Instant Block Confirmation in the Sleepy Model

Vipul Goyal[1,2], Hanjun Li[1], and Justin Raizes[1] *

[1] Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
vipul@cmu.edu, lihanjun1212@gmail.com, jraizes@andrew.cmu.edu
[2] NTT Research

Abstract. Blockchain protocols suffer from an interesting conundrum: owning stake in the Blockchain doesn't necessarily mean that the party is willing to participate in day to day operations. This leads to large quantities of stake being owned by parties who do not actually participate in the growth of the blockchain, reducing its security. Pass and Shi [23] captured this concern in the *sleepy model*, and subsequent work by Pass et al. [5] extended their results into a full Proof of Stake blockchain protocol which can continue to securely progress even when the majority of parties may be offline. However, their protocol requires 10 or more blocks to be added after a transaction first appears in the ledger for it to be confirmed. On the other hand, existing Byzantine Agreement based blockchain protocols such as Algorand [6, 14, 7] confirm transactions as soon as they appear in the ledger, but are unable to progress when users are not online when mandated.

The main question we address is:

> *Do there exist blockchain protocols which can continue to securely progress even when the majority of parties (resp. stake) may be offline, and confirm transactions as soon as they appear in the ledger?*

Our main result shows the answer to this question to be "yes". We present a Proof of Stake blockchain protocol which continues to securely progress so long as more than half of the online stake is controlled by honest parties, and instantly confirms transactions upon appearance in the ledger.

1 Introduction

Blockchain protocols provide significant economic and cryptographic implications, by means of the creation and maintenance of a globally agreed-upon log in an environment with low trust. The two most popular variants of blockchain protocols are Proof of Work (PoW) and Proof of Stake (PoS). Proof of Work unfortunately carries expensive hardware requirements and wastes a large amount of energy. Proof of Stake protocols bypass the wastefulness of Proof of Work by using the amount of stake a user owns in the system as a means for determining whether the user can contribute to its progression.

* The authors were supported in part by the NSF award 1916939, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

However, this approach carries the downside of requiring users who own stake in the system to be active in order for the system to securely progress. Since it is more desirable to own stake in the system than it is to actively participate in it (for instance, shareholders rarely want a say in the day-to-day operations of a company), much of the stake present in the system may be owned by inactive users. Current Proof of Stake protocols such as Algorand [6, 14, 7] are typically unable to effectively deal with this problem, requiring over half of the stake present in the system to be owned by users which are both *active* and honest.

Pass and Shi [23] were the first to address this problem, presenting a protocol which could securely progress even when the majority of stake in the whole system was owned by inactive users, so long as the majority of stake owned by active users was owned by honest users (i.e. the "honest active stake" was over half of the "active" stake). Their protocol was based on the follow-the-longest-chain ideas of Nakamoto [21], which, although revolutionary, bring with them the distinct downside of requiring a block to be "buried" beneath several others before it is confirmed.

On the other hand, Byzantine Agreement-based blockchain protocols allow blocks to be confirmed as soon as they are added to the chain. Unfortunately, it is nontrivial to securely allow progression while the majority of stake may be owned by inactive users and simultaneously scale to millions of users with this style of protocol. Algorand [6, 14, 7] addresses the latter issue by securely selecting a small committee to run a Byzantine Agreement protocol using a Verifiable Random Function (VRF) [19]. As we will see later, it is difficult to directly extend this committee approach to the heavily inactive setting, which will pose our main challenge.

We present a new blockchain system, using Algorand as a starting point, which allows transactions to be securely added to the ledger *regardless of the amount of stake owned by currently active users*. In comparison, Algorand in its base state cannot progress if even 40% of stake is owned by inactive users. Furthermore, basing the mechanism on Byzantine Agreement *prevents forks*, eliminating the major weakness of Sleepy Consensus [23] and its extension Snow White [5]. While Sleepy Consensus and Snow White require 10 or more blocks to be added after a transaction first appears in the ledger for it to be confirmed, our approach allows transactions to be confirmed *immediately upon appearing in the ledger*.

2 Technical Roadmap

2.1 Starting Point: Algorand

At a high level, in Algorand, a block is added to the chain by randomly selecting a committee, which then runs a Byzantine Agreement protocol to decide on the next block in a consistent manner. Since the probability of being in the committee is proportional to the number of coins a user possesses (users may appear multiple times in the committee if they own multiple coins), is unlikely for a majority corrupt committee to be selected.

Concretely, each coin is associated with a Verifiable Random Function (VRF) [19] for each round, and if its VRF is above some threshold, the owner is granted the right to participate one additional time in the current committee. This threshold is based on the number of coins (amount of stake) in the system, and is set to ensure an average committee size. The larger the actual committee size, the less likely it is to be majority-corrupt, which would allow the adversary to break security. Therefore, Algorand does not allow a committee which is too small to add blocks to the chain.

Unfortunately, in the sleepy model, parties do not know how much stake is online (owned by online users) at any time, so it is not clear how to set the threshold so as to frequently choose a committee large enough for secure progress. Furthermore, since the adversary may control just under half of all online stake and can selectively deliver corrupt messages to subsets of honest parties, it is difficult to even estimate the amount of online stake by using messages from other parties. Therefore, we will need a different approach for selecting a committee.

