Dynamically Available Common Subset

Yuval Efron[1] and Ertem Nusret Tas[2]

[1] Columbia University
[2] Stanford University
ye2210@columbia.edu, nusret@stanford.edu

Abstract. Internet-scale consensus protocols used by blockchains are designed to remain operational in the presence of unexpected temporary crash faults (the so-called *sleepy* model of consensus) – a critical feature for the latency-sensitive financial applications running on these systems. However, their leader-based architecture, where a single block proposer is responsible for creating the block at each height, makes them vulnerable to short-term censorship attacks, in which the proposers profit at the application layer by excluding certain transactions. In this work, we introduce an atomic broadcast protocol, secure in the sleepy model, that ensures the inclusion of all transactions within a constant expected time, provided that at least one participating node is non-censoring at all times. Unlike traditional approaches, our protocol avoids designating a single proposer per block height, instead leveraging a so-called dynamically available common subset (DACS) protocol – the first of its kind in the sleepy model. Additionally, our construction guarantees deterministic synchronization – once an honest node confirms a block, all other honest nodes do so within a constant time, thus addressing a shortcoming of many low-latency sleepy protocols.

1 Introduction

Consensus is the problem of reaching agreement on a growing *log* of values among a group of nodes, some of which may be corrupted and controlled by an *adversary*. It is one of the most fundamental problems in the distributed computing literature and the backbone of modern blockchain systems. Before blockchains, most of the consensus literature considered settings with *consistent participation*, where the set of participants (called nodes) is publicly known and fixed throughout the execution of the protocol [LSP19,DS83,BT85,CL99]. Over the last fifteen years, growing interest in blockchain systems motivated research on *Internet-scale* consensus protocols, executed by a large, but less reliable set of nodes [Nak08,BHK+20,PS17b,KRDO17,DPS19,DGKR18]. Desiderata for these protocols are often informed by the economic considerations of the applications running on them (cf., ebb-and-flow property [NTT21], accountable

The authors contributed equally and are listed alphabetically. Contact: ye2210@columbia.edu, nusret@stanford.edu.

safety [SWN+21a]). In this work, we focus on one such requirement, called *short-term censorship resistance*, a crucial property for financial applications such as on-chain auctions and automated market makers [Pra24,FPR23,BGR24].

Internet-scale consensus. Blockchains such as Bitcoin and Ethereum aim to support a highly decentralized set of nodes, both in number and geographical location, not all of which might be active at all times. Therefore, the consensus protocols underlying these systems must remain secure amidst *unexpected temporary crash faults*. This requirement was formalized under the names *dynamic availability* [KRDO17] and *sleepy model* of consensus [PS17b]. In this work, we consider the synchronous sleepy model with a known message delay upper-bound Δ, where the set of potential nodes (of size N) is publicly known, and at any given time t, only an *unknown* $n_t \leq N$ number of nodes is *awake* (*i.e.*, active) to the discretion of the adversary.

Short-term censorship resistance. The majority of both Internet-scale (cf. Table 1) and PBFT-style consensus protocols (*e.g.*, [CL99,BKM18,YMR+19a]) can be characterized as *leader-based*: Time is divided into intervals, and in each interval, a leader node is designated to propose the next batch of values (called *block*) to be added to the log. The *monopoly* power of the leaders over the block contents makes the protocols vulnerable to *short-term censorship*. For illustration, consider the state machine replication formulation of the consensus problem, in which the nodes receive *transactions* of a blockchain application as input. Due to the economic incentives of these applications [But15,Pra24,FPR23,BGR24], leader nodes can often profit from actively excluding certain transactions from their blocks, significantly delaying the time for the transactions to enter the log. Not only is this behavior highly profitable, but also the nodes acting in this way can often not be held accountable, *i.e.*, one cannot produce a transferable *proof* that censorship took place. Therefore, protocols cannot deter short-term censorship via financial penalties that are deployed against other types of attacks (*e.g.*, forking, safety violations [SWN+21b,BLR24]).

To explicitly capture short-term censorship attacks, we consider a setting where an overwhelming *majority* of the honest awake nodes engage in censorship by excluding certain transactions from their blocks. Intuitively, an honest-but-censoring node follows the protocol faithfully, except that whenever given the opportunity to construct blocks, it proposes blocks excluding the censored transactions[3]. Then, our goal is to design a protocol that guarantees low ($O(\Delta)$) latency for the inclusion of transactions within the log, even when all but a constant number of the honest awake nodes are censoring. We informally refer to such a protocol as having $O(\Delta)$-*censorship resistance*.

[3]When the protocol enforces external validity rules, it might require the inclusion of certain transactions in a given block. In this case, the honest-but-censoring nodes include these transactions. However, such rules cannot be enforced for many transaction types (*e.g.*, bids in on-chain auctions).

1.1 Technical Contributions

Our main result can be stated as follows.

Theorem 1 (Informal). *There exists a consensus protocol secure in the synchronous sleepy model with $O(\Delta)$ expected latency and censorship resistance, as long as at any given time, a majority of the awake nodes are honest.*

Namely, even if all but one of the honest awake nodes are censoring, if the majority of the awake nodes are honest (can be censoring or not) at all times, then our protocol guarantees the inclusion of all inputs within the log in $O(\Delta)$ expected time.

Our protocol is *optimal* in terms of its latency and security resilience, and is the *first* to achieve $O(\Delta)$ censorship resistance in the synchronous sleepy model (for a detailed comparison to existing work, cf. Table 1). In the optimistic case, *i.e.*, when all awake nodes are honest (but can be censoring except one), our protocol achieves a latency of 4Δ. As no atomic broadcast protocol can remain secure under both the sleepy and partially-synchronous (or asynchronous) network models [LR23], synchrony is justified as a necessary assumption for Internet-scale consensus, implying the optimality of $O(\Delta)$ latency and censorship resistance. Similarly, no protocol can satisfy security in the Δ-synchronous sleepy model, when the adversary can control half or more of the awake nodes [PS17a, Theorem 3], implying the optimality of the resilience.

Deterministic synchronization. In many synchronous protocols [AMN+20,CS20], once the first honest node adds a value to the log (*i.e.*, confirms a value), *all* honest nodes confirm that value within $O(\Delta)$ time. In these protocols, a node typically confirms a value only upon collecting sufficient evidence. Therefore, by forwarding this evidence to all other nodes, it can convince them to confirm the same value in Δ time. This approach becomes far more complex to implement in the sleepy model, as the number of participating nodes at any point in time is unknown; and thus what might appear as sufficient evidence in the view of one node, may be insufficient for another. Evidently enough, no consensus protocol with $O(\Delta)$ expected latency in the sleepy model offers deterministic synchronization, instead satisfying a weaker property called *randomized synchronization*: Once the first honest node confirms a value, all honest nodes add the same value to their logs within $O(\lambda)$ time with probability $1-\exp(-\lambda)$, where λ is a security parameter. In this context, we design the *first* low latency consensus protocol in the sleepy model with deterministic synchronization, in which once an honest node confirms a value, all other honest nodes follow suit within $O(\Delta)$ time.

1.2 Technical Overview

Our main construction is an Atomic Broadcast (ABC) [CASD95a] protocol with $O(\Delta)$-censorship resistance in the synchronous sleepy model. The primary building block of the ABC protocol and the main technical contribution is a low

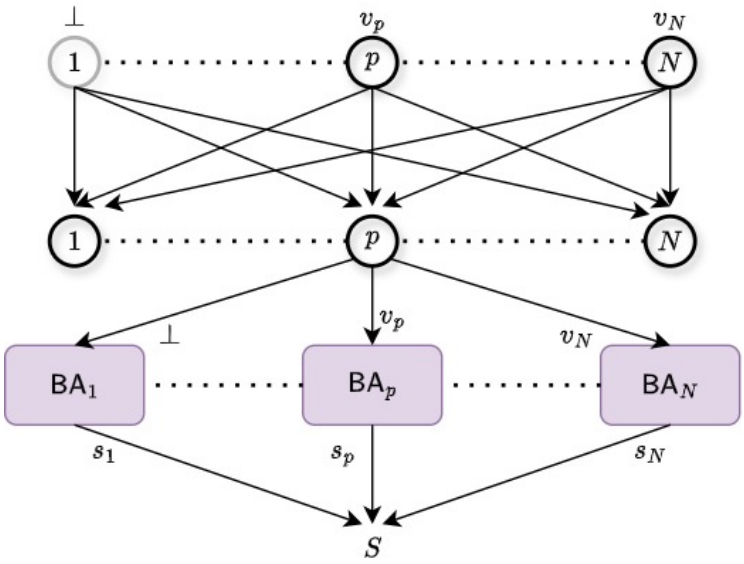

Fig. 1: Intuitive overview of the structure of a single iteration of our DACS protocol. First, every awake honest node multicasts its input value to all nodes, at which point each node inputs the value it received from node i to BA_i. Each asleep node, *e.g.*, node 1 in the figure, is treated by the other nodes as if it sent $\perp$, and that value is input to the corresponding BA. See Section 7.1 for the full formal details.

latency (*i.e.*, $O(\Delta)$) *Dynamically Available Common Subset (*DACS*)* protocol (Figure 1), which can be thought of as the sleepy model analogue of the well-known Asynchronous Common Subset (ACS) primitive [BKR94,CKPS01]. In DACS, the nodes aim to come to agreement on a set of values that includes the inputs of all honest nodes awake at the beginning of the protocol. We describe the technical details of the DACS construction from the bottom up, starting with the primitives used within the construction. The first such primitive is Consistent Graded Agreement (CGA), our variant of the Graded Agreement (GA) primitive.

Consistent Graded Agreement. (Section 4 and Section 5) In graded agreement (GA), each node inputs a value to the protocol and outputs a pair consisting of a value and a binary grade $g \in \{0, 1\}$, intuitively indicating its confidence in the output value. The usual security desiderata for the problem include graded delivery and validity. Graded delivery states that if an honest node is confident about a value (*i.e.*, outputs it with grade 1), then all other honest nodes output that value. Validity requires that if all honest nodes input the same value, then all honest nodes output that value with grade 1. We present two novel GA constructions for a primitive called Consistent Graded Agreement (CGA), which

satisfies *consistency* in addition to graded delivery and validity. Consistency stipulates that all honest nodes outputting a value $v \neq \perp$ for a special value $\perp$, output the same value v. Our CGA protocols requires 4 rounds and have $O(n^2)$[4] message complexity. We compare our constructions to similar primitives found in the literature in Section 2. The CGA primitive is a key building block of the next component of our construction, the well-known Byzantine Agreement (BA) problem.

Byzantine Agreement. (Section 6) We propose the *first* BA protocol in the sleepy model with $O(\Delta)$ expected latency. Furthermore, our protocol has deterministic synchronization. Our BA is iteration-based, and each iteration is comprised of two instances of the Consistent Graded Agreement (CGA) protocol, followed by a leader election step (See Figure 2). Nodes start the first CGA instance of the first iteration with their original inputs to the BA protocol. The first CGA in each iteration is called the *decision* CGA, from which an output with grade 1 triggers a node to *decide* that value for the BA protocol. The goal of the second CGA is to preserve the agreement among the nodes in case a node made a decision during the first CGA. The leader election step selects a leader uniformly at random from among the awake nodes, in an attempt to bring the network into a favorable position before entering the next iteration. If the leader is honest, all honest nodes decide and terminate in the next iteration. Therefore, when the honest nodes constitute over half of the awake nodes, with probability at least $\frac{1}{2}$, all honest nodes decide and terminate in the next iteration, implying an expected latency of $O(\Delta)$. Moreover, due to the second CGA, once the first honest node decides a value, all honest nodes decide that value after the next iteration (*i.e.*, within $O(\Delta)$ time), implying deterministic synchronization. With BA in hand, we can now describe our DACS protocol.

