie occurred, is the probability that exactly $\frac{n}{2}$ reporters voted according to the $EPO(p)$ and is calculated differently depending on the configuration of reporters. Finally, P_{Min}, the probability that reporter i is in the minority, is simply that probability that they were not in the majority and that there was no tie:

$$P_{Min} = 1 - P_{Maj} - P_{Tie} \quad (16)$$

For a voter to be rewarded or penalized, the oracle outputs denoted by u and u' for a query with questions p and p' should converge to different answers.

$$\mathrm{P}(u \neq u') = \mathcal{M}(n, \texttt{True})\mathcal{M}(n', \texttt{False}) + \mathcal{M}(n, \texttt{False})\mathcal{M}(n', \texttt{True}) \quad (17)$$

Combining both equations 15 and 17, we can compute the probability that i receives a reward/penalty:

$$P_{Rew} = P_{Maj} \cdot P(u \neq u')$$

Similarly,

$$P_{Pen} = P_{Min} \cdot P(u \neq u')$$

Finally, if the reward size is g and the penalty size is h, then a reporter's expected payoffs are $gP_{Rew} - hP_{Pen}$.

3) Expected Rewards for Lazy Reporting: The previous section shows that honest reporting provides for positive expected payoffs. The main challenge for any oracle protocol is to disincentivize lazy reporting. In the lazy case, it is clear to see $P(u \neq u') \approx 0$. This forces expected payoffs to also be zero, which causes lazy reporting to be less efficient than honest reporting.

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

Lemma 2. *In a scenario where all other users are honest, honest reporting has an expected $\beta_p > 0.5$ for an arbitrary reporter i with no information about the PO of other reporters. Additionally, lying has an expected $\beta_p < 0.5$ for such a reporter.*

Proof. First, note that by definition of $EPO(p)$, it must always be the case that $P(\Gamma = EPO(p)) \geq 0.5$ when all reporters are honest. The only circumstance in which $P(\Gamma = EPO(p)) = 0.5$ is an extreme case where $P(\Gamma = \mathtt{T}) = 0.5$ (*i.e.,* the EPO is unknown). Assuming most submitters ask questions which have an answer, it is reasonable to conclude that $P(\Gamma = EPO(p)) > 0.5$. This means that from the perspective of a specific reporter i with incomplete information about the PO of other reporters, $P(PO_i = EPO(p)) > 0.5$, and so $\beta_p > 0.5$ if i reports honestly. Additionally, if an honest reporter has probability z of reporting the $EPO(p)$, then a lying reporter has a probability $1 - z$ of reporting the $EPO(p)$ and thus $\beta_p < 0.5$. □

Lemma 3. *In a scenario where all other users are honest, lazy reporting has an expected β_p of 0.5 for an arbitrary reporter i with incomplete information.*

Proof. If submitters act honestly, then we can conclude they are creating an equal number of `True` and `False` queries, implying $P(EPO(p) = \mathtt{T}) = 0.5$. Thus, on any particular query, a reporter always votes `True` *(or* `False`*)* has a probability of 0.5 of reporting the $EPO(p)$, which implies $\beta_p = 0.5$. □

Theorem 4. *Honest reporting is a Nash equilibrium.*

Proof. By Lemmas 2 and 3, the pure strategy of honest reporting has a strictly better expected β_p than the pure strategies of lying and lazy reporting. This means that no mixed strategy can achieve a greater expected β_p than honest reporting. Since expected payoffs increase monotonically with increasing β_p, honest reporting is a strictly best response in a scenario where all other users are honest. □

C. Discussion

In this subsection, we discuss the trade-offs with both versions of the protocol. In the base protocol, submitters do not need any prior knowledge of the responses for queries as there is no requirement for the questions to be antithetic. The only restriction is to provide questions related to binary markets. This constraint also becomes a significant factor when considering the rate at which queries get posted to the oracle. There is only a limited number of queries a submitter can handle at a time. Also, in the future, with the simplified query structure, the submitter role can be automated using natural language processing or artificial intelligence tools.

