Fully-Fluctuating Participation in Sleepy Consensus

Yuval Efron
Columbia University, New York, NY, USA

Joachim Neu
a16z Crypto Research, New York, NY, USA

Toniann Pitassi
Columbia University, New York, NY, USA

Abstract

Proof-of-work allows Bitcoin to boast security amidst arbitrary fluctuations in participation of miners throughout time, so long as, at any point in time, a majority of hash power is honest. In recent years, however, the pendulum has shifted in favor of proof-of-stake-based consensus protocols. There, the sleepy model is the most prominent model for handling fluctuating participation of nodes. However, to date, no protocol in the sleepy model rivals Bitcoin in its robustness to drastic fluctuations in participation levels, with state-of-the-art protocols making various restrictive assumptions. In this work, we present a new adversary model, called *external adversary*. Intuitively, in our model, corrupt nodes do not divulge information about their secret keys. In this model, we show that protocols in the sleepy model can meaningfully claim to remain secure against fully fluctuating participation, without compromising efficiency or corruption resilience. Our adversary model is quite natural, and arguably naturally captures the process via which malicious behavior arises in protocols, as opposed to traditional worst-case modeling. On top of which, the model is also theoretically appealing, circumventing a barrier established in a recent work of Malkhi, Momose, and Ren.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Security and privacy → Distributed systems security

Keywords and phrases Sleepy Consensus, fully-fluctuating dynamic Participation

Digital Object Identifier 10.4230/LIPIcs.AFT.2025.17

Acknowledgements We thank Joseph Bonneau, Javier Nieto, and Ling Ren for fruitful discussions.

1 Introduction

Byzantine-fault tolerant (BFT) consensus, the problem of reaching agreement on a value amongst n nodes, of which a fraction are corrupted and controlled by an *adversary*, is one of the most fundamental problems in distributed computing, with research spanning over four decades [19, 26]. The last decade has seen a significant growth in work and interest in the problem, mainly motivated by the pivotal role that consensus protocols play in the design and analysis of blockchains [23, 17, 7, 1, 25, 13].

Traditionally, it is assumed that apart from the corrupt nodes, all other nodes follow the protocol at all times. This assumption, while justified for applications such as data centers, is arguably optimistic in the context of blockchains, where the participants in the protocol may come and go at arbitrary points in time. Specifically, the desiderata for many blockchain protocols, such as Bitcoin and Ethereum, requires maintaining security even in the presence of *many unexpected temporary crash faults*, in which some subset of the nodes may go offline for an undetermined interval of time. In particular, the protocol may experience alternating periods of high participation and low participation.[1] The seminal

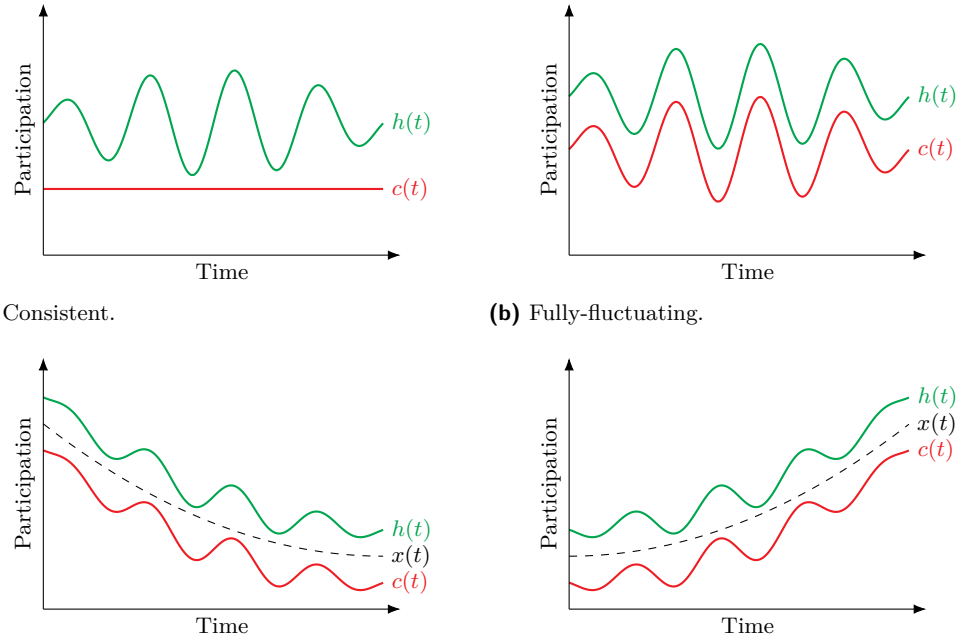
[1] Throughout, we assume a fixed set of *potential* participants *known* to the protocol; i.e., the set of nodes allowed to partake is fixed, and fluctuation in participation occurs w.r.t. that set. *Reconfiguration* of the nodes partaking in the protocol is orthogonal to the *unexpected* and *wide* fluctuations in participation considered here, and reconfiguration is expressly *not* considered in this work.

Bitcoin protocol [23], also known as proof-of-work Nakamoto consensus, was the first protocol to achieve secure consensus under such circumstances, allowing participants in the protocol to come and go as they please so long as an honest majority (of hash-based mining power) is maintained at any point in time. Its deftly formalized proof-of-work framework allowed for protocol design which is completely agnostic to the total number of participating nodes (corrupt or honest!) *across* present, past, or future. For example, if tomorrow the honest participation in Bitcoin drops to a level below the corrupt participation *today*, it would not render Bitcoin insecure (so long as corrupt participation decreases accordingly). The same holds for if corrupt participation *tomorrow* exceeds honest participation *today* (assuming honest participation increases accordingly).

As the years went on, proof-of-work ceded its spot in the limelight to proof-of-stake, which presently is the dominant approach to Sybil-resistance among blockchain protocols. In the translation, however, resilience to fluctuation in participation was lost. In the context of proof-of-stake, a formal model that captures fluctuating participation is the *sleepy* consensus model, introduced in [24] by Pass and Shi. This model is the focus of our work. In the sleepy model, there is a fixed set of n nodes, identified via a public key infrastructure (PKI). Besides being classified as *honest* or *corrupt*, nodes can alternate (adversarially) between two states at each point in time: *awake*, or *asleep*. Awake nodes can participate in the protocol normally, i.e., perform local computation, send and receive messages. On the other hand, asleep nodes cannot perform any computation and cannot engage in the protocol in any way, until they wake up. While a proof-of-stake instantiation of Nakamoto's protocol was proven secure [24] in the sleepy model, the proof assumes significant restrictions on fluctuating participation. Namely, that honest participation at *any* time exceeds corrupt participation at *all* times. Such a concession perhaps felt inherent, with many follow-up works in the sleepy model making the same assumption [24, 16, 12, 2, 25, 8, 13]. This intuition for the necessity of such an assumption was formalized in [21], which showed that indeed any consensus protocol that does not employ time-based cryptography (e.g., verifiable delay functions [5], henceforth VDF) must heavily restrict fluctuating participation.

The impossibility [21] is established by the following *key transfer* attack. Consider a consensus protocol that makes use of a PKI setup, where it is assumed that prior to the initiation of the protocol, all nodes were privately handed secret-key/public-key pairs. Pondering on the PKI assumption for a moment, however, one observes that the notion of an *asleep corrupt node* makes little sense, as any corrupt node that is awake for even a single point in time, and whose behaviour is completely controlled by the adversary, can simply broadcast its secret key to all other corrupt nodes, and then be put to sleep. Thus allowing essentially a *single* awake corrupt node to simulate a behaviour indistinguishable from all corrupt nodes being awake. Such an attack clearly renders a protocol vulnerable to the case where honest participation decreases over time.

What remains is perhaps an unfortunate state of affairs, in which proof-of-stake protocols seem to inherently carry the burden of rigid participation constraints. The goal of this work is to challenge this status quo. We do so by revisiting the adversary model. Specifically, [21] has already observed that such a strong adversary model capable of executing the key transfer attack may be a tad unrealistic: "The (standard) adversary assumption, where nodes can extract all corrupt nodes' private states, is too strong in practice. In particular in the proof-of-stake protocols, it is highly unlikely that corrupt nodes hand off their secret keys to the adversary (or other corrupt nodes) at the risk of losing their entire stake." [21] They conclude by raising the problem of achieving sleepy consensus under fully-fluctuating corrupt nodes in a "more realistic adversarial model."



(a) Consistent.

(b) Fully-fluctuating.

(c) Decreasing.

(d) Increasing.

Figure 1 Examples of possible participation rates for each of the discussed settings. The graphs depict the number of participating nodes at any point in time, with green curves depicting honest participation, and red curves depicting corrupt participation. Note that in all four settings, honest participation exceeds corrupt participation at any given point in time. On top of that, in the consistent case, note that *any* point on the honest curve dominates the corrupt curve at *all* points in time. Denoting by $h(t)$ and $c(t)$ the honest and corrupt participation at time t, the increasing (decreasing) setting can be characterized by the existence of a monotone increasing (decreasing) function $x(t)$ such that $c(t) < x(t) < h(t)$ holds at all times. The fully-fluctuating case poses no additional restrictions.

Motivated by similar principles, we consider in this work a mildly relaxed adversary model, which we refer to as an *external adversary*. Informally, an external adversary is external to the network and the protocol; the adversary can observe the protocol from a birds eye view. In turn, corrupt nodes are not inherently malicious, but opportunistic: they are recruited by the external adversary to perform arbitrary behavior in the protocol, but won't divulge information about their own secret key to any other node. We believe it is much more likely that a malicious entity would be able to "rent" keys in the protocol, by means of recruiting nodes to engage in Byzantine behavior, rather than convince nodes to give up their secret keys. Stating a formal model that accurately captures the above intuition turns out to be quite simple, but does requires some care, and we tend to it in Sec. 2. Not only does this model still capture a rich plethora of attack vectors pertinent in practice, but it is also theoretically appealing: Disallowing corrupt nodes to share information about their secret keys with one another is precisely the minimal relaxation required to render the key transfer attack infeasible.