DACS *construction.* (Section 7) From a high level, each DACS instance consists of N parallel executions of BA, the outputs of which constitute the output set of the DACS instance (See Figure 1). Each BA instance is associated with a node in the network, and its goal is to ensure agreement on that node's contribution to the set for the current DACS instance. However, recall that each BA instance has $O(\Delta)$ expected latency; thus with this construction, our DACS protocol would incur an $\Omega(\log(N))$ latency blowup, as it would be equal to the latency of the *slowest* BA instance. We circumvent this by *correlating* the leader election step of the BA instances, specifically by choosing the *same leader* for all BA instances in each iteration. Then, *all* BA instances terminate within $O(\Delta \cdot k)$ time, with probability at least $1 - \frac{1}{2^k}$, yielding $O(\Delta)$ expected latency for DACS. Although the message complexity of DACS is $O(Nn^2)$ due to the N instances of BA, employing aggregate signatures and a trick from [YMR+19b], we show that this can be reduced down to $O(n^3)$.

Finally, our ABC protocol is comprised of repeated sequential executions of DACS, each commencing $O(\Delta)$ time after its predecessor (with appropriately set

[4]Henceforth, n refers to an upper bound on the number of awake nodes during the execution of a protocol/primitive, the identity of which will be clear from context.

constants). Contents of the final log output by each node are ordered first by the order of the DACS instances, and second by a deterministic ordering given a particular output S of a DACS instance. We provide a detailed comparison of our ABC protocol to other protocols in the literature in Table 1 and Section 2.

2 Related Work

Table 1: Overview of previous work and our contributions for Atomic Broadcast protocols in the synchronous sleepy model, along with some of their corresponding properties. Below, the message complexity column refers to the amortized message complexity per block, ρ refers to the fraction of corrupt nodes, CR refers to the censorship resistance parameter (*i.e.*, expected latency given majority censoring nodes), γ refers to the participation rate of all nodes, Δ is the upper bound on network delay, and λ is the security parameter.

Paper	Message complexity	ρ	CR	Synchronization	Latency
[Nak08,GKL15]	$O(1)$	$\frac{1}{2}$	$O(n\Delta/\gamma)$	Randomized[1]	$O(\frac{\lambda\Delta}{\gamma})$
[DPS19,BGK$^+$18][2]	$O(1)$	$\frac{1}{2}$	$O(n\Delta/\gamma)$	Randomized[1]	$O(\frac{\lambda\Delta}{\gamma})$
[GLR21a]	$O(n)$	$\frac{1}{2}$	$O(n\Delta)$	Randomized	$O(\lambda\Delta)$
[BKT$^+$19]	$O(n)$	$\frac{1}{2}$	$O(n\Delta/\gamma)$	Randomized	$O(\frac{\Delta}{\gamma})$
[MMR23a,DZ23b]	$O(n^2)$	$\frac{1}{2}$	$O(n\Delta)$	Randomized	$O(\Delta)$
[DNTT22]	$O(n)$	$\frac{1}{2}$	$O(n\Delta)$	Randomized	$O(\lambda\Delta)$[3]
BRAID on [MMR23b]	$O(Nn)$	$\frac{1}{2}$	$O(\Delta \log N)$	Randomized	$O(\Delta \log N)$
This paper	$O(n^2)$	$\frac{1}{2}$	$O(\Delta)$	Deterministic	$O(\Delta)$

1. It is possible to achieve deterministic synchronization by requiring the honest nodes to multicast the observed blocks. However, this makes message complexity $O(n)$.
2. Here, we also consider [PS17b,KRDO17,DGKR18].
3. The latency is $O(\Delta)$ optimistically, *i.e.*, when the participation is high.

2.1 Asynchronous Common Subset & Interactive Consistency

Protocols solving asynchronous common subset (ACS) have been constructed by several works [BKR94,CKPS01,MXC$^+$16]. Honeybadger BFT builds a practical asynchronous ABC protocol by running ACS iteratively [MXC$^+$16]. For validity, an ACS protocol with $N = 3f + 1$ nodes requires each set output by an honest node to contain the inputs of at least $N - f$ nodes and $N - 2f$ honest nodes. This is the best possible guarantee achievable under asynchrony due to the indistinguishability of the corrupt and delayed nodes. In contrast, our DACS protocol exploits the synchrony assumption to ensure that each set output by an honest node contains the inputs of all other honest nodes. As we work in the sleepy

network model, and no common subset protocol can satisfy security under both the sleepy and partially-synchronous (or asynchronous) network models [LR23], synchrony is a necessary assumption to solve the DACS problem. Furthermore, the synchrony assumption, together with our leader correlation trick allows to shave a $\log N$ factor in latency in order to design a DACS protocol with $O(\Delta)$ expected latency, this is in contrast to state of the art ACS protocols, which have latency $\Theta(\log N)$.

The common subset problem has also been studied in the more standard synchronous non-sleepy setting, where it is usually referred to as the Interactive Consistency (IC) problem. We refer the reader to [CDG+24] for a deeper discussion on the related work in IC.

2.2 Sleepy Consensus

Variants of the consensus problem have been formulated over the past 40 years [LSP19,DS83,BT85,CL99]. Recent years have been marked by increased attention to many of these formulations in the sleepy model.

Atomic Broadcast. The vast majority of the work on consensus in the sleepy model has focused on the Atomic Broadcast (ABC) problem. Nakamoto consensus [Nak08], introduced as part of the Bitcoin protocol, was the first to achieve security [GKL15] given large numbers of unexpected and temporary crash faults. Whereas Bitcoin uses a proof-of-work based sybil resistance mechanism, Nakamoto consensus was subsequently adopted by proof-of-stake protocols [PS17b,KRDO17,DPS19,DGKR18]. These works also formalized the aforementioned network model under the names *dynamic availability* [KRDO17] and the *sleepy network model* [PS17b].

Despite its security, Nakamoto consensus suffers from a latency of $\Omega(\lambda\Delta/\gamma)$, where λ is the security parameter, Δ is the delay bound, and $\gamma = n/N$ is the fraction of awake nodes over all nodes. This motivated a long line of research aiming to reduce latency, shaving-off either the λ [BKT+19,FGKR18,DNTT22,DZ23a] or the γ [GLR21b] terms. The desired $O(\Delta)$ expected latency was eventually achieved by [MR22a], with follow-up works [MMR22a,MMR23b,MMR22b] further reducing the constant expected latency. Assuming a 1/2-bounded adversary, [MMR23b] has the best known expected latency at 14Δ.

Building on classical BFT protocols, [MR22a,MMR22a,MMR23b,MMR22b] inherit their view-based structure with a single leader proposing a batch of transactions (*i.e.*, a block) at each view. The leader has absolute control over the contents of its view's block. Therefore, these protocols suffer from *short-term censorship*: when all but a constant number of the honest nodes censor a transaction by excluding it from their blocks, they cannot ensure its inclusion with latency $o(\lambda)$, as it takes in expectation $\Omega(\lambda)$ slots for a non-censoring honest node to be the leader.

Byzantine Agreement. Thie first Byzantine agreement protocol secure in the sleepy model is due to Garay et al. [GKL15]. The protocol is based on Nakamoto

consensus [Nak08], and thus has a latency of $\Omega(\lambda\Delta/\gamma)$, message complexity of $O(1)$ and randomized synchronization. Goyal et al. [GLR21b] later improved on the latency, which was reduced to $\Omega(\lambda\Delta)$, at the expense of message complexity that became $O(n)$. We design a BA protocol with $O(\Delta)$ expected latency, $O(n^2)$ message complexity and deterministic synchronization. The work of [GL23] designs a BA protocol with $O(\Delta)$ latency for the sleepy model where the adversary is restricted to be non-equivocating, and messages are time-stamped.

Graded Agreement. There exist several formulations and implementations of the graded agreement problem in the sleepy model, under varying assumptions. The works of [GLR21a,MR22b,DSTZ24b,DLZ24,DSTZ24a] all implement a constant round graded agreement protocol under varying *stable participation* assumptions, requiring that some subset of the honest nodes remains consistently awake for a sufficiently long interval of time. Malkhi et al. [MMR23a] design a graded agreement protocol that makes no assumption about stable participation, but lacks the *consistency* property that usually accompanies the primitive. To the best of our knowledge, we design the first graded agreement protocol in the sleepy model with no stable participation assumptions that satisfies both validity and consistency. Our protocol takes 4 rounds, and has message complexity of $O(n^2)$.

2.3 FOCIL

FOCIL (fork choice conditional inclusion lists) was proposed as a protocol add-on to combat short-term censorship on Ethereum [TMDM24]. In each Ethereum slot, a subset of the nodes called the inclusion committee builds and propagates to the slot leader a set of censored transactions called the inclusion list (IL). The nodes then check if the block proposed for that slot contains the transaction in the IL up to the block size limit (*i.e.*, gas limit), otherwise refusing to vote for the block. Without FOCIL, the adversary can censor a transaction tx by paying to a rational slot leader slightly more than tx to exclude it from the block [Pra24]. In FOCIL, once tx is part of the IL, the adversary must fill up either IL or the block with its transactions to exclude tx, paying a lot more than tx in the process.

Censorship is only one protocol deviation made possible by the leaders' monopoly over the block contents. More generally, the leaders enjoy last-mover advantage, enabling them to extract more surplus from the blockchain applications [Pra24]. Although FOCIL raises the bar for a successful censorship attack, it does not alleviate the problems due to the last-mover advantage. In contrast, both issues can be addressed by consensus protocols with multiple concurrent leaders in each slot (*i.e.*, multi-proposer schemes). First, to avoid censorship, a transaction tx can offer to pay a large amount T upon inclusion by one block only, and some $t << T$ upon inclusion by multiple blocks [FPR23]. Then, tx ends up paying only $\Theta(t)$ in equilibrium, whereas the adversary must pay T to each leader to exclude tx. Second, timelock or threshold encryption schemes can ensure that no leader observes the block contents before any other leader. Then, competition among these equally-positioned leaders can prevent the surplus of on-chain activity from flowing to any single one, thus mitigating the last-mover

advantage. Ensuring such competition requires a careful design of transaction fee mechanisms, which is beyond the scope of this work.

2.4 BRAID

BRAID is a candidate multi-proposer scheme. To reduce the latency of censored transactions, it entails running multiple instances of a view-based blockchain protocol in parallel, each executed by the whole set of awake nodes. At any view v, BRAID selects a different leader for each protocol instance to ensure an honest, non-censoring leader for one of the instances. Then, the final block confirmed for view v consists of the blocks confirmed for that view by each parallel instance. This ensures the inclusion of a censored transaction as long as one of the leaders is a non-censoring honest node.

Assuming that all but a constant number of the honest nodes is censoring, BRAID has to run $I = \Omega(N^\epsilon)$, $\epsilon \in [0, 1]$, instances at all times to achieve an expected latency of $o(N^{1-\epsilon})$ for the censored transactions[5]. Moreover, BRAID nodes confirm the meta-block consisting of the blocks at a certain view v, only after confirming a view v block for *all* protocol instances. Therefore, BRAID instantiated with the protocols in [MR22a,MMR22a,MMR23b,MMR22b] has $O(\log(I)\Delta) = O(\log(N)\Delta)$ expected latency, as these protocols achieve constant latency only *in expectation*. In these protocols, the nodes are guaranteed to confirm only the blocks proposed by the honest leaders and their prefixes (regardless of whether the leader is censoring or not). As the leaders at different views of a given protocol instance are selected uniformly at random, for a given view v, it takes in expectation $O(\log(I)\Delta)$ time for each of the I instances to have an honest leader at view v or higher. In contrast, our DACS protocol achieves a constant expected latency by correlating the leaders across different BA instances. This idea is not applicable to BRAID since correlated leaders across different protocol instances defeats the purpose of censorship resistance.

The construction described above for BRAID has a message complexity of $O(N^\epsilon n^2)$ as it consists of more than N^ϵ protocol instances and each instance incurs a complexity of $O(n^2)$. This can be reduced to $O(n^3)$ using aggregate signatures in a manner similar to that of our DACS protocol.

Finally, BRAID based on [MR22a,MMR22a,MMR23b,MMR22b] inherits the probabilistic synchronization of these protocols.

3 Model & Definitions

A function $f(x)$ is said to be negligible in the security parameter λ if $f(x) = o(1/x^d)$ for all $d > 0$. Time proceeds in discrete rounds denoted by t. We denote the set $\{1, \ldots, n\}$ by $[n]$. For two sequences of values $u = (a_1, \ldots, a_n)$ and $v = (b_1, \ldots, b_m)$, we write $u \preceq v$ if $n \leq m$ and $a_i = b_i$ for all $i \in [n]$.