On the other hand, the simplified version of the protocol induces inherent balance in the queries without which there is a possibility that one of the certifier reward pools (either R_{T} or R_{F}) can get disproportionately large, possibly incentivizing certifiers to vote dishonestly on future queries. Removing the role of certifiers makes it a simple interface for reporters to participate in the process. The proposed protocol is most effective for scenarios in which voters are able to answer any proposition, and sufficiently many voters are available to answer. As indicated by Figure 4, the ideal scenario would have at least 10-20 reporters per proposition in order to achieve strong incentives for voter honesty and a high probability

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

V. Design and Implementation

This section outlines a prototype design of the proposed protocol. We selected Ethereum [10] as the underlying blockchain for our prototype. By building on top of Ethereum, integrating key decentralized technologies, the application becomes a user-friendly ecosystem that increases the adoption of blockchain technology as a whole. The main reasons for selecting the Ethereum as a code base are its flexibility, its open-source nature and the overall availability of client API implementations.

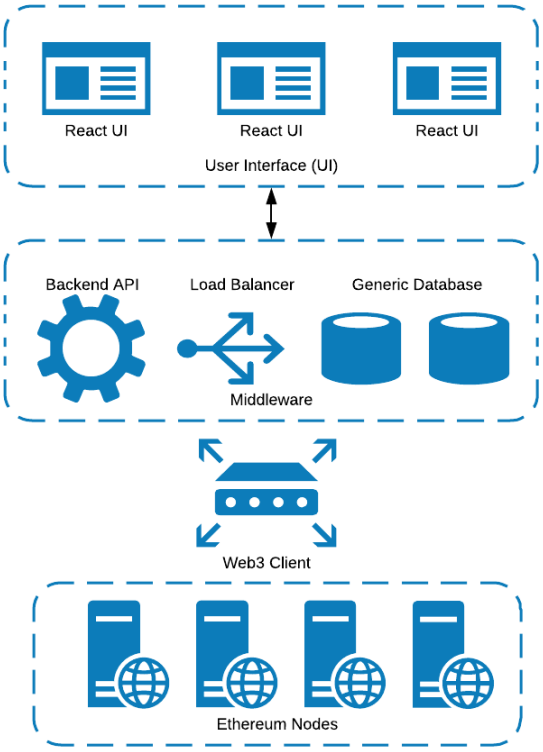


Fig. 5: High-level Architecture

A. High-level Architecture

We present the basic components for the prototype in Figure 5. Our design introduces three main components: a web-based user interface through which users like submitters/reporters interact with the oracle, a middleware consisting of an off-chain database, load balancer and back-end API, and an Ethereum client. These components have been executed on servers, combining to create a coherent distributed system. Any provider back-end and user interface implementation can participate in the system through our smart contracts.

1) Web-based user interface: Frictionless user interaction was a big focus for our development efforts. The user interface portal facilitates the submission of queries and viewing the oracle outcome. In addition, a notification service broadcasts events to appropriate users. The notification service is used to alert submitters after their query moves to a completed state and reporters are notified once the oracle is ready to distribute rewards.

2) Middleware: We constructed multiple utilities, bundled as a middleware, to facilitate the system's operation. Middleware abstracts the communications with the blockchain and exports a function-call API. The user interfaces can thus avoid the hurdles of working directly with the blockchain. One such hurdle is verifying that each sent transaction is accepted with high confidence by the network. The middleware handles uncertainties of when transactions are mined and deals with cases when they are discarded. Middleware interacts with an Ethereum client to excercise low-level formatting and parsing of the Ethereum protocol.

3) Ethereum Client: This component implements the full functionality of joining and participating in the Ethereum blockchain network. This handles a broad set of tasks, such as connecting to the peer-to-peer network, encoding and sending transactions and keeping a verified local copy of the blockchain. For our prototype implementation we use `Geth` and `Web3` client.

We implement a service to locate all of the relevant contracts using address lookup. This service runs continuously within the client to monitor real-time changes to the smart contracts. In the event of an update, the service signals the middleware to issue a user notification and, if necessary, sync the local database.

B. Data-flow and Smart Contract Structure

Indicated in Figure 6 is the direction of data flow in our system architecture. Data is entered into the blockchain starting from User Interface (UI) on the left. As shown in the diagram, data is also shared back to the users from blockchain. Middleware which includes the back-end library and the off-chain database supports in persisting the data transfer to and from blockchain.

To implement all the functionalities of our protocol, the system is structured on the blockchain by implementing four contracts. In Figure 6 we illustrate the contract structures and relationships. We coded our smart contracts in Solidity using Remix, a website that has a compiler to test contract functionality.

