Delphi: Efficient Asynchronous Approximate Agreement for Distributed Oracles

Akhil Bandarupalli*, Adithya Bhat†, Saurabh Bagchi*, Aniket Kate*‡, Chen-Da Liu-Zhang§¶, Michael K. Reiter‖**
*Purdue University *{abandaru, sbagchi, aniket}@purdue.edu*
†Visa Research *haxolotl.research@gmail.com*
§Lucerne University of Applied Sciences and Arts *chen-da.liuzhang@hslu.ch*
‖Duke University *michael.reiter@duke.edu*
‡Supra Research, ¶Web3 Foundation, **Chainlink Labs

2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) | 979-8-3503-4105-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/DSN58291.2024.00051

***Abstract*—Agreement protocols are crucial in various emerging applications, spanning from distributed (blockchains) oracles to fault-tolerant cyber-physical systems. In scenarios where sensor/oracle nodes measure a common source, maintaining output within the convex range of correct inputs, known as convex validity, is imperative. Present asynchronous convex agreement protocols employ either randomization, incurring substantial computation overhead, or approximate agreement techniques, leading to high $\tilde{\mathcal{O}}(n^3)$ communication for an n-node system.**

This paper introduces Delphi, a deterministic protocol with $\tilde{\mathcal{O}}(n^2)$ communication and minimal computation overhead. Delphi assumes that honest inputs are bounded, except with negligible probability, and integrates agreement primitives from literature with a novel weighted averaging technique. Experimental results highlight Delphi's superior performance, showcasing a significantly lower latency compared to state-of-the-art protocols. Specifically, for an $n = 160$-node system, Delphi achieves an 8x and 3x improvement in latency within CPS and AWS environments, respectively.

I. Introduction

In the distributed oracle problem setting, a set of n nodes each with an input coming from its sensor device, aim to closely agree on a common output while tolerating a minority of Byzantine faults. This problem is getting increased interest recently as it captures many scenarios including fault-tolerant distributed cyber-physical systems (CPS) [28], [38], [39], [44], [52], oracle networks [16], [20], or agreement on physical sensor inputs such as the average global temperature. Such systems are typically deployed in compute-starved environments supported by paltry infrastructure and asynchronous networks with unpredictable delays. In this context, to ensure that it accurately represents the real-world variable, the output of an ideal system is close to the range of honest inputs. The convex agreement problem [21] captures this requirement by ensuring that the output is in the convex hull (or min-max range) of honest inputs.

Current solutions to the convex agreement problem can be categorized into two different classes. In the first, protocols that use randomization using common coins [9]. Generally, such coins are computationally expensive and require complex cryptographic setup assumptions. As an example, the most efficient and popular implementation of a common coin [14] requires $\mathcal{O}(n)$ bilinear pairing computations per coin [13], [18]. Each pairing requires 1000 times more time and energy than symmetric key primitives used in TLS. Furthermore, it also requires a threshold setup, which involves running an expensive Asynchronous Distributed Key Generation (ADKG) procedure amongst nodes [23], [36].

In the second, protocols use asynchronous approximate agreement (AAA) [24]. These AAA protocols are deterministic and, therefore, avoid the expensive computational cost of generating common coins. The state-of-the-art approximate agreement protocols [1] incurs high communication costs of $\mathcal{O}(n^3)$ bits per round and take $\mathcal{O}(\log(\frac{\delta}{\epsilon}))$ rounds to achieve outputs that are ϵ-close to each other when the range of honest inputs is δ.

Towards addressing these inefficiencies, we make a key observation regarding input from honest parties for the oracle problems. In the CPS settings, the ambient noise causing the difference between inputs follows distributions like Normal and Lognormal [48], [49]. Even prices of stocks and cryptocurrencies are often modeled using distributions like Pareto and Loggamma, where samples are mostly concentrated within a small distance [20]. As a result, given that honest parties often measure inputs that are close to each other, we assume that honest inputs of size ℓ are sampled from a *thin-tail* distribution.

Building on the above observations and the corresponding assumption, we present a protocol that is efficient in both computation and communication. We introduce Delphi, a signature-free deterministic protocol with $\mathcal{O}(\ell n^2 \log(\lambda \log n))$ communication, where λ is a statistical security parameter. Delphi's outputs are ϵ-close and are at most within a distance $\delta = O(\epsilon)$ from the convex range of honest inputs. See Table I for a comparison to prior works.

A. Solution Overview

Our starting point is a protocol BinAA that achieves Approximate Agreement with binary inputs, i.e. either 0 or 1. In this case, we observe that the parties output values within ϵ distance with $\mathcal{O}(n^2 \log \frac{1}{\epsilon})$ communication.

Next, we divide the real input space into intervals of size ρ. Then, for each interval, each honest node i participates in BinAA with input 1 if its input v_i is within ρ-distance to the interval, or 0 otherwise. Note that if all honest inputs

2158-3927/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN58291.2024.00051

Table I: Comparison of relevant Asynchronous Convex BA protocols

Protocol	Communication Complexity (in bits)	Rounds	Computation		Agreement Distance	Validity	Setup
			Sign	Verf			
HoneyBadgerBFT [42]	$\mathcal{O}(ln^3)$	$\mathcal{O}(\log(n))$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	0	$[m, M]$	DKG
Dumbo2 [35]	$\mathcal{O}(ln^2 + \kappa n^3)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	0	$[m, M]$	HT-DKG
FIN [27]	$\mathcal{O}(ln^2 + \kappa n^3)$	$\mathcal{O}(1)$	$\mathcal{O}(\log(n))$	$\mathcal{O}(n \log(n))$	0	$[m, M]$	DKG
WaterBear [51]	$\mathcal{O}(ln^3 + exp(n))$	$\mathcal{O}(exp(n))$	0	0	0	$[m, M]$	Auth. channels
Abraham et al. [1]‖	$\mathcal{O}(ln^3 \log(\frac{\delta}{\epsilon}) + n^4)$	$\mathcal{O}(\log(\frac{\delta}{\epsilon}))$	0	0	ϵ	$[m, M]$	Auth. channels
DELPHI¶‖	$\mathcal{O}(ln^2 \frac{\delta}{\epsilon}(\log(\frac{\delta}{\epsilon} \log \frac{\delta}{\epsilon}) + \log(\lambda \log n)))$	$\mathcal{O}(\log(\frac{\delta}{\epsilon} \log \frac{\delta}{\epsilon}) + \log(\lambda \log n))$	0	0	ϵ	$[m - \delta, M + \delta]$	Auth. channels

V_h is the set of all honest inputs, l is the size of each input, $m = \min(V_\text{h}), M = \max(V_\text{h})$, $\delta = M - m$ is the range of honest inputs,ϵ is the range of outputs, and κ is the cryptographic security parameter. $l < \kappa$ in practice. **Validity**: The protocol's output o is guaranteed to be within this range. ‖ **Agreement Distance** These protocols are Approximate Agreement protocols, where honest outputs have a range at most ϵ, where $|o_i - o_j| \leq \epsilon$. ¶**Probability distribution** DELPHI requires the input range $\delta \leq \Delta$, where Δ is an upper bound. DELPHI utilizes the fact that honest inputs come from a distribution, and sets $\Delta = f(n, \lambda)$ such that $\delta \leq \Delta$ with probability negligible in λ. We report DELPHI's communication complexity in the case where honest inputs are from a Normal distribution, where $f(n, \lambda) = \mathcal{O}(\lambda \log n)$. For Lognormal and Pareto distributions, $f(n, \lambda) = \mathcal{O}(\lambda n), \mathcal{O}(2^\lambda n)$, where DELPHI increases as $\mathcal{O}(ln^2 \log \lambda n)$ and $\mathcal{O}(l\lambda n^2 \log \lambda n)$ [29].

are within ρ distance, there exists an interval where all honest nodes output 1 (since all honest nodes input 1 to that interval). Moreover, all intervals that are further than distance ρ to any honest input will have output 0.

Next, the nodes perform a weighted average among the intervals, where each interval has a representative value called *checkpoint* (e.g. the midpoint value of the interval), and the output of BINAA for each corresponding interval is the weight. We observe that only intervals that are ρ-close to an honest input can have a non-zero output, and the overall average remains ρ-close to the honest range. Moreover, the weights among honest nodes remain ϵ-close, ensuring that the weighted averages are also close.

This technique fails when honest nodes' inputs are further than ρ distance, in which case all weights are 0. We address this by running our technique over multiple levels, each with increasing values of ρ_l. The starting level $l = 0$ has $\rho_0 = \epsilon$, and higher levels l have $\rho_l = 2^l \rho_0$. We then introduce a novel multi-level weighted average scheme and design levels and their corresponding ρ_l values based on the input distribution so that the averages are close when the honest inputs are close. Finally, we ensure that levels with high ρ_l values do not contribute to the average , while ensuring that our protocol incurs $\tilde{\mathcal{O}}(n^2)$ bits of communication per round.

Data Analysis and Performance Evaluation. We evaluate DELPHI in two applications: (a) A network of oracle nodes reporting the price of Bitcoin, and (b) A distributed system of surveillance Drones detecting and locating unauthorized vehicles in an area using an object detection program. We analyze the nodes' inputs in each application by collecting data over a prolonged period and configure DELPHI based on this analysis. We indeed observe that measurements can be best explained using loggamma and gamma distributions in the oracle network and object detection settings, both of which are indeed thin-tailed distributions. We also conduct experiments in two testbeds: (a) A geo-distributed testbed on AWS for the oracle network application and (b) A distributed CPS testbed comprised of Raspberry Pi devices for Drone-based object detection. For $n = 160$, DELPHI takes $\frac{1}{3}$rd and $\frac{1}{8}$th the time taken by FIN [27], the State-of-the-art Asynchronous Convex BA protocol, and $\frac{1}{6}$th and $\frac{1}{8}$th the time taken by Abraham et al. [1], the best AAA protocol, in the AWS and CPS testbed, respectively. We also analyze the practical implications of the Validity relaxation of DELPHI.

II. PRELIMINARIES

A. System Model

We assume a system of n nodes $\mathcal{P} := \{1, \ldots, n\}$ connected by pairwise authenticated channels and an asynchronous network. We consider an adaptive adversary $\mathcal{A}$, who can corrupt $t < \frac{n}{3}$ nodes at any time during the execution of the protocol. $\mathcal{A}$ also controls the network between honest nodes and can arbitrarily delay and reorder messages but cannot drop them. We consider a node honest if it is never faulty.

Nodes in the system measure a physical state variable τ and aim to agree on a value representative of τ. A few examples of such state variables are the price of a cryptocurrency like Bitcoin, the physical location of objects like cars, and the temperature in a given area. Each node measures τ using an on-board data source, which provides the node with a value v_i, an estimate of τ. Such data sources include the price feeds provided by currency exchanges, cameras indicating the position of an object, and sensors sensing temperature. These sources have been known to have a finite accuracy and hence measure τ with an accuracy error, conventionally modeled using a probability distribution. In line with this error, we assume node i's input v_i is independently sampled from a random variable X, with a probability distribution $p(x)$. Nodes and data sources do not know any details about p. We model the inputs v_i as floating point numbers with a finite precision. We also assume $v_i \in [s, e]$, where s and e are very small and large finite numbers, respectively, defined at a system level.

