A Decentralized Truth Discovery Approach to the Blockchain Oracle Problem

Yang Xiao*, Ning Zhang†, Wenjing Lou‡, Y. Thomas Hou‡
*University of Kentucky, Lexington, KY, USA
†Washington University in St. Louis, St. Louis, MO, USA
‡Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

***Abstract*—When a blockchain application runs on data from the real world, it relies on an oracle mechanism that transports data from external sources to the blockchain. The blockchain oracle problem arises around the need to procure trustworthy data from external sources. Previous works have addressed data authenticity/integrity by building a secure channel between blockchain and external sources while employing a decentralized oracle network to avoid a single point of failure. However, the truthful data challenge, which emerges when legitimate external sources submit fraudulent or deceitful data, remains unsolved. In this paper, we introduce a new decentralized truth-discovering oracle architecture called DECENTRUTH to address the truthful data challenge using a data-centric approach. DECENTRUTH aims to elevate the "truthfulness" of external data input by enabling decentralized oracle nodes to discover and reach consensus on truthful values of common data objects from multi-sourced inputs in an off-chain manner. It harmonizes techniques in both the data plane and consensus plane—truth discovery (TD) and asynchronous BFT consensus—and enables nodes to finalize the same estimated truths on data objects with high accuracy, amid the harsh asynchronous network condition and presence of Byzantine sources and nodes. We implemented DECENTRUTH and evaluated its performance in a simulated oracle service scenario. The results demonstrate significantly higher Byzantine resilience and long-term data feed accuracy of DECENTRUTH, compared to existing median-based aggregation methods.**

***Index Terms*—Blockchain oracle problem, trustworthy data feed, decentralized consensus**

I. INTRODUCTION

Blockchain technology is known for enabling distrustful entities to exchange value and curate financial ledgers without involving a central authority. Smart contract, one prominent blockchain application native to many known platforms (e.g., Ethereum [1], Polkadot [2], EOSIO [3]), has redefined how independent parties reach contractual agreement and execute business logic without a trusted intermediary, culminating in a new world of decentralized applications (DApps). Meanwhile, blockchain applications make the most compelling case when they operate on information about the real world. An investment smart contract needs to fetch asset prices as well as the platform token's exchange rate from external markets; an insurance smart contract needs to collect the usage and environment data of the insured objects to determine premiums and payouts [4]; a supply chain DApp feeds on the merchandise location data for automating ownership transfer and on-delivery payment [5]. In many cases, the external world also collects information from the blockchain such as what types of external data are most inquired [6]. It is evident that the ability of blockchain applications to communicate with the external world is crucial to unleashing their true economic potential.

The Oracle Problem in Blockchain. In current blockchain systems, the mechanism to fetch data from external sources is known as *data feed*, or more commonly *oracle*, when the data feed is instantiated as a standalone service. Data from external sources, however, are foreign to the blockchain system and outside the jurisdiction of built-in consistency measures. They are in contrast to the transaction data that are inherently generated within the blockchain and verifiable by the native consensus. The lack of trustworthy mechanism for blockchains to communicate with the external world—to secure high-fidelity external data in particular—is known as the *oracle problem*. The oracle problem has become a major hindrance to blockchain's wider utility in the real-world economy [7].

Existing work has tackled the blockchain oracle problem from the source authenticity (i.e., data are from legitimate sources) and data integrity (i.e., no tampering during transportation) perspectives [8], [9], [10], [11]. They generally involve enabling authenticated communication between the blockchain and an external data source, following either the third-party model that provides oracle service as an independent entity [8], [9] or the first-party model that allows data sources to act as oracles directly [10], [11]. They generally assume that the oracle service is always reliable and that external sources can provide definitive and truthful input on a given data object. However, in a decentralized blockchain system, it can be difficult to convince users to trust one oracle service for providing a specific data feed as it is subject to a single point of failure. It is also risky to trust single sources for critical data (financial DApps in particular)—as authenticated sources may also suffer from Byzantine influence and provide erroneous data, which could bring catastrophe to the blockchain applications. The above risks were embodied in two recent attacks (October 2022) on two cryptocurrency lending platforms, Mango Markets and Moola Market, which involved the compromise of oracle services followed by manipulation of price data, resulting in more than 100 million US dollars lost [12].

Philosophically, relying on individually trusted oracles and data sources not only invites targeted attacks but also defeats the purpose of a decentralized blockchain application. As a result, mainstream blockchain oracle services [13], [14], [15], [16] have gradually adopted the *decentralized oracles* (DO) model where a consortium of independent and reputed oracle nodes collect data from distributed external sources with respect to common data objects. Data feed proposals from different oracle nodes are aggregated on-chain as the final data feed to the applications.

The Truthful Data Challenge. While it is commendable that existing DO solutions have taken advantage of oracle and source redundancy to tackle the single point of failure, they are still susceptible to the influence of low-quality or compromised external data. There generally lacks a data-plane solution to attaining high-quality data feed in the presence of unreliable or malicious data sources. Some of the DO solutions adopt the heuristic that each oracle node is incentivized to select so-called "premium" data sources for their own good [17] while a global reputation scheme can be used to select the most reliable oracle nodes [11], [15]. Also, the on-chain aggregation (such as taking the average or median [13], [16]) may filter out outliers in the oracles' data proposals. Nonetheless, the reputation-based incentive on source selection does not exclude the presence of authenticated but malicious sources. And the lightweight on-chain aggregation mechanism does not provide effective resilience to sources who dynamically supply Byzantine data to oracle nodes. To provide a trustworthy data feed service to the blockchain, we argue that an oracle solution should incorporate a data-centric mechanism to extract the *truthful data* (close to the inputs provided by honest sources) out of multi-sourced data and achieve resilience against adversely affected sources or oracle nodes.

We observe that truth discovery (TD), a data mining technique that independently evolved [18], [19], [20], poses an ideal data-centric solution to the truthful data challenge. TD jointly estimates the ground truths of data objects and source reliability from potentially conflicting multi-sourced inputs. Unreliable sources are assigned low reliability degrees that will penalize them in the weighted aggregation step. When adapted to an oracle system, this approach can potentially extract trustworthy information from the noisy multi-sourced data, as long as the multi-sourced data exhibits certain statistical patterns and there is a "ground truth" of the data object. However, the traditional TD assumes the algorithm is executed (or eventually aggregated) by one trusted server. To instantiate TD for blockchain oracles, novel adaptations are needed to decentralize the TD workflow, accommodate streaming data inputs, and react to potentially malicious sources and decision-making nodes.

A New Oracle Model. Eyeing the potential of TD in addressing the truthful data challenge and providing a holistic solution to the blockchain oracle problem, we propose the *Decentralized Truth Discovering Oracles (DTDO)* model to enable blockchain applications to procure truthful data from external sources of varying quality without introducing a central point of trust. This model builds upon a network of decentralized oracle nodes, who connect to each other to form a dedicated *off-chain* network for performing collective truth discovery on multi-sourced data. Each node collects inputs for a common list of data objects from its external sources. The nodes agree on the same value for each data object on a batch basis. These values, called the *truth estimates*, are committed as final data feed to the blockchain by each node. For producing the same truthful estimates across all nodes with high reliability while preserving decentralization, the model entails the combined use of data-plane and consensus mechanisms to solve the following challenges. First, sources may arbitrarily deviate their inputs due to abnormality or adversarial influence which can also be adaptive. Second, the off-chain network of nodes in the worst case may operate in an asynchronous, Byzantine-ridden situation—communication between nodes is subject to indefinite delay and some nodes may alter their communicated messages arbitrarily.