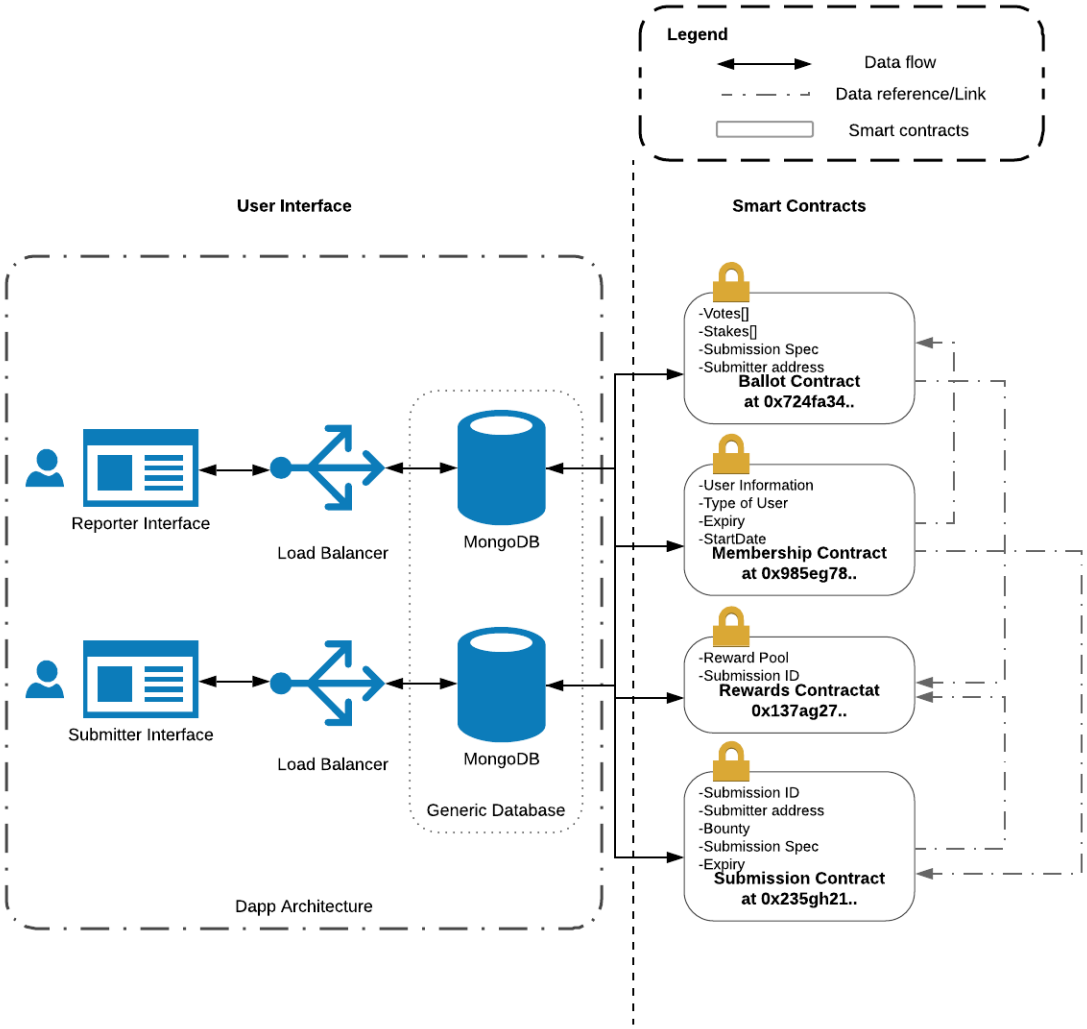


Fig. 6: Data-flow and Smart Contract structure

1) Membership Contract (MC): This contract maps the participant identification strings to their Ethereum address identity (equivalent to a public key). We intentionally use strings rather than the cryptographic public key identities directly, allowing the use of an already-existing form of ID. Instructions coded into the contract can regulate registering new identities or changing the mapping of existing ones. MC also identifies users by two types: submitters and reporters.

2) Submissions Contract (SC): This contract holds a list of references to the submissions, representing all the queries posted by a particular submitter. It also persists information regarding the bounty and duration for a query. SC implements functionality to enable user notifications. Each submission stores a status variable, indicating whether the submission is pending or completed. SC refers to MC for submitter identification.

3) Ballot Contract (BC): BC is one of the crucial contracts in our prototype. It holds the functionality required to arrive at an outcome. Based on the voting and staking information available, this contract sets the status of a query in SC. Reporters submit their response via BC. BC interacts with MC in order to obtain voters and submitter information.