[5]Recall that N is the total number of nodes in the sleepy model.

3.1 Model

Adversary. The adversary $\mathcal{A}$ is a PPT algorithm, which upon observing a given protocol Π, can statically corrupt and subsequently control a subset of the nodes, hereafter called *corrupt* nodes. They surrender their internal states to the adversary and can deviate from the protocol arbitrarily (Byzantine faults). The remaining nodes are called *honest* and execute the protocol as specified. We denote the total number of nodes corrupted by the adversary throughout the execution by f.

Environment and network. Nodes exchange messages over a peer-to-peer *synchronous* network. The adversary can delay the messages sent by the honest nodes up to Δ rounds and deliver them in any order. At each round, it can insert a polynomial number of messages to any honest node. The *environment* is an external entity to the protocol that at any point in time, can send *transactions* from a universe I to nodes in the network. The timing and content of these messages is to the discretion of the adversary.

Sleepy model of consensus. There are N nodes in total, identified by a public key infrastructure (PKI). At any round t, the adversary can adaptively[6] select a subset of the honest nodes to be *asleep*, whereas the remaining nodes are *awake*. Asleep nodes are temporarily crashed: they do not execute the protocol or send messages. Messages sent to an asleep node are buffered and delivered to the node in an adversarially selected order once it is awake. Upon being awake, nodes know the current round number[7]. Once corrupted, nodes stay awake throughout the entire execution. The number of awake nodes at round t is denoted by n_t[8]. We say that the adversary is ρ-bounded if for all rounds t, $f/n_t < \rho$. We assume that the adversary is 1/2-bounded unless stated otherwise, *i.e.*, $n_t \geq 2f + 1$ at all rounds t.

Cryptographic primitives. Our protocols use a cryptographic hash function $H(.)$, a digital signature scheme and a verifiable random function (VRF), with public keys given by the PKI. A message m signed by a node p is denoted by $\langle m \rangle_p$. The VRF function evaluated by a node p on input m is denoted by $\mathsf{VRF}_\mathsf{p}(m)$ and outputs a deterministic pseudorandom value ρ and a proof π that attests to the correct VRF evaluation by node p on message m.

Censorship. As discussed in the introduction, we propose a formal model to capture the *short-term censorship* phenomena, which plagues modern blockchain protocols. While the intuition from the introduction about nodes being honest but censoring is helpful to consider, defining the concept of an honest-but-censoring node in a *protocol agnostic* fashion, *i.e.*, as part of the model, without

[6] The adaptive adversary is not strongly rushing, i.e. it *can not* observe the contents of a message sent by a node, intercept it, and then put the node to sleep.

[7] We assume synchronized clocks. The latency Δ will be larger under a bounded clock skew and will require occasional clock synchronization.

[8] We drop the subscript when the round is clear from the context.

making assumption on the structure or behavior of any protocol, seems like an elusive task. We thus resort to defining a property which we argue captures the essence of censorship, and is well defined for any primitive equipped with the *termination* property and whose interface has the honest nodes commence the protocol with input values from some known universe, and output a set of values from said universe (*e.g.*, Byzantine Agreement, Graded Agreement, ACS, DACS). Specifically, in the context of the sleepy model, we say that a protocol Π for such a primitive has *censorship resistance* if the output of Π always contains the inputs of all honest nodes. We then say that Π has ℓ-censorship resistance where ℓ is the *latency* (See section 3.2) of Π.

Message complexity. A standard measure in the literature for the communication cost of a protocol is *bit* complexity, i.e. the number of bits sent by honest nodes on the communication links throughout the execution. Note that in a network with N nodes, an honest node multicasting a message of size $O(\lambda)$ to all nodes incurs $O(N\lambda)$ bit complexity. In the context of the sleepy model, however, when $n << N$, we would want to capture the communication cost in terms of the number of awake nodes. As such, we focus on *message complexity*, which is the number of different *messages* sent by honest nodes over the communication network throughout the execution. We define a message to be any string of length $O(\lambda)$. The bit complexity of our protocol is thus obtained by simply multiplying the message complexity by $N\lambda$.

3.2 Consensus Primitives

Definition 1 (Atomic Broadcast (ABC)). *In atomic broadcast [CASD95b], nodes receive values $v \in \mathcal{V}$ from the environment and output a growing sequence of values called the* log. *Let $\mathsf{L}^{\mathsf{p}}_r$ denote the log output by a node p at round r. A secure ABC protocol satisfies the following properties:*

- ***Safety.*** *For all honest nodes $\mathsf{p}_1, \mathsf{p}_2$ and rounds r_1, r_2, either $\mathsf{L}^{\mathsf{p}_1}_{r_1} \preceq \mathsf{L}^{\mathsf{p}_2}_{r_2}$, or $\mathsf{L}^{\mathsf{p}_2}_{r_2} \preceq \mathsf{L}^{\mathsf{p}_1}_{r_1}$. For any honest node p and rounds $r_1 \leq r_2$, $\mathsf{L}^{\mathsf{p}}_{r_1} \preceq \mathsf{L}^{\mathsf{p}}_{r_2}$.*
- ***Liveness.*** *If all[9] awake honest nodes receive an input v from the environment by round r, then there is a finite time r' such that v is in all the logs output at or after round $r' + r$ by the awake honest nodes.*

We define a version of the graded agreement problem [MR22a,MMR23b] (similar to Gradecast [KK06]) with an additional consistency requirement.

Definition 2 (Consistent Graded Agreement (CGA)). *In consistent graded agreement, each node takes as input a value $v \in \mathcal{V}$ and outputs a value $v' \in \mathcal{V} \cup \{\perp\}$ along with a grade $g \in \{0, 1\}$, subject to the following constraints:*

- ***Consistency.*** *If two honest nodes output $(v, *)$ and $(v', *)$ for $v, v' \neq \perp$, then $v = v'$.*

[9] Even if a single awake honest node were to receive an input value, it can multicast it to all current and future awake honest nodes.

- ***Graded delivery.*** *If an honest node outputs $(v', 1)$, then all honest nodes that produce an output, output $(v', *)$.*
- ***Integrity.*** *If an honest node outputs $(v', *)$, then at least one honest node awake at time $t = 0$ has input v'.*
- ***Validity.*** *If all honest nodes awake at time $t = 0$ have v as input, then all honest nodes that produce an output, output $(v, 1)$.*
- ***Termination.*** *There exists time T such that every honest node awake at any time $\geq T$ outputs and halts.*

We next define the well known problem of byzantine agreement (BA) [LSP19].

Definition 3 (Byzantine Agreement (BA)). *In byzantine agreement, each node has an input value $v \in \mathcal{V}$, and every node can output a value $v' \in \mathcal{V}$, subject to the following constraints:*

- ***Agreement.*** *There exists a value $v \in \mathcal{V}$, such that all honest nodes that output a value, output v.*
- ***Validity.*** *If all honest nodes awake at time $t = 0$ hold the same input value v, then all honest nodes that output a value, output v.*
- ***Termination.*** *There exists time T such that every honest node awake at any time $\geq T$ outputs and halts.*

3.3 Dynamically Available Common Subset

We next formalize *dynamically available common subset (*DACS*)*, the cornerstone of our protocol. It can be thought of as the analogue of the well-known ACS primitive [BKR94,CKPS01,MXC+16] in the sleepy network model. The formal definition is as follows.

Definition 4 (Dynamically Available Common Subset (DACS)). *In dynamically available common subset, each node has an input value v_i and can output a subset $S \subseteq \mathcal{V}$, subject to the following constraints:*

- ***Agreement.*** *There exists a set $S \subseteq \mathcal{V}$ such that all honest nodes that output a set, output S.*
- ***Validity.*** *If an honest node awake at time $t = 0$ holds the input v, then for any honest node that outputs a set S, $v \in S$.*
- ***Termination.*** *There exists a time T such that every honest node awake at any time $\geq T$ outputs and halts.*

We say that a protocol is ρ-secure protocol if it satisfies its respective properties for all ρ-bounded adversaries. If it satisfies a property prop (for some property *e.g.*, agreement) for all ρ-bounded adversaries; then we say it satisfies ρ-prop (*e.g.*, ρ-agreement). Given a ρ-secure protocol Π and an adversary $\mathcal{A}$, we denote by $L(\Pi_{\mathcal{A}})$ the random variable (over the randomness used by the honest nodes and the adversary) indicating the termination time T in the presence of $\mathcal{A}$.

We define the *expected latency* of Π with respect to $\mathcal{A}$ by $L_{\mathcal{A}}(\Pi) = \mathbb{E}[L(\Pi_{\mathcal{A}})]$. We define the *expected latency* of Π to be

$$L(\Pi) = \sup_{\frac{1}{2}-\text{bounded } \mathcal{A}} L_{\mathcal{A}}(\Pi).$$

Similarly, we denote by $\mathcal{C}(\Pi_{\mathcal{A}})$ the expected message complexity of Π w.r.t. $\mathcal{A}$, and by $\mathcal{C}(\Pi)$ the supremum of $\mathcal{C}(\Pi_{\mathcal{A}})$ over all $\frac{1}{2}$-bounded adversaries $\mathcal{A}$.

4 Consistent Graded Agreement

We now describe the CGA protocol and prove its security.

4.1 The CGA Protocol

The CGA protocol builds on the graded agreement protocol of [MMR23b]. We give a step-by-step description of the protocol below. The start time is normalized to be $t = 0$.

1. **Round 1: Echo.** Each node awake at time $t = 0$ multicasts its input value v_i via an echo message, $\langle \mathsf{echo}, v_i \rangle$ (echo_{v_i} for short). At any future round, all awake nodes multicast all observed echo messages if they have not multicast that message before. If two different messages of any type by a node p were observed, then we say that p is an equivocating node.
2. **Round 2: Tally.** Each node awake at time $t = 1$ does the following:
   - Construct F_v - The set of received echo_v messages.
   - S - The set $\{F_v \mid v \in \mathcal{V}, |F_v| \neq 0\}$ of tallies for the values v. Here, we denote F_v by $S[v]$.
   - Multicast the tally message $\langle \mathsf{tally}, S \rangle$.
3. **Round 3: Group.** Each node awake at time $t = 2$ does the following:
   - D - The dictionary that has $v \in \mathcal{V}$ as its keys, and the set of echo_v messages by non-equivocating nodes as the value $D[v]$ (*i.e.*, no echo message by an equivocating node is included in D). If $D[v] = \emptyset$, then v is omitted from D.
   - Multicast the group message $\langle \mathsf{group}, D \rangle$.
4. **Round 4: Vote.** Each node awake at time $t = 3$ does the following:
   - E_4 - The number of nodes such that an echo message by the node was received.
   - F - The number of nodes from which a group message was received.
   - C_v - The set of non-equivocating nodes p such that over $E_4/2$ echo_v messages by non-equivocating nodes appear in the set $D_{\mathsf{p}}[v]$ within the dictionary D_{p} received from the node p.
   - If $|C_v| > F/2$ for some value $v \in \mathcal{V}$, multicast a vote message $\langle \mathsf{vote}, v \rangle$.
5. **Round 5: Output.** Each node awake at time $t \geq 4$ does the following:
   - E_5 - The number of nodes such that an echo message by the node was received.

- V - The number of nodes from which a vote message was received.
- V_v - The number of nodes from which a vote message for value v was received.
- S'_{p} - In each tally message $\langle \mathsf{tally}, \{F_v \mid v \in \mathcal{V}, |F_v| \neq 0\}\rangle_{\mathsf{p}}$ from the nodes p, remove the equivocating nodes' messages from the sets F_v to obtain F'_v and reconstruct the set $S'_{\mathsf{p}} = \{F'_v \mid v \in \mathcal{V}, |F'_v| \neq 0\}$ with the new F'_v.
- M_v - For each value v such that a non-empty set F'_v appears in a reconstructed S'_{p}, calculate the median M_v of $|S'_{\mathsf{p}}[v]|$ across the nodes p from which a tally message was received.
- If $M_v > E_5/2$ for a value v, output $(v, 1)$. Else, if $V_v > V/2$ for v, output $(v, 0)$. Else, output $(\perp, 0)$.

4.2 Security Analysis

In the analysis below, we denote a value A in a node p's view by A^{p} unless stated otherwise.