The settings. Pondering on the notion of fluctuating participation of nodes in the protocol, one can identify several interesting settings to consider. See also Fig. 1.

1. **Consistent participation.** (Fig. 1a) This is the most common assumption in the context of the sleepy model [24, 16, 12, 2, 25, 8, 13], where it is assumed that new (i.e., after time 0) corrupt nodes may never wake up, and corrupt nodes may never go to sleep. Additionally, an honest majority is assumed to hold at all times. In other words, for *any* point time, honest participation exceeds corrupt participation at *all* times.
2. **Increasing participation.** (Fig. 1d) In this setting, participation of corrupt nodes is guaranteed to be non decreasing as time progresses. In other words, new corrupt nodes may wake up throughout the protocol, but may no go to sleep once woken up. Note that in this setting, even assuming an honest majority at any point in time, it could be that as time goes on, corrupt participation exceeds honest participation in *earlier* points in time. As a matter of fact, this setting has received a fair amount of attention recently [22, 21, 9, 10, 11], in several works showcasing efficient consensus protocols for this setting.
3. **Decaying participation.** (Fig. 1c) In this setting, participation of corrupt nodes is guaranteed to be non increasing as time progresses. In other words, corrupt nodes may go to sleep throughout the protocol, but can never be newly woken up. In this setting, even with an honest majority at any point in time, it could be that as time goes on, corrupt participation in the *past* exceeds *current* honest participation.
4. **Fully-fluctuating.** (Fig. 1b) This is the most general setting in the context of the sleepy model. In this setting, participation of both honest and corrupt nodes can both increase and decrease over time, to the discretion of the adversary. In particular, any of the scenarios mentioned in the previous settings can also occur in this setting.

1.1 Main results

Our results establish that assuming an external adversary allows protocols in the sleepy model to be endowed with resilience to fluctuating participation of nodes, rivaling Nakamoto in its graceful handling of unknown participation patterns.

(1) Sleepy consensus under decaying participation. Our first result establishes the utility of the external adversary model by showcasing a separation from the standard adversary model. Namely, consider the case where participation of both honest and corrupt nodes decreases over time. In particular, for any point in time, *future* honest participation may be eclipsed by *current* corrupt participation. Recent work [21] showed that in this setting, no protocol can be secure in the presence of an adversary that can carry out key transfer.

Our first result shows that against an external adversary, a consensus protocol resilient to decaying participation is not only feasible, but can be made highly efficient and requires only standard cryptographic assumptions.

▶ **Theorem** (Informal, see Thm. 6)**.** *Assuming a PKI, a VRF, and an external adversary, we present a randomized protocol solving consensus in the decaying participation setting with expected constant latency.*

(2) Consensus under fully-fluctuating participation. Thm. 6 establishes the strength of the external adversary model from the lens of protocol design, allowing us to obtain provable security guarantees known to be impossible under the standard notion of an adversary, that is able to carry out the key transfer attack. We now turn our attention to the more general setting of *fully-fluctuating* participation, in which participation of honest and corrupt nodes can both increase and decrease over time, to the discretion of the adversary. Designing a

protocol secure in the fully-fluctuating setting in a relaxed adversary model was explicitly stated as an open problem in [21]. We show that with the help of VDFs, one can design an efficient consensus protocol for the fully-fluctuating participation setting, secure against an external adversary.

▶ **Theorem** (Informal, see Thm. 8)**.** *Assuming a PKI, a VRF, a VDF, and an external adversary, we present a randomized protocol solving consensus that tolerates fully-fluctuating participation with expected constant latency.*

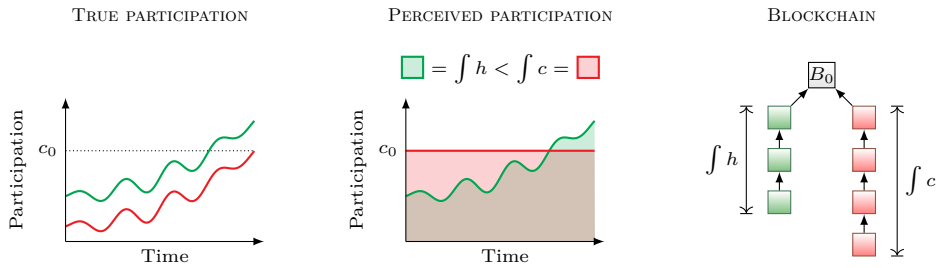
Tab. 1 summarizes our contributions and gives full context of previous work using the formal notation we introduce in Sec. 2. See Sec. 6 for an in-depth overview of related work.

1.2 Key techniques

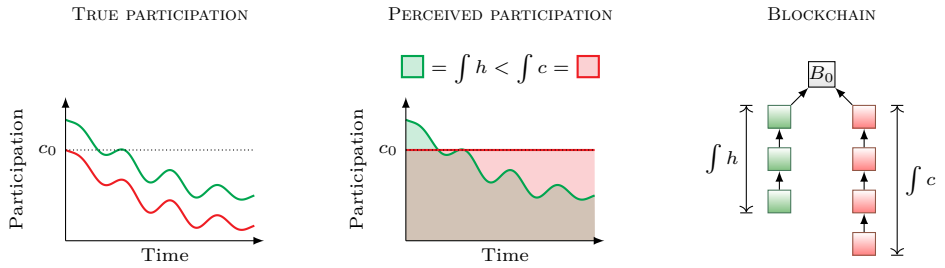
The challenge. Even in non-sleepy synchrony settings [27], any consensus protocol must assume that the adversary is limited in corrupting only a minority of the participants. The sleepy model, being a generalization of the traditional setting, clearly must also abide by an assumption of similar flavour. Note further that in the sleepy model, the awake/asleep status of every node at every point in time is also dictated by the adversary. Thus the changing nature of node participation in the sleepy model makes such an assumption non-trivial to state. Arguably the minimal assumption one could make, is that at each point in time, honest participation exceeds corrupt participation. E.g., this is the assumption for which Nakamoto consensus [23] is proven secure in the proof-of-work setting. In the translation to proof-of-stake, however, such a minimal assumption forces any purported protocol to be resilient against the following two attack vectors. We emphasize that these attacks remain a concern in the external adversary model, and are described to go through within the confines of the model. To envisage the attack vectors in a tangible manner, we consider here as a running example the proof-of-stake version of Nakamoto consensus.

- **Backward simulation.** (Fig. 2a) Consider the following. The adversary commences the protocol at time 0 with low participation of both corrupt and honest nodes. As time goes on, the adversary increases participation of corrupt (and respectively, honest) nodes until at some point in time t, corrupt participation at time t exceeds *honest* participation at timeslot 0 by a sufficiently large amount. At this point, the corrupt nodes, via a costless simulation attack, can construct a private chain whose length exceeds the length of the honest chain constructed thus far. From the point of view of a newly woken up honest node at time t, the longest chain rule will cause it to build on top of the corrupt chain, violating safety. Such an attack is a concern in periods of *increasing participation.*
- **Forward simulation.** (Fig. 2b) Alternatively, consider the following scenario. Participation starts out high, and as time goes on, the adversary puts corrupt (and honest, respectively) nodes to sleep until at some time t, the number of awake corrupt nodes at time 0 exceeds *honest* participation at time t by a sufficiently large amount. Assume further that the leader order is *predictable* from time 0. The corrupt nodes awake at time 0 can simulate an execution from time 0 to time t with a consistent participation rate equal to the number of corrupt nodes *awake at time* 0. From the point of view of a newly woken up honest node at time t, the longest chain rule will cause it to build on top of the corrupt chain, violating safety. Such an attack is a concern in periods of *decreasing participation.*

In this work, we devise several tools to deal with such attack vectors.



(a) Backward simulation attack.



(b) Forward simulation attack.

Figure 2 Examples of backward and forward simulation attacks. (b) Consider the presented *decaying* participation pattern amongst nodes. Let $c_0 = c(0)$, and let t be a timeslot such that $\int_0^t c_0 > \int_0^t h(t)$. Assume that corrupt players awake at time 0 can predict the leader order up to timeslot t. Then the corrupt players awake at time 0 can costlessly simulate a valid execution of the protocol with chain length $\int_0^t c_0$. As $\int_0^t c_0 > \int_0^t h(t)$, newly woken up honest nodes at timeslot $t+1$ will confirm blocks from the corrupt chain, thus violating safety. (a) A similar attack can be carried out by an adversary under *increasing* participation, but "in reverse", using the players awake at a time slot t in which c_0 corrupt players are awake such that $\int_0^t c_0 > \int_0^t h(t)$ holds, this time costlessly simulating a chain as if they were all awake from time 0.

Wakeness vectors. We define a new primitive in the context of the sleepy model, which might be of independent interest, which we call *wakeness vectors*. Intuitively, wakeness vectors allow nodes to have an estimate on the timeslots in which other nodes were awake throughout the execution a protocol. A similar idea was explored by Bar-Joseph, Keidar, and Lynch [4] in the context of crash failures. Both of our protocols (for the decaying setting, and the fully-fluctuating setting) begin with establishing that the respective models we consider allow the nodes to implement wakeness vectors of sufficient quality. The construction of these vectors employs different techniques in each of the settings. See Sec. 4 and Sec. 5 for an in-depth overview of the main ideas behind the construction of wakeness vectors for the decaying and fully-fluctuating settings, respectively, along with the formal proofs. Armed with this primitive, we then show how can be used to to secure protocol against forward/backward simulation attacks, and enhance protocols in the sleepy model to remain secure amidst decaying/fully-fluctuating participation.

General compiler. For our protocols (Thms. 6 and 8) we actually prove a more general statement than discussed above. We identify key properties, which most existing protocols in the literature satisfy, that are necessary for a protocol to be compatible with our techniques, and then prove that any such protocol can be augmented to be secure amidst fully-fluctuating

Table 1 Overview of previous work and our contributions for atomic broadcast protocols in the synchronous sleepy setting, along with the corresponding properties/assumptions. Below, resilience refers to the fraction of corrupt nodes, f^* refers to the number of *actual* corruptions in a given execution, γ refers to the participation rate of all nodes, and λ is a security parameter; $O(\cdot)^*$ refers to $O(\cdot)$ latency in the optimistic case. The specific adversary restrictions in each of the above results are as follows. See Sec. 6 for further details.