2.2 Selecting a Committee

To reliably choose a committee which is large enough to be majority honest with high probability, we will build on the idea of using the n parties with the highest VRFs seen as the committee. Though this approach ensures committee sizes are always large enough to make them majority honest with high probability, is obvious that *not all parties are guaranteed to see the same committee.*

In Algorand, players do not necessarily see the full committee, but all honest parties will accept messages from any member of the committee, even if they are received later. In contrast, using our committee selection procedure, the fact that one party accepts a party i as a committee member does *not* imply that another party who receives i's message accepts i as a committee member. For instance, if the adversary controlled 40% of the online stake, we would expect the true list of the top n VRFs to be about 40% corrupt. If the first party receives no corrupt VRFs, its committee will be completely honest. However, a second party receiving all of the corrupt VRFs will replace the lower 40% of the top n honest VRFs with corrupted VRFs. Furthermore, the second party cannot simply extend its committee to include the first party's committee, since it has no way of determining that the first party is not corrupt, and corrupt parties may pretend to see arbitrary committees.

As it is very difficult to entirely patch this flaw and guarantee that all parties see the same committee, we will instead focus our efforts on ensuring that the committees which are seen by various honest parties are "close enough".

2.3 Consensus with Different Committees

Standard Byzantine Agreement protocols rely on all parties knowing the same committee (even if they cannot directly communicate), and break down when this is not the case.

Somewhat surprisingly, we show that the general design of Algorand's binary Byzantine Agreement protocol does work with our committee selection, despite being designed for parties all using the same committee. However, this is not immediately obvious, and there is an additional nuance to iron out: to ensure that parties continue to do the same steps at the same times, parties should not halt at different steps in the Byzantine Agreement protocol. With Algorand's committee selection, when a party halts during an execution of the Byzantine Agreement protocol it is easy to produce a certificate which will convince the other parties to halt. This certificate simply consists of all of the committee messages received during the halting round, and since in Algorand, all parties will always accept the same committee members, parties receiving this certificate will also see a valid halt. However, as discussed previously, with our committee selection, parties *do not necessarily agree* on the committee, and so may reject some of the messages which caused the halting party to halt. To remedy this problem, we will use a binary Byzantine Agreement algorithm from [20] (based on the same design as Algorand's BA), which also has the advantage of reducing our requirements to only $> \frac{1}{2}$ honest online stake (from Algorand's $> \frac{2}{3}$). However, it is also not immediately obvious that the algorithm from [20] works with our committee selection, and we will need to show this formally.

Byzantine Agreement is not the only building block of Algorand which requires consistent views of the committee. Algorand uses a Graded Consensus protocol, which requires consistent views of the committee, to transform binary Byzantine Agreement into multivalued Byzantine Agreement protocol, allowing for *blocks* to be consistently decided on. Informally, Graded Consensus [6, 7] allows players to output a value and a grade, indicating how confident they are that all other players output the same value. In the multivalue Byzantine Agreement construction, binary Byzantine Agreement is used to decide on a default value (the empty block), or a variable value, and the non-default value should only be output if some honest party *knows* that all other honest parties are using the same value. Since we ultimately want to decide the next block to add, not the next bit to add, we will also show that the Graded consensus algorithm of [20] with only minor modifications surprisingly still works with our notion of "close enough" committees, despite being designed for the scenario where there is a single committee which all parties accept messages from.

2.4 Summary of Challenges and Theorem Statement

In summary, the challenges we must overcome are:

- Ensuring the committees seen by all honest players are "close enough". The concrete properties we achieve are described in Lemma 1.
- Showing that the binary Byzantine Agreement algorithm from [20] works when parties use different committees which are "close enough".
- Showing that the Graded Consensus protocol from [20] works with minor modifications when parties use different committees which are "close enough".

Theorem 1. *(Informal) If less than half the online stake is adversarially owned, there exists a blockchain protocol in the sleepy model which, with overwhelming probability, does not fork and enters transactions into the ledger at a constant rate on average.*

3 Related Work

Sleepy Consensus Pass and Shi [23] initiated the study of consensus in the sleepy model, where parties may be either awake or asleep at any given point in time. Upon waking, a previously sleeping user receives both all messages actually sent during the time it was asleep, and some set of adversarially generated messages, intermingled. They showed that it is possible to achieve consensus in this setting if and only if the number of awake honest parties are strictly greater than the number of adversarial parties (which are always awake). However, their protocol requires many blocks to be added before a transaction can be confirmed with high confidence. Snow White [5], a blockchain built using the sleepy consensus protocol, requires 10 additional blocks to be added for 99% confidence when the adversary controls only 16.5% of the online stake. If the adversary controls 30%, the number of blocks required jump to 33, and the authors only note that Snow White is theoretically capable of dealing with 49% corruption, without providing a concrete number of blocks to wait. We aim to achieve immediate confirmation as soon as a transaction appears in the chain, without waiting for additional blocks to be added.