DECENTRUTH. We introduce the DECENTRUTH architecture as concrete instantiation of the DTDO model. DECENTRUTH combines techniques from two lines of research: online incremental TD [21], [22], [23] and asynchronous Byzantine fault tolerant (BFT) consensus [24], [25], [26]. We design a novel composite batch incremental TD process (CBI-TD) as the data-plane solution. Nodes are able to consistently produce truth estimates from a common subset of local truth proposals for every batch of objects, and perform reliability tracking on their local sources and peer nodes for achieving Byzantine resilience. For the consensus plane, we devise a consensus protocol called weight-prioritized asynchronous common subset (WP-ACS) that enables nodes to propose its local truth estimates and jointly decide on the aforementioned common subset of proposals amid the harsh asynchronous network condition. Priority is given to the proposals from nodes with higher historical weights computed by CBI-TD. The combination of the two techniques realizes a decentralized oracle service with strong guarantees on Byzantine resilience and data-plane accuracy.

In summary, we make the following contributions:

- We formulate a decentralized truth discovering oracle (DTDO) model to address the truthful data challenge for blockchain oracles. It allows blockchain applications to obtain truthful data from potentially untrustworthy sources, while preserving the decentralization property.
- We introduce the DECENTRUTH architecture to realize the DTDO model while maintaining truth discovery accuracy and Byzantine resilience. DECENTRUTH is composed of two components, CBI-TD and WP-ACS, which harmonize with each other to realize online incremental truth discovery and consensus on global truth estimates.
- We show that DECENTRUTH achieves Byzantine resilience under a practical adversary model, including adaptive Byzantine corruption on legitimate sources and nodes as well as network synchrony.
- We implemented DECENTRUTH and evaluated its performance in an emulated oracle service scenario. The result shows that our system presents a practical ap-

proach towards the truthful data challenge with effective Byzantine resilience and long-term estimation accuracy, outperforming the median-based aggregation mechanism in a well-known decentralized oracle scheme.

II. BACKGROUND AND RELATED WORK

A. Existing Solutions to the Blockchain Oracle Problem

Earlier solutions have addressed the authenticity and integrity part of the blockchain oracle problem. Town Crier [8] is a third-party oracle service for Ethereum smart contracts. It builds on a smart-contract front end and a trusted execution environment (TEE) back end, realizing a secure channel for transporting data from HTTPS-enabled websites to client contracts. DECO [9] realizes a similar authenticated data feed functionality but without trusted computing hardware. It relies on the participation of independent oracles and zero-knowledge proofs after a multiparty authentication process. PDFS [10] and API3 [11] assign the oracle function to data sources directly, representing a "first-party" approach, which essentially tries to incorporate data sources into the blockchain's decentralized trust model. However, the above solutions, third-party and first-party alike, lack effective countermeasures against low-quality and dishonest (authenticated) sources that cannot be prevented from providing bad data into the blockchain. Also, the oracle service itself has to be trusted, posing a single point of failure [7].

Existing commercial oracle services tend to adopt the decentralized oracles (DO) model to introduce redundancy to the oracle nodes. Chainlink [13] is currently the most popular DO solution that comprises of 21 independent and reputed oracle nodes, each is able to provide the Town Crier and DECO functionalities. Other DO solutions like Band Protocol [16], WINkLink [15], and UMA [14] builds on a derivative business logic such as a reputation or reward system to promote the honest participation of oracle nodes. However, how to deal with the authenticated but unreliable sources, which culminates in the truthful data challenge, still remains largely unsolved. In recent proposals, Astraea [27] and Cai et al. [28] use a smart contract to implement a stake-and-vote mechanism to select the most favorable external data. Chainlink [13] and Band Protocol [16] use lightweight on-chain aggregation mechanisms on multi-sourced data, notably taking the median, to rid the outliers in oracle data proposals. These mechanisms potentially add to the expensive on-chain computation and also do not represent an effective data-plane solution to countering fraudulent or deceitful data from individual sources. We stress that in order to solve the truthful data challenge, a data-plane mechanism that extracts truthful information from multi-sourced data should be a native feature of a DO service in order to achieve resilience against potentially Byzantine sources and oracle nodes.

B. Online and Distributed Truth Discovery

Emerged as an independent research, truth discovery (TD) provides a potential data-plane methodology to address the truthful data challenge. We identify two lines of TD research that partially inspired our data-plane design.

Online TD aims to instantiate TD on streaming input in a data-driven fashion, in which data are continuously generated by sources and fed to the TD algorithm. They commonly adopt lightweight mechanisms to handle streaming inputs in an online or recursive fashion [29], [30], [31]. Li et al. [32] propose an online incremental TD scheme that is able to estimate the truths and source reliability degrees with consistent accuracy when the reliability of sources evolves over time. Though not addressing Byzantine sources, these solutions provide valuable lessons on maintaining TD accuracy in continuous operation.

Distributed TD aims to scale TD to larger data volume and source diversity. To accommodate large data volumes in crowdsourcing tasks, Ouyang et al. [21] decompose the original TD problem into several small-scale tasks that can be run in parallel before aggregation. Wang et al. [23] extend the TD problem to a two-stage distributed setting, wherein TD servers handle local data sources while a central server aggregate the local results. These parallel and distributed TD solutions still rely on a central server to perform task allocation and final aggregation. Tian et al. [33] instantiate a TD mechanism using Ethereum smart contract. Fu et al. [34] propose a decentralized TD formulation based on maximum likelihood estimation and P2P gossiping. The inter-node consistency on discovered truths and adversarial influence on nodes are not considered.

III. SYSTEM MODEL

To tackle the truthful data challenge of blockchain oracles, we propose the *decentralized truth discovering oracles* (DTDO) model where a decentralized network of oracles procure truthful data from distributed sources.

A. Network and Task Model

Consider a network of N oracle nodes ("nodes" hereafter) tasked with discovering the true values of common data objects for a blockchain application. We assume there exist S external sources that provide data inputs on the objects to the nodes. Each node $n \in [N]$ ($[N] := \{1, ..., N\}$) has access to a subset of the data sources denoted $\mathcal{S}_n \subset [S]$. $\mathcal{S}_n$ thus represents the "local sources" that report to node n. Nodes may communicate with each other via *asynchronous* but authenticated off-chain channels. That is, messages between any two nodes are guaranteed eventual delivery, but message delays are unpredictable. We choose the asynchrony assumption for it captures the unpredictable network conditions in the off-chain realm, where nodes may communicate with each other in an ad hoc network and the communication link is subject to arbitrary delay in the worst case. Messages are guaranteed eventual delivery with authenticity and integrity provided by TLS communication.