Paper	T_f	T_b	Resilience	VDF	Latency	Adversary restrictions
[23, 15]	0	0	$1/2$	✗	$O(\frac{1}{\gamma}\lambda\Delta)$	Proof of work
[24, 8, 2]	∞	∞	$1/2$	✗	$O(\frac{1}{\gamma}\lambda\Delta)$	–
[16]	∞	∞	$1/2$	✗	$O(\lambda\Delta)$	–
[3]	∞	∞	$1/2$	✗	$O(\frac{\Delta}{\gamma})^*$	–
[18]	∞	∞	$1/3$	✗	$O(n)$	Nodes announce going to sleep
[4]	0	0	$1/3$	✗	$O(f^*)$	Crash failures
[14]	0	0	$1/2$	✗	$O(\Delta)$	Time-stamped messages
[21, 10]	∞	$O(\Delta)$	$1/2$	✗	$O(\Delta)$	–
[21]	$O(\Delta)$	$O(\Delta)$	$1/3$	✗	$O(\Delta)$	Message timeout
[11]	$O(\frac{\Delta}{\gamma})$	$O(\frac{\Delta}{\gamma})$	$\frac{1}{1+e} \simeq 0.268$	✓	$O(\frac{1}{\gamma}\lambda\Delta)$	External adversary
[9]	∞	$O(\Delta)$	$1/2$	✗	$O(\Delta)^*$	–
This paper	$O(\Delta \log^2 \lambda)$	∞	$1/2$	✗	$O(\Delta)$	External adversary
This paper	$O(\Delta)$	$O(\Delta)$	$1/2$	✓	$O(\Delta)$	External adversary

or decaying participation, in the presence of an external adversary. In the next section (Sec. 2), we pour rigor into above discussion by formally defining the models and problems we consider in this work.

2 Preliminaries

Sleepy model. We work in the extended sleepy model that allows fully-fluctuating participation of corrupt nodes as defined in [21]. Specifically, we operate in the permissioned setting, i.e., there exists a set $\mathcal{P}$ of size n which contains the identifiers of all the nodes, which are public and known to all nodes. Without loss of generality, we will assume that $\mathcal{P} = [n]$. At each timeslot t, any node can exist in one of two states, *awake* or *asleep*. The number of awake nodes at timeslot t is denoted by $0 < n_t \leq n$. Asleep nodes at time t are prohibited from sending messages or executing any code at time t, and all messages delivered to them at time t are buffered until the next time step $t' > t$ in which they are awake. Given a protocol Π, there are two types of nodes, *honest* and *corrupt*. Corrupt nodes are controlled by the *adversary* (defined shortly) and can behave in any manner under the restrictions imposed on a given adversary. Honest nodes are the nodes not controlled by the adversary, and they execute Π at all timeslots. As for message delivery, we assume a synchronous network, with synchrony parameter Δ that is fixed and known to all nodes: for every timeslot t, if a node is awake at timeslot t, then we assume that they have received all messages sent to them prior to timeslot $t - \Delta$. We assume that the adversary has control over message delivery timing so long as it abides by the Δ-synchrony assumption. As to not over encumber notation, we omit Δ from the list of inputs to protocols and algorithms discussed in this work. As mentioned earlier, asleep nodes receive all messages sent to them by timeslot $\max(t', t + \Delta)$, where $t' > t$ is the first timeslot when they are awake after timeslot $t + \Delta$. Throughout this paper, we assume that communication is point-to-point.

The environment. We consider an external party to the protocol, *the environment*, which has the role of providing inputs to nodes from some universe I of inputs.

Oracles. We adopt the notion of oracles to model ideal versions of cryptographic primitives, formally defined in [20]. Such a modeling choice assists us in easily formalizing external adversary model we consider in this work, and to more cleanly state our results. Specifically, the description of a protocol also specifies a set of oracles $\mathcal{O} = \{O_1, ..., O_z\}$. At each timeslot t, each node p_i may issue a finite amount of *queries* to an oracle O_j, to which it receives a response at some timestep. An oracle's behaviour may depend on the nodes identity i, the current timeslot t, the input query q, and the internal randomness of the oracle, if the oracle models a random function. More formally, a deterministic oracle O is modeled by a function mapping tuples $(p, m, t) \to (m, t)$ such that $O(p, m, t) = (m', t')$ implies that if p queried O with input m at timeslot t, then p obtained a response m' at timeslot t'. A random oracle is modeled by a distribution over deterministic oracles. Intuitively, in this work, oracles are going to model trusted hardware access to cryptographic primitives. From here on, we denote protocols by the tuple $(\Pi, \mathcal{O})$ where Π is an algorithm executed by honest nodes, and $\mathcal{O}$ is the set of oracles employed by Π. When $\mathcal{O}$ is clear from context, we refer to a protocol as Π.

Adversary and fluctuating participation. Throughout this paper, we focus on the *static* adversary model, i.e., the identities of corrupt nodes are chosen by the adversary prior to the initiation of the protocol, and prior to any randomness being drawn, and in particular, nodes that begin the protocol as honest, cannot be corrupted later. Denote by f the total number of corrupt nodes, and by $\mathcal{F}$ the set of all corrupt nodes. To formalize the notion of fully-fluctuating participation of corrupt nodes, we adopt the terminology introduced in [21]. Specifically, let $\mathcal{F}_t$ be the set of corrupt nodes awake at timeslot t, and let $f(t, T_\mathrm{f}, T_\mathrm{b})$ be defined as follows.

$$f(t, T_\mathrm{f}, T_\mathrm{b}) = |\bigcup_{t-T_\mathrm{f} \leq \tau \leq t+T_\mathrm{b}} \mathcal{F}_\tau|$$

When $T_\mathrm{f}, T_\mathrm{b}$ are clear from context, we at times refer to $f(t, T_\mathrm{f}, T_\mathrm{b})$ by f_t. To provide some intuition for this notation, the parameters $T_\mathrm{f}, T_\mathrm{b}$ indicate how resilient the protocol has to be to forward simulation and backward simulation, respectively. In particular, for a given corrupt node q, if q is awake at time t, then is it considered awake for T_b time-steps *prior* to being awake, and for T_f steps *after* being awake. For a given $T_\mathrm{b}, T_\mathrm{f}$, unless otherwise stated, we always assume that $f(t, T_\mathrm{f}, T_\mathrm{b}) < \frac{n}{2}$ for all t.

Furthermore, the adversary has complete control over the environment and over the set of nodes awake at any timeslot t. While the choice of corrupt nodes occurs at the beginning of the protocol and is thus static, the same does not hold for the awake/asleep status. Namely, the set of nodes awake at each timeslot t can be adaptively chosen by the adversary based on the current transcript (i.e., the contents of all messages sent by honest nodes up to, but not including, the current timeslot) of the protocol and internal state of the corrupt nodes.

External adversary. The view of cryptography via oracles (introduced in [20]) allows us to define the external adversary model in a clean manner, in which (polynomial time) corrupt nodes do not know their secret key, and in particular, due to idealized cryptographic guarantees of the primitives implemented by the oracles, can not produce signatures on behalf of other *asleep corrupt nodes*, except for with negligible probability. This is achieved by not distributing the secret keys of nodes to them, but only allowing them oracle access

to cryptographic functionalities that make use of said keys. More formally, in the external adversary model, given a protocol $(\Pi, \mathcal{O})$, each node p (honest or corrupt) can only issue queries to any oracle $O \in \mathcal{O}$ of the form $(p, \cdot, \cdot)$ throughout the execution of Π. In particular, corrupt nodes, even when colluding, can not obtain fresh oracle queries using the keys of *asleep* corrupt nodes.

This is in contrast to the *standard* adversary in which each node $p \in \mathcal{F}$ can access all the secret of corrupt nodes. Rephrased in oracle terms: Every $p \in \mathcal{F}$ can query any $O \in \mathcal{O}$ with queries of the form $(q, \cdot, \cdot)$, for all $q \in \mathcal{F}$, regardless of the asleep/awake status of q.

Protocols and executions. We are now ready to define the notion of a *protocol* and an *execution* of a protocol. A protocol is specified by a pair $(\Pi, \mathcal{O})$, where Π is the algorithm run by the honest nodes, and $\mathcal{O}$ is the set of oracles employed by Π. An execution of a protocol $(\Pi, \mathcal{O})$ is determined by a 4-tuple, $E = (\Pi, \mathcal{O}, \mathcal{A}, r)$, where $\mathcal{A}$ is an adversary, and the vector r specifies the random coins of all nodes and the adversary. An execution refers further to the contents of all messages sent and received by nodes at each timeslot. Given an execution E, we denote thus by E^t the contents of the execution up to timeslot t, i.e., all random coins and messages received by nodes up to timeslot t. Given an execution E, a timeslot t, and a node i awake at timeslot t, we denote by E_i^t the *view* of node i of the execution E, which includes the internal state of i, the protocol $(\Pi, \mathcal{O})$, and all messages received by i and the timeslots in which they were received. For a given adversary $\mathcal{A}$, we define the *transcript of* Π *at time* t to be all the messages sent by honest nodes up to time t, and the internal state of the adversary $\mathcal{A}$ at time t, and we denote it by $\mathrm{Tr}_{\mathcal{A}}^t$. Throughout the entire paper, we consider a finite execution horizon of $\mathrm{T}_{\mathrm{horizon}} = \mathrm{poly}(n, \lambda)$. We assume that $\mathrm{T}_{\mathrm{horizon}}$ is known in advance to all nodes and in particular can be taken into account in the protocol design. We say that an execution is *admissible* in the $(T_{\mathrm{f}}, T_{\mathrm{b}}, \rho)$-sleepy model iff for all timesteps $t \geq 0$ it holds that $f(t, T_{\mathrm{f}}, T_{\mathrm{b}}) < \rho n_t$.