Blockchain Protocols Both Snow White [5] and Ouroboros Genesis [3] can securely progress in the sleepy model, but requires transaction to be buried below many blocks before being confirmed. Algorand [7] enables immediate confirmation upon a transaction appearing in the chain, but is unable to deal with large quantities of stake being offline. Algorand's method of dealing with offline parties is to allow parties to determine if they will be part of any committees for the next, say, week's worth of blocks. If the user will not be part of a committee for the next week, they can go inactive with good conscience. We wish to treat the more general case where parties *cannot* commit to being active at particular points in time. This may be due to vacations, network interruptions, or simply the user not wanting to participate in the chain's progression. Nevertheless, Algorand forms a strong starting point for our protocol.

Other blockchain protocols include Ouroboros [17], Ouroboros Praos [8], Ouroboros Genesis [3], and the well-known Bitcoin protocol [21]. These all fall into one or more of the categories discussed previously (PoW, long confirmation times, majority online). Thunderella [24] is able to give instant confirmation during the optimistic case of the committee being over $\frac{3}{4}$ honest, but slows down significantly when this is not the case.

Accelerating Meta-solutions [3] Prism [4] and Parallel Chains [13] give meta-solutions which can be applied to existing solutions to significantly improve

[3] We were made aware of these works by helpful reviewers.

transaction throughput and/or confirmation times. In particular, the results of Parallel Chains can be applied to sleepy PoS blockchains such as Snow White and Ouroboros variants, enabling blockchains which progress quickly in the sleepy model. This matches our target application. However, our work additionally contributes to the understanding of how BFT protocols can be adapted to the sleepy setting.

Prism, on the other hand, is specifically for PoW blockchains, and comes with the associated energy waste.

Byzantine Agreement with Unknown Participants Alchieri et. al. [2, 1] characterize the possibility of solving Byzantine Agreement when participants do not know all other participants. Their main theorem identifies necessary and sufficient properties for the knowledge graph (defined by the set of parties and which other parties each knows) under which there exists a Byzantine Agreement algorithm robust against an adversary which can corrupt up to t parties.

Though Alchieri et. al. provide protocols solving this problem, their results are unfortunately insufficient for our setting. We will additionally require security against an adversary which controls many more corrupt parties than the size of any honest view - the whole set of corrupt players, not just the small fraction of those selected to be committee members. This is in contrast to the adversary considered by Alchieri et. al., may only corrupt up to t parties *total*, where all honest views are at least size $2t + 1$.

3.1 Comparison of Confirmation Times and Communication Complexity

Confirmation Times The time for a transaction to be confirmed in the blockchain depends on both the time to first enter the ledger and the time for it to stabilize in the chain. Our solution entirely eliminates the latter, which can be very large in longest-chain style protocols such as Snow White or Ouroboros. Parallel Chains is able to eliminate that weakness in those protocols, at the cost of increased computation in proportion to the speedup, since parties have to track/participate in multiple blockchains.

Since Algorand is the most similar to our protocol and its performance is well understood by the community, it will form our main comparison point. In both our protocol and Algorand, transactions must be present at block proposal to be included, so transaction confirmation time is at most twice the time for a block to be added. The main contribution to Algorand's block addition time is the number of Byzantine Agreement rounds, the expectation of which is given by the expected number of rounds until an honest leader is elected. In this, our protocol matches Algorand, since we borrow Algorand's leader election. However, due to the complexities of adapting to the sleepy model, the constant time for each Byzantine Agreement round is significantly higher for our protocol. In particular, it takes 4 messages to receive each committee member's message, as opposed to just 1 in Algorand. This leads to our protocol's block addition time being roughly $4x$ Algorand's. Algorand's best case scenario is significantly faster than

our protocol's, since it is able to terminate the Byzantine Agreement almost immediately when the next block is proposed by an honest party.

Communication Complexity One of the major strengths of current PoS blockchain solutions is the subquadratic communication complexity. For a committee size of n and a total number of online users N, current solutions usually achieve a communication complexity of $O(nN)$, or $O(n)$ in the broadcast channel model. Our protocol unfortunately requires a higher communication complexity of $O(N^2)$, or $O(N)$ in the broadcast channel model.

However, future work may be able to mitigate this downside by integrating our protocol as a fallback option to base Algorand. When participation is high, Algorand provides low communication complexity, and during periods of highly sporadic participation, our protocol can be used to provide progress when Algorand would otherwise stall. Using our protocol only when participation is low will also help mitigate the communication complexity slightly, since a lower number of online users incurs a lower communication cost. More work will be required to integrate the two protocols without introducing security issues.

Additionally, the approach of following the highest n VRFs means that after receiving only a few messages, many others become entirely obsolete. Future work may be able to exploit this at the gossip network level by heuristically forwarding only the top n messages seen.

4 Definitions

4.1 Blockchain Execution Model

To allow us to capture the ability of users to be inactive, we adopt the sleepy execution model of Pass and Shi [23], as extended by Bentov et. al. [5], with one major difference: we consider a more powerful adversary who can *instantly* corrupt any honest party. In the sleepy model, generally, parties may be either awake or asleep (corrupt parties are assumed to always be awake). The two states differ in that parties may only receive messages when they are awake.