4) Rewards Contract (RC): We isolated rewards and penalty functions into a separate contract to enable easy upgrades or fixing of any future issues. As soon as BC updates the status of a submission, methods in RC are used to distribute rewards based on the outcome of the voting. RC maintains the Reward Pool which is the total amount of stake collected from the reporters.

C. Discussion

An implementation of the detailed architecture can be executed on a local PC or even a mobile phone. The local databases of these hardware tools can be one of many lightweight database implementations. The databases can function merely as cache storage of the user data. Missing data can be retrieved from the network at any time by following the Submissions Contract.

The prototype discussed in this section was built on a local PC environment. The user interface was made using the `React` framework and `Sass` styling, while we used `Sketch` for

Fig. 7: Tools used in Prototype Implementation

creating the wireframes. The whole middleware component of our architecture was implemented using `NodeJs`. As to the Ethereum interactions, we used the `Truffle` tool suite extensively. Smart contracts were coded using `RemixIDE` and we used built-in verification enabled in the same tool for unit testing. Further, `Ganache` and `Heroku` were utilized to deploy the application. Lastly, for project management and source control we used `Asana` and `github`. A summary of tools used can be found in Figure 7.

VI. Comparison with Prior Art

Now, we discuss existing oracle solutions to analogize the requirement for our decentralized oracle. `Oraclize` [15] retrieves data from any data source and publishes it to a blockchain smart contract along with a verifiable cryptographic proof. But the data is provided to the blockchain by a centralized server that handles requests for off-chain information. `TLSnotary` [16] and `TownCrier` [17] attest to facts on websites accessed using `Hyper Text Transfer Protocol Secure` (HTTPS) protocol cryptographically. The assumption here is that any information exchanged over `TLS` (Transport Layer Security) is checkable, and it doesn't guarantee that all users will see the same information at all times. Further, they both use the `Intel SGX` (Secure Guard Extensions) [18] hardware in order to protect the attestations against malicious actors, *e.g.,* against malware running on the attesting system. This system is highly effective, so long as users trust that the underlying hardware does not contain backdoors or exploits [19, 20]. `ChainLink` [21] aims to provide a cross-chain portal to internet-available information, *i.e.,* data available on websites, through their centralized system. All three protocols, `Oraclize`, `TownCrier` and `ChainLink` violate the permissionless property of a decentralized oracle that is a desired property to which the proposed system described here adheres.

`Augur` [22], a crowdsourced prediction market place, allows internal token holders to predict or dispute the outcomes in a multi-phase procedure. Users of this platform are required to use native platform tokens in order to report predictions to the oracle. Members do not have the flexibility of moving in and out of the markets at their will, which would restrict the usability of the ecosystem. For example, a user chosen as a Designated Reporter cannot drop out of voting on their choice, upon which they will be penalized. Also, in case of a fork, the burden rests with users to make decisions as to which branch of `Augur` they would prefer to move all of their earned currency to. Both of these issues hinder the usability of their decentralized oracle.

An important feature of our proposed extension protocol is a re-balancing of incentives. A submitter receives a penalty for creating an imbalance of $EPO(p) = \texttt{T}$ and $EPO(p) = \texttt{F}$ queries. Additionally, we are able to reduce the size of penalties without sacrificing incentive compatibility. The simplicity of the new protocol is also advantageous. It is not enough that a protocol guarantees optimal rewards for honesty; its users must be convinced of this fact. Otherwise, they may act according to an incorrect belief that a dishonest strategy is optimal. Furthermore, a simpler protocol exhibits a simpler formal analysis. While making fewer assumptions, stronger guarantees are proven, and extensions or critical adjustments to system parameters are easily evaluated.

The proposed protocols are most effective for scenarios in which voters are able to answer any query, and sufficiently many voters are available to answer. As indicated by Figure 4, the ideal scenario would have at least 10-20 voters per query in order to achieve strong incentives for reporter honesty and a high probability of correctness, depending on the level of agreement among the reporters. It is also important to note that our oracle model involves several idealizations in order to lend itself to a more tractable analysis. In particular, by assuming that a reporter's strategy i depends only on PO_i, we implicitly disregard strategies that make use of information contained in the query itself. For instance, a reporter may try to guess which of the queries p and p' is the positive statement and which is the negation, and always vote `True` on the positive and `False` on the negation. If voters are able to guess correctly with high probability then this strategy can be a Nash equilibrium with large payoffs. However, in many scenarios it is possible to construct query statements in such a way that guessing accurately is extremely difficult.