The task of the nodes is to reach an agreement on the same truth estimate of high fidelity for a growing list of data objects. This process is executed in epochs, with each epoch e committed to a batch of objects $\mathcal{B}_e$ with fixed batch size B. This batch configuration accommodates the scenario that the blockchain application needs to periodically take action upon every cycle of updates. Specifically for each epoch e, node $n \in [N]$ receives inputs from its local sources, denoted $\{x^n_{s,i}\}_{s \in \mathcal{S}_n, i \in \mathcal{B}_e}$. $x^n_{s,i}$ is null if source s does not provide input on object i. The input data can be of any type. In this paper, we focus on continuous data for a consistent narrative (e.g., temperature measurement and stock price). At the end of an epoch, all nodes need to reach a consensus on the same truth estimates $\{\hat{x}_i\}_{i \in \mathcal{B}_e}$ for this batch, which should be close to the ground truths.

As a motivating example, consider an investment DApp that needs to feed on hourly closing prices of a portfolio of stocks from different market sources. The price of a stock at a certain hour is an object. The quote on the price-hour provided by a source is an input. By the end of each hour, all honest nodes agree on the same quotes for the same collection of stocks, which are then committed to the DApp's smart contract.

B. Threat Model

We assume an honest majority of sources and oracle nodes. Specifically, the adversary can corrupt up to f_s fraction of authenticated sources and f_n fraction of nodes at any time, with $f_s < 0.5$ and $f_n < 1/3$, and exert Byzantine influence on corrupted sources and nodes. In the data plane, Byzantine sources may provide arbitrary values within the input space on any object to their corresponding nodes. Similarly, Byzantine nodes may send arbitrary or even conflicting information to peer nodes. In line with the asynchronous network assumption, a strong adversary may also introduce arbitrary delays to the communication link between any nodes. We further assume that adversarial corruptions are adaptive in that the adversary can corrupt different subsets of sources and nodes throughout time. The corrupted sources and nodes may act innocuously at first but suddenly turn Byzantine at some points. An honest source provides inputs close to ground truths with high consistency, measured by the reciprocal of the standard deviation of its inputs. An honest node strictly follows the predefined protocol and does not disseminate conflicting information.

The threshold assumption on Byzantine sources and nodes is in line with the existing DO formulations [13], [16] as the sources should be authenticated by all nodes, and the nodes operate with valid credentials. The data-plane goal is not to eliminate the influence of Byzantine sources. A rather practical goal is to maximally mitigate such influence, attaining high accuracy of data feed.

IV. BUILDING BLOCKS

A. Baseline TD

Given a list of objects $\mathcal{O}$ and a collection of inputs $\{x_{s,o}\}_{s \in \mathcal{S}, o \in \mathcal{O}}$ from source group $\mathcal{S}$ on each object in $\mathcal{O}$, TD aims to jointly estimate the truths behind the objects $\{\hat{x}_o\}_{o \in \mathcal{O}}$ and the reliability degrees of sources $\{r_s\}_{s \in \mathcal{S}}$ so that the estimates approximate the ground truths as close as possible.

Prior wisdom provided different mathematical formulations for the TD problem, with nuanced assumptions on issues including input data generation model, data format, source reliability consistency and dependence, and correlation between objects [20]. The formulations commonly take the form of a joint optimization problem and the solutions resemble an iterative procedure, in which truth aggregation (Eq. (1)) and the source reliability degree estimation (Eq. (2)) take place alternately until a certain convergence criterion is met.

$$\hat{x}_o = \frac{\sum_{s \in \mathcal{S}} \mathbb{1}_{s,o} r_s x_{s,o}}{\sum_{s \in \mathcal{S}} \mathbb{1}_{s,o} r_s} \qquad \forall o \in \mathcal{O} \tag{1}$$

$$r_s = g\left(\sum_{o \in \mathcal{O}} d(x_{s,o}, \hat{x}_o) \right) \qquad \forall s \in \mathcal{S} \tag{2}$$

where $\mathbb{1}_{s,o}$ returns 1 if source s provided input on object o (0 otherwise). $d(\cdot,\cdot)$ is a distance measure between input $x_{s,o}$ and the current truth estimate $\hat{x}_o$. $g(\cdot)$ is a monotonically decreasing function. Choices on $d(\cdot,\cdot)$ and $g(\cdot)$ vary among different solutions. Having source reliability in the estimation loop allows the TD algorithm to capture the consistency of a source's inputs on all objects and assign high reliability degrees to sources with consistently accurate inputs.

When instantiating TD for our DTDO model, the baseline TD cannot be used directly since it works on a static, monolithic dataset in a centralized fashion. We will use parts of the baseline TD algorithm in the TD component of our system which also takes inspiration from online and distribution TD approaches (see §II-B for discussions). It is also worth noting that some real-world data objects lack a "ground truth", which poses a challenge to TD deployment and accuracy evaluation. We will explore this kind of data in future work.

B. BFT Consensus and ACS

The BFT consensus problem has been extensively studied in the distributed systems and blockchain community [35], [36]. In the simplest form, it describes a network of N nodes working to agree on a common value, while up to F nodes may behave maliciously (Byzantine) by sending arbitrary values to other nodes [37]. The consensus goal is reachable if $N \geq 3F + 1$ [38].

We are interested in one special type of BFT consensus called *asynchronous common subset (ACS)* which was first formulated by Ben-Or et al. [24]. "Asynchronous" refers to the condition that messages within the network can be delayed arbitrarily, though the eventual delivery is guaranteed, in line with our network setting in §III-A. In such a network of N nodes, up to F nodes can be Byzantine and each node n has a proposal $\mathcal{P}_n$. The goal of ACS is to allow all correct nodes to agree on a common subset of proposals, denoted CSP. An ACS protocol can be composed of two BFT sub-protocols, namely *reliable broadcast (*RBC*)* and *binary agreement (*BA*)*. RBC is a consensus primitive that allows a node to safely disseminate its proposal to peer nodes [39]. When node n starts instance RBC$[n]$ with proposal $\mathcal{P}_n$, the following properties are guaranteed: *1) Agreement:* If any two correct nodes deliver $\mathcal{P}$ and $\mathcal{P}'$, then $\mathcal{P} = \mathcal{P}'$; *2) Validity:* If the leader n is correct, then all correct nodes will eventually deliver $\mathcal{P}_n$. BA is a lightweight consensus primitive that allows nodes to agree on a binary value $\in \{0, 1\}$. BA achieves the following properties: *1) Termination:* If all correct nodes receive input, then each of them will end up with a decision. *2) Agreement:* If any correct node decides b, then all other correct nodes will decide b. *3) Validity:* If any correct node decides b, then b must be the input of at least one node. If all correct nodes have the same input b, then b must be the final decision. To achieve termination in an asynchronous network, BA needs to make random decisions at times when seeing conflicting or insufficient information. In known BA implementations [40], [41], a cryptographic scheme called *common coin* (COMCOIN) [42] is used to provide such randomness. When at least $F+1$ nodes execute the COMCOIN(k) protocol, all nodes will receive the same coin toss result $coin_k \in \{0,1\}$ for object k.

An ACS protocol's agreement, termination, and validity properties follow from its composing RBC and BA protocols. In §V-B we will use RBC, BA, and COMCOIN to compose our customized WP-ACS scheme, a key DECENTRUTH component that facilitates local truth consensus.