(Weakly) Synchronized Clocks We assume all player clocks differ by at most a constant at all times. As noted by Pass and Shi [23], the clock offset can be generically transformed into a network delay. Therefore, without loss of generality, we will consider players to have synchronized clocks.

Network Delivery The adversary is responsible for delivering messages between players. We assume that all messages sent by honest players are received by all awake honest players within Δ time steps, but that the adversary may otherwise delay or reorder messages arbitrarily. It must be emphasized that the adversary can *exactly* control the precise time that an honest player receives a message.

Sleeping players do not receive messages until they wake, whereupon they receive all messages they would have received had they not slept. Note that this may include a polynomial number of adversarially inserted messages, and the ordering of all messages received upon waking may be adversarially chosen.

Corruption Model Corrupt parties may deviate arbitrarily from protocol (ie. exhibit Byzantine faults), and are controlled by a probabilistic polynomial time adversary which can see the internal state of corrupt players. At any time, the adversary may instantly corrupt an honest party or cause them to sleep until a future time. However, the adversary is not capable of seeing a message, corrupting the sending party, then erasing the message from the network.

At any time, the adversary may spawn new corrupt users (distinct from parties, which represent a unit of stake). This does *not* increase the amount of adversarially owned stake in the system.

Secure Bootstrapping Assumption As noted by Bentov, Pass, and Shi [23, 5], in this model it is impossible to achieve a secure blockchain protocol using only common knowledge of the initial committee. Therefore we assume a trusted bootstrapping procedure, as Sleepy Consensus [23] and Snow White [5] do. Future work may be able to sidestep the impossibility result using a similar modification to the execution model as Ouroboros Genesis [3].

4.2 Tools

Verifiable Random Functions A Verifiable Random Function (VRF) [19] takes in a secret key sk_i and a seed s, and returns a random number in the range along with a proof. Anyone who knows the public key pk_i associated with sk_i can verify that $VRF(sk_i, s)$ was computed correctly, but cannot compute $VRF(sk_i, s')$ themselves without knowing sk_i. Due to the complexity of instantiating VRFs when players may choose their own seeds, we model them as random oracles, and direct readers to [7] for a more in-depth treatment of the subject.

Byzantine Agreement The standard definition of Byzantine Agreement [25] is given below. We say a party is honest if they behave according to the protocol specification throughout its entire execution.

Definition 1. *A protocol $\mathcal{P}$ achieves **Byzantine Agreement** with soundness s if, in an execution of $\mathcal{P}$, every honest player j halts with probability 1 and the following two properties both hold with probability $\geq s$:*

1. Agreement: *All honest parties output the same value.*
2. Consistency: *If all honest players input the same value v, then all honest players output v.*

If parties input values in $\{0, 1\}$, we say it achieves binary Byzantine Agreement.

Player Replaceability The idea of player replaceability was introduced by Chen and Micali [6] as a means of preventing targeted attacks on committee members from disrupting Algorand's binary Byzantine Agreement protocol. Consider a protocol executing over a very large set of players where a small subset of players (the committee) is chosen to carry out the r'th round of a Byzantine Agreement protocol. Informally, a Byzantine Agreement protocol is player replaceable if the

protocol still achieves agreement and consistency, despite the following conditions: after each round the old committee may be immediately corrupted and a new committee is selected to carry out round $r + 1$.

Graded Broadcast Graded broadcast was introduced in [12], and informally allows parties to receive a message from a dealer and express how confident they are that all other parties received the same message.

Definition 2. *A protocol achieves* ***Graded Broadcast*** *if, in an execution where the dealer D holds value v_D, every player i outputs (g_i, v_i) where $g_i \in \{0, 1, 2\}$ such that:*

1. *If D is honest, then every honest player outputs $(2, v_D)$.*
2. *For any honest parties i and j, $|g_i - g_j| \leq 1$.*
3. *For any honest parties i and j, if $g_i > 0$ and $g_j > 0$, then $v_i = v_j$.*

We say a protocol achieves $\{0, 1\}$-graded broadcast [20] if g_i takes values in $\{0, 1\}$, property 3 holds (2 holds trivially), and if players output $(1, v_D)$ when D is honest. For simplicity, in a $\{0, 1\}$-graded broadcast, we say a party *accepts* a value if it has grade 1, and *rejects* a value if it has grade 0.

Graded Consensus For the reduction of multivalued Byzantine Agreement to binary Byzantine Agreement, we will additionally need the notion of Graded Consensus [6, 7], which is a relaxation of consensus, and extends the concept of graded broadcast [12].

Definition 3. *A protocol $\mathcal{P}$ achieves* ***Graded Consensus*** *if, in an execution of $\mathcal{P}$ where every player i inputs v'_i, every player i outputs a grade g_i and a value v_i such that:*

1. *For any honest players i and j, $|g_i - g_j| \leq 1$*
2. *For any honest players i and j, if $g_i > 0$ and $g_j > 0$, then $g_i = g_j$*
3. *If there exists a value v such that $v'_i = v$ for all honest players i, then $v_i = v$ and $g_i = 2$ for all honest players i*

4.3 Other Notation

Additionally, we will use the following pieces of notation which have not been covered so far:

- H represents the set of all honest parties.
- V_i represents participant i's current view of the committee
- N denotes the total amount of online stake at any time in a blockchain

Proof of Stake Abstraction In a proof of stake blockchain, users are granted voting power proportional to how much currency they own in the blockchain. Hence, we consider each unit of currency to be a party. Users owning multiple units of currency act as multiple parties.