B. Problem Definition

We define the Approximate Agreement for oracles.

Definition II.1. *We denote the set of honest nodes' input values by V_h. A protocol Π_{AA} for n nodes where node i inputs v_i and outputs o_i solves Approximate Agreement with ρ-relaxed validity if it satisfies the following properties.*

1) ***Termination**: Each honest node must eventually produce an output o_i.*
2) ***ρ-relaxed Min-Max Validity**: Each honest node's output must be within the ρ-relaxed interval formed by honest inputs. Let the interval formed by honest inputs is $[m, M]$, where $m = \min(V_h)$ and $M = \max(V_h)$. For every honest node $i \in \mathcal{P}$: $m - \rho \leq o_i \leq M + \rho$*
3) ***ϵ-agreement**: The outputs of any pair of honest nodes i and j are within ϵ of each other, i.e., $|o_i - o_j| < \epsilon$. ϵ is called the agreement distance.*

C. Approximate Agreement for Binary Inputs

Our starting building block BINAA is an Approximate Agreement protocol with convex validity (i.e., 0-relaxed validity) for binary inputs (every honest node has input 0 or 1), and communicating $\mathcal{O}(n^2)$ bits per round, similar to the notion of binary Proxcensus by Ghinea, Goyal and Liu-Zhang [31].

Our protocol builds upon a weaker variant of the Binary Value broadcast primitive defined in Mostefaoui, Moumen, and Raynal [43]. This primitive can be instantiated with (a straightforward adaptation of the) Crusader Agreement protocol by Abraham, Ben-David, and Yandamuri [2], with $\mathcal{O}(n^2)$ bits per round, and terminating in three rounds.

Definition II.2. *A protocol Π_{BV} for n nodes where node i inputs a value v_i and outputs a set of values B_i achieves weak Binary Value broadcast if it satisfies the following properties.*

- ***Termination**: All honest nodes eventually terminate and output a non-empty set B_i, where $|B_i| \geq 1$.*
- ***Justification**: If the output set B_i of an honest node contains value v, then v must have been the input of at least one honest node.*
- ***Weak Uniformity**: The output sets of any pair of honest nodes i and j must have a non-empty intersection.*

$$B_i \cap B_j \neq \emptyset$$

We describe the BINAA protocol in Algorithm 1 and give a brief intuition of its functioning here. BINAA proceeds in iterations, where in each iteration r, each node i participates in an instance of a BV broadcast protocol $\Pi_{BV,r}$ with an input z_i (in the initial iteration, $z_i = v_i \in \{0, 1\}$). Each iteration achieves the weak BV-broadcast primitive defined in Definition II.2.

We first show that the first iteration of the protocol satisfies all three properties specified in Definition II.2. First, the protocol satisfies Termination because honest inputs are binary, and at least one value b would have been possessed by $t + 1$ honest nodes. This ensures that b will be echoed by enough honest nodes and eventually enable honest nodes to terminate. Second, the protocol satisfies Justification because faulty nodes can produce at most t ECHO1/ECHO2 messages on a value not possessed by honest nodes. Finally, the protocol

Algorithm 1 Binary Approximate Agreement (BINAA) protocol

1: **INPUT**: $\mathcal{P}$, $b_i \in \{0, 1\}$, ϵ

▷ Parameter Setup
// r_M is the number of rounds to run and r is the current round
2: $r_M \leftarrow \log_2(\frac{1}{\epsilon})$; $r = 1$
// $E1$ and $E2$ stand for list of ECHO1 and ECHO2 messages received, B_i is the output set for weak BV broadcast
3: $E1_i[r] \leftarrow \{\}$; $E2_i[r] \leftarrow \{\}$; $B_i[r] \leftarrow \{\}$ for $r \in \{1, 2, \ldots, r_M\}$

▷ Begin protocol
4: **on** receiving $\langle$Start,ID.i, $r = 1\rangle$
5: $b_{i,1} \leftarrow b_1$
6: $E1_i[r] \leftarrow E1_i[r] \cup \{(b_{i,1}, i)\}$
7: **SendAll** $\langle$ECHO1, $b_i, r\rangle$ // Send ECHO1 message
8: **on** receiving $\langle$ECHO1, $b_{j,r}, r\rangle$ from node j
9: $E1_i[r] \leftarrow E1_i[r] \cup \{(b_{j,r}, j)\}$
10: **on** receiving ECHO1s from $t + 1$ nodes for a value b in current round r i.e. there exists b such that at least $t + 1$ $(b, k) : k \in \mathcal{P}$ values are in $E1_i[r]$
11: **SendAll** $\langle$ECHO1, $b, r\rangle$ // Bracha amplify value b
12: **on** receiving ECHO1s from $n - t$ nodes for a value b and not previously sending an ECHO2 message in current round r
13: $E2_i[r] \leftarrow E2_i[r] \cup \{(b, i)\}$
14: **SendAll** $\langle$ECHO2, $b, r\rangle$
15: **wait until** one of the following conditions is true for round r:
16: (1) Receiving ECHO1s from $n-t$ nodes for two values b_1, b_2 i.e. $\exists b_1, b_2$ such that $n - t$ $(b_1, k) : k \in \mathcal{P}$ values and $n - t$ $(b_2, k) : k \in \mathcal{P}$ values are in the set $E1_i[r]$
17: (2) Receiving ECHO2s from $n - t$ nodes for a value b i.e. $\exists b$ such that $n - t$ $(b, k) : k \in \mathcal{P}$ values are in the set $E2_i[r]$
18: **on** condition (1) being true
19: $B_i[r] \leftarrow \{b_1, b_2\}$ // End here for Weak BV Broadcast
20: $b_{i,r+1} \leftarrow \frac{b_1+b_2}{2}$
21: $r \leftarrow r + 1$ and **goto** *Line* 4 to begin new round
22: **on** condition (2) being true
23: $B_i[r] \leftarrow \{b\}$ // End here for Weak BV Broadcast
24: $b_{i,r+1} \leftarrow b$
25: $r \leftarrow r + 1$ and **goto** *Line* 4 to begin new round
26: **output** the value b_{i,r_M} after terminating r_M rounds

satisfies weak uniformity because of two main reasons - (a) Each honest node sends at most one ECHO2 message, which implies at most one value b can receive $n - t$ ECHO2s. This ensures that no pair of honest nodes satisfy condition 2 (Line 17) for two different values $b_1 \neq b_2$. (b) A node that terminates this iteration by satisfying the first condition (Line 16) must have at least one value in common with every other honest node. This is because only values $0, 1$ can receive $n - t$ ECHO1s.

Second, at the end of each iteration, each node i updates its value as the average of values in the output set $B_{i,r}$. Given the set of honest inputs is $\{v_0, v_1\}$, the set of honest output sets from Π_{BV} is either $\{\{v_0\}\}$, $\{\{v_0\}, \{v_0, v_1\}\}$, $\{\{v_1\}, \{v_0, v_1\}\}$, or $\{\{v_1\}\}$. The range of honest inputs decreases by at least $\frac{1}{2}$ at each iteration. Further, the updated values in each iteration satisfy the binary input assumption. When run for $\log \frac{1}{\epsilon}$ iterations, this protocol achieves approximate agreement with

- **Real range** δ is the real difference between maximum and minimum honest inputs $\delta = \max(V_\mathsf{h}) - \min(V_\mathsf{h})$.
- **Max range** Δ is the maximum possible difference between maximum and minimum honest inputs $\delta \leq \Delta$.
- **Separator** ρ_l is the length of each interval or difference between adjacent checkpoints in a level l.
- **Checkpoint** μ_k^l or CHECKPOINT(k, y, l) is the kth multiple of the separator ρ_l: $\mu_k^l = k\rho_l$, where k is an integer between $k \in [\frac{s}{\rho_l}, \frac{e}{\rho_l}]$. Level l has checkpoints throughout the space of honest inputs $[s, e]$, where adjacent checkpoints are separated by $\rho_l = 2^l\rho_0$.
- **Weight** ω_k^l is the weight of checkpoint μ_k^l. Nodes achieve Approximate Agreement on these values using BINAA protocol.
- **Level** LEVEL(l, ρ) or $\mathcal{L}[l]$ is the level object. It contains a list of intervals $\mathcal{C}_x^l$ at level l between endpoints s and e. $\rho_l = 2^l\rho_0$ is the distance between adjacent checkpoints.

Figure 1: Description of symbols used in DELPHI

communication complexity of $\mathcal{O}(n^2 \log^2 \frac{1}{\epsilon})$ bits.

III. DESIGN

In this section, we present the design of our protocol DELPHI. We first provide context, explain the limitations of prior solutions, and then discuss our approach.

A. Limitations of prior protocols

Prior Approximate Agreement protocols like Dolev et al. [24] and Abraham et al. [1] proceed in a round-based manner, where in each round, every node collects a set of values from other nodes, and updates its state using a Trimmed mean of this set. In these protocols, reaching Approximate Agreement requires each node to collect $2t + 1$ values in common with other honest nodes. This is for two major reasons. (a) Even with t faulty values, each honest node must ensure that its updated state is within the range of honest inputs, and (b) Even after trimming t greatest and smallest values in their sets, honest nodes must have enough values in common to reduce their range. Dolev et al. [24] achieve this condition using multicasts with $n = 5t + 1$, where a faulty node can make different honest nodes accept different values. Achieving this at $n = 3t + 1$ resilience requires restricting equivocation by faulty nodes, which requires using the Reliable Broadcast (RBC) primitive. RBC has a lower bound of $\mathcal{O}(n^2)$ bits, which results in an overall $\mathcal{O}(n^3)$ communication complexity. Therefore, we observe that this requirement of collecting $2t+1$ values in common with other nodes, which is necessary for achieving strict convex-hull Validity, is the main reason for the $\mathcal{O}(n^3)$ communication complexity of all prior Approximate Agreement protocols.

B. Our Approach

We overcome this $\mathcal{O}(n^3)$ communication bottleneck by proposing DELPHI, an Approximate Agreement protocol that does not require RBC. DELPHI uses a novel approach based on checkpoints and Binary Approximate Agreement, which achieves communication efficiency by trading off a relaxation in the Validity condition, specified in Definition II.1.