V. DECENTRUTH

We introduce the DECENTRUTH architecture as a concrete instantiation of the DTDO model. Fig. 1 illustrates the workflow of DECENTRUTH of N nodes for one epoch. After receiving local inputs for the current epoch e, each node n computes a local estimate $\tilde{x}_i^n$ for every object $i \in \mathcal{B}_e$. These estimates constitute node n's local proposal, denoted $\mathcal{P}_n$. After going through a consensus process called *weight-prioritized asynchronous common subset (WP-ACS)*, all nodes decide on a common subset of proposals denoted CSP, which is subsequently fed to the global TD algorithm. Here "global" means the algorithm and its hyperparameters are pre-determined in all nodes. The global TD at all honest nodes will return the same estimated truths and the node weights. At the epoch end, each node updates the reliability degrees of its local sources based on the newly obtained truth estimates. The updated source reliability degrees will be used for local truth estimation in the next epoch. The local truths estimation, global TD, and source reliability updating constitute a *composite batch incremental TD (CBI-TD)* process that keeps track of source reliability degrees and node weights and make online decisions on global truth estimates. Notably, we focus on the off-chain operation in DECENTRUTH. Realization of the final data feed commitment to blockchain applications is an independent task that we defer to future extension. Next, we elaborate on the two major system components, i.e., CBI-TD and WP-ACS.

A. Component 1: CBI-TD

Each node n maintains four types of internal variables: reliability degree r_s^n, error measure ϵ_s^n and consistency measure κ_s^n of each local source $s \in \mathcal{S}_n$, and node weight w_k of every peer node $k \in [N]$ (all notations are associated to the current epoch e unless otherwise specified). Two sub-tasks are executed: local incremental TD and global TD.

Local Incremental TD is a cross-epoch procedure responsible for generating proposals for the global TD and keeping track of the reliability degrees of local sources. While previous solutions have demonstrated formulating an optimization problem on historical data to realize incremental TD [30], [32], our scheme faces the unique challenge of Byzantine sources and needs a new reliability evaluation method for Byzantine resilience.

For each epoch, the local incremental TD works as follows. After receiving inputs $\{x_{s,i}^n\}_{s\in\mathcal{S}_n, i\in\mathcal{B}_e}$ from local sources $\mathcal{S}_n$, each node $n \in [N]$ computes the local truths using the local source reliability degrees $\{r_s^n\}$ from the last epoch:

$$\tilde{x}_i^n = \frac{\sum_{s\in\mathcal{S}_n} \mathbb{1}_{s,i} r_s^n x_{s,i}^n}{\sum_{s\in\mathcal{S}_n} \mathbb{1}_{s,i} r_s^n} \qquad \forall i \in \mathcal{B}_e \tag{3}$$

Then node n compiles its proposal $\mathcal{P}_n = \{\tilde{x}_i^n\}_{i\in\mathcal{B}_e}$ and provides $\mathcal{P}_n$ to the consensus process and the ensuing global TD. After the global TD delivers the estimated global truths $\{\hat{x}_i\}_{i\in\mathcal{B}_e}$, node n computes an *error measure* ϵ_s^n with mean square error on objects that s has participated, and a *consistency measure* $\kappa_s^n \in (0,1]$:

$$\epsilon_s^n = \frac{\sum_{i\in\mathcal{B}_e} \mathbb{1}_{s,i}(x_{s,i}^n - \hat{x}_i)^2}{\sum_{i\in\mathcal{B}_e} \mathbb{1}_{s,i}} \tag{4}$$

$$\kappa_s^n = \mathit{erfc}(\beta|\epsilon_s^n - \mathring{\epsilon}_s^n|)\cdot(1-\alpha) + \mathring{\kappa}_s^n \cdot \alpha \tag{5}$$

wherein $\mathring{\epsilon}_s^n$ and $\mathring{\kappa}_s^n$ refer to the corresponding measures from the last epoch. $\mathit{erfc}()$ is the complementary error function that is widely used for evaluating statistical accuracy. The intuition behind using $\mathit{erfc}()$ here is that it is a monotonic decreasing function and provides a convenient output range $(0,1]$ for input range $[0,\infty)$. It sharply penalizes input increase when input is close to 0, which helps stage a swift response to Byzantine source inputs. The weighted moving average calculation of κ_s^n is aimed at space efficiency, as it only needs the most recent error and consistency measures. $\alpha \in [0,1)$ is a user-defined decay factor: lower α enables swift reaction to short-term Byzantine behaviors while higher α helps establish long-term judgement on Byzantine sources. $\beta > 0$ is the scale factor that depends on the range of input value. α and β are design choices during implementation.

Finally, node n updates its local source reliability degrees:

$$r_s^n = \frac{\kappa_s^n}{\epsilon_s^n} \qquad \forall s \in \mathcal{S}_n \tag{6}$$

While the error metric ϵ_s^n penalizes source s for generally inaccurate inputs, the consistency measure κ_s^n (when close to 0) penalizes source s specifically for Byzantine influence. We provide analysis on how κ_s^n facilitates the Byzantine resilience of our system in §V-C.

Global TD is executed by every node to obtain the global truth estimates $\{\hat{x}_i\}_{i\in\mathcal{B}_e}$ for epoch e after obtaining the common subset of proposals $CSP = \{\mathcal{P}_k | k \in CS\}$. CS is the corresponding subset of node IDs and $\mathcal{P}_k = \{\tilde{x}_i^k\}_{i\in\mathcal{B}_e}$. As the problem of Byzantine proposals is addressed by the preceding WP-ACS consensus (to elaborate in §V-B), we adopt the conventional optimization-based approach for global TD. With

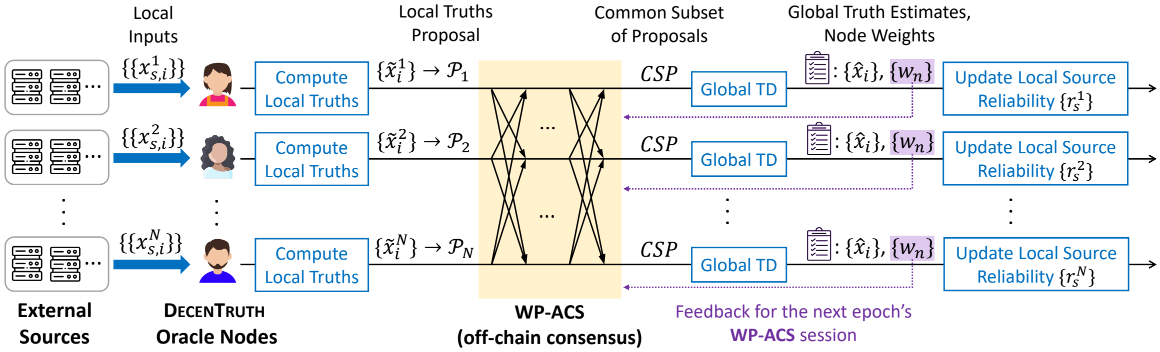


Fig. 1. DECENTRUTH workflow in one epoch. Data-plane operations that constitute CBI-TD are highlighted in blue.

node weights $\{w_k\}_{\in CS}$ and proposed values $\{\tilde{x}_i^k\}_{k\in CS, i\in\mathcal{B}_e}$ we formulate the following optimization problem:

$$\min_{\{w_k\},\{\hat{x}_i\}} \sum_{k\in CS}\sum_{i\in\mathcal{B}_e} w_k d(\tilde{x}_i^k,\hat{x}_i) \quad \text{s.t.} \sum_{k\in CS}\log w_k = 1 \tag{7}$$

$d()$ can be any distance function, such as the square error we used for local TD. We keep the form $d()$ for generality.