Atomic broadcast. We consider the atomic broadcast (AB) problem, in which the nodes are required to agree on a linearizable log of their inputs. For every timeslot t, every honest node i reports a *log*, denoted by $\mathcal{L}_i^t$, of its *decided values* $[x_0, x_1, x_2, ...]$. Honest nodes make decisions about the contents of their log based on their internal state, Π, and the messages they received so far in any given execution. More formally, given the random variable $\mathbf{E}$ over executions defined by Π and $\mathcal{A}$, $\mathcal{L}_i^t(\mathbf{E}_i^t)$ is the random variable that denotes the contents of the log of node i at timeslot t, and it depends only on $\mathbf{E}_i^t$. When i and t are clear from context, we omit them from the superscript and subscript of $\mathcal{L}$.

We say that a protocol Π solves the atomic broadcast problem with probability $1 - \epsilon$ in a given execution if the following is maintained throughout the entire execution horizon.

1. ϵ-*Safety.* For every timeslot t, and every i, j such that p_i and p_j are honest with logs $\mathcal{L}(\mathbf{E}_i^t), \mathcal{L}(\mathbf{E}_j^t)$, with probability 1 we have: $\mathcal{L}(\mathbf{E}_i^t)[0 : m] = \mathcal{L}(\mathbf{E}_j^t)[0 : m]$ where $m = \min\{|\mathcal{L}(\mathbf{E}_i^t)|, |\mathcal{L}(\mathbf{E}_j^t)|\}$.
2. (ϵ, ℓ)-*Liveness.* For all $t \in [\mathrm{T}_{\mathrm{horizon}}]$ the following holds: If input x was sent to an honest node p_i by the environment at timeslot t, then $\Pr[\forall t'' \geq t + \ell \colon x \in \mathcal{L}(\mathbf{E}_j^{t''})] \geq 1 - \epsilon$ holds for all honest nodes p_j.

Cryptographic primitives. Let λ be a security parameter. Throughout the paper, we make use of three cryptographic primitives. Namely, a PKI (for signatures), a VRF, and a VDF. We employ a cryptographic hash function (known and computable by all nodes), which we

model as a random oracle, and denote by H. Similarly to previous work, we denote by $\langle m \rangle_p$ the fact that the message m was signed by node p. We model all three of these primitive in an *idealized* fashion, using oracles $O_{\mathrm{S}}, O_{\mathrm{V}}, O_{\mathrm{D}}$ that are defined as follows:

1. Oracle O_{S} on a given input (p, m, t) where p is a node, t is a timeslot, and m is a message is defined to be $(\langle m \rangle_p, t)$. In other words, p receives $\langle m \rangle_p$ in the same timeslot in which it issued the query. From here on in, unless explicitly said otherwise, we assume that every message being sent in any protocol by an honest node is signed by its sender.
2. Oracle O_{V} is defined as $O_{\mathrm{V}}(p, m, t) = (\mathrm{VRF}_{sk_p}(m), \pi_m, t)$. In other words, p receives $\mathrm{VRF}_{sk_p}(m), \pi_m$ in the same timeslot in which it issued the query. Here, $\mathrm{VRF}_{sk_p}(\cdot)$ is a VRF scheme based on p's secret key sk_p, known by the oracle O_{V}, and π_m is the accompanying proof as per the VRF primitive. Again, to keep the focus of the paper on protocol design, we model the functionality of all the VRFs as random oracles. We further assume that the VRF functions accessible to each node all have the same domain ($\{0,1\}^*$), and the same range ($\{0,1\}^\lambda$).
3. Oracle O_{D} is defined as follows when p sends queries $m_1, \ldots, m_k$ to the oracle:

   $$O_{\mathrm{D}}(p, m_1, \ldots, m_k, t) = (\mathrm{VDF}(m_1), \pi_{m_1}, \ldots, \mathrm{VDF}(m_k), \pi_{m_k}, t).$$

   In other words, p receives $\mathrm{VDF}(m_i), \pi_{m_i}$ at timeslot t. Here, $\mathrm{VDF}(m)$ is the VDF computation result on input string m, and π_m is the accompanying proof as per the VDF primitive. We stipulate the following restriction on O_{D}: For every round t and every node p, p can call O_{D} *at most once* at timestep t. In other words, we capture the delay property of O_{D} by not allowing adaptive queries to O_{D} in any single timestep. This restriction applies to both honest and corrupt nodes, thus in particular, the adversary has no advantage in computing the VDF. We consider this modeling as an idealized model for a VDF. We model our VDF as a random function for the purposes of the analysis. We assume that the VDF accessible to each node has domain $\{0,1\}^*$, and range $\{0,1\}^\lambda$.

This would be the place to mention, that even though we model our cryptographic primitives using oracles that execute their ideal functionality, we still assume that all corrupt nodes are computationally bounded in the sense that there exists a polynomial $z(n)$ that depends on the total number of nodes such that the total number of queries Q_p to oracles performed by any node p in a single timeslot satisfies $Q_p \leq z(n)$. Finally, for an event E we say that E holds *with high probability* (w.h.p.) if $\Pr[E] \geq 1 - \mathsf{negl}(\lambda)$.

3 Tools and definitions

3.1 Wakeness vectors

Consider vectors $v^1, ..., v^n$ whose dimension is $\mathrm{T}_{\mathrm{horizon}}$. These vectors are defined as follows for each timeslot t: For all $i \in [n]$ and all $j \in [t]$, $v^i[j] = 1$ if and only if node p_i was awake at timeslot j, and 0 otherwise. Endowing honest nodes with such vectors should intuitively make protocol design in the sleepy model substantially easier. Our upper bounds are constructed by having nodes implement subroutines that allow them to emulate an approximate version of this ideal functionality. The formal definition is as follows.

▶ **Definition 1.** *Let $(\Pi, \mathcal{O})$ be a protocol and let $\mathbf{E}_{\Pi,\mathcal{A}}$ be the random variable indicating a sample from the distribution over executions of Π defined by adversary $\mathcal{A}$. For a given T, we say that $v_p \in \{0,1\}^T$ is a* d-valid wakeness vector *of node p with respect to $\mathbf{E}$ iff, except with negligible probability, $v_p[j] = 1$ implies that p is awake at a timeslot $j' \geq j - d$. Furthermore, if p is honest, then p being awake at timeslot j implies $v_p[j] = 1$. When v is a 0-valid wakeness vector, we simply refer to it as valid.*

Throughout the paper, we assume without loss of generality (w.l.o.g.) that all honest nodes keep a *local view* of wakeness vectors. I.e., if $(\Pi, \mathcal{O})$ is a protocol, then in any execution E of Π, each honest node p stores $v_p^1, ..., v_p^n$, where $v_p^j \in \{0,1\}^{\mathrm{T_{horizon}}}$ for all j. We say that p *deems* node i to be awake at timeslot t iff $v_p^i[t] = 1$. For an interval I of timeslots, we at times abuse notation and use $v_p^i[I]$ to refer to all coordinates in I in v_p^i. The main two upper bounds we prove in this work can be formalized as a general statement about augmenting atomic broadcast protocols with wakeness vectors. We present the formal lemmas in the appropriate sections.

Both our upper bounds are general compiler results, that augment protocols working in the (∞, ∞, ρ)-sleepy model with wakeness vectors of sufficient quality to support fully-fluctuating participation. These augmentations however don't work for arbitrary protocols. Specifically, there is a key set of properties regarding the structure of a protocol that we must assume in order for our results to go through. It turns out, however, that the vast majority of literature on protocols for the sleepy model already posses these properties. Specifically, we define notions that we refer to as *Statelessness* and *Unpredictability*.

Stateless protocols. Intuitively, a protocol is said to be stateless if an honest nodes action depends on messages it received from nodes it has *deemed* to be awake in recent timeslots. We assume w.l.o.g. that in any protocol Π, honest node keep a local view of wakeness vectors. Note that these vectors are not necessarily d-valid for any d. Such a property depends on the model and protocol in question. We further assume w.l.o.g. that honest nodes always attach the current timeslot to their messages.

▶ **Definition 2.** *Let $(\Pi, \mathcal{O})$ be a protocol. We say that Π is* T-stateless *if the following holds for all honest nodes p and timeslots t and pair of executions E_0, E_1. Let $v_p^1, ..., v_p^n$ be the wakeness vectors viewed by p at timeslot t. Let $T' \leq T$, chosen by p, and consider the set $S = \left\{i \in [n] \mid \exists j \in [t - T', t - 1], v_p^i[j] = 1\right\}$. Then if S is the same in both E_0, E_1, and furthermore, the messages sent from nodes in S to p are the same in both E_0, E_1, then the distribution over actions of p is the same at timeslot t in both E_0, E_1. Furthermore, Π solves atomic broadcast in any execution in which for all honest nodes p and timeslots t, S has a strict honest majority.*

Unpredictability. Intuitively, this property captures the adversary's ability to predict the contents of the log of honest nodes at some future timeslot.

▶ **Definition 3.** *Let $(\Pi, \mathcal{O})$ be a protocol that solves atomic broadcast. We say that Π has* α-unpredictability *if the following holds for all t, and for all adversaries $\mathcal{A}$. Let $\mathrm{Tr}_{\mathcal{A}}^t$ be the transcript of the protocol up to timeslot t, and let $\mathcal{L}_{\mathcal{A}} \leftarrow \mathcal{A}(\Pi, \mathrm{Tr}_{\mathcal{A}}^t, \alpha)$ be an adversarial log, which we refer to as the adversary's* guess. *Then for all honest nodes p_i, it holds that $\Pr[\mathcal{L}(\mathbf{E}_i^t) \prec \mathcal{L}_{\mathcal{A}} \preceq \mathcal{L}(\mathbf{E}_i^{t+\alpha})] = \mathsf{negl}(\lambda)$.*

4 Decaying participation

Efficient protocols for the $T_\mathrm{f} = \infty$ regime (i.e., increasing participation) can be found in [22, 21, 9], with T_b being nearly optimal, namely with $T_\mathrm{b} = O(\Delta)$. In this section, we tackle the yet unstudied $T_\mathrm{b} = \infty$ case, in which participation is decaying. Namely, in the decaying participation model, corrupt nodes are allowed to go to sleep, but may never wake up after timeslot 0. We show that that PKI, VRF, and an external adversary suffice to design protocols in this setting, with $T_\mathrm{f} = O(\Delta\lambda)$, where λ is the security parameter, with error

probability of $O(\frac{1}{2^\lambda})$. This result establishes the benefit of considering an external adversary, as an impossibility result by [21] established that atomic broadcast is impossible to solve in the $T_b < \infty$ case against a standard adversary that can carry out the key transfer attack.