5 The Blockchain Protocol

We will make use of Algorand's leader election (alg 1) in several of our protocols. The following claim is modified from [6, 7] to reflect our treatment of VRFs as random oracles and its usage in the sleepy model.

Proposition 1. *[14, 6, 7] At the end of algorithm 1, if the adversary owns less than $\frac{1}{2}$ the online stake, then with probability $> \frac{1}{2}$, all honest parties output the same message m, which was input by an honest party.*

Proof. If an online honest party has the highest VRF for round r and inputs m, all honest parties output m. Since we model VRFs as random oracles and the adversary owns less than $\frac{1}{2}$ of the online stake, this occurs with probability $> \frac{1}{2}$.

5.1 Committee Selection

The committee view formation algorithm needs to fulfill two different goals. First, views must be majority honest, motivating a uniformly random selection process with proportion to the amount of money (or stake) each user owns. As in Algorand, users may be selected multiple times for the same committee view, so long as they own enough stake. Second, the resulting committee views must be similar enough that our binary Byzantine Agreement protocol will work. Informally, for Byzantine Agreement to work, we need to ensure than any two V_i, V_j overlap on more than half their respective views. The concrete properties we achieve are actually stronger than this, and are described in Lemma 1.

Strawman: Committee Discovery Starting with the base idea of forming a temporary committee consisting of the n highest VRFs seen, a natural first approach is to attempt to discover the temporary committee members which other honest players have selected and take the most commonly selected parties as your final committee, similar to the participant discovery idea from [2, 1]. Intuitively, since each honest temporary view is likely to be majority honest, parties selected by many parties in your temporary view are likely to both be honest and appear in many other honest views. Similarly, parties which are selected by only a few parties in your temporary view are liable to either appear in very few honest views globally, or to have been nominated by dishonest parties. Concretely:

1. All parties send their VRF for the round. Each party i takes the owners of the highest n VRFs received to be its temporary committee V_i^*.

Algorithm 1: Leader Election [14, 6, 7]
Input: message m_i' 1) Propagate $VRF(i, r), sig_i(m_i')$ 2) Set $m_i \leftarrow m_j'$ such that $sig_j(m_j')$ was received and $VRF(j, r)$ was the highest VRF seen **Output:** m_i

Algorithm 2: Committee Selection

Input: m_i, committee size n, round r

1) Propagate $VRF(i,r), sig_i(m_i,r)$
2) Set $V_i^* = \{j : VRF(j,r)$ was one of the highest n valid VRFs received during step 1$\}$.
   Propagate $VRF(j,r), sig_j(m_j,r)$ for each $j \in V_i^*$
3) Let $V_{U,i} = \{j : VRF(j,r)$ was one of the highest n valid VRFs received during step 2.$\}$
   Set $V_i = V_i^* \cap V_{U,i}$.
   Set messages $= \{(j, m_j, sig_j(m_j)) : j \in V_i$ and $sig_j(m_j,r)$ was received$\}$

Output: V_i, messages

2. All parties propagate their temporary committees. Each party i takes its final committee to be $V_i = \{j : j \in V_k^*$ for more than $\frac{n}{2}$ parties $k \in V_i^*\}$.

At the surface level, this seems quite promising - since temporary committees were selected randomly, each temporary committee is highly likely to be over half honest. Any party seen by the honest portion of your temporary committee will end up in your final committee, and the adversary can't add parties your final view which were not seen by at least one honest party, since that would require $> \frac{n}{2}$ corrupt parties to appear in your temporary committee.

Preventing Majority-Corrupt Committees Unfortunately, this strategy is not quite as successful at keeping corrupt parties out of your final committee as it might seem. The core issue is that while no single honest party sees many corrupted parties in its temporary committee, different honest parties may see different corrupted parties, and the adversary may use corrupted votes to reach the threshold for acceptance. Section A in the appendix illustrates a concrete example of the issue.

The key insight to ensuring honest parties end up with committees which are very similar, but don't include too many corrupted parties, is to *only remove* parties from your temporary committee, rather than allowing them to be added (in the example, adding parties resulted in a majority corrupt committee!).

Generally, we will assume that the committee size n is much smaller than the amount of online stake N at any time.

Lemma 1. *At the end of protocol 2, if the adversary controls* $\leq \frac{1}{2} - \epsilon$ *fraction of the online stake, then with probability* $\approx 1 - \left(1 - 4\epsilon^2\right)^{n/2}$ *the following holds:*

1. $\left|\bigcup_{i \in H} V_i\right| \leq n$
2. $\exists$ *set of honest parties* H_C *such that* $|H_C| \geq \frac{n}{2}$ *and* $H_C \subseteq V_i \ \forall i \in H$.