Notation. We divide the space of possible honest inputs between $[s, e]$ among checkpoints separated by a system parameter ρ. A checkpoint $\mu_k = k\rho$ represents the space of values between $[k\rho - \frac{\rho}{2}, k\rho + \frac{\rho}{2}]$, where k is the set of all integers in $(\frac{s}{\rho}, \frac{e}{\rho})$. Nodes run a BINAA protocol instance for all checkpoints in $[s, e]$ to approximately agree on a *representative weight* w_k^l for each checkpoint. We denote the corresponding pseudocode in Algorithm 2. Each level, indexed by the number l, is characterized by the difference between two consecutive checkpoints ρ_l. Further, each level comprises checkpoints in $[s, e]$, where each checkpoint is a multiple of ρ_l.

For ease of understanding, we first describe a basic, single-level version of DELPHI, explain the challenges, and then explain the multi-level version that addresses these challenges.

1) Single level Approximate Agreement: We first describe a basic version of DELPHI at a single level with distance between checkpoints ρ. Nodes run BINAA for all checkpoints in a level. An honest node inputs $v = 1$ to checkpoint $k\rho$ only if $|v_i - k\rho| \leq \rho$, and inputs 0 otherwise (Line 11 in Algorithm 2). Nodes then send out messages for checkpoints and handle incoming messages. Each node then checks whether a round r has terminated for a checkpoint. Upon terminating round r of BINAA of all checkpoints in the level, nodes initiate round $r + 1$. The nodes terminate a BINAA instance after running for r_M rounds (specified later). Upon terminating all BINAA instances, nodes calculate a weighted average of checkpoints $o_i = \frac{\sum_k (k\rho) \times w_k}{\sum_k w_k}$ (Line 18 in Algorithm 2).

Intuition. We observe that when honest inputs are clustered within a distance $\delta \leq \rho$, at least one checkpoint will have weight $w_k = 1$. This is because the Validity property of BINAA ensures that if all honest nodes input the same value to any BINAA instance $\mathcal{B}_k$, then all nodes must output that value. This property gives way to two key observations: (a) If all honest inputs v_i are within ρ distance, the weight $w_k = 1$ for at least one checkpoint μ_k, and (b) No checkpoint $\mu_k : |\mu_k - v_i| > \rho \forall i \in V_\mathsf{h}$ will have a non-zero weight. The first observation guarantees approximate agreement of the weighted average, whereas the second observation enables us to show that the final output will strictly lie between the ρ-relaxed interval of honest inputs $[\min(V_\mathsf{h}) - \rho, \max(V_\mathsf{h}) + \rho]$.

Challenges. The described approach produces a valid result only when honest nodes' inputs are within $\delta \leq \rho$ distance. When $\delta > \rho$, the weights of all checkpoints can be zero, resulting in a division by zero. Moreover, estimating the range δ in a distributed manner also costs a whopping $\mathcal{O}(n^4)$ complexity in prior protocols [1]. We explain this challenge pictorially in Fig. 2.

We assume an upper bound Δ on the range δ, and utilize it to ensure that nodes always output a well-defined weighted average. Prior works like Chakka et al. [20] also utilized this upperbound assumption to achieve BA efficiently, and practically justified this assumption [20]. Moreover, as many applications are modeled using thin-tailed probability distribu-

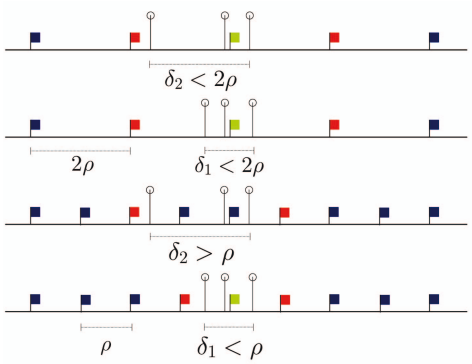


Figure 2: DELPHI **with one level**: The flags denote the checkpoints, and antennae denote honest inputs. Green checkpoints have weight $w_k = 1$ and drive agreement among nodes, and red checkpoints are outside the range of honest inputs that can have a non-zero weight and contribute to Validity relaxation. When $\delta_2 > \rho$, no green checkpoint exists, which results in agreement failure. A higher 2ρ enables nodes to reach agreement with range δ_2, but also adds the weight of farther red checkpoints when the range is δ_1, which affects Validity negatively.

tions, this assumption can be replaced with a statistical security parameter λ. In Section IV, we show that $\Delta = \mathcal{O}(\lambda\delta_{\mathsf{mean}})$ is sufficient to ensure that $\delta \leq \Delta$ with probability $1 - \mathsf{negl}(\lambda)$. We can statically set $\rho = \Delta$ to ensure the honest inputs are clustered within the ρ range, where $\delta \leq \rho$ even in the worst case.

However, this ρ causes an abnormally high Validity relaxation of Δ, even in the average case with much lesser δ. For example, as we study in Section VI, in the case of nodes agreeing on the price of a cryptocurrency, honest nodes' inputs can differ by hundreds of dollars in situations of drastic volatility, which prompts us to assume $\Delta \sim 100\$$, and set $\rho \sim 100\$$ to guarantee termination. This ρ value causes a high Validity relaxation of $100\$$, even in the average case when honest nodes input values close to the ground truth and close to each other with $\delta \sim 10\$$. This high Validity relaxation generates inaccurate results, which misrepresent the ground truth.

2) Multi-level Approximate Agreement: We address this dependence of Validity relaxation on the maximum range Δ by running BINAA across checkpoints over multiple levels. We describe this protocol, DELPHI, in Algorithm 2. We define a level l where checkpoints are separated by the level-specific separator distance $\rho_l = 2^l\rho_0$ (Line 3 in Algorithm 2). ρ_0 is the separator at the starting level 0. Each node inputs 1 to instance $\mathcal{B}_k^l$ iff its input v_i is within distance $2^l\rho_0$ from the checkpoint μ_k^l (Line 3 in Algorithm 2). Upon terminating r_{M} rounds of BINAA instances, we calculate the weighted average of checkpoints using the agreed-upon weights (Line 18 in Algorithm 2).

Intuition. We give a pictorial description of this approach in Fig. 3. Since we know that $\delta \leq \Delta$, there must exist a level $\phi \leq l_{\mathsf{M}}$ such that every level $\geq \phi$ must have at least one value μ_k^l with weight $\omega_k^l = 1$. In cases with relatively small

Algorithm 2 DELPHI protocol

1: **INPUT**: $v_i \in [s, e]$,ρ_0,Δ,ϵ

▷ Setup
// Level and round parameters
2: $l_{\mathsf{M}} \leftarrow \log_2(\frac{\Delta}{\rho_0})$; $\epsilon' \leftarrow \frac{\epsilon}{4\Delta l_{\mathsf{M}} n}$; $r_{\mathsf{M}} \leftarrow \log_2(\frac{1}{\epsilon'})$
3: **for** l in $\{0, 1, \ldots, l_{\mathsf{M}}\}$ **do**
4: **Define** $\rho_l = 2^l\rho_0$
5: **for** k in $\{i \in \mathbb{Z} : \lceil \frac{s}{\rho_l} \rceil \leq i \leq \lfloor \frac{e}{\rho_l} \rfloor\}$ **do**
// Initialize Binary AA instances for all checkpoints
6: $\mathcal{B}_k^l \leftarrow$BINAA ()
▷ BINAA phase
7: **on** receiving $\langle Start, ID.i \rangle$
8: $\mathbb{M} \leftarrow \{\}$
9: **for** l in $\{0, 1, \ldots, l_{\mathsf{M}}\}$ **do**
// Checkpoints separated by ρ_l
10: $\mu_{z-1}^l, \mu_z^l \leftarrow$Two closest checkpoints to input v_i at lev l
11: **Start** BINAA instances $\mathcal{B}_{z-1}^l$ and $\mathcal{B}_z^l$ with input $b_{i,z-1}^l = 1, b_{i,z}^l = 1$ and all other instances with input $b_i = 0$.
12: **Bundle** messages of all BINAA instances together and invoke **SendAll** on the bundled message
▷ Aggregation phase
13: **on** terminating r_{M} rounds for all BINAA instances at all levels:
// 1. Aggregate weights of checkpoints at each level. Each level has a representative value V_l and a weight w_l.
14: **for** l in $\{0, 1, \ldots, l_{\mathsf{M}}\}$ **do**
15: **if** $\exists$ a BINAA instance $\mathcal{B}_k^l$ with output > 0 **then**
16: $w_k^l \leftarrow \mathcal{B}_k^l$.OUTPUT()
17: $\mu_k^l \leftarrow k\rho_l$
18:

$$(V_l, w_l) := \left(\frac{\sum_{\lceil \frac{s}{\rho_l} \rceil \leq k \leq \lfloor \frac{e}{\rho_l} \rfloor} w_k^l \mu_k^l}{\sum_{\lceil \frac{s}{\rho_l} \rceil \leq k \leq \lfloor \frac{e}{\rho_l} \rfloor} w_k^l}, \max_{\lceil \frac{s}{\rho_l} \rceil \leq k \leq \lfloor \frac{e}{\rho_l} \rfloor} w_k^l \right)$$

19: **else**
// As weighted average is undefined when all weights are 0, we handle this case by assigning a custom weight for level l
20: $(V_l, w_l) := (v_i, \epsilon')$
// 2. Cross-level Aggregation: Aggregate weights across levels
21: $w_0' \leftarrow w_0^2$
22: **for** l in $\{1, 2, \ldots, l_{\mathsf{M}}\}$ **do**
23: $w_l' \leftarrow w_l \times |w_l - w_{l-1}|$
24: $o_i := \frac{\sum_{l=0}^{l_{\mathsf{M}}} (w_l' \times V_l)}{\sum_{l=0}^{l_{\mathsf{M}}} w_l'}$
25: **output** o_i

δ, the level ϕ is small. As the checkpoints at higher levels contribute a larger Validity relaxation to the output, we aim to minimize their contribution to the weighted average. We utilize the fact that the maximum weight of a checkpoint in a level $\max w_k^l = 1$ for any level $l \geq \phi$. We set the weight of a level $w_l' = w_l|w_l - w_{l-1}|$, where w_l is the maximum weight of a checkpoint in level l (Line 21 in Algorithm 2). This operation is analogous to a differentiation. Any checkpoint at level $l > \phi$ will not contribute to the output.