Blocks. We employ the notion of a *block*. Discussing blocks allows our black box protocols to be compatible with previous protocols in the literature that employ such a notion. We think of blocks as being built from the inputs sent by the environment to the nodes in the following way. In general, each block B added to the chain is going to have the following structure: Each block has the form $B = (x, \mathsf{H}(B'), v, e, \mathsf{seed})$, where x is the content of the block. You should think of x as all of the inputs sent by the environment to nodes that the block constructor has seen thus far that are not already decided. Here, $\mathsf{H}(B')$ is a hash of the previous block in the chain, with all honest nodes initializing their chain with the *genesis block*, denoted by $B_0 = (\bot, \bot, 0)$. At times, protocols induce a partition of the timeslots into intervals which are referred to as *views* and *epochs*, when that is the case, v indicates the *view* in which the block was created, and e is the *epoch* in which the block was created. When the protocols we discuss do not make use of views and epochs, we simply omit these coordinates from the description of a block. Lastly seed is an ancillary string. Similarly, when we discuss protocols in which the seed entry is not used, we simply omit it from the block description. We denote by $\mathsf{view}(B)$ and $\mathsf{epoch}(B)$ the view and epoch, respectively, in which block B is was (or claimed to be) created. We define the *height* of a block to be its distance in the chain from the genesis block, and we denoted it by $h(B)$.

Note that any block B defines a unique path from B back to the genesis block, and thus defines a unique log. We say that a block B extends a block B' if $B = B'$ if or B' is an ancestor of B. We say that two block B, B' *conflict* with each other if neither one of them extends the other. We say that a block B is valid if its ancestor block B' is valid and $\mathsf{epoch}(B') < \mathsf{epoch}(B)$ or ($\mathsf{view}(B') < \mathsf{view}(B)$ and $\mathsf{epoch}(B') = \mathsf{epoch}(B)$). If the view or epoch are omitted from the block, the above conditions hold vacuously.

Permissible block. On top of being valid, a protocol may impose additional constraints on blocks in order for them to be considered for confirmation. This usually takes the form a *locally computable* (i.e., no communication required) predicate ξ, computable efficiently and locally by every node, so that $\xi(B) = 1$ if B is considered a permissible block, and 0 otherwise. Messages involving impermissible blocks simply get ignored by honest nodes throughout the execution of the protocol.

4.1 Graded proposal election

To obtain the result we require the observation that the protocol of [21] can be endowed with the unpredictability property with a slight modification which we discuss formally later in the section. This modification concerns a procedure called *graded proposal election*, which we define here for completeness.

▶ **Definition 4.** *In* graded proposal election *(GPE), each node p has an input block B_p, and outputs a single pair (B, g) of a block and a grade ($g \in \{0, 1\}$) with the following properties.*

1. Consistency. *If two honest nodes p, q output blocks B, B' respectively, so that $B \neq \bot, B' \neq \bot$, then $B = B'$.*
2. Graded Delivery. *If an honest node p outputs $(B, 1)$, then all honest nodes output $(B, *)$.*
3. Validity. *With probability (w.p.) at least $\frac{1}{2}$, all honest nodes output $(B, 1)$ for some block B that was the input of some honest node p.*
4. Integrity. *If an honest node p outputs $(B, *)$, then either $B = B_0$, or there exists an honest node p' that received the* content *of B from the environment.*

In their paper, Malkhi, Momose, and Ren [21] exhibit a protocol solving GPE in $O(\Delta)$ timesteps in the $(\infty, \infty, \frac{1}{2})$-sleepy model.

4.2 Protocol and proof

We now have all the tools to state our results formally. The following result employs the notions of log unpredictability (Def. 3), and the notion of a stateless protocol (Def. 2). We can now formally state the following lemma, which intuitively says that assuming an external adversary, one can turn any protocol with non-trivial unpredictability and statelessness, into a protocol that can tolerate decaying participation.

▶ **Lemma 5.** *Let $(\Pi, \mathcal{O})$ be a protocol solving atomic broadcast w.p. $1-\epsilon$ and liveness parameter ℓ in the (∞, ∞, ρ)-sleepy model, and assume that:*

- *Π has α-unpredictability, and*
- *Π is α-stateless.*

Then assuming an external adversary, $\rho < \frac{1}{2}$, there is a protocol $(\Pi', \mathcal{O})$ that solves the atomic broadcast problem in the $(O(\alpha), \infty, \rho)$-sleepy model w.p. $1-\epsilon-\mathsf{negl}(\lambda)$ and liveness parameter ℓ . Furthermore, Π' expected latency is upper bounded by that of Π', and incurs an additive $O(\lambda n^2)$ factor of communication per timeslot. In particular, assuming an external adversary, one can implement $O(\alpha)$-valid wakeness vectors in the $(O(\alpha), \infty, \rho)$-sleepy model.

Lem. 5 is established via a reduction, depicted in Alg. 1, that constructs Π' given blackbox access to the given protocol Π in the premise of the lemma. Instantiating Π in the reduction of Lem. 5 with a modification (detailed in Alg. 2) of the atomic broadcast protocol of [21, Alg. 3], we get the following result:

▶ **Theorem 6.** *Assuming PKI, VRF, and an external adversary, there is a protocol Π, solving the atomic broadcast problem w.p. $1-\frac{1}{2^\lambda}$ in the $(O(\Delta\lambda), \infty, \frac{1}{2})$-sleepy model with $O(\Delta)$ expected latency, and liveness parameter $\ell = O(\Delta\lambda)$.*

We next provide an overview of the main ideas that go into the proof of Lem. 5. The key observation for the decaying participation setting against an external adversary is that honest nodes can *always* rely on the messages *claimed* to have been sent in the earlier timeslots, due to the $T_\mathrm{b} = \infty$ assumption. The rough idea then is two-fold.

1. **Random log.** The main challenge in the decaying participation setting is dealing with forward simulation attacks, similarly to Fig. 2b. Our idea for handling such attacks is by introducing *nested* randomness to the log decided by honest nodes. Specifically, along with a proposal for a block, the proposal must include a VRF output, which is computed on the VRF output attached to the parent of the proposed block. In case the proposer is honest, this randomness is unknown to the corrupt nodes until the moment of broadcasting the proposal. As such, corrupt nodes attempting to construct a corrupt chain will not be able to do so with the randomness of honest nodes beyond the latest available block/proposal sent by honest nodes. Thus, the key is to ensure that blocks proposed by *honest* nodes are decided often enough. This endows the log with an unpredictability property of the contents of the log with corrupt nodes awake in the early timeslots, but not in the current timeslot. We show that we can ensure that w.h.p., a block proposed by an honest node gets decided once every $O(\Delta\lambda)$ timeslots, hence our choice of $T_\mathrm{f} = O(\Delta\lambda)$.
2. **Tolerating forward simulation.** In the $T_\mathrm{f} < \infty$ model, we are susceptible to forward simulation attacks, in which corrupt nodes active in early rounds send messages to honest nodes awake far in the future, pretending to have been awake all along. With the

Algorithm 1 Protocol Π' for Lem. 5.

// Each node p, if it is awake, executes the following at all timeslots t. Time is divided into epochs, each of length α, starting from timeslot 0. Denote the current epoch by $e(t)$. Each node initializes a local variable $\mathcal{L}_p^* = B_0$. Initialize $\mathcal{L}_i^0 \leftarrow B_0$. Epoch -1 is the same as epoch 0. Initialize $v_p^1 = ... = v_p^n = 0^{\mathrm{T}_{\mathrm{horizon}}}$.
// Update log
for $e = 0, ..., e(t) - 1$
  $D^* \leftarrow$ # of nodes q so that p received $\langle decide, *, e \rangle$
  for all blocks B *extending a block of epoch* $e - 1$ *in* $\mathcal{L}^*$
    $v_p^i[e] = 1^\alpha$ for every p_i that sent a block B extending a block of epoch $e - 1$ in $\mathcal{L}_p^*$
    $D(B) \leftarrow$ # of nodes q so that p received $\langle decide, B', e \rangle$ for any B' extending B
    if $D(B) > D^*/2$
      Add B and all its ancestors to $\mathcal{L}_p^*$
// Executing Π
Execute Π as instructed at timeslot t, ignoring messages from nodes p_i that do not satisfy $v_p^i[e(t) - 1] = 1^\alpha$
upon a block B is *decided* according to Π
  Decide B and all its ancestors.
  $\mathcal{L}_p^* \leftarrow B$ and all its ancestors.
// Decision messages
Multicast $\langle decide, B, e(t) \rangle_p$, where B is the most recently decided block

Algorithm 2 Augmenting [21, Alg. 3] with unpredictablility.

// $\mathsf{GPE_v}$ invocation
if $r \in [0, 4\Delta]$
  $B \leftarrow (x, H(\mathsf{candidate}), v(t), e(t), O_{\mathrm{V}}(\mathsf{candidate}, t))$ for some $x \in I_p$
  Execute $\mathsf{GPE_v}$ with input B. A block B is *permissible* if it extends lock and has $\mathsf{view}(B) = v(t)$, $\mathsf{epoch}(B) = e(t)$, and $O_{\mathrm{V}}(\mathsf{candidate}, t)$ consists of a valid output of O_{V} for input of the form $(p, \cdot, \cdot)$.

randomness introduced in the first bullet point, honest nodes joining the protocol at a late time can recover the honest log by *inductively* building it, starting from the first block. The inductive argument then intuitively goes as follows.