Proof (Sketch). Property 1 relies on the fact that every honest player sees every honest temporary committee V_i^*. This means that every honest player will

remove at least every VRF in $\bigcup_{i \in H} V_i^*$ beyond the first n. However, they never add additional parties to their view.

For the second property, consider the list of all VRFs of online parties for the round, regardless of what messages they send. Define H_C to be the set of honest parties whose VRF is among the highest n in this list. These will appear in all honest temporary views, and cannot be removed, since there are simply not enough higher VRFs among the online parties.

The size of this set then follows by upper bounding the number of corrupt parties among the highest n VRFs in the complete list. Since there are many more than n online parties, we can approximate this with $Bin(n, \frac{1}{2} - \epsilon)$. A bound by Hoeffding [15] upper bounds the probability that this number is $\geq \frac{n}{2}$.

It is worth noting that the honest core H_C is unknown, though it is guaranteed to exist. We do not know how to find it, but as we will show, it is not necessary to know H_C in order to use it; it is sufficient that it simply exists.

Section B in the appendix discusses the committee size requirements for safety using our committee election process.

5.2 Binary Byzantine Agreement

In this section, we discuss the importance of completing the Byzantine agreement protocol in the same step, as well as why that is difficult to achieve adaptively in the sleepy model. Then, we show that both the binary Byzantine Agreement protocol of Micali and Vaikuntanathan [20] and the $\{0, 1\}$ Graded Broadcast protocol used in it still work with our committee selection procedure, despite being designed for the scenario where all honest parties agree on the committee.

Observe that if players were to complete the Byzantine agreement protocol in different steps, then they would begin to do different steps of the overlying blockchain protocol, with some parties operating "in the future". Though we can easily prevent messages sent for different steps of the protocol interfering with each other, it is not so simple for a party lagging behind to "catch up", nor is it easy to convince a party speeding ahead to let the others catch up. Furthermore, splitting the parties like this opens opportunities for the adversary, who may be able to achieve majority online stake in the "future", where only some honest parties are currently operating, despite having minority online stake overall.

Algorand avoids this issue by relying on the fact that if one party accepts a message from a committee member, all other players will also accept that message (as being from a committee member) if they receive it. Thus, a player who sees messages causing them to halt early can ensure that all other players halt early as well by simply propagating those messages.

This strategy does not work under our committee selection, since it is *not* the case that if i accepts k as a committee member, j will too after learning about k. For instance, if k were corrupt, then during a Committee Selection execution (algorithm 2) it is easy for k to appear in V_i^* but not V_j^* by simply not sending a message to j by the deadline, resulting in k never appearing in V_j. Furthermore, it is extremely important to not expand V_j based on other party's

views, since corrupt parties may claim an arbitrarily corrupt view. Thus, we will use the strategy presented in [20], which ensures all parties exit the Byzantine Agreement execution at the same time.

To achieve resilience against slightly less than half of any committee view being corrupted, we will need a protocol similar to Graded Broadcast. The only difference in our requirements is in property 1: instead of each $i \in H$ being required to output $(1, m_D)$ when D is honest, i is only required to do so when $D \in H_C$. Algorithm 3 describes a parallel version of this, modified from [20].

Proposition 2. *If at all times, $|V_i| \leq n$ for honest i and there exists a set of honest players H_C of size $> \frac{n}{2}$ common to all honest views, then algorithm 3 achieves the following properties:*

1. *If $D \in H_{C,1}$ and sends m_D, then i accepts m_D from D.*
2. *If honest parties i, j accept m_D, m'_D, respectively, from D, then $m_D = m'_D$.*

Proof (Sketch). If $D \in H_{C,1}$, then all members of $H_{C,2}$ receive and forward $sig_D(m_D)$. Each honest player therefore receives this from $> \frac{n}{2}$ sources, and do not ever receive $sig_D(m'_D)$ for $m_D \neq m'_D$.

If $i \in H$ accepts a message, then one of the sources they received it from in step 2 was a member of $H_{C,2}$, so all other parties also receive the message i accepts. This means that no $j \in H$ will accept a different message.

Proposition 3. *If $|V_i \bigcup V_j| \leq n$ for all honest parties i, j during an execution of algorithm 4, then an honest party cannot follow substep (a) in the same step an honest party follows substep (b).*

Proof (Sketch). This would require i and j to see more unique votes combined than exist in the union of their views.

Proposition 4. *If V_i contains at least $\frac{n}{2} + 1$ honest parties and no more than n parties total during an execution of algorithm 4, then if at some step all honest parties agree on a bit b, all honest parties continue to agree on the same bit b.*

Proof. By proposition 2 every honest party i accepts at least $\frac{n}{2} + 1$ votes for b from the honest parties in their view and no more than $n - (\frac{n}{2} + 1)$ votes for $1 - b$. Therefore i sets $v_i = b$ at the end of the step.