C. Optimizing Communication

DELPHI requires running the Binary Approximate Agreement protocol for all intervals between s and e. However,

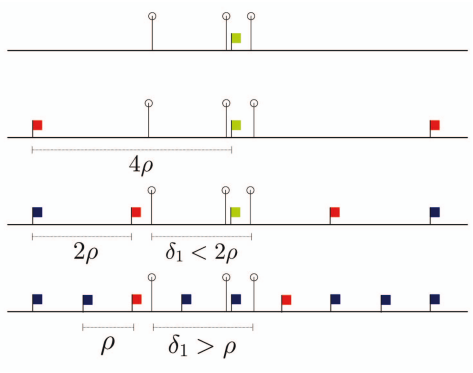


Figure 3: DELPHI **with multiple levels**: As $\delta < 2\rho$, all levels $l \geq 2$ have at least one green flag, which implies $w_l = \max_k w_k$ is 1 for levels 2,3,4. We eliminate contributions of levels 3 and 4 by using weight $w'_l = w_l|w_l - w_{l-1}|$ in the weighted average.

separately sending messages for each BINAA instance would require $(\frac{e-s}{\rho_0})n^2$ messages per round, which is infeasible in practice. We bundle messages from multiple BINAA instances and treat them as a single BINAA instance. For example, we consider a level l with honest inputs between checkpoints k_1, k_2. All honest nodes will input 0 to checkpoints from $[s, k_1]$ and $[k_2, e]$. This implies all these checkpoints require only $\mathcal{O}(n^2)$ communication per round. Further, for cases when $k_2 - k_1 = \mathcal{O}(c)$, the total communication per level is $\mathcal{O}(n^2)$ bits per round. We also perform cross-level optimization. If all honest inputs are within two consecutive checkpoints μ_k^ϕ and μ_{k+1}^ϕ at level ϕ, the BINAA data for levels ϕ to l_M can be represented with constant bits.

IV. ANALYSIS

Overview. We know that nodes approximately agree on the weight of each checkpoint, which implies each node's weights are ϵ'-close for all checkpoints at all levels. The weighted average function enables nodes to combine the weights and checkpoints by offering two key properties: (a) It forces the final output to be within checkpoints with non-zero weight outputs, and (b) As the weights are ϵ'-close, the outputs from the function are ϵ-close, which guarantees approximate agreement with Validity. However, this approach also has a caveat: it becomes undefined when the sum of weights is zero. Hence, we first ensure that the denominator in our weighted average is always greater than $\frac{1}{2}$. Next, we utilize the properties offered by the weighted average scheme to prove approximate agreement and Validity of DELPHI.

A. Termination

We discuss DELPHI's termination. We show that the weighted average in Line 24 in Algorithm 2 always has a non-zero sum of weights.

Theorem IV.1. ***Termination****: Given that the* BINAA *protocol provides Safety, Termination, and Validity, every honest node terminates* DELPHI *and outputs a finite, defined value.*

Proof. We know that every honest node terminates all BINAA instances corresponding to all checkpoints in levels $l = \{0, \ldots, l_\mathsf{M}\}$. Consider the denominator in Line 24 in Algorithm 2. Every honest node starts BINAA for all intervals in every level until l_M. From the Termination property of BINAA, every honest node terminates r_M rounds for all checkpoints until level l_M.

$$S = w_0^2 + \sum_{l=1}^{l_\mathsf{M}} w_l \times |w_l - w_{l-1}|$$

$$\implies S \geq \frac{1}{2}(2w_0^2 + \sum_{l=1}^{l_\mathsf{M}} 2w_l^2 - 2w_l w_{l-1})$$

$$\implies S \geq \frac{1}{2}(w_0^2 + w_{l_\mathsf{M}}^2 + \sum_{l=1}^{l_\mathsf{M}} (w_l - w_{l-1})^2) \geq \frac{w_{l_\mathsf{M}}^2}{2}$$

As honest inputs have a maximum difference $\delta \leq \Delta$, there must exist level $l \leq l_\mathsf{M}$ such that $w_l = 1$, which ensures that the sum of weights is $\geq \frac{1}{2}$. □

B. Validity

The Validity offered by DELPHI depends on two factors: (i) The separator at level 0 ρ_0, and (ii) The rate of increase of interval length across levels. These factors decide the number of levels in DELPHI and therefore influence the Validity.

Lemma IV.2. *Given that the* BINAA *protocol satisfies Agreement, Termination, and Validity, an honest node i's representative weight for level l $w'_{l,i}$ in Line 21 in Algorithm 2 is at most $5\epsilon'$ away from $w'_{l,j} \forall l \in \{0, \ldots, l_\mathsf{M}\} \forall j \in \mathcal{P}$. Further, for all levels $l > \lceil \log(\frac{\delta}{\rho_0}) \rceil$, the weight $w'_{l,i} = 0 \forall i \in \mathcal{P}$.*

$$|w'_{l,i} - w'_{l,j}| \leq 5\epsilon', \forall l \in \{0, \ldots, \lceil \log(\frac{\delta}{\rho_0}) \rceil\} \forall \{i, j\} \in \mathcal{P}$$

$$w'_{l,i} = 0 \forall l > \lceil \log(\frac{\delta}{\rho_0}) \rceil, \forall i \in \mathcal{P}$$

Proof. The weight $w_{l,i}$ in $w'_{l,i}$ is calculated as $w_{l,i} = \max\{\omega_k^l : \mathcal{B}_x^l$. From agreement of BINAA, we know $|w_{l,i} - w_{l,j}| \leq \epsilon' \forall \{i, j\} \in V_\mathsf{h}$, and $|w_{l,i} - w_{l-1,i}| - |w_{l,j} - w_{l-1,j}| \leq 2\epsilon'$. .

$$w'_{l,i} - w'_{l,j} \leq (w_{l,j} + \epsilon') \times (|w_{l,j} - w_{l-1,j}| + 2\epsilon') - w_{l,j}|w_{l,j} - w_{l-1,j}|$$

$$w'_{l,i} - w'_{l,j} \leq \epsilon'(2w_{l,j} + |w_{l,j} - w_{l-1,j}|) + 2\epsilon'^2$$

$$\implies w'_{l,i} - w'_{l,j} \leq 5\epsilon'$$

For level $\phi = \lceil \log(\frac{\delta}{\rho_0}) \rceil$, the separation between checkpoints $\rho_l \geq \delta$, implying $w_l = 1 \forall l \geq \phi, w'_l = 0 \forall l \geq \phi$. This implies there exists at least one checkpoint μ_k^ϕ that is within a ρ_l distance from all honest inputs v_i. From the Validity of BINAA, $\omega_k^\phi = 1 \forall i \in \mathcal{P}, \implies w_{\phi,i} = 1$. This statement is also true for all levels $> \phi$. Therefore, the weight $w'_{l,i} = 0 \forall l > \phi$. □

Theorem IV.3. *Given that the* BINAA *protocol satisfies Termination, and Validity, honest nodes' output values o_i from* DELPHI *are always within the following interval; i.e., $o_i \in [\min(V_{\mathsf{h}}) - \max(\rho_0, \delta), \max(V_{\mathsf{h}}) + \max(\rho_0, \delta)]$*

Proof. From Lemma IV.2, every level $l > \phi$, where $\phi = \lceil \log(\frac{\delta}{\rho_0}) \rceil$, has weight $w'_{l,i} = 0$.

First, we examine all checkpoints μ^l_k that can have a positive weight in Line 18 in Algorithm 2 for levels $l < \phi$. We know that only checkpoints within a ρ_l distance from an honest input v_i can have a positive weight. Therefore, no $\mu^l_k < \min(V_{\mathsf{h}}) - \rho_l$ and $\mu^l_k > \max(V_{\mathsf{h}}) + \rho_l$ can have a weight $\omega^l_k > 0$. This implies that the level's weight $w_{l,i} \in (\min(V_{\mathsf{h}}) - \rho_l, \max(V_{\mathsf{h}}) + \rho_l) \forall l < \phi$. The farthest point which can have a positive weight in o_i until level $l < \phi$ is $\min(V_{\mathsf{h}}) - \delta$ and $\max(V_{\mathsf{h}}) + \delta$, because $\rho_l < \delta, \forall l < \phi$.

Second, we examine checkpoints μ^l_k at $l = \phi$, where $\rho_\phi \in [\delta, 2\delta)$. There exists at least one μ^l_k such that $|\mu^l_k - v_i| \leq \rho_\phi, \forall i \in \mathcal{P}$, which implies $w^\phi_k = 1$. The only other checkpoints that can have a non-zero weight are μ^ϕ_{k-1} and μ^ϕ_{k+1}, which ensures that level ϕ's representative value w_ϕ must be within $[\frac{\mu^\phi_{k-1}+\mu^\phi_k}{2}, \frac{\mu^\phi_k+\mu^\phi_{k+1}}{2}]$. As there must exist honest nodes i, j such that $|\mu^\phi_{k-1} - v_i| < 2\delta$ and $|\mu^\phi_{k+1} - v_j| < 2\delta$, the relaxation can be at most δ, which proves the theorem. □

C. Agreement

We prove that DELPHI achieves ϵ-agreement.

Theorem IV.4. *Given that the* BINAA *protocol satisfies Agreement, Termination, and Validity, outputs of honest nodes o_i after terminating* DELPHI *must be within ϵ distance of each other; i.e., $|o_i - o_j| \leq \epsilon \forall \{i, j\} \in \mathcal{P}$*

Proof. We consider the difference between the weighted averages of two honest nodes $O = o_i - o_j$.

$$O = \left(\frac{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,i} V_{l,i}}{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,i}} \right) - \left(\frac{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,j} V_{l,j}}{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,j}} \right)$$

We assume $w'_{l,j} = w'_{l,i} + c_l$. The equation then transforms to the following.

$$O = \left(\frac{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,i}(V_{l,i} - V_{l,j}) + c_l(V_{avg,i} - V_{l,j})}{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,j}} \right)$$

The term $V_{avg,i} = \frac{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,i} V_{l,i}}{\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,i}}$. From Theorem IV.1, we know that $\sum_{l=0}^{l_{\mathsf{M}}} w'_{l,j} \geq \frac{1}{2}$. Therefore, we rewrite the equation as the sum of the following two terms and then examine each term individually.

$$O \leq \sum_{l=0}^{l_{\mathsf{M}}} \left(w'_{l,i} |V_{l,i} - V_{l,j}| + c_l |V_{avg,i} - V_{l,j}| \right)$$

We split the above sum into two different terms:

$$O_1 = \sum_{l=0}^{l_{\mathsf{M}}} \left(w'_{l,i} |V_{l,i} - V_{l,j}| \right)$$

$$O_2 = \sum_{l=0}^{l_{\mathsf{M}}} \left(c_l |V_{avg,i} - V_{l,j}| \right)$$

We analyze level l's contribution in O_1 denoted by $O_{1,l,i} = w'_{l,i} |V_{l,i} - V_{l,j}|$. Honest nodes can input 1 to at most $\frac{\Delta}{\rho_l}$ checkpoints at level l, bounded by the total number of honest nodes $n - t$. We denote the parameter $|U|_{\max} = \min(\frac{2\Delta}{\rho_0}, n - t)$, where the weighted average can have a non-zero weight for at most $|U|_{\max}$ checkpoints. We refer to these intervals using the terms $\omega_{k,i}$ and μ_k, where $k \in \{1, \ldots, n-t\}$. Additionally, $w'_{l,i} \leq w_{l,i}$ based on Fig. 3. We rewrite the above equation with these variables.