VII. Additional Implementation Details

In this section, we provide some additional improvements that could be included in an implementation of the decentralized oracle.

A. Voting Pools and Threshold Signatures

Both protocols proposed require a large number of reporters to submit their votes in order to achieve correctness up to

95% as shown in Figure 4. This would in turn mean that there will be huge increase in the number of transactions to submit votes. In general, most of the public blockchain platforms have associated transaction fees [23]. Submitters would have to consider this additional cost when posting the bounty amount and make sure it is large enough to offset the fees a reporter must pay. Below, we describe a method to reduce costly transactions while not significantly lowering the oracle's correctness.

A voting pool is like a community consisting of a leader and a number of subscribers. The subscribers can answer queries posted by the leader, who is the one directly communicating with the decentralized oracle. Once answers are submitted, the leader has the control on which answers are to be used and can decide how to disburse the rewards obtained.

1) Perks of being a Leader: A leader can instill several techniques that could help evaluate and improve the correctness of their pool. For example, a leader could perform some initial tests on their subscribers with known queries to level their expertise on the topic or could send the same query to a subscriber multiple times to judge their consistency. With these kinds of mechanisms in place, the leader could increase the overall correctness of their pool which would grow their rewards and yield higher expected payoffs to its subscribers.

2) Subscriber Advantages: As an individual user on a public blockchain, the responsibility of securely maintaining an account rests on the users themselves. Instead, subscribing to a pool service would present a user with a simplified interface. This will also alleviate the requirement for the reporter to post a bond and removes the need to pay transaction fees. Similar to a mining pool [24], participating through a voting pool can lower the risk of high volatility in rewards. In some cases, an honest reporter can lose several rounds at a stretch and if they run out of funds, that would no longer enable them to post the necessary bond in order to continue participating.

3) Threshold signatures: In the above mentioned "basic" voting pool mechanism, there are two specific problems that needs to be highlighted. First, as mentioned above the leader has complete control on which answer they can propose to the blockchain. This would mean that subscribers are to place unconditional trust on the leader and expect positive payoffs. Whereas, the leader can even utilize the staked deposits for their own benefits and show negative returns to avoid payoffs to the subscribers. Second, each query will receive a single vote from the voting pool irrespective of the number of subscribers that are submitting their votes. Other than using the stake amount there should be a way to prove the count of voters on the blockchain without increasing the transactions that are to be committed.

One way to secure the voting pools process is by using distributed Schnorr Signatures as this scheme has been proved secure and unforgeable in [25]. Using this scheme, a secret key is shared between all of the subscribers. To produce a valid signature for the transaction at least t subscribers need to sign using the shared secret key. This also secures the system from an attack by $t-1$ subscribers and the leader by themselves cannot sign the transaction as well. On-chain, the signature can be verified to attest for the count of subscribers who have voted for the query. In [26], a similar Schnorr-like signature scheme is used with reduced gas cost around 15k, including the input parameters, which is feasible to verify in Ethereum and its programming language Solidity.

4) Advantages for the Decentralized Oracle: First, the number of transactions is significantly reduced, which in turn lowers the cost to submitters. Second, though fewer participants are interacting directly with the oracle, each of them potentially has a much higher β_{ip} parameter. As shown in Figure 4 the oracle needs only a small number of reporters provided β is large enough.

B. Reputation and Adjudication systems

In our present model, we do not consider any feedback from the submitters on how satisfied they are and also, currently there is no way for the submitters to dispute the outcome of the voting process. Voters on the other hand are not incentivized to participate in such a feedback/dispute process once the payoffs have been distributed. Introducing a reputation score for all the participants in the system i.e., both voters and submitters would definitely add value to the decentralized voting process.

1) Bootstrapping Calculation of Reputation Score: Over the past years, research in reputation mechanisms has identified various models [27] that can be re-used for our decentralized oracle. We could employ a scoring where in any new user in the system starts with an initial score for reputation and as they participate in multiple instances of voting their reputation would increase or decrease based on the voting outcomes. Also, we can do Bayesian update on the prior reputation score to calculate a new score after each round of voting is completed. Additional incentives in terms of payoffs could be provided to voters depending on their reputation score.

2) Dispute Resolution: In a case where a submitter is not satisfied with the response from the oracle or he is able to provide with some proof that the outcome received may not be correct, the submitter can request for a re-evaluation of the voting process. This can be done in multiple rounds until a concrete resolution has been arrived which is agreed upon by both submitters and voters. A similar methodology has been implemented in Augur [22].