- The correct recovery of the *first* block in the log is guaranteed by the $T_{\mathrm{b}} = \infty$ assumption. Namely, the number of honest nodes awake at the commencement of the protocol is greater than the total number of corrupt nodes that are *ever* going to be awake during the execution. This is the base step of the reduction.
- The randomness introduced to the log and the choice of T_{f} allows honest nodes to safely deduce the next block in the log, given correct recovery up to block k, for any k. Roughly, this follows from two observations. First, any forward simulation of the protocol constructing a corrupt log up to $O(\Delta\lambda)$ timeslots forward is going to get outvoted by honest nodes, due to the $T_{\mathrm{f}} = O(\Delta\lambda)$ assumption. Any attempt by corrupt nodes to forward simulate the protocol for $\omega(\Delta\lambda)$ timeslots is foiled by the fact that any adversarial log is almost certainly different than the honest log, as its suffix contains no blocks proposed by honest nodes, and thus the randomness attached to it is unpredictable to corrupt nodes. Hence, honest nodes awake during the following $O(\Delta\lambda)$

timeslots are still going to outvote the fake log, due to the $T_f = O(\Delta\lambda)$ assumption. This allows honest nodes to safely deduce block $k+1$ of the log by deciding on the block for which they observe a majority of votes.

The proof of Lem. 5 involves augmenting a general protocol with resilience to forward simulation, this augmentation is presented in Alg. 1. Afterwards, in order to obtain Thm. 6, we explain how to augment the protocol of [21, Alg. 3] with randomness and thus endow it with $O(\Delta \log^2 \lambda)$-unpredictability (see Def. 3).

Proof of Lem. 5. Let Π be the protocol given in the premise of Lem. 5. The protocol we construct is described in Alg. 1. The protocol is divided into epochs, each of length α. Each node that wakes up, reconstructs the contents of the log inductively starting from the first epoch. The inductive step is then implemented by each honest node ignoring all messages regarding blocks of epoch e that do not extend a block of epoch $e-1$ confirmed by the node. This is formalized as described in Alg. 1. In Alg. 1, the notation $v_p^i[e]$ corresponds to all coordinates of v_p^i corresponding to epoch e. First we prove the following property of Alg. 1.

▶ **Lemma 7.** *Except for with negligible probability, it holds that for any timeslot t and honest node p, $v_p^1, ..., v_p^n$ are 3α-valid wakeness vectors in all $(T_f = 10\alpha, \infty, \rho)$-sleepy executions. Furthermore, it holds that at the end of iteration k of* update log *(See line 2 of Alg. 1), that $\mathcal{L}_p^*$ contains a block of epoch k of $\mathcal{L}(\mathbf{E}_p^t)$, and contains no blocks that were not decided by at least one honest node.*

Proof. We prove this by induction over the epochs. Denote by t the current timeslot. For $e = 0$, the wakeness vector statement is trivial, i.e., whenever $v_q^i[e=0] = 1^\alpha$ for an honest node q for some i, then node i was awake at timeslot at least 0. For $\mathcal{L}_q^*$, Let B be a block in $\mathcal{L}^*$ that extends the genesis block, this implies that q observed a majority of *decide* messages for B amongst all decide messages it received for blocks extending the genesis block. This includes all the honest nodes awake during epoch 0, by line 16 of Alg. 1. By the $(10\alpha, \infty, \rho)$-sleepy model, the honest nodes awake during epoch 0 exceed the total number of corrupt nodes that are ever going to be awake during the execution. This implies that p received a *decide* message for B from at least one honest node p, i.e., $B \in \mathcal{L}(\mathbf{E}_p^{t'})$ for some timeslot $t' < t$, as required. The log $\mathcal{L}_q*$ contains the genesis block, and thus contains a block of $\mathcal{L}(\mathbf{E}_p^t)$ of epoch 0. For the induction step, consider the claim to be true for all epochs at most $\leq e$, we prove the claim for epoch $e+1$. Let q be an honest node awake at timeslot t of epoch $\geq e+1$ and let i be a node such that $v_q^i[e] = 1^\alpha$. This implies that q received a *decide* message from i extending a block in $\mathcal{L}_q^*$ of epoch $e-1$. Consider the event E in which p_i was last awake during epoch $e' \leq e-5$. By the induction hypothesis, up to epoch $e-1$, all wakeness vectors stored by all honest nodes are 3α valid. In particular, due to the $(T_f = 10\alpha, \infty, \rho)$-sleepy assumption, and due to the α-stateless property of Π, we get that at least up to the end of epoch $e-1$, Π correctly solves the atomic broadcast problem, as by line 11 of Alg. 1, all honest nodes consider messages from a set that has an honest majority. This allows us to invoke the α-unpredictability of Π, together with the second induction assumption, that assures us that $\mathcal{L}_q^*$ contains a block in $\mathcal{L}(\mathbf{E}_q^t)$ of epoch $e-1$ to claim that the probability that i could send to q a block extending a block of epoch $e+1$ in $\mathcal{L}_q^*$ is $\mathsf{negl}(\lambda, n)$. This invokes the external adversary assumption, meaning that a message observed to be from i can be sent only by i (or other corrupt node) at a timeslot in which i is awake. Thus E occurs with at most negligible probability. Applying a union bound over all corrupt nodes and over all timeslots during the execution horizon (both polynomial in n, λ) maintains this negligible probability, and proves the claim for the wakeness vectors. This

guarantees the correctness of Π for epoch $e+1$ by its α-stateless property. This implies that $\mathcal{L}$ is consistent between all honest nodes at timeslot t. Now we must prove that at iteration e, a block of epoch e in $\mathcal{L}(\mathbf{E}_q^t)$ is added to $\mathcal{L}_q^*$, and no blocks that are not decided by some honest node are added.

Assume that for an honest node q no block of epoch e is added to $\mathcal{L}_q^*$ by the end of iteration e. This implies that there exists an honest node p, awake at timeslot $t-1$, that doesn't have any blocks of epoch e in $\mathcal{L}(\mathbf{E}_p^{t-1})$. This again holds by the induction assumption of $\mathcal{L}^*$ for iteration $e-1$, the 3α-validity of the wakeness vectors, the α-unpredictability of Π, and the $(10\alpha, \infty, \rho)$-sleepy execution. All these imply that q is hearing from an honest majority, even when restricted to honest nodes awake at timeslot $t-1$. So if all honest nodes awake at timeslot $t-1$ had a block of epoch e decided, consistency of Π would imply that it is added to $\mathcal{L}^*$ for q at line 9 of Alg. 1.

This event can only happen with negligible probability, as otherwise we contradict the α-unpredictability of Π, as the adversary at round $t-\alpha-1$ had already seen all blocks of epoch at most $e-1$ communicated between honest nodes, including the transcript of the protocol, and all messages received by p, and could thus compute $\mathcal{L}(\mathbf{E}_p^{t-1})$ with non negligible confidence. Thus except for with negligible probability, all honest nodes awake at timeslot $t-1$ have a block of epoch e decided, and thus this block is added to $\mathcal{L}^*$ by q at timeslot t by the behaviour of the protocol and the induction assumption. Lastly, again since q hears from an honest majority, we get that every block B added to $\mathcal{L}_q^*$ was decided by at least one honest node, i.e., added to $\mathcal{L}(\mathbf{E}_p^{t'})$ for some honest node p at a timeslot $t' < t$. ◀

The 3α-validity of the wakeness vectors given by Lem. 7, combined with the $(10\alpha, \infty, \rho)$-sleepy executions, and the α-statelessness of Π, allow us to deduce that Π' (Alg. 1) solves the atomic broadcast correctly as required. Notice that block confirmation in Π' coincides with block confirmation in Π, and thus they have the same latency properties. The only added communication occurs in *decision messages* part of the protocol, and each message is of length $O(\lambda)$, thus the total communication blowup per timeslot is $O(\lambda n^2)$, as required. ◀

Proof of Thm. 6. All that is left to obtain Thm. 6, is to observe that the protocol (specifically, Algorithm 3) of [21] (henceforth [21, Alg. 3]) is $O(\Delta)$-stateless, works in the $(\infty, \infty, \frac{1}{2})$-sleepy model (see Thm. 11). It can be augmented to be $O(\log^2 \lambda)$-unpredictable with an incredibly simple change to the *GPE invocation* part of the protocol, which we describe in Alg. 2. In words, each honest node, when inputting a block B to GPE_v, simply includes in its seed a VRF computation on the current timeslot and the parent block of B. This ensures that the contents of each block contain a component with high entropy, thus making the logs of honest nodes unpredictable.

More formally, the augmented algorithm is described in Alg. 2, with the red lines indicating changes from the original pseudo-code. Due to the simplicity of the change, and not to burden the write up with the technical details of [21, Alg. 3], we provide below a sketch of the proof argument. [21, Alg. 3] is divided into *views*, each consisting of $O(\Delta)$ timeslots, during which an attempt to agree on this next block in the log is executed. This attempt is captured by the GPE procedure, which is executed once every view. By the properties (in particular, validity) of the GPE subroutine (see Def. 4) and the behaviour of [21, Alg. 3], w.p. at least $\frac{1}{2}$, at the end of the GPE subroutine, *all* honest nodes decide a block proposed by an honest node into their log. Conditioned on this event, the contents of $O_\mathrm{V}(\mathsf{candidate}, t)$ are uniformly random and *unpredictable* to any adversary not awake before the beginning of the current view. In particular, the adversary $\mathcal{A}$ attempting at timeslot t to guess the contents of the log of *any* honest node, succeeds w.p. at most $\frac{1}{2^\lambda}$. As different GPE executions are

independent, we get that after $O(\lambda)$ views, condition 3 in Def. 4 is invoked except w.p. $\frac{1}{2^\lambda}$. Thus in total, the probability that an adversary at timeslot t can correctly guess the contents of the log of *any* honest node at timeslot $t + O(\lambda\Delta)$ is at most $\frac{1}{2^\lambda}$, as required. ◀

5 Fully-fluctuating participation

Finally, we move on to consider fully-fluctuating participation regime, where both $T_\mathrm{f}, T_\mathrm{b} < \infty$. We show that assuming an external adversary and a VDF, one can design efficient protocols in this setting.