$$O_{1,l,i} \leq w_{l,i} \left| \frac{\sum_{k=1}^{|U|_{\max}} \omega_{k,i} \mu_k}{\sum_{k=1}^{n-t} \omega_{k,i}} - \frac{\sum_{k=1}^{|U|_{\max}} \omega_{k,j} \mu_k}{\sum_{k=1}^{n-t} \omega_{k,j}} \right|$$

We write $\omega_{k,j} = \omega_{k,i} + c_k$.

$$O_{1,l,i} \leq w_{l,i} \left| \frac{\begin{array}{l}(\sum_{k=1}^{|U|_{\max}} \omega_{k,i} \mu_k)(\sum_{k=1}^{|U|_{\max}} \omega_{k,i} + c_k) \\ - (\sum_{k=1}^{|U|_{\max}} (\omega_{k,i} + c_i) \mu_k)(\sum_{k=1}^{|U|_{\max}} (\omega_{k,i}))\end{array}}{(\sum_{k=1}^{|U|_{\max}} \omega_{k,i})(\sum_{k=1}^{|U|_{\max}} \omega_{k,j})} \right|$$

$$\implies O_{1,l,i} \leq w_{l,i} \left| \frac{\sum_{k=1}^{|U|_{\max}} c_k (V_{l,i} - \mu_k)}{\sum_{k=1}^{|U|_{\max}} \omega_{k,j}} \right|$$

$$\implies O_{1,l,i} \leq \frac{|U|_{\max} w_{l,i} \epsilon' \delta}{\sum_{k=1}^{|U|_{\max}} \omega_{k,j}}$$

As $w_{l,i} = \max\{\omega_{k,i} : k \in \{1, \ldots, |U|_{\max}\}\}$, the term $\frac{w_{l,i}}{\sum_{k=1}^{|U|_{\max}} \omega_{k,j}} \leq 1 + \frac{\epsilon'}{w_{l,j}} \leq 2$. Therefore, the term O_1 is upper bounded as follows: $O_1 \leq 2\Delta l_{\mathsf{M}} \epsilon' |U|_{\max}$.

Now, we examine the second term $O_2 = \sum_{l=0}^{l_{\mathsf{M}}} c_l |V_{avg,i} - V_{l,j}|$. From Lemma IV.2, we know that the level weights $|w_{l,i} - w_{l,j}| \leq \epsilon' \forall l \in \mathbb{L}$. Further, after a level $l > \log_2(\frac{\delta}{\rho_0}) + 1$, we know $w_{l,i} = 0$ and $c_l = 0$. Finally, from the Validity of DELPHI in Theorem IV.3, $w_{avg,i}$ always lies at a distance at most δ from other honest inputs v_i. Finally, the difference $|V_{avg,i} - V_{l,j}| < 3\delta$. This gives $O_2 \leq 6 l_{\mathsf{M}} \Delta \epsilon'$. Overall,

$$O \leq 2\Delta l_{\mathsf{M}} \epsilon' (|U|_{\max} + 3) \leq 4\Delta l_{\mathsf{M}} \epsilon' |U|_{\max}$$

The parameters $|U|_{\max} = \min(\frac{2\Delta}{\rho_0}, n), l_{\mathsf{M}} = \log(\frac{\Delta}{\rho_0})$.

Therefore, DELPHI achieves Approximate Agreement when run for $\mathcal{O}(\frac{\Delta l_{\mathsf{M}} |U|_{\max}}{\epsilon})$ rounds. □

D. Complexity Analysis

We analyze the communication complexity of DELPHI. A level l has at most $\frac{\delta}{\rho_l}$ checkpoints μ^l_k to which honest nodes can input 1. Based on the optimization we described in Section III-C, level l requires $\mathcal{O}(n^2 \min(\frac{\delta}{\rho_l}, n))$ bits of communication per round. Overall, DELPHI uses $\mathcal{O}(n^2 \min(\frac{\delta}{\rho_0}, n l_{\mathsf{M}}))$ bits of communication per

round. Combined over r_{M} rounds, this complexity totals to $\mathcal{O}(n^2 \min(\frac{\delta}{\rho_0}, nl_{\mathsf{M}}) \log(\frac{\Delta l_{\mathsf{M}} \min(\frac{\Delta}{\rho_0}, n)}{\epsilon}))$ bits.

Setting parameters for DELPHI. DELPHI requires setting ρ_0 and Δ as global system parameters. For the scope of this paper, we statically set $\rho_0 = \epsilon$, which ensures the minimum Validity relaxation is ϵ. We set the parameter Δ using the assumption that honest nodes' inputs come from a thin-tailed probability distribution. We theoretically analyze DELPHI 's communication complexity (in bits) under different distributions. We discuss the communication complexity under different Δ, ϵ, and δ in Table II. Empirically, we collect and analyze data from the underlying applications and statistically infer the maximum possible difference between honest inputs. In the absence of such historical data from the application, Δ can be set based on domain knowledge. For example, in a use case where nodes want to agree on the price of Bitcoin, we can set Δ to be the maximum possible price observed so far - for example, $\Delta = 100000\$$. This way of setting Δ degrades the performance of DELPHI by increasing its round complexity.

Analysis under probability distributions. Many applications model real-world events using probability distributions. This analysis enables us to estimate the parameters in use and accordingly benchmark DELPHI's performance. We consider two types of distributions: (a) Thin-tailed distributions like Normal, Gamma, and Lognormal, and (b) Fat-tailed distributions like Pareto. We use the extreme value theory and order statistics to conduct this analysis. We calculate Δ based on a parameter λ, where we ensure that the range $\delta \leq \Delta$ with probability $1 - \mathsf{negl}(\lambda)$.

Thin-tailed distributions are used in error modeling in practice. For instance, Normal, Lognormal, and Gamma distributions have been used in sensor noise modeling and modeling sizes of insurance claims, respectively [12], [29]. The extreme value theory suggests that the difference δ between the maximum and minimum of n randomly drawn samples from both Normal and Gamma distributions follows a Gumbel distribution, with CDF of $F(x) = e^{-e^{-x}}$ [29, page 155], and the mean of the distribution of δ grows as $\mathcal{O}(\log(n))$ for Normal and Gamma distributions. We calculate Δ using λ and n in the CDF. Using this estimation, we observe that $\Delta = \mathcal{O}(\lambda \log(n))$. With this Δ value, DELPHI's communication complexity is $\mathcal{O}(n^2 \frac{\delta}{\epsilon}(\log(\frac{\delta}{\epsilon} \log \frac{\delta}{\epsilon}) + \log(\lambda \log n)))$.

In the case of honest samples coming from distributions like Pareto or Loggamma distributions with shape parameter α, which have relatively fatter tails, the range of honest inputs follows a Frechet distribution, with CDF $F(x) = e^{-x^{-\alpha}}$ [29, page 153]. The mean of this distribution grows as $\mathcal{O}(n^{\frac{1}{\alpha}})$ In this case, we observe that $\Delta = e^{\lambda} n^{\frac{1}{\alpha}}$, which gives DELPHI's communication to be $\mathcal{O}(n^2 \frac{\delta}{\epsilon}(\log(\frac{\delta}{\epsilon} \log \frac{\delta}{\epsilon}) + \frac{\lambda}{\alpha} \log n)))$ bits.

V. APPLICATION

Oracle networks play an important role in the web3 ecosystem. They provide trusted information about real-world events to blockchains, which utilize it to execute logic on smart contracts. Oracle networks empower multiple applications in the DeFi space, like provisioning loans and trading futures, by providing reliable market data about stocks and cryptocurrencies [37]. Oracle nodes also play an important role in decentralized insurance applications like crop insurance, providing metrics about critical variables like forecasted rainfall and potential drought [37]. In these cases, the network must provide values representative of the real-world event, even in the presence of malicious nodes.

Table II: Communication and round complexity of DELPHI under input conditions

Condition Δ	Condition δ	Communication	Rounds
$\mathcal{O}(\epsilon)$	$\mathcal{O}(\epsilon)$	$\mathcal{O}(n^2 \log(\frac{\delta}{\epsilon}))$	$\mathcal{O}(\log(\frac{\delta}{\epsilon}))$
$\mathcal{O}(f(n)\epsilon)$	$\mathcal{O}(\epsilon)$	$\mathcal{O}(n^2(\log(\frac{n\Delta}{\epsilon}) + \log\log(f(n))))$	$\mathcal{O}(\log(\frac{n\Delta}{\epsilon}) + \log\log(f(n)))$
	$\mathcal{O}(\Delta)$	$\mathcal{O}(n^3 \log(f(n))(\log(\frac{n\Delta}{\epsilon}) + \log\log(f(n))))$	$\mathcal{O}(\log(\frac{n\Delta}{\epsilon}) + \log\log(f(n)))$

Δ, ϵ are statically set parameters, as specified in Algorithm 2, whereas range $\delta \leq \Delta$ is a runtime parameter that changes with varying inputs. $f(n)$ is any function growing faster than n. Δ is a constant measured based on the underlying data distribution, whereas ϵ is statically set at a system level. Under Δ like $\Delta = poly(n)\epsilon$, the $\log\log(n)$ term becomes $\log(polylog(n))$.

An oracle network must provide an attested value within the range of honest nodes' inputs to the blockchain, which verifies and includes it in the ledger. Chakka et al. [20] defined this as the Distributed Oracle Agreement (DORA) problem, where nodes must agree on a value within the range of honest inputs, assisted by the external blockchain modeled as an SMR channel. A strawman solution to solving DORA is for each honest oracle to broadcast a signature on its input v_i, collect $n-t$ other signed values, submit this list of $n-t$ values to the SMR channel, wait for the first list in the SMR channel, and output the median of this list. The properties of SMR ensure that all honest oracles see the same first list and reach an agreement. However, this approach requires the SMR channel to verify $\mathcal{O}(n)$ signatures and include a message of size $\mathcal{O}(n\kappa)$ in the ledger, which makes it inefficient and expensive.

Prior approaches like Chainlink's reporting protocol [16] and Chakka et al. [20] generate an aggregated BLS signature of size $\mathcal{O}(n + \kappa)$ on a single value within the range of honest inputs[1]. In Chainlink, nodes use partially synchronous convex BA to reach an agreement and then create a signature on the output [16]. Chakka et al. builds on the described strawman approach by adding a round to generate a succinct signature [20], and uses the SMR channel for agreement. However, both approaches are computationally expensive with $\mathcal{O}(n^2)$ signature verifications per node per attested value (while achieving adaptive security). Refer to Table III for a detailed comparison between these approaches.