This problem can be solved by coordinate descent in an iterative manner. First, we fix the truths estimates $\{\hat{x}_i\}$ and apply the Lagrange multipliers method to Eq. (7) to get the best estimate of the weights $\{w_k\}$. We omit the derivation and directly give the result (wherein c is a constant):

$$w_k = \frac{c}{\sum_{i\in\mathcal{B}_e} d(\tilde{x}_i^k,\hat{x}_i)} \qquad \forall k \in CS \tag{8}$$

Next, we fix the weights and aggregate the truths:

$$\hat{x}_i = \frac{\sum_{n\in CS} w_n \tilde{x}_i^n}{\sum_{n\in CS} w_n} \qquad \forall i \in \mathcal{B}_e \tag{9}$$

Eq. (8) and Eq. (9) are executed alternately until a convergence criterion is met. After that $\{\hat{x}_i\}_{\mathcal{B}_e}$ are committed to the blockchain as the global truths. Moreover, throughout the iterations we keep the weights of nodes outside CS, i.e., $\{w_z\}_{z\notin CS}$, unchanged for consistency. Thus one more step is needed to normalize the weights of those in CS: $w_k \leftarrow \frac{w_k}{\sum_{l\in CS} w_l}\left(1-\sum_{z\notin CS} w_z\right), \forall k \in CS$.

B. Component 2: WP-ACS

WP-ACS is the consensus component that enables nodes to agree on a common set of proposals CSP in the asynchronous network before proceeding to global TD. We define $F := \lfloor \frac{N-1}{3} \rfloor$ as the maximum possible number of Byzantine nodes—a design parameter rather than the actual Byzantine node population (which is Nf_n). WP-ACS is designed to take advantage of the global TD for improving the quality of its output. In specific, WP-ACS relies on global TD for feedback on node weights from the last epoch. Proposals from nodes with lower weights should have a reduced chance of being included in CSP. The ensuing global TD in turn can benefit from the improved quality of CSP. This feedback loop mechanism is essential for DECENTRUTH to penalize the proposals made by malfunctioning/Byzantine nodes.

Algorithm 1: WP-ACS (for each epoch by node n)

```
Variables: $\mathcal{L}$, $dSet \leftarrow \perp$, $coin$
Input: Local proposal $\mathcal{P}_n$, node weights $\{w_k\}_{k\in[N]}$
Output: Common subset of proposals $CSP$ and indices $CS$
Assign $\mathcal{L} \leftarrow \{l | w_l$ is among the top $N-F$ of $\{w_k\}_{k\in[N]}\}$
Start RBC$[n]$ with $\mathcal{P}_n$ as input
while true do
    if receiving the delivery of $\mathcal{P}_k$ from RBC$[k]$ and input has not been provided to BA$[k]$ then
        if if $k \in \mathcal{L}$ then
            Provide input 1 to BA$[k]$
        else
            $dSet \leftarrow dSet \cup \{k\}$
    if having provided inputs to at least $N-2F$ BA instances with identifiers $n \in \mathcal{L}$ and $dSet \neq \perp$ then
        for $k \in dSet$ do
            $coin_k \leftarrow$ COMCOIN$(k)$
            Provide input $coin_k$ to BA$[k]$
        $dSet \leftarrow \perp$
    if having received outputs of value 1 from at least $N-F$ BA instances then
        Provide input 0 to each of the BA instances that has not been provided input
    if all $N$ BA instances have output a value then
        $CS \leftarrow \{k | $BA$[k]$ outputs $1\}$
        $CSP \leftarrow \{\mathcal{P}_k | k \in CS\}$ (wait for RBC$[k]$ to deliver $\mathcal{P}_k$ if not received yet)
        Return $CS$, $CSP$
```

Following the above intuition, we propose the WP-ACS protocol in Algorithm 1. WP-ACS makes use of the RBC, BA, COMCOIN primitives as described in §IV-A. In the beginning, every node computes the priority list $\mathcal{L}$ using the node weights $\{w_n\}_{n\in[N]}$ from the last epoch: $\mathcal{L}$ contains the identifiers of the top $N-F$ nodes ranked by weights, and their proposals are tagged 'trustworthy'. Nodes outside $\mathcal{L}$ are considered potentially Byzantine and their proposals are tagged 'questionable'. When a node receives the delivery of proposal $\mathcal{P}_k$ from RBC$[k]$ and has not provided input to BA$[k]$ yet, it provides input 1 to BA$[k]$ if k is in $\mathcal{L}$. If k is outside $\mathcal{L}$, it is added to the deferred action set $dSet$.

Only when at least $N - 2F$ BA instances identified in $\mathcal{L}$ have been provided input, will the protocol provide $coin_k$ to BA$[k]$ for $k \in dSet$. Here $coin_k \in \{0,1\}$ is the common coin received from ComCoin(k). When at least $N - F$ BA instances output 1, all the remaining BA instances are provided input 0 to facilitate a timely conclusion (per the "termination" requirement). When all N BA instances have provided an output, CSP is assigned to the proposals whose corresponding BA output 1. CSP shall contain at least $N-2F$ 'trustworthy' proposals (i.e., identified in $\mathcal{L}$) plus at most F 'questionable' ones. Detailed analysis is provided in Propositions 2, 3.

The deferred coin-toss treatment leaves a 1/2 chance for the 'questionable' proposals that finish RBC early to be included in CSP. Our scheme does not reject 'questionable' proposals altogether (i.e., provide input 0 to their BA instances) due to the following consideration. If $\mathcal{P}_k$ falls into the questionable proposals due to its accidental bad inputs, $\mathcal{P}_k$ should be still given a chance, though reduced, of being considered in the ensuing global TD in which w_k can be re-evaluated. This essentially provides honest nodes a recovery path from short-term deterioration in source data quality.

C. Analyses

We show DecenTruth's resilience against Byzantine sources and nodes under the threat model in §III-B.

***Proposition** 1 (Byzantine source resilience):* In the continuing operation, the accuracy of truth estimates can recover from degradation caused by suddenly turned Byzantine sources.

Proof Sketch: When source $s \in \mathcal{S}_n$ turns Byzantine, its inputs arbitrarily deviate from the ground truths. When updating local source reliability degrees, the consistency measure κ_s^n evaluates the variation of reliability of source s across different epochs (Eq. (5)). A low κ_s^n captures that source s is no longer bound to the fundamental assumption of consistent reliability. When source s starts to behave Byzantine, node n will assign a near-to-zero κ_s^n in the subsequent epochs, resulting in a near-to-zero reliability degree r_s^n which minimizes the impact of source s' subsequent inputs and thus maintains the global TD's accuracy. We remark that such quick reaction and accuracy are built on the honest majority assumption on all sources and nodes in the threat model, which ensures that probabilistically more than half of the local truth proposals to WP-ACS are consistently close to the ground truths. ■

***Proposition** 2 (Byzantine node resilience, static case):* If the adversary corrupts up to $F := \lfloor \frac{N-1}{3} \rfloor$ nodes in a static manner who start behaving Byzantine at some point, then in the long run, with overwhelming probability, at least $\frac{3}{4}$ of the common subset of proposals (CSP) for each epoch will come from uncorrupted nodes.