▶ **Theorem 8.** *Assuming a PKI, a VRF, a VDF, and an external adversary, there exists a randomized protocol that with high probability (w.h.p.) solves atomic broadcast with expected latency of $O(\Delta)$, and liveness parameter $\ell = O(\Delta \log \frac{1}{\epsilon})$ in all admissible $(O(\Delta), O(\Delta), \frac{1}{2})$-sleepy executions.*

The proof of Thm. 8 is driven by two lemmas. The first lemma (Lem. 9, stated below) is to show that there is an efficient protocol that implements valid wakeness vectors, assuming PKI, VRF, and VDF oracles, and an external adversary:

▶ **Lemma 9.** *Assuming a PKI, a VRF, a VDF, and an external adversary, there is a protocol with $O(\lambda n^2 \log^2 n)$ communication complexity per timeslot that w.h.p. implements valid wakeness vectors for all timeslots t over an execution horizon of length $\mathrm{T}_{\mathrm{horizon}}$.*

The second lemma (Lem. 10, stated below) is a general result that gives a black-box simulation showing how to convert *any* stateless protocol for atomic broadcast in the $(\infty, T_\mathrm{b}, \rho)$-sleepy model into an atomic broadcast protocol in the stronger $(T_\mathrm{b}, T_\mathrm{b}, \rho)$-sleepy model:

▶ **Lemma 10.** *Let $(\Pi, \mathcal{O})$ be a protocol solving atomic broadcast w.p. $1 - \epsilon$ in the $(\infty, T_\mathrm{b}, \rho)$-sleepy model with liveness parameter ℓ. Furthermore, assume that Π is T_b-stateless (Def. 2). Then, assuming a PKI, a VRF, an external adversary, $\rho < \frac{1}{2}$, and valid wakeness vectors, there exists a protocol $(\Pi', \mathcal{O})$ solving atomic broadcast in the $(T_\mathrm{b}, T_\mathrm{b}, \rho)$-sleepy model w.p. $1 - \epsilon$ and liveness parameter ℓ. Furthermore, Π' has expected latency upper bounded by that of Π and incurs an additive $O(\lambda n^2 \log^2 n)$ communication cost per timeslot.*

We prove Lem. 10 and Lem. 9 by presenting a protocol (Alg. 3) that both implements wakeness vectors using VDFs in the external adversary model, and uses a given protocol Π as a black box. In a high level, every node at each timeslot has three responsibilities:

1. Extracting decided log so far from received messages. This computation is internal, i.e. involves no communication.
2. Extension of wakeness vectors.
3. Running Π.

The idea is to build, along with the blockchain of inputs of nodes, an object we refer to as *VDF chains*, from which one can extract valid wakeness vectors for honest nodes to convince other honest nodes of them being awake at certain timeslots. This allows the honest nodes that wake up after a long period of sleepiness to only consider messages from nodes that were awake in recent timeslots to realize the current state of the blockchain; the T_b interval in both forward simulation and backwards simulation give us the needed majority to make sure the honest nodes can perform these tasks successfully.

The proofs of wakeness allow honest nodes to determine the identities of of nodes that were awake in the previous view, and take only their messages into account in deciding on the state of the log, using the $(T_\mathrm{b}, T_\mathrm{b}, \frac{1}{2})$-sleepy assumption regarding the permissible executions.

Algorithm 3 Constructing protocol Π' from Π.

// The following is executed by every honest node p at every timeslot t in which it is awake. Let $T = \text{poly}(\lambda, n)$ be the execution horizon. Initialize $v_p^i \leftarrow 0^T$ for all $i \in [n]$. $\mathcal{L}_i^0 \leftarrow B_0$.
// Wakeness vector updates
Let $(v_p^1, ..., v_p^n)$ be the local view of p from the last timeslot r in which p was awake
for $r < t' \leq t$
  for all $i \in [n]$
    if p observes a valid t'-depth value with prover p_i
      $v_p^i[t'] \leftarrow 1$ // p deems p_i awake at timeslot t'
    else
      $v_p^i[t'] \leftarrow 0$ // p deems p_i asleep at timeslot t'
// VDF chain extension
$P \leftarrow$ set of nodes q so that p received a valid depth-$(t-1)$ value val_q from q
$S \leftarrow O(\log^2(n \cdot t))$ uniformly random elements (q, val_q) sampled from P
for all $(q, \mathsf{val}_q) \in S$
  Compute $O_{\text{D}}(p, \mathsf{val}_q, t)$ and multicast $\langle \text{VDF}(\mathsf{val}_q) \rangle_p$
// Running the protocol
Execute Π as instructed, and in particular, decide when Π decides, while ignoring messages from nodes p_i for which $v_p^i[t - T_{\text{b}} : t - 1] = 0$

Thm. 8 follows from Lem. 9 and Lem. 10, by instantiating the protocol Π to be the protocol from [21]. Specifically, [21, Alg. 3] proves the following theorem, despite not stating the stateless property explicitly.

▶ **Theorem 11** (Theorem 1, [21]). *Assuming a PKI and a VRF, there exists a $O(\Delta)$-stateless protocol Π solving atomic broadcast w.p. 1 with expected latency $O(\Delta)$, and liveness parameter $\ell = O(\Delta \log \frac{1}{\epsilon})$, in any admissible $(\infty, O(\Delta), \frac{1}{2})$-sleepy execution.*

Sec. 5.1 is devoted to the proofs of Lem. 9, and Lem. 10.

5.1 Constructing wakeness vectors from VDFs

This section is devoted to proving Lems. 9 and 10. Thm. 8 then follows. In the external adversary model, where corrupt nodes may only access their secret keys via oracles, a VDF assumption allows any honest node p to prove w.h.p. the following statement for any timeslot t, if it is true: "node p was awake at some timeslot $t' \geq t$". In order to prove this statement with VDF, we need to define a few notions first.

▶ **Definition 12.** *Let val be some value in the range of the VDF function, and let p be some node. We say that val is a valid* depth-v *value w.r.t. p if there exists values $\mathsf{val}_0, ..., \mathsf{val}_{v-1}, \mathsf{val}_v$, where $\mathsf{val}_v = \mathsf{val}$, and nodes $p_0, ..., p_{v-1}, p_v$, where $p_v = p$, such that following holds.*

1. *For all $i \in [v]$, $\text{VDF}(\mathsf{val}_i) = \mathsf{val}_{i+1}$.*
2. *Node p has received $\langle O_{\text{D}}(p_i, \mathsf{val}_i, \cdot) \rangle_{p_i}$ for all $i \in [v]$.*

We call p_{v-1} the prover, and val the v-proof.

We first note that if p received a valid depth-v value val from p_{v-1}, then p knows that w.h.p. p_{v-1} sent that value at a timeslot $v' \geq v$, and in particular was awake at a timeslot $v' \geq v$. We give a formal proof of this below.

▷ Claim 13. In the presence of a external adversary, if a node p receives a valid depth-t value val from a node p_{t-1}, then except with negligible probability, p_{t-1} sent val at timeslot $t' \geq t$, and was *awake* at a timeslot $t'' \geq t-1$.

Proof. We prove this by induction on t. For $t = 0$, the claim is trivial since a node that wasn't awake at any time ≥ 0 can not send any messages. Assume that node p received a valid depth-t value val from q, for q that was not awake at any timeslot $\geq t$. Denote the corresponding nodes and values by $p_1, ..., p_{t-1} = q$, $\mathsf{val}_0, ..., \mathsf{val}_{t-1}$. Note that in particular, val_{t-1} is a valid depth-$(t-1)$ value w.r.t. p, sent by p_{t-2}, and thus by the induction assumption, it was sent by p_{t-2} at timeslot $t'' \geq t-1$.

Now assume that val was sent by p_{t-1} at a timeslot $r < t$, this means that p_{t-1} had not yet received val_{t-1} from p_{t-2}. Denote by Q the total number of queries to the VDF oracle made by any node by timeslot r, that number is bounded by $r \cdot n \cdot p(n, \Pi)$, in particular, the probability that val_{t-1} appeared as the valid output of a VDF oracle at timeslot $\leq r$ is at most $\frac{r \cdot n \cdot p(n,\Pi)}{2^n}$, and thus as long as $r = o(2^n)$, w.h.p. p_{t-1} could not have computed $O_{\mathrm{D}}(p_{t-1}, \mathsf{val}_{t-1}, \cdot)$ before timeslot r, for any $r < t$. Thus w.h.p. p_{t-1} could have computed $O_{\mathrm{D}}(p_{t-1}, \mathsf{val}_{t-1}, \cdot) = \mathsf{val}$ only after p_{t-2} sent val_{t-1}, which occurred at timestep at least $t-1$, i.e., p_{t-1} could have only queried $O_{\mathrm{D}}(p_{t-1}, \mathsf{val}_{t-1}, \cdot)$ at timestep at least $t-1$, and received a response at timestep at least t due to the behavior of O_{D}, thus p_{t-1} sent val at timestep at least t, and was thus awake at timestep at least $t-1$, as required. ◁

Proof of Lem. 9. The full algorithm is depicted in Alg. 3, specifically in the *wakeness vector updates* and *VDF chain extension* parts.

With Clm. 13 in mind, the main observation is that to prove the lemma, it suffices to prove that at every timeslot, every honest node extends a valid VDF chain of at least one other *honest* node. Notice that this is necessary as well, as otherwise, consider the following case involving 3 nodes p_1, p_2, p_3.

1. Node p_3 is corrupt and awake at timeslot 1, node p_1 is honest and awake at timeslot 2, node p_2 is honest and awake at timeslot 3.
2. Node p_3 computes $\mathsf{val}_1 = O_{\mathrm{D}}(p_3, \mathsf{val}_0, 1)$, and sends it to p_1, but not to p_2.
3. By instructions of some protocol Π, p_1 computes $\mathsf{val}_2 = O_{\mathrm{D}}(p_1, \mathsf{val}_1, 2)$ and broadcasts it.