We solve the DORA problem in a computationally efficient manner using DELPHI. We introduce an additional step after

[1]Message size can be reduced to $\mathcal{O}(\kappa)$ with threshold BLS signatures, but require a threshold setup

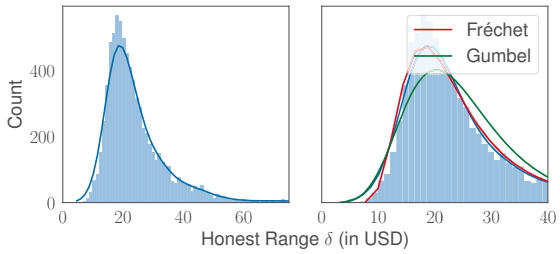


Figure 4: **Bitcoin price range histogram**: We plot a histogram of observed $\delta = |\max(V_h) - \min(V_h)|$ in US Dollars on the x-axis and the number of times the δ values in the bin appeared in the two weeks on the y-axis. We also fit different probability distributions and observe that Fréchet and Gumbel distributions, the two distributions used to model extreme order values, are the closest fit, with Fréchet being the better fit.

reaching Approximate Agreement, where nodes round off their input to the closest integer multiple of ϵ. Next, they broadcast a signature on their rounded values, wait for $t+1$ signatures on a given value, and aggregate them to form a succinct signed message.

Intuition. After rounding, honest nodes' outputs must be within at most two adjacent ϵ-checkpoints, which implies at least $t+1$ signatures must be broadcasted for at least one checkpoint. Further, no other value can receive $\geq t+1$ signatures. Therefore, DELPHI solves the DORA problem without using any additional computation than what is required to send the output to the SMR channel. The rounding operation incurs a further ϵ relaxation to the Validity.

VI. EVALUATION

We evaluate DELPHI in the context of two applications: (a) an oracle network reporting the price of a cryptocurrency, and (b) a group of surveillance drones agreeing on the location of an object. We analyze the input data in each application and accordingly configure DELPHI to maximize efficiency while ensuring security.

A. Oracle Network

In the first application, we consider an oracle network that reports the trading price of Bitcoin, a prominent cryptocurrency, in US Dollars. We configure the network to produce one price report every minute. Nodes in the Oracle network aim to agree on the current price of Bitcoin. Once a minute, each node measures the trading price of Bitcoin by querying one or a set of prominent cryptocurrency exchanges and computing the median of responses. Nodes then input this price, v_i, into an instance of DELPHI protocol and report the output as Bitcoin's trading price for that minute.

Range analysis. We configure DELPHI based on a comprehensive analysis of the difference between honest inputs or range $\delta = \max(V_h) - \min(V_h)$, where V_h is the set of inputs of honest nodes. We collected price feeds of Bitcoin from 10 prominent exchanges: `Binance`, `Coinbase`, `Crypto.com`, `Gate.io`, `Huobi`, `Mexc`, `Poloniex`, `Bybit`, `Kucoin`, and `Kraken`, for a two-week timespan from July 7th to July 21st, 2023, at a granularity of one reading every minute. We then calculated the maximum difference between the prices reported by these exchanges each minute, giving us one value per minute for two weeks. This value represents the difference between honest nodes' inputs or δ because each node queries at least one exchange. We then plot a histogram of all these δ values based on their frequency of occurrence in Fig. 4.

We observe that beyond $\delta = 30\$$, higher δ values become increasingly improbable. The δ values are below $100\$$ for 99.2% of the time and below $300\$$ for 100%. We also fit various probability distributions for this data and observed the Frechet distribution with $\alpha = 4.41$ and a scale factor of 29.3 to be the best fit for this data, implying the underlying distribution of inputs is a Loggamma distribution. We configure DELPHI's parameters using these probability distribution parameters. We derive maximum range $\Delta = 2000\$$ such that the range $\delta > \Delta$ only with probability $p = 1 - 10^{-10}$, which guarantees a statistical security of $\lambda = 30$ bits. In this context and under these distribution parameters, assuming we run one instance of DELPHI every minute, this assumption gets broken once every 2000 years. We set $\rho_0 = \epsilon = 2\$$.

B. Distributed CPS

In the second application, we consider a group of drones flying in an area, aiming to detect and localize objects. Such systems have gained increased prominence in light of recent advances in autonomous and semi-autonomous CPS [5], [8]. The drones are equipped with cameras to take photos of the area and are guided by the GPS navigation system. Each drone detects cars in the area using a deep learning-based object detection program like EfficientDet [45]. This program takes an image as input and outputs a rectangular bounding box of relative coordinates in the image containing the object. Each drone uses the center of this bounding box and its own geo-location provided by the GPS to estimate the geo-location of the target object. Given that the bounding box coordinates measured by drone i is $\vec{L_{\mathsf{BB,i}}}$, its height is hm, and the drone's 2D location from GPS is $\vec{L_{\mathsf{GPS,i}}}$, the estimate of the target's 2D location $\vec{L_{\mathsf{T,i}}}$ is calculated as $\vec{L_{\mathsf{T,i}}} = \vec{L_{\mathsf{BB,i}}} + \vec{L_{\mathsf{GPS,i}}}$. The drones use this estimated location as an input to DELPHI protocol to agree on the geo-location of each target in the area. As input $\vec{L_{\mathsf{T,i}}} = (x, y)$ is a 2D vector, drones use two instances of DELPHI to agree on each coordinate, $\vec{L_{\mathsf{T,i}}}.x$, $\vec{L_{\mathsf{T,i}}}.y$, individually.

Range Analysis. We analyze the coordinate-wise ranges $\delta_x = |\max(\vec{L_{\mathsf{T,i}}}.x) - \min(\vec{L_{\mathsf{T,j}}}.x)|$ and $\delta_y = |\max(\vec{L_{\mathsf{T,i}}}.y) - \min(\vec{L_{\mathsf{T,j}}}.x)|$ by first analyzing the distribution of the error distance $\vec{d_i} = |\vec{L_{\mathsf{T,i}}} - \vec{L_{\mathsf{GT}}}|$. $\vec{d_i}$ quantifies the accuracy of the estimate $\vec{L_{\mathsf{T,i}}}$ with the true location $\vec{L_{\mathsf{GT}}}$ of the object. We then use extreme value theory on this distribution to estimate the ranges δ_x,δ_y. We consider cars to be the objects that need to be localized. We assume the drones are deployed at height $h = 45$m or 150ft. The vectors $\vec{L_{\mathsf{BB,i}}}$ and $\vec{L_{\mathsf{GPS,i}}}$ both contribute to the parameter $\vec{d_i}$.

Table III: Comparison of relevant oracle reporting protocols

Protocol	Network	Communication Complexity (in bits)	Adaptively Secure?	Computation Sign	Computation Verf	Rounds	Validity
Chainlink [16]	p sync.	$\mathcal{O}(ln^3 + \kappa n^3)$	✗	$\mathcal{O}(1)$	$\mathcal{O}(n)$	4	$[m, M]$
DORA [20]*	async	$\mathcal{O}(ln^2 + \kappa n^2)$	✗	$\mathcal{O}(1)$	$\mathcal{O}(n)$	3	$[m, M]$
DELPHI*	async	$\mathcal{O}(ln^2 \frac{\delta}{\epsilon}(\log(\frac{\delta}{\epsilon} \log \frac{\delta}{\epsilon}) + \log(\lambda \log n)))$	✓	0	0	$\mathcal{O}(\log(\frac{\delta}{\epsilon} \log \frac{\delta}{\epsilon}) + \log(\lambda \log n))$	$[m - \delta - \epsilon, M + \delta + \epsilon]$

V_h is the set of all honest inputs, l is the size of each input, $m = \min(V_\text{h})$, $M = \max(V_\text{h})$, $\delta = M - m$ is the range of honest inputs, and κ is the cryptographic security parameter. $l < \kappa$ in practice. **SMR Channel** In all protocols, oracles produce an attested output and submit it to the blockchain for consumption by smart contracts [16, Section 5.4]. This step is common for all protocols, so we do not count its cost in this table. **Adaptive Security** Chainlink [16] and DORA [20] can be made adaptively secure at the expense of $\mathcal{O}(n)$ factor increase in communication and computation complexities. Chainlink also incurs a $\mathcal{O}(n)$ increase in round complexity. *__Agreement__ These protocols can produce multiple outputs with the specified Validity condition. DORA [20] can produce at least $\mathcal{O}(n)$ possible outputs, whereas DELPHI can produce at most two possible outputs. The external blockchain orders them and consumes the first output.

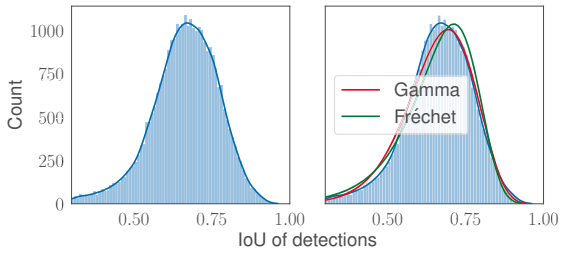


Figure 5: **IoU histogram for Drone-based object detection**: We plot the IoU values of detections output by the detection program and analyze their incidence by bucketing them into bins. These values follow a Gamma distribution, which has a thin tail.

Intersection-over-Union(IoU) analysis. We empirically estimate the error induced by the object detector in $\vec{L_{\text{BB,i}}}$ by measuring and plotting the Intersection-over-Union(IoU) I_i of the predictions. The IoU measures the overlap between the detector's output bounding box and the ground truth bounding box of the object. A higher IoU I_i implies a more accurate detection. We characterize this IoU distribution by first training a model of EfficientDet [45] on a training dataset consisting of over 100000 cars [5], [19], [25] and then testing it on a test dataset with 80000 cars. We then plot a histogram of all these IoU values based on their frequency of occurrence in Fig. 5. We observe that EfficientDet produces detections with IoU following a Gamma distribution with mean $I_{\text{mean},i} = 0.87$. Further, EfficientDet outputs a detection with $I_i < 0.6$ IoU only in 0.37% of the cases.

We translate the IoU I_i into the error distance vector $\vec{d_i}$. The x and y coordinates of this vector are bounded by $\vec{d_i}.x, \vec{d_i}.y \leq (1 - I_i) \times l_{\text{diag}}$, where l_{diag} is the length of the diagonal of the ground truth bounding box. We calculate l_{diag} assuming by using the standard dimensions of 5m length and 2m width of a car. This formula gives us the distribution of the vector $\vec{d_i} = (5.3 \times (1 - I_i), 5.3 \times (1 - I_i))$, with a mean of 0.7m.

For $\vec{L_{\text{GPS}}}$, the US FAA report on GPS accuracy documents the horizontal location error induced by GPS over 417 million samples [15]. The accuracy is within 5m 99.99% percent of the time, with an average accuracy of 1.3m.