Proof Sketch: For any proposal $\mathcal{P}_k$ marked 'questionable' ($k \notin \mathcal{L}$), the WP-ACS algorithm's deferred coin tossing treatment dictates that the chance of accepting $\mathcal{P}_k$ into CSP is capped by $\frac{1}{2}$, representing the case that RBC$[k]$ is among the first $N - f$ RBC instances to deliver and $coin_k = 1$. In this case, the expected number of 'questionable' proposals in CSP maxes at $\frac{F}{2}$ with overwhelming probability (for large N). Meanwhile, since there will be at least $N-F$ proposals in CSP, the expected ratio of 'questionable' proposals in CSP is thus capped by $\frac{F}{2(N-F)} < \frac{1}{4}$.

For any corrupted node that has started behaving Byzantine (e.g., providing tampered proposals), if its proposal is excluded from CSP for the current epoch, the chance of accepting its proposal in the next epoch will be capped by $\frac{1}{4}$ with overwhelming probability. On the other hand, if a Byzantine proposal $\mathcal{P}_z$ happens to be included in CSP, the global TD will assign a low node weight w_z as long as $\mathcal{P}_z$ deviates significantly from those of honest nodes. And its next-epoch proposal will be marked 'questionable' again. When more epochs pass, the group of Byzantine nodes will converge with the group that constantly provides 'questionable' proposals, resulting in a long-term CSP with at least $\frac{3}{4}$ proposals from the uncorrupted nodes. ■

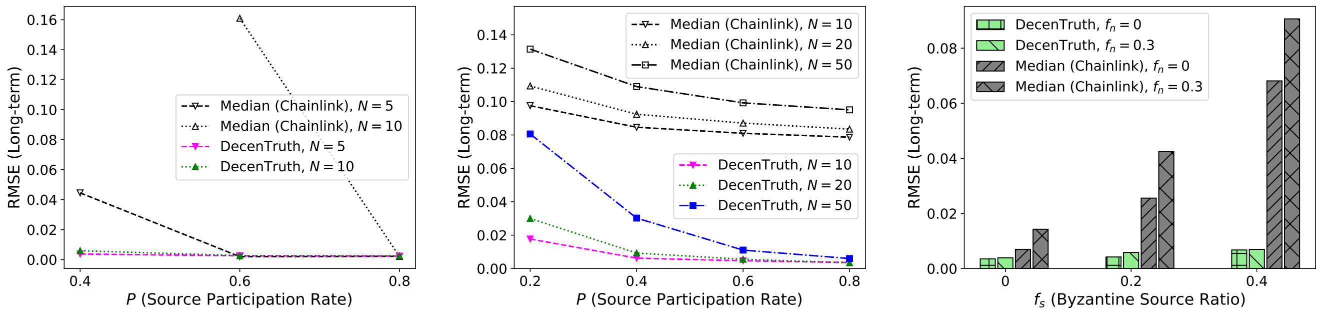
***Proposition** 3 (Byzantine node resilience, adaptive case):* If the adversary can corrupt any targeted nodes (up to F) on an epoch-to-epoch basis, then at least half of proposals in CSP will come from honest nodes.

Proof Sketch: We consider the worst case where the adversary can adaptively corrupt F nodes that all belong to the 'trustworthy' category (corresponding to those in $\mathcal{L}$), and its network scheduling capability can make the RBC instances of corrupted nodes delivery ahead of honest nodes. This results in a Byzantine proposal ratio in CSP of $\frac{F}{N-F}$, which is less than $\frac{1}{2}$. This means the subsequent global TD can still rely on the honest-majority proposals in CSP for truth aggregation and assign lower weights to the corrupted nodes. ■

Complexity. The communication overhead solely comes from the protocol messages of WP-ACS. The communication complexity of WP-ACS follows from its composing primitives, namely RBC [43], BA [41] and ComCoin [40], yielding complexity of $O(N|\mathcal{P}| + \lambda N^2 \log N)$ bits per node. $|\mathcal{P}|$ is the size of a proposal in bits and λ is a security parameter of ComCoin. The computation complexity is contributed by the CBI-TD in the data plane and the cryptographic computations in WP-ACS' RBC and BA components. In the data plane alone, the local incremental TD and global TD yield $\Theta(\frac{S}{N})$ and $\Theta(N)$ truth evaluations per epoch, resulting in $\Theta(\frac{S}{N} + N)$ data-plane complexity at each node for each epoch.

VI. IMPLEMENTATION AND EVALUATION

We implemented a DecenTruth node prototype in Python. The CBI-TD component contains the local incremental TD and global TD modules as in Fig. 1. For the WP-ACS component, we adopted the implementation of RBC and BA primitives from HoneyBadgerBFT [25] while directly instantiating ComCoin with pre-distributed secret shares for reducing cryptographic overhead. The node-to-node message format includes a node identifier, protocol instance identifier, message type (RBC or BA and subtypes), and a 1500-byte payload for truth proposal. We implemented an environment simulator env that can simulate data inputs under Byzantine influence to each node and also introduce delays for any node-to-node message. For comparison, we also implemented a



(a) RMSE vs P (Nasdaq-100 dataset)

(b) RMSE vs P (synthetic dataset)

(c) RMSE vs f_s, f_n (synthetic dataset)

Fig. 2. Long-term RMSE results. Other parameters: (a)(b) $f_s = 0.4, f_n = 0.3$. (c) $N = 20, P = 0.5$.

reference routine called *Median* to emulate Chainlink's [13] aggregation scheme whereby for each data object, each node's proposal is the median of the inputs from local sources and the median of all proposals serves as the final data feed.

Evaluation Setting. We evaluated DECENTRUTH's performance as an oracle service through simulation experiments on an AWS c5.12xlarge instance (48 vCPU, 96GB memory, Ubuntu 20.04 LTS). For each experiment, $N \in \{5, 10, 20, 50\}$ DECENTRUTH nodes ran in parallel along with one env instance. The maximum number of Byzantine nodes F, a design parameter, was fixed to $\lfloor \frac{N-1}{3} \rfloor$. Packet delay between any two nodes was randomly sampled from the exponential distribution of rate $\frac{1}{mdelay}$, simulating volatile communication delays, with mean delay $mdelay \in \{0.2, 0.4, 0.8\}$sec. env was pre-loaded with the datasets and randomly assigned the data to N source groups, each group being local to a node. The data were provided to the nodes during runtime on a batch basis. To simulate a source participation rate of $P \in (0, 1)$, every input was discarded with probability $1 - P$. For comparison, we instantiated the *Median* nodes in the same machine that hosted DECENTRUTH nodes.