Lemma 1 (Graded delivery). *The* CGA *protocol satisfies* $1/2$*-graded delivery: If an honest node outputs* $(v, 1)$*, then for all* $t \geq 4$*, all honest nodes awake at time* t *output* $(v, *)$*.*

Proof. Suppose an honest node p outputs $(v, 1)$. Let M_v^{p} and E_5^{p} denote the values of M_v and E_5 respectively in p's view at time $t = 4$. By definition, $M_v^{\mathsf{p}} > E_5^{\mathsf{p}}/2$. Let C_v^{q} and E_4^{q} denote the values of C_v and E_4 respectively in an honest node q's view at time $t = 3$. Since q forwards all of the echo messages counted in E_4, we have $E_5^{\mathsf{p}} \geq E_4^{\mathsf{q}}$.

Since the adversary is $1/2$-bounded, the median M_v^{p} in p's view is upper bounded by at least one honest node p_0's tally $|S'_{\mathsf{p}_0}[v]|$ for v after p removes the equivocating nodes. Since p_0 forwards all of the echo_v messages received by time $t = 1$, these messages are received by all honest nodes awake at time $t = 2$, and the forwarded echo_v messages by non-equivocating nodes are included in the dictionary D of all honest nodes awake at time $t = 2$. Moreover, if a node p_1 is detected as an equivocating node by q at time $t = 3$, then p_1's messages are also not in any set $S[v]$ in p's view at time $t = 4$; since q forwards all observed echo messages at least once. Therefore, all messages within $S'_{\mathsf{p}_0}[v]$ are also in the set $D[v]$ for each dictionary D received by q from an honest node, and for each such dictionary D, it holds that

$$|D[v]| \geq |S'_{\mathsf{p}_0}[v]| \geq M_v^{\mathsf{p}} > E_5^{\mathsf{p}}/2 \geq E_4^{\mathsf{q}}/2$$

Since for each dictionary D received by q from an honest node, $|D[v]| \geq E_4^{\mathsf{q}}/2$, and the adversary is $1/2$-bounded, it holds that $C_v^{\mathsf{q}} \geq F^{\mathsf{q}}$. Therefore, q sends a vote message for v.

Since the reasoning above holds for any honest node awake at time $t = 3$, all honest nodes awake at time $t = 3$ sends a vote for v. By the $1/2$-bound on the adversary, each node awake at any time $t \geq 4$ outputs $(v, 0)$, if it has not already output $(v, 1)$. □

Lemma 2 (Consistency). *The* CGA *protocol satisfies* $1/2$*-consistency: If two honest nodes output* $(v, *)$ *and* $(v', *)$ *respectively, where* $v, v' \neq \perp$*, then* $v = v'$*.*

Proof. Lemma 1 shows that if for an honest node, the condition for outputting $(v, 1)$, *i.e.*, $M_v^{\mathsf{p}} > V/2$, is satisfied; then for all honest nodes, the condition for outputting $(v, 0)$ is also satisfied, *i.e.*, $V_v > V/2$. Thus, to prove consistency, it suffices to show that the condition for outputting $(v, 0)$, *i.e.*, $V_v > V/2$, cannot be satisfied for two different values $v \neq v'$ in the views of any (potentially the same) honest nodes. Towards contradiction, suppose $V_v > V/2$ in the views of two (potentially the same) honest nodes p_1 and p_2. Then, as the adversary is $1/2$-bounded, two (potentially the same) honest nodes p and q must have sent vote messages for v and v' at time $t = 3$. Therefore, it must be that $C_v^{\mathsf{p}} > F^{\mathsf{p}}/2$ and $C^{\mathsf{q}} > F^{\mathsf{q}}/2$ at time $t = 3$.

As the adversary is $1/2$-bounded, $C_v^{\mathsf{p}} > F^{\mathsf{p}}/2$ and $C^{\mathsf{q}} > F^{\mathsf{q}}/2$ respectively imply the existence of distinct honest nodes p_1 and p_2 such that p_1's dictionary D_1 is received by p, p_2's dictionary D_2 is received by q, and after p and q remove the equivocating nodes from the dictionaries, (i) $|D_1[v]| > E_4^{\mathsf{p}}/2$, and (ii) $|D_2[v']| > E_4^{\mathsf{q}}/2$. Since p_1 and p_2 are honest, p and q both receive all of the echo_v messages within $D_1[v]$ and the $\mathsf{echo}_{v'}$ messages within $D_2[v']$. Hence, after the equivocating nodes are removed, the sets $D_1[v]$ and $D_2[v']$ are disjoint, and

$$\min(E_4^{\mathsf{p}}, E_4^{\mathsf{q}}) \geq |D_1[v]| + |D_2[v']|$$

Now, summing the inequalities in (i) and (ii) gives

$$|D_1[v]| + |D_2[v']| > (E_4^{\mathsf{p}} + E_4^{\mathsf{q}})/2 \geq \min(E_4^{\mathsf{p}}, E_4^{\mathsf{q}})$$

However, this is a contradiction, implying that no two honest nodes output $(v, *)$ and $(v', *)$ such that $v \neq v'$ and $v, v' \neq \perp$. □

Lemma 3 (Integrity). *The* CGA *protocol satisfies* $1/2$*-integrity: If an honest node outputs* $(v, *)$*, then at least one honest node has input* v*.*

Proof. Lemma 1 shows that if for an honest node, the condition for outputting $(v, 1)$, *i.e.*, $M_v^{\mathsf{p}} > E_5^{\mathsf{p}}/2$, is satisfied; then for all honest nodes, the condition for outputting $(v, 0)$ is also satisfied, *i.e.*, $V_v > V/2$. Thus, to prove integrity, it suffices to show that if the condition for outputting $(v, 0)$, *i.e.*, $V_v > V/2$, is satisfied in the view of an honest node, value v must be the input of an honest node. Suppose $V_v > V/2$ in the view of an honest node. Then, as the adversary is $1/2$-bounded, an honest node p must have sent a vote message for v at time $t = 3$ Therefore, it must be that $|C_v| > E_4/2$ in p's view at time $t = 3$. By the definition of C_v, p must have received over $F/2$ dictionaries by unique nodes such that for each of these dictionaries, $|D[v]| > E_4/2$ after p removes the equivocating nodes. As the adversary is $1/2$-bounded, this cannot happen unless an honest node sends an echo_v message, *i.e.*, v is the input of an honest node. □

Lemma 4 (Validity). *The* CGA *protocol satisfies* $1/2$*-validity: If all honest nodes have* v *as input, then all honest nodes output* $(v, 1)$*.*

Proof. Let M_v^{p} and E_5^{p} denote the values M_v and E_5 observed by an honest node p at time $t \geq 4$. Since the adversary is 1/2-bounded, the median M_v^{p} must be lower bounded by at least one honest node p_0's tally $S'_{\mathsf{p}_0}[v]$ for v after p removes the equivocating nodes. Let n_h denote the number of honest nodes awake at time $t = 0$. As honest nodes never send equivocating echo messages, it holds that $S'_{\mathsf{p}_0}[v] \geq n_h$. Thus, $M_v^{\mathsf{p}} \geq S'_{\mathsf{p}_0}[v] \geq n_h > f$. Moreover, for E_5^{p}, it holds that $n_h \leq E_5^{\mathsf{p}} \leq n_h + f$, which together with the inequalities above implies that $M_v^{\mathsf{p}} > E_5^{\mathsf{p}}/2$. Therefore, p outputs $(v, 1)$. □

4.3 Performance

The protocol has a round complexity of 4 and message complexity of $\Omega(n^2)$ when the number of awake nodes is n. The message complexity is due to the group step, where each honest awake node multicasts a dictionary to $n-1$ other nodes, where the total size of the keys and values is $O(n)$.

5 An Alternative Consistent Graded Agreement Protocol

5.1 The CGA Protocol

The start time is again normalized to $t = 0$. The protocol works as follows:

1. **Round 1: Echo.** Each node awake at time $t = 0$ multicasts its input value v_i via an echo message, $\langle \mathsf{echo}, v_i \rangle$.
2. **Round2: Forward1.** Each node awake at time $t = 1$ does the following:
   - Construct F_v - The set of echo messages received for value v, $v \in \mathcal{V}$.
   - Multicast $\langle \mathsf{forward1}, \{F_v \mid v \in \mathcal{V}\} \rangle$[10].
3. **Round 3: Vote1.** Each node awake at time $t = 2$ does the following:
   - E^* - The set of nodes from which an echo message was received.
   - F^* - The set of nodes from which a $\mathsf{forward1}$ message was received.
   - C_v - The set of nodes for which an echo_v message appeared in more than $|F^*|/2$ of the $\mathsf{forward1}$ messages, and no equivocation was detected.
   - If $|C_v| > |E^*|/2$ for some value $v \in \mathcal{V}$, multicast $\langle \mathsf{vote1}, v \rangle$. Else, multicast $\langle \mathsf{novote} \rangle$.
4. **Round 4: Forward2.** Each node awake at time $t = 3$ does the following.
   - W_v - The set of $\mathsf{vote1}$ messages received for value v, $v \in \mathcal{V}$.
   - W_{nv} - The set of novote messages received.
   - Multicast $\langle \mathsf{forward2}, \{W_v \mid v \in \mathcal{V}\}, W_{nv} \rangle$[11].
5. **Round ≥ 5: Output.** Each node awake at time $t = 4$ does the following.
   - V^* - The set of nodes for which a $\mathsf{vote1}$ or novote message was received.
   - W^* - The set of nodes for which a $\mathsf{forward2}$ message was received.
   - D_v - The set of nodes for which a $\mathsf{vote1}_i$ message appeared in more than $|W^*|/2$ of the $\mathsf{forward2}$ messages, and no equivocation was detected.
   - If $|D_v| > |V^*|/2$ for some value $v \in \mathcal{V}$, output $(v, 1)$. Else, if $|D_v| > |D_u|$ for all $u \neq v$ for some $v \in \mathcal{V}$, output $(v, 0)$. Else output $(v_i, 0)$.

[10] Sets F_v which are empty are omitted from the message.

[11] Similarly, W_v which are empty are omitted from the message.

5.2 Security Analysis and Performance

For the analysis, we first start with proving validity.

Lemma 5. *Assume that all honest nodes awake at time $t = 0$ commence* GA *with the same input v. Then every honest node awake at time ≥ 4 outputs $(v, 1)$.*

Proof. Assume all honest nodes awake at $t = 0$ input the same value v. By the honest majority assumption, and the behavior of the protocol we get that for every honesttt p, $\langle \mathsf{echo}, v \rangle_p$ is included in all forward1 messages created by honest nodes at time $t = 1$, and in parituclar $p \in C_v$ for all honest nodes awake at time $t = 2$. Since this holds for all honest nodes p, we get then that $|C_v| > |E^*|/2$ for all honest nodes awake at $t = 2$, which in particular means they all cast $\langle \mathsf{vote1}, v \rangle$. By a similar argument, all awake nodes at time ≥ 4 have that $p \in D_v$ for any honest p awake at time $t = 2$, and so $|D_v| > |V^*|/2$. Thus any honest node awake at time ≥ 4 outputs $(v, 1)$, as required. □

Before we prove graded delivery, we prove the following lemma.

Lemma 6. *Denote by S the set of values v for which there exists a honest node that sent $\langle \mathsf{vote1}, v \rangle$. Then $|S| \leq 1$.*

Proof. Assume towards a contradiction that $|S| \geq 2$, and let p, q be two honest nodes awake at $t = 2$, s.t. p sent $\langle \mathsf{vote1}, v \rangle$, and q sent $\langle \mathsf{vote1}, u \rangle$, for $u \neq v$. Denote by C_v^p, C_u^q the corresponding sets C as defined in Round 3 of the protocol that triggered these messages.

Claim. It holds that $C_v^p \cap C_u^q = \emptyset$.

Proof. Let $a \in C_v^p$, in particular this means, that there exists an honest node a, awake at time $t = 1$, that included $\langle \mathsf{echo}, v \rangle_a$ in its forward1 message, that was sent to all nodes. If it was the case that $a \in C_u^q$, then all honest nodes would witness an equivocation from a, and would thus discard it from their C set, as instructed. This concludes the proof. □

Denote by E_p^*, E_q^* the E^* variables witnesses by p, q respectively at time $t = 2$. Next, we prove the following.