Distribution fitting. Overall, the parameters $\vec{d_i}.x$,$\vec{d_i}.x$ have an expected value of 2m and do not exceed $d = 10.5$m 99.99% of the time. However, analyzing the combined probability distribution is challenging because both $\vec{L_{\text{GPS}}}$ and $\vec{L_{\text{BB}}}$ are from different distributions. We perform asymptotic analysis by upper bounding the $\vec{L_{\text{GPS}}}$ distribution using a Gamma distribution of the same scale as $\vec{L_{\text{BB}}}$. This approximation allows us to combine $\vec{L_{\text{GPS}}}$ and $\vec{L_{\text{BB}}}$ into a Gamma distribution with $scale = 0.18$ and $shape = 30.77$. For large n, extreme value theory suggests that the range of n independently drawn inputs from a Gamma distribution is a Gumbel distribution [29, page 155]. From the analysis in Section IV-D, a maximum range $\Delta = \mathcal{O}(\lambda \delta_{\text{mean}})$. In this context, we set $\rho_0 = \epsilon = 0.5$m and $\Delta = 50$m.

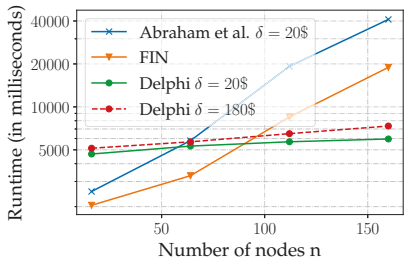
C. Implementation and Testbed Details

We implement DELPHI in Rust with the `tokio` library as our asynchronous runtime [6] [2]. We use Hash-based Message Authentication Codes (HMAC) with the `SHA256` Hash function and shared symmetric keys to implement authenticated channels. We evaluate both protocols in two testbeds - (a) A geo-distributed testbed on AWS for the oracle network application and (b) A distributed testbed on Raspberry Pi devices for the object detection application.

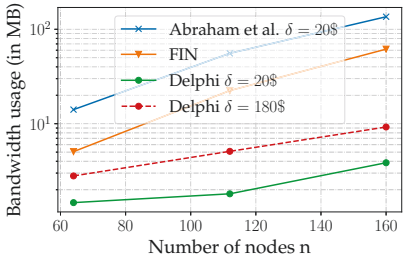
AWS testbed. We evaluate DELPHI on Amazon Web Services (AWS) with varying nodes $n = 16, 64, 112$, and 160. We run both protocols on `t2.micro` nodes, each with 1 CPU core and 2GB RAM. We create a geo-distributed testbed of n nodes to simulate execution over the internet. We distribute the nodes equally across 8 regions: N. Virginia, Ohio, N. California, Oregon, Canada, Ireland, Singapore, and Tokyo.

Embedded Device Testbed. We set up an embedded testbed consisting of 15 Raspberry Pi 4-B devices connected with the help of a network switch functioning as a router. These devices are used in most commercially available Drones and other CPS. Each device has 2 GB of RAM and 4 CPU cores. We emulate a large network of drones by running multiple processes of DELPHI on each device. We evaluate both protocols with varying nodes $n = 43, 85, 127, 169$.

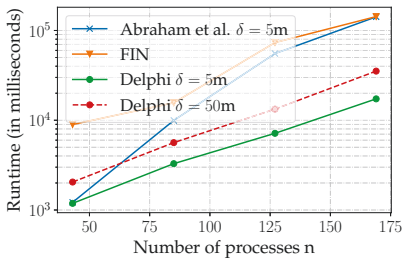
[2] Available at https://github.com/akhilsb/delphi-rs



(a) **Runtime vs n on AWS**: We report the runtime of nodes reaching Approximate Agreement on the price of Bitcoin. δ is the range of honest inputs in US Dollars. DELPHI's config is $\rho_0 = 10\$, \Delta = 2000\$, \epsilon = 2\$$. DELPHI's runtime increases much slowly with n than FIN and Abraham et al., and does not vary much with range δ.



(b) **Network bandwidth vs n on AWS**: We report the amount of data used by each protocol to reach an agreement on the price of Bitcoin. DELPHI's config is $\rho_0 = \epsilon = 2\$, \Delta = 2000\$$. DELPHI's bandwidth usage is an order of magnitude lesser than FIN and Abraham et al. and also grows at a slower pace.



(c) **Runtime vs n on Embedded testbed**: We report the time taken to reach Approximate Agreement on the location of a car seen by a group of drones. δ is the range of honest inputs in meters. DELPHI's config is $\Delta = 50m, \rho_0 = \epsilon = 50cm$. DELPHI terminates an order faster than FIN and Abraham et al. In contrast with AWS, a higher range δ takes more time to terminate in this testbed.

Figure 6: Scalability results of DELPHI

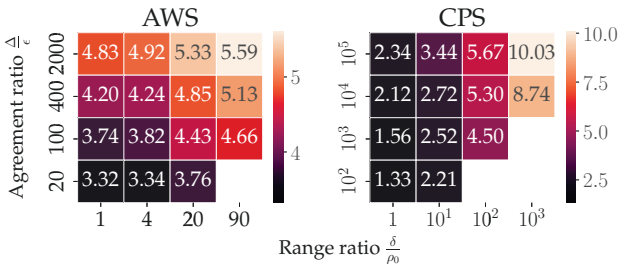


Figure 7: **Runtime patterns of DELPHI on AWS and CPS**: We study the runtime (in seconds) of DELPHI with varying agreement ratio $\frac{\Delta}{\epsilon}$ which controls the round complexity, and range ratio $\frac{\delta}{\rho_0}$, which controls the per-round communication. On AWS, the round complexity substantially influences the runtime compared to per-round communication. Whereas in CPS, we see the opposite pattern where the per-round volume of communication has more influence over runtime.

Comparison to prior works. We compare DELPHI to FIN [27], the State-of-the-Art ACS protocol, and Abraham et al. [1], the current best approximate agreement protocol. We also implement FIN and Abraham et al. in Rust [6].

D. Results

We run the three protocols - DELPHI, Abraham et al., and FIN - in the described applications, each in their respective testbed. We configure DELPHI according to the calculated parameters for each application. We run the oracle network in the geo-distributed AWS testbed and the drone-based object detection in the CPS testbed. We measure the runtime and consumed network bandwidth of all three protocols. We evaluate DELPHI under two δ values: $\delta = 20\$, 180\$$ in the oracle network, and $\delta = 5m, 50m$ in the CPS testbed to measure the average case and worst case runtime of DELPHI. We plot our findings in Fig. 6.

Scalability Results. We plot and compare the runtimes of the protocols in the oracle network in Fig. 6a. DELPHI takes only $\frac{1}{3}$rd and $\frac{1}{6}$th the time taken by FIN and Abraham et al. at $n = 160$, which demonstrates its scalability. The reason for DELPHI's higher runtime at small n is the high round complexity of DELPHI coupled with the round trip time of a geo-distributed network. However, DELPHI's runtime scales much better mainly because of its computational efficiency and practical communication complexity. Even at a higher δ value, DELPHI considerably outperforms both prior works because of its low message complexity. We also plot the network bandwidth consumed by each protocol in Fig. 6b. DELPHI consumes an order of magnitude lower bandwidth in both cases of δ than FIN.

We also plot the runtime of the protocols in the drone-based object detection application in Fig. 6c. The devices in this testbed have lower computing power than the AWS testbed, increasing the weight of a protocol's computational efficiency. At $n = 169$ processes, DELPHI's runtime is $\frac{1}{8}$th that of FIN and Abraham et al. in the average case and the worst case of $\delta = 5, 50$m, respectively. Unlike in the AWS testbed, where the round complexity played an important role at low n, DELPHI outperforms FIN in this application at all n because of their computational efficiency. We also observe that DELPHI's runtime is impacted significantly by the range δ, which is in contrast with DELPHI's runtime in AWS.

Runtime analysis of DELPHI. We investigate the major factors affecting DELPHI's runtime in both testbeds. We consider two parameters - (a) The agreement ratio $\frac{\Delta}{\epsilon}$ and (b) The range ratio $\frac{\delta}{\rho_0}$, which control the round complexity and per-round communication complexity. In Fig. 7, we plot a heatmap of the runtime of DELPHI with $n = 64$ and $n = 85$ processes in the AWS and CPS testbeds. We observe that the round complexity is the dominant driver of runtime in AWS. However, we observe the opposite in CPS, where per-round communication volume has a greater influence. The limited network bandwidth of the devices in CPS is the rate-limiting factor. Hence, the parameters ϵ and ρ_0 are crucial for DELPHI's performance in

AWS and CPS, respectively.

E. Validity Relaxation

In the oracle network application, DELPHI's output is δ distance away from the range of honest inputs. As we observe in Fig. 4, the mean δ is 25\$. Taking the current price of Bitcoin as 40000\$, DELPHI adds 0.05% error to the price value in expectation than FIN and Abraham et al. Concretely, DELPHI's output is 25\$ away from the average of honest inputs in expectation. For FIN and Abraham et al., it is 12.5\$ away from the average in expectation. Further, this error is less than 0.5% for more than 99.2% of the time. In the Drone surveillance application, the mean difference along each coordinate is $\delta_{\mathsf{mean}} \leq ln(n) \times 0.18 = 0.92$m. Essentially, the output of DELPHI is at most 1.3m farther away from the target object than FIN and Abraham et al. in expectation. Concretely, DELPHI's output is 2.6m away from the average of honest inputs in expectation. Whereas for FIN and Abraham et al., the output is 1.3m away from the average of honest inputs in expectation.

VII. RELATED WORK

Many agreement protocols were proposed with the weak Validity condition [3], [22], [43], which states that if all honest nodes have the same input v, then the output must be v. Binary BA protocols where nodes input only 0 or 1 offer this form of Validity. Prominent asynchronous binary BA protocols [3], [22], [43] have an $\mathcal{O}(n^2)$ communication complexity (even sub-quadratic [11]) and require $\mathcal{O}(1)$ common coins. However, such protocols are unsuitable for applications with multi-valent inputs, where honest nodes often input multiple values.

Validated BA protocols have been proposed, which have an external validity property, ensuring that the output satisfies an external predicate. Validated BA (VABA) [4] and Multi-valued Validated BA (MVBA) [17], [27], [40] protocols have been proposed with external Validity with a communication complexity of $\mathcal{O}(\kappa n^2)$, and multi-valent common coins used for electing nodes as leaders. However, local Validity assessment cannot be done for real-world data like price feeds of a stock/cryptocurrency or the location of an object in an area.

Convex Validity enables nodes to agree on a value within the range of honest inputs. One way to achieve Convex Validity is to compute the Asynchronous Common Subset (ACS) [10], which enables agreement on a set of $n-t$ inputs. The median of the ACS is guaranteed to be within the range of honest inputs. Current ACS protocols can be classified as either being BKR-style [10] or MVBA-style [17]. BKR-style ACS such as HoneyBadgerBFT [42], Beat [26], and PACE [50] use RBC and n binary BA instances. However, these protocols are computationally expensive, requiring $\mathcal{O}(n)$ common coins to terminate. A concurrent work HashRand [7] overcomes this bottleneck by building common coins from computationally efficient Hash functions. However, the described ACS protocols still cost a high $\mathcal{O}(\kappa n^3)$ bits.