We do not cover the detailed analysis on reputation mechanism as it is out of scope of this paper and can be explored as future work for the proposed decentralized oracle.

C. Sealed Votes

To ensure the security of the oracle it is necessary for votes to remain secret until propositions are closed. One reason is to prevent voters from simply tallying existing votes and choosing to agree with the current majority (instead of honestly reporting their private beliefs). One popular method for sealed voting is a cryptographic commitment scheme [28]. When placing a vote, a reporter sends $Hash(v, r)$ where v is their vote and r is a privately chosen random number. Once a proposition is closed, the reporter reveals v and r, allowing the oracle (and any other participants) to verify that the reporter committed to this vote. Unmodified, this scheme is not enough for the set purposes. An attacker could replay a vote, a reporter could choose to never reveal their vote, or a reporter could publicly announce their vote before the query closes. We propose the following techniques for dealing with these issues:

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

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

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

D. Random Numbers

Introducing randomization for reporters when selecting the queries has the effect of evenly distributing the reports over all queries, and makes it more costly for a reporter (or group of reporters) to collude and force the output for a single query.

Substantial study has been done on randomization in decentralized blockchain platforms, as all users must agree on the exact same random number (implying that its selection must be deterministic) and yet they must not be able to predict or manipulate it. A popular method for accomplishing this uses a `RANDAO` [13]. This technique is executed in two non-overlapping phases: the committing phase and the revealing phase. In the committing phase, users send hashes of privately-generated random numbers. In the revealing phase, the users reveal their private numbers, which are combined into a final result. This ensures that users cannot predict the output during the commit phase (though they can influence it) and that they cannot influence the output during the reveal phase (though they can predict it).

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

Note that an oracle could efficiently use both sealed votes and a `RANDAO` by combining the two techniques. The rew values used for sealed votes can be re-purposed as the basis for random number generation.

VIII. Applications

Imparting the ability to confirm external facts with the use of a decentralized oracle into smart contracts will give rise to a considerable increase in practical applications for blockchain systems. This section gives an overview of candidate use cases based on the proposed protocols.

A. Machine Learning and Data Annotation

Traditional prediction marketplaces survive on annotation and labeling of huge amounts of data [29]. Various incentive structures exist for current crowd-sourcing platforms, which expend efforts from individuals to perform human intelligence tasks [30, 31]. Lack of reliable compensation mechanisms has affected the quality of labeling in this industry [32]. There is no way to determine the correctness of the information provided by these platforms. A decentralized oracle that can incentivize honest workers and enrich trusted, reliable data labeling will potentially reduce the costs and improve quality of these existing solutions.

B. Data Availability

A core issue for decentralized applications using off-chain resources [33] is the data availability problem [34]. Chances of intermittent downtime of these systems (*e.g.*, an off-chain platform is unavailable) affect the essence of decentralization. Our proposed protocol can also be used as a data availability oracle for all such applications.

C. Adjudication Mechanisms

Negotiations between parties that require an adjudication mechanism can instead use a decentralized oracle. In this case, the oracle will essentially serve as a public jury. Decentralized applications that deal with real-world resources, such as legal agreements, transfers of assets, griefing behavior in on-line games, token-curated registries, etc., can make use of this system.

IX. Conclusion

This work proposes two novel crowdsourced protocols for a decentralized oracle. Depending on the version of the protocol, submitters post queries (antithetic queries if the simplified protocol is used) into the system, while reporters (or certifiers) answer the queries by placing a certain amount of monetary stake. Depending on the reporting outcomes calculated by the oracle, rewards and penalties for all users are assessed. We analyze the game theoretical structure, oracle correctness, and reporter payoffs in an honest voting scenario and show the existence of an honest Nash equilibrium for both versions of protocol. The base protocol introduces a high confidence voting process which involves both reporters and certifiers, while the other version provides a simplified version of the base protocol in order to avoid the complexities involved in implementing certifiers. Both protocols provide the same level of security guarantees required by blockchains. We suggest that the choice of the protocol should be made based on the required complexity and also the number of participants in

the system. In case of large pool of voters, the simplified protocol would be efficient. The base protocol could be used when there are enough risk-seeking participants. We present a detailed architecture that could be implemented on Ethereum blockchain. Additionally, we specify a number of features which can increase the cost to force an outcome, reduce transaction costs and provide pseudo-random number generation, and allow for secret voting on a public decentralized blockchain platform. In the future, we plan to test the prototype in a real-world environment. This would allow for empirical analysis of performance and costs so to confirm the theoretical analysis presented in this paper. Additionally, we plan to adapt the architecture so that varied blockchain networks can use the application. We further plan to extend the smart contracts to include a factory design pattern in order to incorporate upgradability.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] S. Malik, S. S. Kanhere, and R. Jurdak, "Productchain: Scalable blockchain framework to support provenance in supply chains," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Nov. 2018, pp. 1–10. DOI: 10.1109/NCA.2018.8548322.