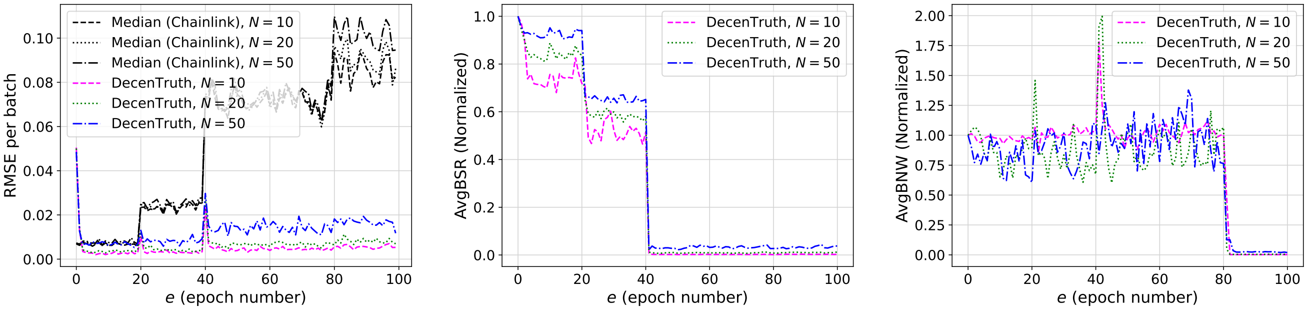
Dataset. We used the Nasdaq-100 dataset, a part of the *Stock* dataset [44] that is widely used in TD research, in selected experiments. Nasdaq-100 contains the daily closing price inputs from 55 sources on 100 Nasdaq stock symbols in the 21 trading days of July 2011. We also generated a synthetic dataset containing inputs from 1,000 sources on 10,000 data objects with normalized value, with batch size B fixed to 100, simulating a bigger Nasdaq-100 dataset. As a result, the 10,000 data objects were delivered in 100 epochs. Each input $x_{s,i}$ from source s on object i was randomly sampled from $N(\dot{x}_i, \delta_s)$ bounded by $[0, 1]$, with $\dot{x}_i$ being the ground truth. δ_s was a predetermined value randomly sampled from $U(0, 0.5)$, representing the intrinsic unpredictability of source s. Building on top of a large number of data objects and sources in this synthetic dataset, we are able to evaluate the performance of DECENTRUTH upon malicious modification by Byzantine sources at different settings.

To evaluate our system's Byzantine resilience, we imposed Byzantine behaviors on the synthetic dataset. Byzantine sources and nodes could arbitrarily deviate their output data but still within the $[0, 1]$ range. For each experiment run iterating through 10,000 objects (with batch size B), three "Byzantine mutinies" took place sequentially. f_s ratio of all sources eventually turned Byzantine with the first half turning at epoch 20 and the second half turning at epoch 40. f_n ratio of all N nodes turn Byzantine at epoch 80. Lastly, for the moving average scheme in Eq. 5 we heuristically chose the decay factor $\alpha = 0.9$ and scale factor $\beta = 100$ after exploring DECENTRUTH with the synthetic data for the fastest recovery from a Byzantine source mutiny (we will explore more rigorous tuning methods in future work).

A. Data-plane Performance and Byzantine Resilience

We first evaluated the long-term TD accuracy of DECENTRUTH and its Byzantine resilience. For each run, we collected the last 10% batches of estimated truths and computed their root-mean-square error (RMSE) against the ground truths. RMSE quantifies the system's overall TD (in)accuracy. Fig. 2(a) shows the data-plane performance of DECENTRUTH on the Nasdaq-100 dataset. When the source participation rate is low (i.e., $P = 0.4$), the *Median* method under $N = 10$ fails to converge while DECENTRUTH still does. For higher P where both DECENTRUTH and *Median* converge, the RMSE results are significantly lower than those in Fig 2(b) where the synthetic dataset was used. This is because we did not introduce Byzantine mutinies to the Nasdaq-100 dataset. Both Fig 2(b) and Fig. 2(c) also show that DECENTRUTH outperforms *Median* under every evaluation case by a significant margin. Meanwhile, larger N leads to higher RMSE of DECENTRUTH especially when source participation rate P is low, as shown in Fig. 2(b). Here we give a possible explanation. For low P and large N, each node receives a small number of inputs and a significant portion of objects are not covered by any input at all. The proposals received by global TD will thus find less common ground in approaching ground truths. Fig. 2(c) shows the influence of Byzantine sources and nodes at a fixed N. Higher Byzantine source ratio (f_s) and Byzantine node ratio (f_n) both contribute to higher RMSE, but are limited in scale compared to the impact of low P, as Fig 2(b) shows.

To provide deeper insight into DECENTRUTH's resilience to adaptive Byzantine influence and the data-plane performance in real-time, we measured the per-batch RMSE, average Byzantine source reliability degree (AvgBSR), average



(a) RMSE result across epochs

(b) Avg. Byzantine source reliability degree

(c) Avg. weight of Byzantine nodes

Fig. 3. Temporal data-plane performance showing DECENTRUTH's Byzantine resilience (parameters: $P = 0.5, f_s = 0.4, f_n = 0.3$). The synthetic dataset was used. The first (second) half of Byzantine sources turned active at epoch 20 (40). All Byzantine nodes turned active at epoch 80.

Byzantine node weight (AvgBNW) throughout the 100 epochs. Fig. 3(a) demonstrates DECENTRUTH's quick comeback in accuracy at the start and the two Byzantine source mutinies (at epoch 20 and 40), which is in contrast to *Median*'s deteriorating performance. Fig. 3(b) further demonstrates DECENTRUTH's capability in detecting Byzantine behaviors and assigning low reliability degrees to Byzantine sources. A similar swift response is also observed for the weights of Byzantine nodes (turned active at epoch 80) as illustrated in Fig. 3(c). AvgBNW quickly reduces to near-zero, demonstrating the system's swift response to compromised nodes and ability to retain high TD accuracy.

Fig. 3(a) also shows that experiments with larger N experienced deeper accuracy degradation, as is indicated by the blue curve's steeper increase in RMSE after each Byzantine mutiny. A plausible reason is that a larger N means fewer local sources per node; individual nodes are more likely to encounter a situation of Byzantine majority among local sources.

B. Consensus Runtime

We evaluated the consensus runtime of WP-ACS under different networking scenarios. Fig. 4 shows the consensus runtime results under different N and $mdelay$. It shows the consensus runtime grows linearly in $mdelay$ for small N and grows quadratically in N. And N tends to have a bigger performance impact as it increases. We speculate a two-fold reason: (1) larger N leads to higher cryptographic computation overhead at each node; (2) the total processing capacity of our simulation environment was limited so that the cryptographic computation overhead could easily overshadow the communication overhead. For practical deployment, we recommend N be controlled in a reasonable size in order to limit the consensus latency while ensuring a two-thirds honest majority (similar to Chainlink [13] which has 21 oracle nodes). In future work, we plan to use lightweight cryptography to instantiate the RBC and BA components of WP-ACS in a more efficient manner, with lessons from the asynchronous BFT consensus research [26], [45], [46], [47]. Lastly, the data-plane runtime results (of CBI-TD) are obviated in this paper. We observed that its cumulative computation time of one epoch was generally within one second.

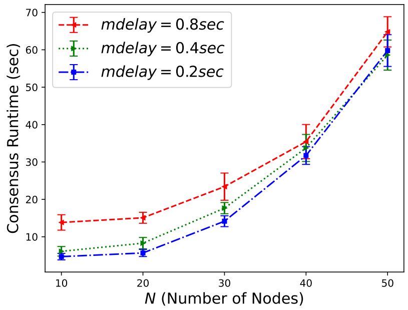


Fig. 4. WP-ACS consensus runtime for one epoch (parameters: $P = 0.5, f_s = 0.4, f_n = 0.3$).