Claim. $C_v^p \cup C_u^q \subseteq E_p^*$, and $C_v^p \cup C_u^q \subseteq E_q^*$.

Proof. We prove $C_v^p \cup C_u^q \subseteq E_p^*$. The proof of $C_v^p \cup C_u^q \subseteq E_q^*$ follows an identical line or arguing. $C_v^p \subseteq E_p^*$ is immediate by the behavior of the protocol. Thus all that is left is to prove $C_u^q \subseteq E_p^*$. Let $a \in C_u^q$. This implies that $\langle \mathsf{echo}, u \rangle_a$ appeared in more than $|F_q^*|/2$ of the forward1 messages received by q, which in particular means that there exists a honest node b, awake at time $t = 1$, that included $\langle \mathsf{echo}, u \rangle_a$ in its F_u set, and in particular sent this set to all nodes via a forward1 message. Thus, all honest nodes awake at time $t = 2$, including p have received the forward1 message from b and thus received $\langle \mathsf{echo}, u \rangle_a$, which in particular means that $a \in E_p^*$, as required. □

Assume w.l.o.g. that $|E_p^*| > |E_q^*|$. Recall that $|C_v^p| > |E_p^*|/2$, and $|C_u^q| > |E_q^*|/2$. Since $C_v^p \cap C_u^q = \emptyset$, we thus have that $|C_v^p \cup C_u^q| = |C_v^p| + |C_u^q| > |E_q^*|/2 + |E_q^*|/2 = |E_q^*|$. This contradicts $C_v^p \cup C_u^q \subseteq E_q^*$. Thus concluding the proof of the lemma. □

We can now prove graded delivery.

Lemma 7. *Assume there exists a honest node p awake at time $t \geq 4$ that output $(v, 1)$ for some $v \in \mathcal{V}$. Then all honest nodes awake at time ≥ 4 output $(v, *)$.*

Proof. We begin by observing that an identical line of arguing as in the claim above, it holds for any two honest p, q awake at time ≥ 4 that $D_v^p \cap D_u^q = \emptyset$, and that $D_v^p \cup D_u^q \subseteq V_p^*, V_q^*$, for any $u \neq v$. Assume p output $(v, 1)$. This in particular means that there exists a honest a, awake at $t = 2$, that sent a $\langle \mathsf{vote1}, v \rangle_a$ message. Which by lemma 6 implies that no other honest nodes sent a $\mathsf{vote1}$ message for any other value. Let q be a honest node awake at time ≥ 4 and assume towards a contradiction that $|D_u^q| \geq |D_v^q|$ for some $u \neq v$. Let $H_{\mathsf{vote1},v}$ be the set of honest nodes awake at time $t = 2$ that cast a $\langle \mathsf{vote1}, v \rangle$ message. Let H be the set of honest nodes awake at time $t = 2$, and let B denote the set of corrupt nodes. Let $B_v^p = D_v^p \backslash H_{\mathsf{vote1},v}$. It holds that $H_{\mathsf{vote1},v} \subseteq D_v^q$, and that $D_v^p \cap D_u^q = \emptyset$. We further have that $D_v^p \cup D_u^q \subseteq V_p^*$. Notice now that

$$H \cup B_v^p \cup D_u^q \subseteq V_p^*$$

, and this is a disjoint union. Thus

$$\begin{aligned} |D_v^p| &> |V_p^*|/2 \geq (|H| + |B_v^p| + |D_u^q|)/2 \\ &\geq (|H| + |B_v^p| + |D_v^q|)/2 \geq (|H| + |B_v^p| + |H_{\mathsf{vote1},v}|)/2 \end{aligned}$$

Note that $|B_v^p| + |H_{\mathsf{vote1},v}| = |D_v^p|$. Thus in total we get that $|D_v^p| > (|H| + |D_v^p|)/2$. This boils down to $|D_v^p| > |H|$. On the other hand, we have that

$$|D_v^p| = |B_v^p| + |H_{\mathsf{vote1},v}| \leq |B_v^p| + |D_v^q| \leq |B_v^p| + |D_u^q|$$

Recall now that D_u^q contains no nodes from H. Also recall that $D_v^p \cap D_u^q = \emptyset$, and so in particular $B_v^p \cap D_u^q = \emptyset$. Thus

$$|B_v^p| + |D_u^q| = |B_v^p \cup D_u^q| \leq |B| < |H|$$

This is a contradiction, and thus every node q awake at time ≥ 4 observes $|D_v^q| > |D_u^q|$ and thus outputs $(v, *)$. □

The message complexity of Π_{GA} is $O(n^2)$, and its round complexity is 4.

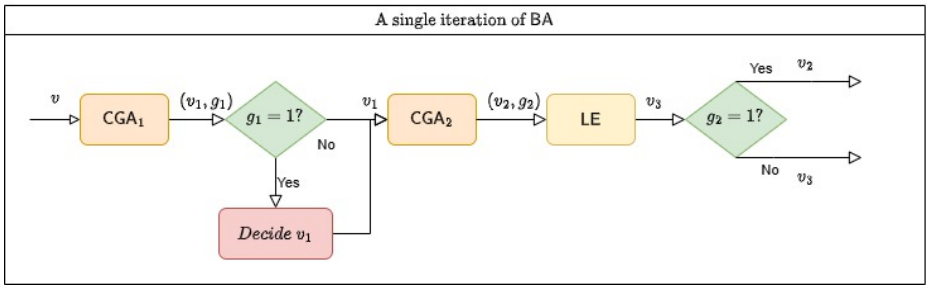

Fig. 2: Overview of the structure of a single iteration of our BA protocol. The input to an iteration is the output to the former iteration, or the input to the BA instance for the first iteration. The CO task is captured by CGA_2 and the leader election (LE) tasks.

6 Byzantine Agreement

In this section, we present a multi valued Byzantine Agreement (BA) protocol with $O(1)$ expected latency in the sleepy model (Figure 2). Specifically, we bootstrap our BA construction by interlacing CGA executions with leader election. We begin by defining the *conciliator* primitive.

Definition 5. *In the conciliator task, each node has an input value $v_i \in \mathcal{V}$ and produces an output $o_i \in \mathcal{V}$, subject to the following constraints:*

- ***Validity.*** *If all honest nodes awake at time $t = 0$ have the same input v, then all honest nodes that output a value, output v.*
- ***Probabilistic agreement.*** *With probability (w.p.) at least $\frac{1}{2}$, all honest nodes that output a value, output the same value v.*
- ***Termination.*** *There exists T such that every honest node awake at time $\geq T$ outputs and terminates.*

If a conciliator protocol Π is ρ-valid, ρ-terminating and has ρ-probabilistic agreement we say that Π is a ρ-secure conciliator (CO) protocol.

We now present a constant round CO protocol Π_{CO} which is $\frac{1}{2}$-secure.

1. **Rounds 1-5:** Each honest node awake at time $0 \leq t \leq 4$ executes Π_{CGA} that starts at time $t = 0$ with its input v_i. Furthermore, if awake at time $t = 4$, after obtaining the output (o_i, g_i) from Π_{CGA}, each honest node multicasts $\langle \mathsf{leader}, \mathsf{VRF}(\mathsf{p}, t), o_{\mathsf{p}} \rangle$.
2. **Round 6.** After obtaining the output (o_i, g_i) from Π_{CGA}, let *val* be the largest value of $\mathsf{VRF}(\mathsf{p}, 4)$ observed from all received leader messages, and let $\langle \mathsf{leader}, \mathsf{VRF}(\mathsf{p}, 4), v_{\mathsf{p}} \rangle$ be the witness message. Each node awake at time ≥ 5 does the following:
   - If $g_i = 1$, output o_i.
   - Else, output v_{p}.

We now prove security.

Lemma 8. *Π_{CO} is a $\frac{1}{2}$-secure CO protocol.*

Proof. Let $\mathcal{A}$ be some $\frac{1}{2}$-bounded adversary. Π_{CO} is clearly $\frac{1}{2}$-terminating. For validity, assume all honest nodes awake at $t = 0$ input the same value v. Then, due to Π_{CGA} being $\frac{1}{2}$-valid, all nodes awake at time ≥ 4 obtain output $(v, 1)$ from Π_{CGA}, which in turn implies that they output v from Π_{CO}, as required. Lastly, for $\frac{1}{2}$-probabilistic agreement, we prove that when the highest $\mathsf{VRF}(\cdot, t)$ value is attained by an honest node q awake at $t = 4$ (an event that occurs with probability at least $\frac{1}{2}$, by the honest majority assumption and the modeling of the VRF as a random oracle), all honest nodes awake at time ≥ 5 output the same value. We now have two cases.

- There exists an honest p, awake at time ≥ 5, that has $g_{\mathsf{p}} = 1$ and outputs o_{p}. By the consistency of Π_{CGA}, all honest nodes awake at time ≥ 4 output $(o_{\mathsf{p}}, *)$ from Π_{CGA}. In particular, q has $o_{\mathsf{q}} = o_{\mathsf{p}}$. Thus, all honest nodes awake at time ≥ 5 receive the leader message from q, and by our conditioning, $\mathsf{VRF}(q, 4)$ is the highest VRF value seen by all honest nodes awake at time ≥ 5. Therefore, all honest nodes output $o_{\mathsf{q}} = o_{\mathsf{p}}$, as required.
- No honest node awake at time ≥ 5 outputs grade 1 from Π_{CGA}, in which case all honest nodes output from Π_{CO} according to item 2 in Round 6. Since all honest nodes awake at time ≥ 5 receive the leader message from q, and by our conditioning, $\mathsf{VRF}(q, 4)$ is the highest VRF value seen by all honest nodes awake at time ≥ 5, all honest nodes awake at time ≥ 5 output o_{q}, as required.

□

We are now ready to describe our BA protocol, Π_{BA}. The protocol consists of interlacing executions of CGA and CO. Intuitively, the CGAs purpose is to detect when the network is in a favorable condition (all honest nodes hold the same value), and trigger a decision, while the goal of the CO is twofold in attempting to bring the network to a favorable condition and preserving preexisting agreement in case some honest node decided in the preceding CGA. More formally, our protocol proceeds in iterations where iteration i consists of an execution of CGA, denoted by $\mathsf{CGA}[i]$, followed by a CO, denoted by $\mathsf{CO}[i]$. Formally, the protocol Π_{BA} is as follows. Denote by L_c, L_{ga} the constant latencies of $\Pi_{\mathsf{CO}}, \Pi_{\mathsf{CGA}}$, respectively. Denote by t the current timeslot. The following is executed by every honest node p with input v_{p} whenever it is awake. See fig. 2 for an illustration of the structure of the protocol.

1. Compute $i \leftarrow \lfloor t/(L_c + L_{ga}) \rfloor$.
2. While true:
   (a) If $t \in [(L_c + L_{ga})i, (L_c + L_{ga})i + L_{ga}]$, run as in $\mathsf{CGA}[i]$ with input being the output of $\mathsf{CO}[i-1]$ or v_{p} if $i = 0$.
   (b) If $r \in [(L_c + L_{ga})i + (L_{ga} + 1), (L_c + L_{ga} + 1)i]$, let (o_i, g_i) be the output obtained from $\mathsf{CGA}[i]$.

i. If $g_i = 1$, then output o_i as BA decision. Participate in the protocol for one more iteration with inputs to all subroutines being fixed to o_i, and then halt.
ii. Else, continue.

(c) If $r \in [(L_c + L_{ga})i + (L_{ga} + 1), (L_c + L_{ga} + 1)i]$ run as in $\mathsf{CO}[i]$ with input being p's output in $\mathsf{CGA}[i]$.

Our goal for the reminder of the section is to prove the following theorem.