[3] A. E. C. Mondragon, C. E. C. Mondragon, and E. S. Coronado, "Exploring the applicability of blockchain technology to enhance manufacturing supply chains in the composite materials industry," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, Apr. 2018, pp. 1300–1303. DOI: 10.1109/ICASI.2018.8394531.

[4] M. Raikwar, S. Mazumdar, S. Ruj, S. S. Gupta, A. Chattopadhyay, and K. Lam, "A blockchain framework for insurance processes," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Feb. 2018, pp. 1–4. DOI: 10.1109/NTMS.2018.8328731.

[5] M. Z. A. Bhuiyan, A. Zaman, T. Wang, G. Wang, H. Tao, and M. M. Hassan, "Blockchain and big data to transform the healthcare," in *Proceedings of the International Conference on Data Processing and Applications*, ser. ICDPA 2018, Guangdong, China: ACM, 2018, pp. 62–68, ISBN: 978-1-4503-6418-8. DOI: 10.1145/3224207.3224220. [Online]. Available: http://doi.acm.org.myaccess.library.utoronto.ca/10.1145/3224207.3224220.

[6] I. Eyal, "Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities," *Computer*, vol. 50, no. 9, pp. 38–49, 2017, ISSN: 0018-9162. DOI: 10.1109/MC.2017.3571042.

[7] "Blockchains: How they work and why they'll change the world," 2017. [Online]. Available: https://spectrum.ieee.org/computing/networks/blockchains-how-they-work-and-why-theyll-change-the-world.

[8] "Why many smart contract use cases are simply impossible," 2016. [Online]. Available: https://www.coindesk.com/three-smart-contract-misconceptions.

[9] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS â15, Denver, Colorado, USA: Association for Computing Machinery, 2015, 706â719, ISBN: 9781450338325. DOI: 10.1145/2810103.2813659. [Online]. Available: https://doi.org/10.1145/2810103.2813659.

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

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

[12] R. Kraut, "Plato," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Fall 2017, Metaphysics Research Lab, Stanford University, 2017. [Online]. Available: https://plato.stanford.edu/entries/plato/#PlaCenDoc.

[13] *Randao*, https://github.com/randao/randao, 2016.

[14] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Symposium on Foundations of Computer Science*, 1999, pp. 120–130.

[15] Oraclize.it, http://www.oraclize.it, Accessed: 2018-02-01.

[16] *TLSnotary – a mechanism for independently audited https sessions*, https://tlsnotary.org/TLSNotary.pdf, 2014.

[17] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, *Town crier: An authenticated data feed for smart contracts*, Cryptology ePrint Archive, Report 2016/168, https://eprint.iacr.org/2016/168, 2016.

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

[19] Intel, *Intel active management technology, intel small business technology, and intel standard manageability escalation of privilege*, https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00075&languageid=en-fr, Accessed: 2018-02-01, 2017.

[20] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the intel sgx kingdom with transient out-of-order execution," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18, Baltimore, MD, USA: USENIX Association, 2018, pp. 991–1008, ISBN: 978-1-931971-46-1. [Online]. Available: http://dl.acm.org/citation.cfm?id=3277203.3277277.

[21] S. Ellis, A. Juels, and S. Nazarov, *Chainlink a decentralized oracle network*, https://link.smartcontract.com/whitepaper, 2017.

[22] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, *Augur: A decentralized oracle and prediction market platform*, http://www.augur.net/whitepaper.pdf, 2018.

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

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

[25] D. R. Stinson and R. Strobl, "Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates," in *Proceedings of the 6th Australasian Conference on Information Security and Privacy*, ser. ACISP '01, London, UK, UK: Springer-Verlag, 2001, pp. 417–434, ISBN: 3-540-42300-1. [Online]. Available: http://dl.acm.org.myaccess.library.utoronto.ca/citation.cfm?id=646038.678297.

[26] Chainlink. (2019). Threshold signatures in chainlink, [Online]. Available: https://blog.chain.link/threshold-signatures-in-chainlink/.