The MVBA-style ACS uses RBC with MVBA to overcome the high runtime of BKR-style ACS. In exchange for running n binary BAs, MVBA uses a more complex multi-valent common coin for proposal election (PE), implemented using $\mathcal{O}(\log(n))$ binary coins [17]. Prior MVBA-style protocols like Dumbo2 [35] and SpeedingDumbo [34] trade computational complexity for communication efficiency and therefore use $\mathcal{O}(n^2)$ signatures per ACS. FIN-ACS [27] is the most computationally efficient ACS protocol thus far, which uses RBC and a multi-valent coin for PE. FIN-ACS has an overall computational complexity of $\mathcal{O}(\log(n))$ coins and uses $\mathcal{O}(\kappa n^3)$ bits, which we argue is also very expensive in our setting. Information-theoretic approaches like WaterBear [51] do not use threshold signatures but require $\mathcal{O}(exp(n))$ communication to terminate.

Approximate Agreement (AA) protocols have been proposed as an alternative to the compute-intensive exact agreement protocols. Dolev et al. [24] proposed the first asynchronous AA protocol with $n = 5t+1$ resilience, with $\mathcal{O}(n^2)$ communication per round and $\mathcal{O}(\log_2(\frac{\delta}{\epsilon}))$ rounds. Abraham et al.'s [1] protocol improved this resilience to $n = 3t + 1$ but requires $\mathcal{O}(n^3)$ communication per round. Followup works explored AA over a partially connected network [47], multi-dimensional AA [41], [46], and AA with asynchronous fallback [32], [33]. However, Abraham et al.'s per-round communication of $\mathcal{O}(n^3)$ bits has not been improved so far. Binary AA is closely related to the notion of *proxcensus* [30], [31], a generalization of graded consensus.

VIII. CONCLUSION

The paper introduced Delphi, a deterministic asynchronous convex BA protocol designed for applications like distributed oracles in blockchains and fault-tolerant cyber-physical systems. Addressing the challenge of maintaining convex validity in sensor/oracle nodes measuring the same source, Delphi minimizes communication overhead to $\tilde{\mathcal{O}}(n^2)$ while keeping computation costs low. Unlike existing protocols relying on randomization or approximate agreement techniques with high computational or communication costs, Delphi combines agreement primitives and a novel weighted averaging technique. Experimental results demonstrate Delphi's superior performance, achieving an 8x and 3x improvement in latency over state-of-the-art protocols in CPS and AWS environments for a system with $n = 160$ nodes.

IX. ACKNOWLEDGMENTS

We thank the reviewers and our shepherd Darya Melnyk for the helpful feedback and suggestions to improve this draft significantly. We thank Manoj Patil and Saaransh Jakhar at Supra for collecting data on cryptocurrency price feeds. We also thank Azam Ikram and Mihir Patil for help with experiments on the CPS testbed. This work was supported in part by NIFA award number 2021-67021-34252, the National Science Foundation (NSF) under grant CNS1846316 and National Science Foundation Cyber Physical Systems (CPS). C. Liu-Zhang's research was supported by Hasler Foundation Project #23090 and ETH Zurich Leading House RPG-072023-19.

REFERENCES

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In *Proceedings of the 8th International Conference on Principles of Distributed Systems*, OPODIS'04, page 229–239, Berlin, Heidelberg, 2004. Springer-Verlag.

[2] Ittai Abraham, Naama Ben-David, and Sravya Yandamuri. Asynchronous agreement part 4: Crusader agreement and binding crusader agreement https://decentralizedthoughts.github.io/2022-04-05-aa-part-four-ca-and-bca/. https://decentralizedthoughts.github.io/2022-04-05-aa-part-four-CA-and-BCA/, 2021.

[3] Ittai Abraham, Naama Ben-David, and Sravya Yandamuri. Efficient and adaptively secure asynchronous binary agreement via binding crusader agreement. In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 381–391. ACM, 2022.

[4] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.

[5] A. Bandarupalli, S. Jain, A. Melachuri, J. Pappas, and S. Chaterji. Vega: Drone-based multi-altitude target detection for autonomous surveillance. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pages 209–216, Los Alamitos, CA, USA, jun 2023. IEEE Computer Society.

[6] Akhil Bandarupalli. Delphi: Efficient Asynchronous Approximate Agreement for Distributed Oracles, March 2024. Available on Zenodo at https://doi.org/10.5281/zenodo.10896706.

[7] Akhil Bandarupalli, Adithya Bhat, Saurabh Bagchi, Aniket Kate, and Michael Reiter. Hashrand: Efficient asynchronous random beacon without threshold cryptographic setup. Cryptology ePrint Archive, Paper 2023/1755, 2023. https://eprint.iacr.org/2023/1755.

[8] Akhil Bandarupalli, Dhruv Swarup, Noah Weston, and Somali Chaterji. Persistent airborne surveillance using semi-autonomous drone swarms. In *Proceedings of the 7th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, Dronet'21, page 19–24, New York, NY, USA, 2021. Association for Computing Machinery.

[9] Michael Ben-Or. Another advantage of free choice (extended abstract) completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30, 1983.

[10] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '94, page 183–192, New York, NY, USA, 1994. Association for Computing Machinery.

[11] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, pages 353–380. Springer, 2020.

[12] Philip J Boland. *Statistical and probabilistic methods in actuarial science*. CRC Press, 2007.

[13] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 31–46, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[14] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 416–432. Springer, 2003.

[15] Satellite Navigation Branch. Global positioning system standard positioning service performance analysis report. https://www.nstb.tc.faa.gov/reports/2020_Q4_SPS_PAN_v2.0.pdf, 2021.

[16] Lorenz Breidenbach, Christian Cachin, Alex Coventry, Ari Juels, and Andrew Miller. Chainlink off-chain reporting protocol. *URl: https://blog. chain. link/off-chain-reporting-live-on-mainnet*, 2021.

[17] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference*, pages 524–541. Springer, 2001.

[18] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantipole: Practical asynchronous byzantine agreement using cryptography (extended abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, page 123–132, New York, NY, USA, 2000. Association for Computing Machinery.

[19] Yaru Cao, Zhijian He, Lujia Wang, Wenguan Wang, Yixuan Yuan, Dingwen Zhang, Jinglin Zhang, Pengfei Zhu, Luc Van Gool, Junwei Han, et al. Visdrone-det2021: The vision meets drone object detection challenge results. pages 2847–2854, 2021.

[20] Prasanth Chakka, Saurabh Joshi, Aniket Kate, Joshua Tobkin, and David Yang. Dora: Distributed oracle agreement with simple majority. In *Proceedings of the 43rd International Conference on Distributed Computing Systems(ICDCS'23)*, pages 1–13. IEEE, 2023.

[21] Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. Meeting in a convex world: Convex consensus with asynchronous fallback. *IACR Cryptol. ePrint Arch.*, page 1364, 2023.

[22] Tyler Crain. A simple and efficient asynchronous randomized binary byzantine consensus algorithm. *CoRR*, abs/2002.04393, 2020.

[23] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2518–2534. IEEE, 2022.

[24] Danny Dolev, Nancy A Lynch, Shlomit S Pinter, Eugene W Stark, and William E Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM (JACM)*, 33(3):499–516, 1986.

[25] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 370–386, 2018.

[26] Sisi Duan, Michael K. Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 2028–2041, New York, NY, USA, 2018. Association for Computing Machinery.

[27] Sisi Duan, Xin Wang, and Haibin Zhang. Fin: Practical signature-free asynchronous common subset in constant time. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 815–829. Association for Computing Machinery, 2023.

[28] El Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). *Advances in Neural Information Processing Systems*, 34:25044–25057, 2021.

[29] Paul Embrechts, Claudia Klüppelberg, and Thomas Mikosch. *Modelling extremal events: for insurance and finance*, volume 33. Springer Science & Business Media, 2013.

[30] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 355–362, 2021.

[31] Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. Round-optimal byzantine agreement. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 96–119. Springer, 2022.

[32] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Optimal synchronous approximate agreement with asynchronous fallback. In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 70–80. ACM, 2022.

[33] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Multidimensional approximate agreement with asynchronous fallback. In Kunal Agrawal and Julian Shun, editors, *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023, Orlando, FL, USA, June 17-19, 2023*, pages 141–151. ACM, 2023.

[34] Bingyong Guo, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Speeding dumbo: Pushing asynchronous BFT closer to practice. In *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022.

[35] Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. *Dumbo: Faster Asynchronous BFT Protocols*, page 803–818. Association for Computing Machinery, New York, NY, USA, 2020.

[36] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. *Asynchronous Distributed Key Generation for Computationally-Secure*

Randomness, Consensus, and Threshold Signatures. Association for Computing Machinery, New York, NY, USA, 2020.

[37] Chainlink Labs. 77+ smart contract use cases enabled by chainlink. https://blog.chain.link/smart-contract-use-cases/#post-title, 2019.

[38] Jiani Li, Waseem Abbas, Mudassir Shabbir, and Xenofon Koutsoukos. Byzantine resilient distributed learning in multirobot systems. *IEEE Transactions on Robotics*, 2022.

[39] Shancang Li, Shanshan Zhao, Po Yang, Panagiotis Andriotis, Lida Xu, and Qindong Sun. Distributed consensus algorithm for events detection in cyber-physical systems. *IEEE Internet of Things Journal*, 6(2):2299–2308, 2019.

[40] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *Proceedings of the 39th symposium on principles of distributed computing*, pages 129–138, 2020.

[41] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K Garg. Multidimensional agreement in byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.

[42] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42, 2016.

[43] Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous binary byzantine consensus with t< n/3, o (n2) messages, and o (1) expected time. *Journal of the ACM (JACM)*, 62(4):1–21, 2015.

[44] Hyongju Park and Seth A Hutchinson. Fault-tolerant rendezvous of multirobot systems. *IEEE transactions on robotics*, 33(3):565–582, 2017.

[45] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, pages 10781–10790, 2020.

[46] Nitin H Vaidya and Vijay K Garg. Byzantine vector consensus in complete graphs. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 65–73, 2013.

[47] Nitin H Vaidya, Lewis Tseng, and Guanfeng Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 365–374, 2012.

[48] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

[49] Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 63–70. IEEE, 2005.

[50] Haibin Zhang and Sisi Duan. Pace: Fully parallelizable bft from reproposable byzantine agreement. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 3151–3164, New York, NY, USA, 2022. Association for Computing Machinery.

[51] Haibin Zhang, Sisi Duan, Boxin Zhao, and Liehuang Zhu. Waterbear: Practical asynchronous bft matching security guarantees of partially synchronous bft. In *USENIX Security Symposium'23*, pages 1–16, 2023. https://eprint.iacr.org/2022/021.

[52] Wenbing Zhao, Congfeng Jiang, Honghao Gao, Shunkun Yang, and Xiong Luo. Blockchain-enabled cyber–physical systems: A review. *IEEE Internet of Things Journal*, 8(6):4023–4034, 2020.