Theorem 2. *Π_{BA} is a $\frac{1}{2}$-secure BA protocol. Furthermore Π_{BA} satisfies deterministic synchronization, has message complexity of $O(n^3)$ and expected latency of $O(\Delta)$ against any $\frac{1}{2}$-bounded adversary.*

Proof. We begin with validity. Assume that all honest nodes awake at time $t = 0$ hold the same input value v. This implies by the validity of CGA that all honest nodes awake at time ≥ 4 output $(v, 1)$ from $\mathsf{CGA}[0]$. Then, all honest nodes awake at time ≥ 4 output v, as required. Next, for agreement, let p be the first honest node to decide a value v. This implies that there exists an iteration i for which p outputs $(v, 1)$ from $\mathsf{CGA}[i]$, and no other honest node outputs from CGA with grade 1 prior to that. Note that this implies (by $\frac{1}{2}$-consistency of Π_{CGA}) that any honest node that outputs from $\mathsf{CGA}[i]$ outputs either $(v, 0)$ or $(v, 1)$. As even the nodes that output $(v, 1)$ continue to participate in the protocol for an additional iteration, security guarantees required for $\mathsf{CO}[i]$ and $\mathsf{CGA}[i + 1]$ are satisfied. Moreover, by the $\frac{1}{2}$-validity of CO, all honest nodes that output from $\mathsf{CO}[i]$, output v. Now, by $\frac{1}{2}$-validity of CGA, we have that all honest nodes that output from $\mathsf{CGA}[i + 1]$ output $(v, 1)$, and thus output v. In conclusion, all honest nodes that output a value, output the same value v. For termination, we prove the following claim.

Lemma 9. *For any $i \geq 1$, with probability at least $1 - \frac{1}{2^i}$, all honest nodes awake at or after the end of $\mathsf{CGA}[i]$ decide a value.*

Proof. We prove this by induction on i. By $\frac{1}{2}$-probabilistic agreement, w.p. at least $\frac{1}{2}$, all honest nodes that output from $\mathsf{CO}[0]$, output the same value, and thus by $\frac{1}{2}$-validity of $\mathsf{CGA}[1]$, all nodes awake at or after the end of $\mathsf{CGA}[1]$ immediately decide a value. Now we assume the statement holds for up to iteration i, and prove it for iteration $i + 1$. Observe that each invocation of CO uses independent randomness from all previous invocations, and thus we get by the $\frac{1}{2}$-probabilistic agreement property of $\mathsf{CO}[i]$, that conditioned on no honest node deciding up to the end of iteration i, w.p. at most $\frac{1}{2}$, not all honest nodes commence $\mathsf{CGA}[i + 1]$ with the same value. Thus, using the induction assumption we get that the probability that not all honest nodes commence $\mathsf{CGA}[i + 1]$ with the same value is upper bounded by $\frac{1}{2} \cdot \frac{1}{2^i} = \frac{1}{2^{i+1}}$. This implies that w.p. at least $1 - \frac{1}{2^{i+1}}$, all honest nodes commence $\mathsf{CGA}[i + 1]$ with the same value, and decide at its end. □

Lemma 9 proves $\frac{1}{2}$-termination, *i.e.*, w.p. 1, there exists T such that any honest node awake at time $\geq T$ outputs and halts. Furthermore, Lemma 9, combined

with the behavior of the algorithm, gives that $\Pr[T > \text{end of iteration } i] \leq \frac{1}{2^{i-1}}$ for all $i \geq 1$. Thus, as required, we get that

$$\mathbb{E}[T] = \sum_{j=0}^{\infty} \Pr[T > j] = O(\Delta) \sum_{i=0}^{\infty} \Pr[T > \text{end of iteration } i]$$
$$< O(\Delta) \cdot (1 + \sum_{i=1}^{\infty} \frac{1}{2^{i-1}}) = O(\Delta)$$

For deterministic synchronization, consider the first honest node to decide a value (*e.g.*, v) at some iteration i. Then, this node must have output v with grade 1 at the end of $\mathsf{CGA}[2i-1]$ at time $t = 4\Delta + 9\Delta(i-1)$. Then, for any time $s \geq t$, all honest nodes awake at time s also output v from $\mathsf{CGA}[2i-1]$, either with grade 1, thus deciding v at iteration i, or with grade 0, by the graded agreement of CGA. Then, all honest nodes awake at time t input it to $\mathsf{CGA}[2i]$. By the validity of CGA, for any time $s \geq t + 4\Delta$, all honest nodes awake at time s output v with grade 1 for $\mathsf{CGA}[2i]$. Therefore, all honest nodes awake at time $t + 5\Delta$ input the same value v to $\mathsf{CGA}[2i+1]$, and by the validity of CGA, for any time $s \geq t + 9\Delta$, all honest nodes awake at time s output v with grade 1 from $\mathsf{CGA}[2i]$, thus deciding at iteration $i+1$, within as early as 9Δ time of the first deciding honest node.

Finally, $O(n^2)$ message complexity follows from the $O(n^2)$ message complexity of CGA. □

7 DACS and Atomic Broadcast

Finally, we describe our DACS and atomic broadcast protocols. By Definition 4, DACS has censorship resistance, and thus, its censorship resistance parameter corresponds to its latency.

7.1 DACS

We first showcase a DACS protocol with $O(\log N)$ expected latency. Afterwards, we demonstrate how this protocol can be modified with a novel trick to obtain a DACS protocol with $O(\Delta)$ expected latency. The DACS protocol Π_{DACS} follows directly from Byzantine agreement and formally proceeds as follows:

1. **Round 1:** Each honest node p awake at time 0 with input v multicasts $\langle \mathsf{content}, z \rangle_{\mathsf{p}}$.
2. **Round ≥ 2.** Each honest node p awake at time $t \geq \Delta$ participates in N instances of BA, denoted by $\mathsf{BA}[i]$, that commence at $t = \Delta$. Each node p inputs a value z_i to the instance $\mathsf{BA}[i]$ if p received a unique message $\langle \mathsf{content}, z_i \rangle_i$ from node i. Otherwise, it inputs $\perp$.
3. **Decision.** Let $v_1^{\mathsf{p}}, \ldots, v_N^{\mathsf{p}}$ denote the outputs of an honest node p once it decides and terminates all of the N BA instances. Node p outputs the set $S_{\mathsf{p}} = \{v_1^p, \ldots, v_N^p\}$.

We now prove the following.

Lemma 10. *Π_{DACS} is a $\frac{1}{2}$-secure* DACS *protocol, satisfies deterministic synchronization, has $O(\Delta \log n)$ expected latency and $O(Nn^2)$ message complexity.*

Proof. We prove the security, deterministic synchronization, latency and complexity claims below.

Security: By 1/2-agreement of BA, there exists a value $v_i \in \mathcal{V}$ for each $i \in [N]$ such that each honest node that outputs a value for $\mathsf{BA}[i]$, outputs v_i. Therefore, there exists a set $S = \{v_1, \ldots, v_N\}$ such that all honest nodes that output a set, output S, proving 1/2-agreement for DACS. For validity, let I^* denote the set of indices i such that node i is honest and awake at time $t = 0$. Note that for all $i \in I^*$, all honest nodes input the same value v_i multicast by node i to $\mathsf{BA}[i]$. Then, by 1/2-validity of BA, all honest nodes that output a value for $\mathsf{BA}[i]$, output v_i, implying that for any honest node that outputs a set S, $v_i \in S$. This shows 1/2-validity for DACS.

Finally, since by 1/2-termination, for each $\mathsf{BA}[i]$, $i \in [N]$, there exists a time T_i such that every honest node awake at any time $\geq T_i$ outputs and halts in $\mathsf{BA}[i]$, there exists a time $T = \max_{i \in [N]}(T_i)$ such that every honest node awake at any time $\geq T$ outputs and halts in DACS. This shows 1/2-termination for DACS.

Latency: Let n denote the total number of awake nodes at time $t = 0$. Note that the termination of DACS requires the completion of all of the N BA instances. For the (honest) nodes p that are asleep at time $t = 0$, each honest node inputs $\perp$ to the associated BA instance. Similarly, for each honest node p that is awake at time $t = 0$, each honest node inputs the unique value v_{p} multicast by p to the instance $\mathsf{BA}[\mathsf{p}]$. Therefore, any awake honest node decides these BA instances after one round of CGA, *i.e.*, by time $t = 4\Delta$ time, and terminates by time $t = 13\Delta$. As for the other $k < n/2$ BA instances, termination is only guaranteed after the election of an honest leader. Since the leader of each BA instance is selected independently from the other instances and is honest with probability 1/2, latency of these k instances are upper bounded by independent identically-distributed random variables X_i, $i \in [k]$, which take value $(13+9j)\Delta$, $j \geq 0$, with probability 2^{-j}. Given the geometric random variables X_i, the expected value of the maximum of these random variables can be found as

$$\mathbb{E}[\max_{i \in [k]}(X_i)] = 13\Delta + 9\Delta\mathbb{E}[\max_{i \in [k]}(Y_i)] \leq 13\Delta + 9\Delta\mathbb{E}[\max_{i \in [n]}(Y_i)] = O(\log(n)\Delta),$$

which gives $O(\log(n)\Delta)$ expected latency.

Deterministic synchronization: An honest node outputs a set only after it decides all of the N BA instances $\mathsf{BA}[i]$, $i \in [N]$. As BA satisfies deterministic synchronization with a delay of at most 9Δ, once an honest node decides a value for a BA instance at some time t, for any time $s \geq t + 9\Delta$, all honest nodes awake at time s decide a value for the same BA. Therefore, if the first honest node to decide a value for all of the N instances $\mathsf{BA}[i]$, $i \in [N]$, does so by some time t, then, for any time $s \geq t + 9\Delta$, all honest nodes awake at time s decide a

value for all of these BA instances, *i.e.*, outputs a set. This shows deterministic synchronization for DACS with a delay of 9Δ.

Message complexity: After round 1, which incurs $O(n)$ message complexity, the awake nodes participate in N concurrent BA instances. Since each BA instance has message complexity of $O(n^2)$, DACS has a total message complexity of $O(Nn^2)$. □

7.2 Reducing the Latency of DACS

The DACS protocol has logarithmic latency due to the independence of the leader election processes within the concurrent BA instances. Therefore, to achieve constant expected latency, we make the leader election process *correlated* across the BA instances. In other words, at each time $t = 8\Delta + 9\Delta i$, $i \geq 0$, a *single* node p is selected to act as the leader in *all* of the CO protocols across *all* BA instances that have not terminated in p's view. More formally, honest nodes when electing a leader for each CO instance, send only a single VRF value, that applies for all CO instances simultaneously, and each honest node deems the highest VRF value observed to be the leader for all CO instances. Then, we can assert the following.

Lemma 11. *Π_{DACS} with correlated leaders is a $\frac{1}{2}$-secure DACS protocol with deterministic synchronization, $O(\Delta)$ expected latency and $O(Nn^2)$ message complexity.*

Proof. Proof of security, deterministic synchronization and message complexity claims is identical to the proof of Lemma 10.

As for latency, all awake honest nodes again decide the BA instances corresponding to the asleep and awake honest nodes after one round of CGA, *i.e.*, by $t = 4\Delta$ and terminate by $t = 13\Delta$. As for the other $k < n/2$ BA instances, termination is again only guaranteed after the election of an honest leader. Since all BA instances have the same leader at any given time $t = 8\Delta + 9\Delta i$, $i \geq 0$, and this leader is honest with probability $1/2$, latency of all of these k instances is upper bounded by the same random variable X, which takes value $(13+9j)\Delta$, $j \geq 0$, with probability 2^{-j-1}. Then, the expected latency is constant and can be found as

$$\mathbb{E}[X] = 13\Delta + 9\Delta = 22\Delta$$

□

7.3 Reducing the Message Complexity of DACS

The DACS protocol incurs $O(Nn^2)$ message complexity since it entails running N concurent BA instances, each with complexity $O(n^2)$. To reduce this to $O(n^3)$, we borrow a trick from HotStuff [YMR+19b] and use aggregate signatures. This is obtained by modifying the PKI setup, as follows. There are now N public verification keys for each of the N nodes, one for each of the N possible BA instances (*i.e.*, the PKI contains N^2 keys). With these keys, the nodes can aggregate their signatures on the echo and vote messages for the value $v = \perp$

across all N BA instances. For efficient verification, they attach a bit map to the aggregate signature to indicate the public keys of the instances that contributed to the signature[12].

Equipped with the aggregate signatures, we modify the BA and CGA protocols run by an honest node as follows. For simplicity, we only state the differences compared to the original BA and CGA protocols.