Note now that p_2 by timeslot 3 did not receive $\mathsf{val}_1 = O_{\mathrm{D}}(p_3, \mathsf{val}_0, 1)$, and thus in particular, w.h.p. val_2 is not considered a valid depth-2 value w.r.t. to p, even though p_1 was indeed awake at timeslot 2.

The trivial solution to this would be to instruct each honest node p to extend the VDF chains it received from all nodes, thus extending all honest node's chains. However, this incurs a multiplicative n cost in communication. A better approach, and the one we implement in Alg. 3, is to let each honest node p sample $O(\log^2(n \cdot t))$ of the nodes it received valid depth-t VDF computations from, and extend those chains, thus guaranteeing that except with negligible probability, at least one of the chosen nodes is honest (due to the honest majority assumption in any admissible execution) and that for all t, except with negligible probability all honest nodes at all timeslots t observe valid chains from all honest nodes for the appropriate timeslots they were awake.

We prove this by induction on the timeslot. The base case of $t = 0$ clearly holds as all honest nodes have valid wakeness vectors. Assume correctness for t and consider time $t+1$, let p be an honest node awake at time t. By the induction assumption, p had valid wakeness vectors, and so p set $v_p^i[t-1] = 1$ for all honest nodes i awake at time $t-1$. In particular, by the behavior of the protocol, this implies that p observed valid depth-$(t-1)$ w.r.t. all honest nodes i awake at time $t-1$. Furthermore, the validness of the wakeness vectors of p

at timeslot t, the set of nodes i for which $v_p^i[t-1] = 1$ has an honest majority. As such, by lines 11, 12 of Alg. 3, when p chooses $O(\log^2 n)$ random nodes amongst these, the probability of an honest node *not* being chosen is at most $2^{\log^2 n} = \mathsf{negl}(n)$. Thus w.h.p. p extends in round t a valid depth-$(t-1)$ chain sent by an honest node, which implies that at round $t+1$, all honest nodes see a valid depth-t honest chain from p, and thus set p as being awake in their wakeness vectors, as required. This proves the completeness of the wakeness vectors at time $t+1$, i.e., that for all honest q awake at time $t+1$, and all $t' < t+1$ and all honest p awake at time t', we have that $v_q^p[t'] = 1$. The soundness of the wakeness vectors at time $t+1$ is implied by Clm. 13. This completes the induction argument and concludes the proof. ◀

Proof of Lem. 10. Denote the given protocol in the premise of Lem. 10 by Π. Lem. 9 proves that *wakeness vector updates* and *VDF chain extension* correctly implement valid wakeness vectors. Which in turns implies, due to the $(T_\mathrm{b}, T_\mathrm{b}, \rho)$-sleepy model, that Π is executed by every honest node by only considering messages from an honest majority at every timeslot. By the T-stateless property of Π, we thus have that Π solves the atomic broadcast problem correctly. Thus Π' inherits ϵ-Safety, and (ϵ, ℓ)-Liveness from Π. Protocol Π' decides blocks when Π decides them, thus the expected latency of Π' can not be worse than that of Π, and implementing *wakeness vector updates* and *VDF chain extension* requires an additive $O(\lambda n^2 \log^2 n)$ overhead in communication at every timeslot, as required. ◀

6 Additional related work

The early protocols designed for the synchronous sleepy model [24, 8, 2, 13] follow the longest chain approach of Nakamoto [23]. While inheriting its simplicity and elegance, they tend to suffer from high latency in the worst case. Furthermore, all of these protocols assume fixed and non-fluctuating participation of corrupt nodes. In this setting, the works of [3, 16] are geared at breaking the $\Omega(\frac{\lambda}{\gamma})$ latency barrier. Prism [3] designs a protocol whose latency is independent of λ under a certain optimistic condition, while [16] designs a protocol whose latency is independent of the participation rate γ (see Tab. 1 for the concrete bounds). Momose and Ren [22] designed the first protocol to achieve near optimal latency, namely $O(\Delta)$ in expectation. This was followed up by [21] in which the latency was further improved, as well as modifying the protocol to work in the *growing* participation setting [9]. The protocols of [22, 21] differ from the earlier protocols in the sleepy model, by emulating the PBFT [6, 28] approach to consensus, as opposed to longest chain based techniques.

Adversary restrictions. On the road to improve latency and supporting fluctuating participation of corrupt nodes, many works have imposed restrictions on adversary capabilities. The early work of [4] designs a protocol that can be instantiated in the fully-fluctuating sleepy model which is secure only against crash failures, with the *early decision* property. Namely, the latency of the protocol is proportional to the *actual* number of crash failures in the network. The work of [14] designs a $O(\Delta)$ expected latency protocol in the sleepy model that remains secure against fully-fluctuating participation, but restricts the adversary to only send time-stamped messages, i.e., each message sent by the adversary includes the *real* timeslot in which it was sent. In particular, this implies that the adversary is not allowed to equivocate (send conflicting messages to honest nodes in a given timeslot). The work of [18] considers a model where honest nodes know whether or not they will be awake or asleep in the next timeslot, and as such may use this information during the execution of the protocol (e.g., to announce their impending sleepiness to the rest of the nodes). The paper of [11] does not discuss fully-fluctuating participation of corrupt nodes, but their

protocol, when instantiated in the external adversary model we introduce in this paper, is secure against fully-fluctuating participation, and makes use of a VDF [5]. An early version of [21] showcased a protocol that supports fully-fluctuating participation and has optimal latency. They assume authenticated channels (i.e., identity of message sender is attached to each message), and further assume that messages that do not get delivered after Δ timeslots (because the recipients are asleep) get dropped from the network and never get delivered. This essentially prevents the adversary from performing forward simulation attacks.

Permissionless setting. In this paper we consider the setting where the total set of nodes allowed to participate in the protocol is fixed and known to all nodes. The work of Lewis-Pye and Roughgarden [20] explores a generalized version of the sleepy model in which the set of allowed participants in the protocol changes over time (via stake), and their identities are not known in advance to nodes.

PoSAT. It is worth noting that an existing consensus protocol is already resilient against fully-fluctuating node participation, under our external-adversary model! Specifically, despite the model not being discussed or defined directly in the paper, the PoSAT protocol [11], which makes use of time-based cryptography (specifically, VDFs) to achieve consensus in the growing participation setting, can be proven secure in the fully-fluctuating participation setting against an external adversary. Both the resilience and latency of the protocol are sub-optimal, however, with resilience $\rho = \frac{1}{1+e} \simeq 0.268$ and latency $O(\frac{1}{\gamma}\lambda\Delta)$, where γ is the participation rate of nodes, i.e., $\frac{1}{\gamma} > 1$, and λ is the security parameter. Our protocol obtains optimal resilience $\frac{1}{2}$ and expected constant latency, respectively. See Sec. 5 for details.

References

1. Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A blockchain protocol based on reconfigurable byzantine consensus. In *OPODIS*, volume 95 of *LIPIcs*, pages 25:1–25:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.OPODIS.2017.25.
2. 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. doi:10.1145/3243734.3243848.
3. 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. doi:10.1145/3319535.3363213.
4. Ziv Bar-Joseph, Idit Keidar, and Nancy A. Lynch. Early-delivery dynamic atomic broadcast. In *DISC*, volume 2508 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002. doi:10.1007/3-540-36108-1_1.
5. Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018. doi:10.1007/978-3-319-96884-1_25.
6. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002. doi:10.1145/571637.571640.
7. Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. DBFT: efficient leaderless byzantine consensus and its application to blockchains. In *NCA*, pages 1–8. IEEE, 2018. doi:10.1109/NCA.2018.8548057.
8. 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 *Lecture Notes in Computer Science*, pages 23–41. Springer, 2019. doi:10.1007/978-3-030-32101-7_2.

9 Francesco D'Amato, Joachim Neu, Ertem Nusret Tas, and David Tse. Goldfish: No more attacks on ethereum?! In *FC (1)*, volume 14744 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2024. doi:10.1007/978-3-031-78676-1_1.

10 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. doi:10.48550/arXiv.2310.11331.

11 Soubhik Deb, Sreeram Kannan, and David Tse. Posat: Proof-of-work availability and unpredictability, without the work. In *Financial Cryptography (2)*, volume 12675 of *Lecture Notes in Computer Science*, pages 104–128. Springer, 2021. doi:10.1007/978-3-662-64331-0_6.

12 Yuval Efron and Ertem Nusret Tas. Dynamically available common subset. Cryptology ePrint Archive, Paper 2025/016, 2025. URL: https://eprint.iacr.org/2025/016.

13 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. URL: https://eprint.iacr.org/2018/1119.

14 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. doi:10.4230/LIPICS.DISC.2023.41.

15 Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015. doi:10.1007/978-3-662-46803-6_10.

16 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. doi:10.1007/978-3-662-64331-0_4.

17 Vincent Gramoli. From blockchain consensus back to byzantine consensus. *Future Gener. Comput. Syst.*, 107:760–769, 2020. doi:10.1016/J.FUTURE.2017.09.023.

18 Pankaj Khanchandani and Roger Wattenhofer. Byzantine agreement with unknown participants and failures. In *IPDPS*, pages 952–961. IEEE, 2021. doi:10.1109/IPDPS49936.2021.00104.

19 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

20 Andrew Lewis-Pye and Tim Roughgarden. Permissionless consensus. *CoRR*, abs/2304.14701, 2023. doi:10.48550/arXiv.2304.14701.

21 Dahlia Malkhi, Atsuki Momose, and Ling Ren. Towards practical sleepy BFT. In *CCS*, pages 490–503. ACM, 2023. doi:10.1145/3576915.3623073.

22 Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. In *CCS*, pages 2295–2308. ACM, 2022. doi:10.1145/3548606.3559347.

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

24 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 380–409. Springer, 2017. doi:10.1007/978-3-319-70697-9_14.

25 Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018. doi:10.1007/978-3-319-78375-8_1.

26 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188.

27 Srivatsan Sridhar, Ertem Nusret Tas, Joachim Neu, Dionysis Zindros, and David Tse. Consensus under adversary majority done right. In *Financial Cryptography*, 2025. URL: https://eprint.iacr.org/2024/1799.

28 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. doi:10.1145/3293611.3331591.