Algorithm 3: $\{0, 1\}$ Graded Broadcast [20]

Input: v'_i, n, round r

1) $V_{i,1}, ms_1 \leftarrow$Committee Selection(v'_i, n, r)
2) $V_{i,2}, ms_2 \leftarrow$Committee Selection(ms_1, n, r)
3) For each $k \in V_{i,1}$, accept m_k if $sig_k(m_k)$ was received from $> \frac{n}{2}$ members of $V_{i,2}$ and no other $sig_k(m'_k)$ was received.i

Output: Accepted Messages, $V_{i,1}$

Algorithm 4: Byzantine Agreement [20]

Input: v_i, n

for $i \leftarrow 0$ **to** k **do**

1) Set mc = Leader Election($b \leftarrow$Uniform($\{0,1\}$))
2) $\{0,1\}$-Graded Broadcast(v_i)
   a) **if** *If #(0) accepted* $> \frac{n}{2}$ **then** set $v_i = 0$
   b) **else if** *If #(1) accepted* $> \frac{n}{2}$ **then** set $v_i = 1$
   c) **else** set $v_i = mc$

Output: v_i

Proposition 5. *If* $|V_i \bigcup V_j| \leq n$ *for all honest parties* i, j *and* V_i *contains at least* $\frac{n}{2}+1$ *honest parties during an execution of algorithm 4, then with probability at least* $\frac{1}{4}$*, at the end of step 2 all honest parties are in agreement.*

Proof (Sketch). If an honest party sets $v_i = b$, then all others either set b or set mc, which matches b with probability $\frac{1}{2}$ and is agreed upon with probability $\frac{1}{2}$.

Lemma 2. *If the following properties hold, then algorithm 4 achieves binary Byzantine Agreement with soundness* $> 1 - \frac{3}{4}^k$.

1. $|V_i \bigcup V_j| \leq n$ *for all honest parties* i, j.
2. $\exists$ *a set of honest players* H_C *of size* $> \frac{n}{2}$ *such that* $H_C \subseteq V_i \ \forall i \in H$.

Proof. Consistency follows immediately from proposition 4.

By proposition 4, agreement will hold at the end of an execution of algorithm 4 if it holds at the start of any step. By proposition 5, the probability that this does not occur during any of the k steps is $< \frac{3}{4}^k$.

5.3 Block Proposal

In the binary Byzantine Agreement protocol, parties decide whether or not to add a particular block to the chain. To extend this to deciding *which* block to add to the chain, if any, parties will first attempt to decide a block to vote on during the binary Byzantine Agreement execution. Intuitively, if one honest party i believes the vote is about whether or not to add a block B to the chain, and another honest party j believes the vote is about whether or not to add a different block B' to the chain, then the outcome of the vote should be that no block is added - otherwise, i will add B and j will add B'!

To ensure a nonempty block is only added when all honest parties agree on it, Algorand uses a Graded Consensus protocol before running the binary Byzantine Agreement protocol. Roughly, this ensures that honest parties will decide to add a nonempty block B as a result of the binary Byzantine Agreement execution only if some honest party *knows* that all honest parties think the vote is about whether or not to add B.

We will show that the graded consensus algorithm from [20] surprisingly still works with the notion of "close enough" committees achieved by Algorithm 2

(laid out in Lemma 1). The only modification necessary is the threshold required for a set of signatures to be consistent, since local views of the committee may be different sizes. Algorithm 5 describes the modified graded consensus algorithm.

Algorithm 5: Graded Consensus

Input: v'_i, n

1) messages$_1$, $V_{i,1} \leftarrow$ {0,1}-Graded Broadcast
2) $m_{i,2} \leftarrow \perp$
   if *accepted* $> \frac{n}{2}$ *signatures for* v' *in step 1* **then**
   $m_{i,2} \leftarrow sig_i(v', 2)$
   messages$_2$, $V_{i,2} \leftarrow$ {0,1}-Graded Broadcast$(m_{i,2})$
3) $m_{i,3} \leftarrow \perp$
   if $> \frac{n}{2} + 1$ *signatures* $sig_j(v'', 2)$, *for* $j \in V_{i_2}$ **then**
   $m_{i,3} \leftarrow sig_i(\{sig_j(v'', 2) : sig_j(v'', 2)$ was accepted from $j \in V_{i,2}\})$
   messages$_2$, $V_{i,3} \leftarrow$ {0,1}-Graded Broadcast$(m_{i,3})$
4) A signature set is *consistent* if it contains $> |V_{i,2}| - \frac{n}{2}$ signatures from members of $V_{i,2}$.
   if $> \frac{n}{2}$ *consistent signatures sets for* $(v''', 2)$ *were accepted in step 3* **then**
   Output $(2, v''')$
   else if ≥ 1 *consistent signature sets for* $(v''', 2)$ *was accepted in step 3* **then**
   Output $(2, v''')$
   else
   Output $(0, \perp)$

Output: (g_i, v_i)

Proposition 6. *If each view change satisfies* $|V_i \bigcup V_j| \leq n$ *and there exists a set of honest parties* H_C *such that* $|H_C| > \frac{n}{2}$ *and* $H_C \subseteq V_i$ *for any honest players* i, j *then algorithm 5 achieves graded consensus.*

Proof (Sketch). For the first property, if $g_i = 2$ for an honest player i, then i receives more consistent sets of signatures in step 3 than half its view. One of these must have been from a member of $H_{C,3}$, so all honest parties also receive a set. Since two honest views cannot contain more than n unique parties in their union, all honest parties consider this set consistent.