Changes in BA: In any BA instance $\mathsf{BA}[i]$, where the node p multicasts a leader message with a value $z \neq \perp$, it attaches a *justification* $\langle \mathsf{content}, z \rangle_i$ to the message: $\langle \mathsf{leader}, \mathsf{VRF}(\mathsf{p}, 4), z, \langle \mathsf{content}, z \rangle_i \rangle_{\mathsf{p}}$. For the BA instances, where the node would like to multicast a leader message with the value $z = \perp$, it creates a message $\langle \mathsf{leader}, \mathsf{VRF}(\mathsf{p}, 4), \perp \rangle_{\mathsf{p}}$, aggregates these messages across the instances and multicasts a single aggregate leader message for the value $\perp$ (recall that at any given iteration, the leader is the same across all BA instances). Non-justified leader messages, *i.e.*, leader messages for values $v \neq \perp$ without a justification are treated as *invalid* and ignored.

Changes in CGA:

1. **Round 1: Echo.** Suppose that the input to CGA at some instance $\mathsf{BA}[i]$ is $z \neq \perp$. Then, the node sends the echo message $\langle \mathsf{echo}, z, \langle \mathsf{content}, z \rangle_i \rangle$ augmented by the justification $\langle \mathsf{content}, z \rangle_i$ within the instance $\mathsf{BA}[i]$. For the BA instances where the input to CGA is $z = \perp$, it creates an echo message for the value $\perp$, aggregates these messages across these BA instances and multicasts a single $\mathsf{echo}_\perp$ message. Within each $\mathsf{BA}[i]$, non-justified echo messages, *i.e.*, echo_v for values $v \neq \perp$ without a justification, and the tally and group messages containing non-justified echo_v with $v \neq \perp$ are treated as *invalid* and ignored. Similarly, any echo message sent by a node q for the value $\perp$ that is not an aggregate of the $\mathsf{echo}_\perp$ messages for the N instances is treated as *invalid* and ignored. If a node sends two different aggregate $\mathsf{echo}_\perp$ messages, it is treated as an equivocating node.
2. **Round 2: Tally.** For each instance $\mathsf{BA}[i]$, each awake node constructs the set $S = \{F_v \mid v \in \mathcal{V}, |F_v| \neq 0\}$ as usual, except that $F_\perp$ is the same across all instances and contains the same aggregate $\mathsf{echo}_\perp$ messages. The node multicasts $F_\perp$ once, rather than for each BA instance.
3. **Round 3: Group.** For each instance $\mathsf{BA}[i]$, each awake honest node constructs the dictionary D as usual, except that $D[\perp]$ is the same across all instances and contains the same aggregate $\mathsf{echo}_\perp$ messages by the non-equivocating nodes. The node multicasts $D[\perp]$ once, rather than for each BA instance.
4. **Round 4: Vote.** Each awake honest node aggregates all vote messages for the value $\perp$ across all BA instances into a single message $\mathsf{vote}_\perp$. The node multicasts $\mathsf{vote}_\perp$ once, rather than for all BA instances. It also attaches the justification $\langle \mathsf{content}, v \rangle_i$ to its vote messages for $v \neq \perp$ within the instance $\mathsf{BA}[i]$: $\langle \mathsf{vote}, v, \langle \mathsf{content}, v \rangle_i \rangle$. Within $\mathsf{BA}[i]$, vote messages for the values $v \neq \perp$ without a justification are treated as invalid and ignored. Similarly, any

[12] Although the aggregate signature with the bit map has size $O(\lambda) + N$, in practice, $N = O(\lambda)$.

vote message sent by a node q for the value $\perp$ that is not an aggregate of the $\mathsf{vote}_\perp$ messages for the BA instances is treated as *invalid* and ignored.

5. **Round 5: Output.** Same as in the original CGA protocol.

Finally, we argue the security of the modified CGA, BA and DACS protocol.

Theorem 3. *Π_{DACS} equipped with correlated leaders and the modified* CGA *protocol is a $\frac{1}{2}$-secure* DACS *protocol with deterministic synchronization, $O(\Delta)$ expected latency and $O(n^3)$ message complexity.*

Proof. We first argue by induction that an honest node can always justify the leader, echo and vote messages for the values $v \neq \perp$. In the first CGA of a BA instance $\mathsf{BA}[i]$, an honest node sets its input as a value $z \neq \perp$ only upon observing a unique message $\langle \mathsf{content}, z \rangle_i$, implying that it can justify its echo message for z. In the later CGA instances within $\mathsf{BA}[i]$, the input is set to be either the output value z of an earlier CGA instance, or the value z multicast in a leader message. In the first case, the output value of the earlier CGA must be the input value of an honest node by the integrity of CGA, and by an inductive argument, this honest node can justify its echo message for the value z within the earlier CGA instance. Hence, all honest awake nodes observe the justification for z by the time the latter CGA instance commences. By the same argument, honest nodes can always justify their leader messages, since the values multicast in these messages are set as the output of the previous CGA instance. In the second case, an honest node sets its input to the latter CGA as z only if the leader message is justified, implying that the honest node observes the justification for z by the time the latter CGA instance commences. Finally, by the proof of Lemma 3, an honest node sends a vote message for the value z only if it observes justified echo messages for this value, implying that the node observes the justification for z by the time it sends the vote message.

Next, we note that adding justifications to echo and vote messages does not change the security analysis of CGA, BA or DACS; since honest nodes never send non-justified messages, and the non-justified messages sent by the corrupt nodes can simply be treated as messages that were never sent. Similarly, considering a node that sends two different aggregate $\mathsf{echo}_\perp$ messages as an equivocating node does not change the security analysis of CGA, BA or DACS; since honest nodes never send different aggregate $\mathsf{echo}_\perp$ messages, and the corrupt nodes that sent those messages can be treated as nodes that sent equivocating messages, as these messages serve as evidence of equivocation. Therefore, proofs of the security and latency claims are identical to the proofs in Lemma 10 and Lemma 11.

As for message complexity, the BA instances corresponding to the corrupt and honest nodes awake at time $t = 0$ incur a total message complexity of $O(n^3)$. In the remaining BA instances for the honest nodes asleep at time $t = 0$, no leader, echo or vote message for a value $v \neq \perp$ can be justified. Therefore, covering all of these instances, all honest nodes awake at times $t = \Delta, 2\Delta$ or 3Δ send a *single* aggregated $\mathsf{echo}_\perp$ message at time $t = \Delta$, a *single* tally message with $F_\perp$ containing $O(n)$ aggregated $\mathsf{echo}_\perp$ messages, and a single group message with $D[\perp]$, containing $O(n)$ aggregated $\mathsf{echo}_\perp$ messages, at times $t = 2\Delta$ and

3Δ respectively. Similarly, all honest nodes awake at time $t = 4\Delta$ send at most a single aggregated vote message for the value $\perp$. In these instances, leader, echo and vote messages for values $v \neq \perp$ as well as the tally and group messages that contain them are deemed invalid and ignored[13]. Finally, honest awake nodes send a single aggregate leader message across these instances. Therefore, the total message complexity of these instances is collectively equal to the complexity of a single instance, *i.e.*, $O(n^2)$, making the total complexity $O(n^3)$. □

7.4 Atomic Broadcast

The atomic broadcast (ABC) protocol Π_{ABC} is comprised of consecutive DACS instances, each commencing 4Δ time after its predecessor. Here, $\mathsf{DACS}[i]$ indicates the i-th DACS instance starting at $i = 0$. Let $\mathsf{L}_r^{\mathsf{p}}$ denote the log in the view of an honest node p at time t and $\mathsf{Input}_t^{\mathsf{p}}$ denote the set of values input to p by time t. Then, at each time t, each honest node p awake at t does the following (let $k := \lceil t/(4\Delta) \rceil$):

1. Node p participates as instructed in the instances $\mathsf{DACS}[i]$, $i \in \{0, \ldots, k\}$, that have not yet terminated in its view.
2. Let m_{p} denote the largest number such that p decided all of the instances $\mathsf{DACS}[i]$, $i \in \{0, 1, \ldots, m_{\mathsf{p}}\}$, by time t. Let S_i, $i \in \{0, 1, \ldots, m_{\mathsf{p}}\}$, denote the set of values output by p for the instance $\mathsf{DACS}[i]$, $i \in \{0, 1, \ldots, m_{\mathsf{p}}\}$. Define $\hat{S}_i$ as the sequence obtained by ordering the values within S_i according to some deterministic rule (*e.g.*, by the output of some cryptographic hash function applied to the values). Then, p sets $\mathsf{L}_r^{\mathsf{p}}$ as $(\hat{S}_0, \ldots, \hat{S}_{m_{\mathsf{p}}})$.
3. If $t = 4k$, then p orders the values within $\mathsf{Input}_r^{\mathsf{p}} \setminus \left(\cup_{i=0}^{m_{\mathsf{p}}} S_i\right)$ according to some deterministic order and inputs this sequence as its 'value' to the instance $\mathsf{DACS}[k]$.

Theorem 4. *Π_{ABC} is a $\frac{1}{2}$-secure ABC protocol with deterministic synchronization, $O(\Delta)$ expected latency and $O(n^3)$ expected message complexity.*

Proof. We prove the security, deterministic synchronization, latency and complexity claims below.

Deterministic synchronization: Since DACS satisfies deterministic synchronization with a delay of 9Δ, if an honest node p outputs a log $\mathsf{L}_t^{\mathsf{p}} = (\hat{S}_0, \ldots, \hat{S}_\ell)$ at time t, *i.e.*, decides the instances $\mathsf{DACS}[i]$, $i \in \{0, \ldots, \ell\}$, by time t, then for any time $s \geq t + 9\Delta$, all honest nodes q awake at time s must have decided the instances $\mathsf{DACS}[i]$, $i \in \{0, \ldots, \ell\}$, by time s, and thus output a log that extends $\mathsf{L}_t^{\mathsf{p}}$ at time s, *i.e.*, $\mathsf{L}_t^{\mathsf{p}} \preceq \mathsf{L}_s^{\mathsf{q}}$. This proves deterministic synchronization for Π_{ABC} with latency 9Δ.

[13] Although the corrupt nodes might attempt to flood the honest ones with invalid messages, we can stipulate each node to list the aggregated echo and vote messages for $\perp$ at the first position within the network package, followed by the justification of other messages; and thus enable the honest nodes to ignore the invalid messages without downloading much data.

Security: By 1/2-agreement of DACS, for any two honest nodes p and q, the output of the instances $\mathsf{DACS}[i]$, $i \in \{0, 1, \ldots, \min(m_{\mathsf{p}}, m_{\mathsf{q}})\}$, is the same for both nodes. Therefore, for any two honest nodes p and q awake at times t and s respectively, it holds that either $\mathsf{L}_t^{\mathsf{p}} \preceq \mathsf{L}_s^{\mathsf{q}}$ or vice versa, *i.e.*, Π_{ABC} satisfies 1/2-safety.

For liveness, consider a value v input to the honest nodes at some time t. Since at all times, there is an awake honest node that includes v as part of its input value, either (i) v will be part of the value input to the instance $\mathsf{DACS}[i^*]$ by some honest node p awake at time $4i^*$, where $i^* = \arg\min_{4i>t}(i)$, or (ii) for this node, $v \in \mathsf{L}_{4i^*}^{\mathsf{p}}$. Next, we consider the following cases:

- **Case (i):** Recall that DACS satisfies 1/2-termination, and more specifically its latency is upper bounded by the random variable T that takes the value $(13 + 9j)\Delta$, $j \geq 0$, with probability 2^{-j-1}. Next, we observe that there exists a random variable T with constant expectation, *i.e.*, $\mathbb{E}[T] = O(\Delta)$, such that for any time $s > 4i^* + T$, all honest nodes awake at time s decides all instances $\mathsf{DACS}[i]$, $i \in \{0, 1, \ldots, i^*\}$, by time s (the precise calculation of $\mathbb{E}[T]$ is presented below). Now, by 1/2-validity of DACS, $v \in S_{i^*}$. Hence, for any time $s > 4i^* + T$ and any honest node q awake at time s, $v \in \mathsf{L}_s^{\mathsf{q}}$.
- **Case (ii):** For any time $s > 4i^* + 9\Delta$, for any honest node q awake at time s, $v \in \mathsf{L}_s^{\mathsf{q}}$ by deterministic synchronization.