[27] L. M. B. Cabral. (2005). The economics of trust and reputation: A primer, [Online]. Available: http://pages.stern.nyu.edu/~lcabral/reputation/Reputation_June05.pdf.

[28] O. Goldreich, *Foundations of Cryptography: Volume 1*. New York, NY, USA: Cambridge University Press, 2006, ISBN: 0521035368.

[29] K. Burke, *Humans help train their robot replacements*, http://www.autonews.com/article/20170827/OEM06/170829822/data - annotation - self - driving, Accessed: 2018-02-01, 2017.

[30] *Crowdflower*, https://www.crowdflower.com.

[31] *Amazon mechanical turk*, https://www.mturk.com.

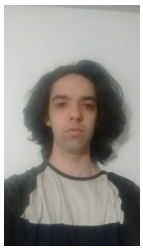
[32] F. A. Schmidt, "The good, the bad and the ugly: Why crowdsourcing needs ethics," in *Cloud and Green Computing*, 2013, pp. 531–535. DOI: 10.1109/CGC.2013.89.

[33] J. Teutsch and C. Reitwießner, *A scalable verification solution for blockchains*, https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf, 2017.

[34] V. Buterin, *A note on data availability and erasure coding*, https://github.com/ethereum/research/wiki/A - note - on - data - availability - and - erasure - coding, Accessed: 2018-02-01, 2017.

Keerthi Nelaturu received a B.E. in with a specialization in Computer Science Engineering from the Osmania University in 2008. She then received a M.Sc. in Computer Science Engineering from the University of Ottawa in 2015. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. Her research interests include decentralized blockchain oracles and smart contract verification/synthesis.

John Adler received a B.A.Sc. in Engineering Science with a specialization in Electrical and Computer Engineering from the University of Toronto in 2013. He then received a M.A.Sc. in Electrical and Computer Engineering from the University of Toronto in 2017. His research interests include decentralized blockchain oracles and blockchain scalability. He is currently with LazyLedger and Fuel Labs.

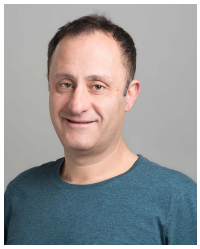


Marco Merlini received a B.Sc. in Electrical Engineering from the University of New Brunswick in 2018. He is currently pursuing an M.A.Sc. degree at the University of Toronto. His research focus is on debugging distributed FPGA applications.

Ryan Berryhill received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2014, and the M.A.Sc. degree in computer engineering from the University of Toronto, Toronto, ON, USA, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His current research interests include inductive formal verification and automated formal debugging of digital systems.

Neil Veira received a B.A.Sc. degree in Engineering Science with a major in Electrical and Computer Engineering from the University of Toronto in 2017. He then received a M.A.Sc. in Electrical and Computer Engineering from the University of Toronto in 2019. His research focus has been on applications of machine learning and data science techniques to hardware verification algorithms. He is currently with SoundHound Inc. He is a member of IEEE.

Zissis Poulos received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 2011, an M.A.Sc degree in Electrical and Computer Engineering from the University of Toronto in 2014, and a Ph.D. degree in Electrical and Computer Engineering from the University of Toronto in 2018. He is currently a Postdoctoral Fellow at Rotman School of Management at the University of Toronto. His research interests include applied machine learning in finance, deep learning acceleration, statistical diagnosis and debugging of VLSI systems, modeling and optimization of information/influence diffusion in social graphs, and distributed ledger technologies. He is a member of IEEE and ACM.



Andreas Veneris received a Diploma in Computer Engineering and Informatics from the University of Patras in 1991, an M.S. degree in Computer Science from the University of Southern California, Los Angeles in 1992 and a Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign in 1998. In 1998 he was a visiting faculty at the University of Illinois until 1999 when he joined the Department of Electrical and Computer Engineering and the Department of Computer Science at the University of Toronto where today he is a Professor. Since 2018 he is a Connaught Scholar for his contributions to blockchain technology. His research interests include CAD for debugging, verification, synthesis and test of digital circuits/systems, crypto-economics, decentralized blockchain technology, and combinatorics. He has received several teaching awards, a best paper award and a Ten Year Best Paper Retrospective Award. He is the author of one book and he holds several patents. He is a member of IEEE, ACM, AMS, AAAS, Technical Chamber of Greece, Professionals Engineers of Ontario and The Planetary Society