VII. CONCLUSION

Eyeing on truthful data challenge facing blockchain oracles, we propose a new decentralized truth-discovering oracles model (DTDO) to enable blockchain applications to obtain truthful data on from potentially malicious external sources while preserving the decentralized purpose of a blockchain application. As a concrete instantiation of this model, we introduce DECENTRUTH harmonizing techniques from two domains, namely truth discovery (TD) and asynchronous BFT consensus, while addressing challenges in system resilience under the threat of Byzantine sources and nodes. We implemented DECENTRUTH and evaluated its performance in an emulated oracle service scenario. The result demonstrates that DECENTRUTH outperforms the median-based aggregation mechanism used in existing solutions in terms of TD accuracy and Byzantine resilience by a significant margin.

In future work, we will work on the following improvements: (1) improving the scalability of DECENTRUTH by developing more efficient WP-ACS consensus, (2) addressing the more challenging data-plane scenario when there is no "ground truth" of each data object, and (2) defending against the adversary who can slowly "poison" the data sources.

ACKNOWLEDGMENT

This work was supported in part by the US National Science Foundation under grants CNS-1916902, CNS-1916926, CNS-2154929, and CNS-2154930, and the Office of Naval Research under grant N00014-19-1-2621.

REFERENCES

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[2] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, vol. 21, pp. 2327–4662, 2016.

[3] EOS.IO, "EOS.IO technical white paper v2," 2018.

[4] S. Ellis, A. Juels, and S. Nazarov, "Chainlink a decentralized oracle network," *Retrieved March*, vol. 11, p. 2018, 2017.

[5] Chainlink, "77+ smart contract use cases enabled by chainlink." https://blog.chain.link/smart-contract-use-cases/, 2022.

[6] M. Bartholic, A. Laszka, G. Yamamoto, and E. W. Burger, "A taxonomy of blockchain oracles: The truth depends on the question," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–15, IEEE, 2022.

[7] Chainlink, "What is the blockchain oracle problem?." https://blog.chain.link/what-is-the-blockchain-oracle-problem/, 2022.

[8] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 270–282, 2016.

[9] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1919–1938, 2020.

[10] J. Guarnizo and P. Szalachowski, "Pdfs: practical data feed service for smart contracts," in *European Symposium on Research in Computer Security*, pp. 767–789, Springer, 2019.

[11] B. Benligiray, S. Milic, and H. Vänttinen, "Decentralized APIs for Web 3.0," *API3 Foundation Whitepaper*, 2020.

[12] T. Geron, "Crypto oracles are a hidden vulnerability." https://www.protocol.com/newsletters/protocol-fintech/data-oracles-crypto, 2022.

[13] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021.

[14] UMA, "UMA Protocol." https://docs.umaproject.org/, 2022.

[15] WINkLink, "Introduction to WINkLink." https://doc.winklink.org/v1/doc/en/, 2022.

[16] Band Protocol, "Bandchain whitepaper." https://docs.bandchain.org/whitepaper/, 2022.

[17] Chainlink, "The 3 levels of data aggregation in chainlink price feeds." https://blog.chain.link/levels-of-data-aggregation-in-chainlink-price-feeds/, 2020.

[18] X. Yin, J. Han, and S. Y. Philip, "Truth discovery with multiple conflicting information providers on the web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796–808, 2008.

[19] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han, "Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 745–754, 2015.

[20] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A survey on truth discovery," *ACM Sigkdd Explorations Newsletter*, vol. 17, no. 2, pp. 1–16, 2016.

[21] R. W. Ouyang, L. M. Kaplan, A. Toniolo, M. Srivastava, and T. J. Norman, "Parallel and streaming truth discovery in large-scale quantitative crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2984–2997, 2016.

[22] D. Zhang, D. Wang, N. Vance, Y. Zhang, and S. Mike, "On scalable and robust truth discovery in big data social media sensing applications," *IEEE Transactions on Big Data*, vol. 5, no. 2, pp. 195–208, 2018.

[23] Y. Wang, F. Ma, L. Su, and J. Gao, "Discovering truths from distributed data," in *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 505–514, IEEE, 2017.

[24] M. Ben-Or, B. Kelmer, and T. Rabin, "Asynchronous secure computations with optimal resilience," in *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, pp. 183–192, 1994.

[25] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 31–42, 2016.

[26] S. Duan, M. K. Reiter, and H. Zhang, "Beat: Asynchronous bft made practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2028–2041, 2018.

[27] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pp. 1145–1152, IEEE, 2018.

[28] Y. Cai, N. Irtija, E. E. Tsiropoulou, and A. Veneris, "Truthful decentralized blockchain oracles," *International Journal of Network Management*, p. e2179, 2021.

[29] D. Wang, T. Abdelzaher, L. Kaplan, and C. C. Aggarwal, "Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pp. 530–539, IEEE, 2013.

[30] Z. Zhao, J. Cheng, and W. Ng, "Truth discovery in data streams: A single-pass probabilistic approach," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1589–1598, 2014.

[31] J. Kim, B. Tabibian, A. Oh, B. Schölkopf, and M. Gomez-Rodriguez, "Leveraging the crowd to detect and reduce the spread of fake news and misinformation," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 324–332, 2018.

[32] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han, "On the discovery of evolving truth," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 675–684, 2015.

[33] Y. Tian, J. Yuan, and H. Song, "Secure and reliable decentralized truth discovery using blockchain," in *2019 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–8, IEEE, 2019.

[34] L. Fu, J. Xu, S. Qu, Z. Xu, X. Wang, and G. Chen, "Seeking the truth in a decentralized manner," *IEEE/ACM Transactions on Networking*, 2021.

[35] H. Attiya and J. Welch, *Distributed computing: fundamentals, simulations, and advanced topics*, vol. 19. John Wiley & Sons, 2004.

[36] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[37] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, pp. 203–226, ACM, 2019.

[38] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[39] G. Bracha, "Asynchronous byzantine agreement protocols," *Information and Computation*, vol. 75, no. 2, pp. 130–143, 1987.

[40] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography," *Journal of Cryptology*, vol. 18, no. 3, pp. 219–246, 2005.

[41] A. Mostefaoui, H. Moumen, and M. Raynal, "Signature-free asynchronous byzantine consensus with t< n/3 and o (n2) messages," in *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pp. 2–9, 2014.

[42] M. O. Rabin, "Randomized byzantine generals," in *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pp. 403–409, IEEE, 1983.

[43] C. Cachin and S. Tessaro, "Asynchronous verifiable information dispersal," in *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pp. 191–201, IEEE, 2005.

[44] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, "Truth finding on the deep web: Is the problem solved?," *arXiv preprint arXiv:1503.00303*, 2015.

[45] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo: Faster asynchronous bft protocols," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 803–818, 2020.

[46] Y. Lu, Z. Lu, Q. Tang, and G. Wang, "Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited," in *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pp. 129–138, 2020.

[47] T. Crain, C. Natoli, and V. Gramoli, "Red belly: a secure, fair and scalable open blockchain," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 466–483, IEEE, 2021.