This implies that Π_{ABC} satisfies 1/2-liveness with constant expected latency.

Upper-bounding the expected latency:

Termination time of the DACS instances is upper bounded by independent, identically-distributed random variables X_i which take value $(13 + 9j)\Delta$, $j \geq 0$, with probability 2^{-j-1}. Let T_i denote the random variable $\max_{j \in \{0,1,\ldots,i\}}(X_j - 4\Delta(i-j))$, and observe that T_i upper-bounds the difference between the time $4\Delta i$ (*i.e.*, the start time of the instance $\mathsf{DACS}[i]$), and the time all DACS instances $\mathsf{DACS}[j]$, $j \in \{0, 1, \ldots, i\}$, terminate. Define D_k as $(X_{i-k} - 4\Delta k)$ for $k \in \{0, 1, \ldots, i\}$. We note that for any time $t \geq 4\Delta i + 13\Delta$,

$$\Pr[D_k \leq t] \geq 1 - 2^{-\lfloor 1 + (t - 4\Delta i - 13\Delta + 4\Delta k)/9\Delta \rfloor}.$$

For $t \geq 4\Delta i + 13\Delta$, renaming $s = (t - 4\Delta i - 13\Delta)/\Delta$, we can write

$$\Pr[D_k \leq t] \geq 1 - 2^{-(s\Delta + 4\Delta k)/9\Delta} = 1 - 2^{-(s+4k)/9}.$$

Therefore, there exists a value $q = 1/2$ such that $\Pr[D_k \leq s\Delta + 4\Delta i + 13\Delta] \geq 1 - q^{(s+4k)/9}$ for $s \geq 0$. This implies

$$\begin{aligned}
\mathbb{E}[T_i] &= \sum_{t=4\Delta i}^{\infty} (1 - \Pr[T_i \leq t]) = \sum_{t=0}^{\infty} (1 - \prod_{k=0}^{i} \Pr[D_k \leq t]) \\
&\leq 13\Delta + \Delta \sum_{s=0}^{\infty} (1 - \prod_{k=0}^{i} (1 - q^{(s+4k)/9})) \\
&\leq 13\Delta + \Delta \sum_{s=0}^{\infty} (1 - \prod_{j=0}^{\infty} (1 - q^{(s+4k)/9}))
\end{aligned}$$

for all $i \geq 0$.

As $1 - \frac{1}{2}x \geq e^{-x}$ for all $x \in [0, 1]$, it holds that

$$\prod_{k=0}^{\infty}(1 - q^{(s+4k)/9}) \geq \prod_{k=0}^{\infty} e^{-2q^{(s+4k)/9}} = e^{-2q^{(s/9)}\sum_{k=0}^{\infty} q^{(4k/9)}} = e^{-2q^{(s/9)}\frac{1}{1-q^{(4/9)}}},$$

Combined with $x \geq 1 - e^{-x}$ for $x \geq 0$, this implies that for all $i \geq 0$,

$$\begin{aligned}\mathbb{E}[T_i] &\leq 13\Delta + \Delta\sum_{s=0}^{\infty}(1 - e^{-2q^{(s/9)}\frac{1}{1-q^{(4/9)}}}) \\ &\leq 13\Delta + \frac{2\Delta}{1-q^{(4/9)}}\sum_{s=0}^{\infty} q^{(s/9)} = 13\Delta + \frac{2\Delta}{(1-q^{(4/9)})(1-q^{(1/9)})},\end{aligned}$$

which is constant in Δ.

Expected message complexity:

Expected message complexity is equal to the expected number of DACS instances that can be running at any point in time multiplied with the complexity of the DACS instance itself. Consider the instances DACS$[k]$, $k \in \{0, 1, \ldots, i\}$, for some $i \geq 0$. Define I_k as the indicator random variable that DACS$[i-k]$ is still running at time $4\Delta i$ and M_i as the number of DACS instances still running at time $t = 4\Delta i$. Note that

$$\mathbb{E}[M_i] = \mathbb{E}\left[\sum_{k=0}^{i} I_k\right] = \sum_{k=0}^{i}\mathbb{E}[I_k] = \sum_{k=0}^{i}\Pr[D_k \geq 4\Delta i].$$

Since for $k > 4$,

$$\Pr[D_k \geq 4\Delta i] \leq 2^{-(4\Delta k - 13\Delta)/9\Delta} = 2^{-(4k-13)/9},$$

it holds that $\sum_{k=0}^{i}\Pr[D_k \geq 4\Delta i]$ is monotonically increasing in i, and,

$$\lim_{i\to\infty}\sum_{k=0}^{i}\Pr[D_k \geq 4\Delta i] \leq 4 + \lim_{i\to\infty}\sum_{k=5}^{i} 2^{-(4k-13)/9},$$

which is a constant. Therefore, $\mathbb{E}[M_i] < C$ for some constant C for all $i \geq 0$, and the expected message complexity of atomic broadcast is $O(n^3)$. □

We note that in the optimistic case, where all awake nodes are honest, the protocol achieves an expected latency of 4Δ, and there is a single DACS instance running at all times.

Acknowledgments. We thank Joachim Neu for many fruitful discussions and Max Resnick for bringing this problem to our attention. ENT is supported by the Stanford Center for Blockchain Research.

References

AMN+20. Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync HotStuff: Simple and practical synchronous state machine replication. In *SP*, pages 106–118. IEEE, 2020.

BGK+18. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *CCS*, pages 913–930. ACM, 2018.

BGR24. Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction fee mechanism design in a post-mev world. *IACR Cryptol. ePrint Arch.*, page 331, 2024.

BHK+20. Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining GHOST and Casper. arXiv:2003.03052v3 [cs.CR], 2020.

BKM18. Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. arXiv:1807.04938v3 [cs.DC], 2018.

BKR94. Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In James H. Anderson, David Peleg, and Elizabeth Borowsky, editors, *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, Los Angeles, California, USA, August 14-17, 1994*, pages 183–192. ACM, 1994.

BKT+19. Vivek Kumar Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *CCS*, pages 585–602. ACM, 2019.

BLR24. Eric Budish, Andrew Lewis-Pye, and Tim Roughgarden. The economic limits of permissionless consensus. *CoRR*, abs/2405.09173, 2024.

BT85. Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, October 1985.

But15. Vitalik Buterin. The problem of censorship, 2015.

CASD95a. Flaviu Cristian, Houtan Aghili, H. Raymond Strong, and Danny Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. *Inf. Comput.*, 118(1):158–179, 1995.

CASD95b. Flaviu Cristian, Houtan Aghili, H. Raymond Strong, and Danny Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. *Inf. Comput.*, 118(1):158–179, 1995.

CDG+24. Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. DARE to agree: Byzantine agreement with optimal resilience and adaptive communication. In *PODC*, pages 145–156. ACM, 2024.

CKPS01. Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and Efficient Asynchronous Broadcast Protocols. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 524–541. Springer, 2001.

CL99. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186. USENIX Association, 1999.

CS20. Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT*, pages 1–11. ACM, 2020.

DGKR18. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT (2)*, volume 10821 of *LNCS*, pages 66–98. Springer, 2018.

DLZ24. Francesco D'Amato, Giuliano Losa, and Luca Zanolini. Asynchrony-resilient sleepy total-order broadcast protocols. In *PODC*, pages 247–256. ACM, 2024.

DNTT22. Francesco D'Amato, Joachim Neu, Ertem Nusret Tas, and David Tse. No more attacks on proof-of-stake ethereum? *IACR Cryptol. ePrint Arch.*, page 1171, 2022. In Financial Cryptography and Data Security 2024.

DPS19. Phil Daian, Rafael Pass, and Elaine Shi. Snow White: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography*, volume 11598 of *LNCS*, pages 23–41. Springer, 2019.

DS83. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

DSTZ24a. Francesco D'Amato, Roberto Saltini, Thanh-Hai Tran, and Luca Zanolini. 3-slot-finality protocol for ethereum. *CoRR*, abs/2411.00558, 2024.

DSTZ24b. Francesco D'Amato, Roberto Saltini, Thanh-Hai Tran, and Luca Zanolini. Tob-svd: Total-order broadcast with single-vote decisions in the sleepy model, 2024.

DZ23a. Francesco D'Amato and Luca Zanolini. Recent latest message driven GHOST: balancing dynamic availability with asynchrony resilience. *IACR Cryptol. ePrint Arch.*, page 279, 2023. In CSF'24.

DZ23b. Francesco D'Amato and Luca Zanolini. Streamlining sleepy consensus: Total-order broadcast with single-vote decisions in the sleepy model. *CoRR*, abs/2310.11331, 2023.

FGKR18. Matthias Fitzi, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Parallel Chains: Improving throughput and latency of blockchain protocols via parallel composition. Cryptology ePrint Archive, Paper 2018/1119, 2018.

FPR23. Elijah Fox, Mallesh M. Pai, and Max Resnick. Censorship resistance in on-chain auctions. In *AFT*, volume 282 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

GKL15. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)*, volume 9057 of *LNCS*, pages 281–310. Springer, 2015.

GL23. Eli Gafni and Giuliano Losa. Brief announcement: Byzantine consensus under dynamic participation with a well-behaved majority. In *DISC*, volume 281 of *LIPIcs*, pages 41:1–41:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

GLR21a. Vipul Goyal, Hanjun Li, and Justin Raizes. Instant block confirmation in the sleepy model. In *Financial Cryptography (2)*, volume 12675 of *Lecture Notes in Computer Science*, pages 65–83. Springer, 2021.

GLR21b. Vipul Goyal, Hanjun Li, and Justin Raizes. Instant block confirmation in the sleepy model. In *Financial Cryptography (2)*, volume 12675 of *LNCS*, pages 65–83. Springer, 2021.

KK06. Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Annual International Cryptology Conference*, pages 445–462. Springer, 2006.

KRDO17. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO (1)*, volume 10401 of *LNCS*, pages 357–388. Springer, 2017.

LR23. Andrew Lewis-Pye and Tim Roughgarden. Byzantine generals in the permissionless setting. In *FC (1)*, volume 13950 of *LNCS*, pages 21–37. Springer, 2023.

LSP19. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.

MMR22a. Dahlia Malkhi, Atsuki Momose, and Ling Ren. Byzantine consensus under fully fluctuating participation. Cryptology ePrint Archive, Paper 2022/1448, Version 20221024:011919, 2022.

MMR22b. Dahlia Malkhi, Atsuki Momose, and Ling Ren. Instant finality in Byzantine generals with unknown and dynamic participation, 2022.

MMR23a. Dahlia Malkhi, Atsuki Momose, and Ling Ren. Towards practical sleepy BFT. In *CCS*, pages 490–503. ACM, 2023.

MMR23b. Dahlia Malkhi, Atsuki Momose, and Ling Ren. Towards practical sleepy BFT. In *CCS*, pages 490–503. ACM, 2023.

MR22a. Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. In *CCS*, pages 2295–2308. ACM, 2022.

MR22b. Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. In *CCS*, pages 2295–2308. ACM, 2022.

MXC+16. Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *CCS*, pages 31–42. ACM, 2016.

Nak08. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

NTT21. Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-Flow protocols: A resolution of the availability-finality dilemma. In *SP*, pages 446–465. IEEE, 2021.

Pra24. Pranav Garimidi. A look down ethereum's roadmap: The cases for focil and multi-proposer schemes, 2024.

PS17a. Rafael Pass and Elaine Shi. Rethinking large-scale consensus. In *CSF*, pages 115–129. IEEE Computer Society, 2017.

PS17b. Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT (2)*, volume 10625 of *LNCS*, pages 380–409. Springer, 2017.

SWN+21a. Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. Bft protocol forensics. In *Computer and Communication Security (CCS)*, Nov 2021.

SWN+21b. Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *CCS*, pages 1722–1743. ACM, 2021.

TMDM24. Thomas Thiery, Barnabe Monnot, Francesco D'Amato, and Julian Ma. Fork-choice enforced inclusion lists (focil): A simple committee-based inclusion list proposal, 2024.

YMR+19a. Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *PODC*, pages 347–356. ACM, 2019.

YMR+19b. Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: BFT Consensus with Linearity and Responsiveness.

In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.