The second property starts with the observation that if no honest party signs $(v, 2)$ in step 2, then no honest party receives a consistent signature set for $(v, 2)$ in step 4, preventing them from outputting v with a non-zero grade. This follows from honest views containing only $|V_{2,i}| - |H_C|$ corrupted parties. Then, we show that two different values will not be signed by honest parties in step 2, since this would require two honest parties to accept more unique messages in step 1 than exist in the union of their views.

The third property uses the fact that all honest parties accept all messages from members of H_{C_1} during the {0,1} graded broadcast, so each these messages

are signed by members of $H_{C,2}$ in step 2. This causes every honest party to send its own set, and honest signature sets are consistent for every honest party, leading to an output grade of 2 for v.

To propose a block, every player will begin by creating a candidate block and broadcasting it alongside a VRF and a short description of the block, such as its hash. The block and its description will be propagated separately, so as to allow fast propagation of the description, which is much shorter than the block itself.

After waiting for the network delay to complete, all players will begin Graded Consensus (alg. 5) using the block description with the highest associated VRF seen as their input. Finally, the block voted on during Byzantine Agreement will be the value output from algorithm 5. This is summarized in algorithm 6.

5.4 Putting It All Together

In this section, we describe how our modifications fit into Algorand as a whole and present our final result.

Our final protocol takes the following parameters, which must be common to all honest players.

- n: the committee size
- k: the number of iterations for the binary Byzantine Agreement
- Δ_N: the network delay

Note that a good parameter choice for n can be determined given a desired safety parameter and an assumed maximum fraction of online stake controlled by the adversary ($\frac{1}{2} - \epsilon$). For this reason, the safety parameter and ϵ may be consider to be parameters in place of the committee size n.

It is also worth noting that the parameter n provides explicit bounds on the committee size, independent of the amount of online stake [4] and in contrast to Algorand, where it is only an expected committee size. By lemma 1, every honest committee view will have size at least $\frac{n}{2} + 1$, but no more than n, regardless of the amount of online stake. In contrast, committee sizes in Algorand may vary drastically if the amount of online stake is over or underestimated.

Theorem 1. *If less than half of all online stake is adversarially owned and all honest parties input the same parameters to an execution of algorithm 6, the following properties hold:*

1. *With overwhelming probability, all honest players output the same block B.*
2. *With probability $> \frac{1}{2}$, B is not empty (i.e. contains transactions).*

Proof (Sketch). The first property holds when all honest players output 0 in the Byzantine Agreement protocol, or if all honest players output 1 in the Byzantine Agreement protocol and the same block in the Graded Consensus protocol. With

[4] Ignoring the possibility of less than n units of stake being online at all, which we consider to be an extreme corner case.

Algorithm 6: Next Block

Input: committee size n, Byzantine Agreement iterations k, network delay Δ_N, current log L

// `During every subroutine, wait for` Δ_N `time between steps`

1) Wait for Δ_N time to receive transactions.
   Construct a block $B_i^{(3)}$ from the transactions received.
2) $B_i'' \leftarrow$ Leader Election$(B_i^{(3)})$
3) $(g_i, B_i') \leftarrow$ Graded Consensus(B_i'')
   During the Graded Consensus execution, ignore invalid blocks.
   if $g_i = 2$ **then** $v_i' \leftarrow 1$
   else $v_i' \leftarrow 0$
4) $v_i \leftarrow$ Byzantine Agreement(v_i')
   if $v_i = 1$ **then** $B_i \leftarrow B_i'$
   else $B_i \leftarrow$ empty block

Output: B_i

probability $\left(\frac{3}{4}^k + 2k\left(1-4\epsilon^2\right)^{n/2}\right)$ and $6\left(1-4\epsilon^2\right)^{n/2}$ respectively, these cases do *not* occur, and we can apply a union bound.

Whenever an honest leader is elected, they propose a non-empty block and that block is unanimously chosen by honest parties. An honest leader is elected with probability $\frac{1}{2}$.

References

1. Alchieri, E.A.P., Bessani, A., Greve, F., d. S. Fraga, J.: Knowledge connectivity requirements for solving byzantine consensus with unknown participants. IEEE Transactions on Dependable and Secure Computing **15**(2), 246–259 (2018)
2. Alchieri, E.A.P., Bessani, A.N., da Silva Fraga, J., Greve, F.: Byzantine consensus with unknown participants. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5401, pp. 22–40. Springer (2008). https://doi.org/10.1007/978-3-540-92221-6_4, https://doi.org/10.1007/978-3-540-92221-6_4
3. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 913–930. ACM (2018). https://doi.org/10.1145/3243734.3243848, https://doi.org/10.1145/3243734.3243848
4. Bagaria, V.K., Kannan, S., Tse, D., Fanti, G.C., Viswanath, P.: Prism: Deconstructing the blockchain to approach physical limits. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November

11-15, 2019. pp. 585–602. ACM (2019). https://doi.org/10.1145/3319535.3363213, https://doi.org/10.1145/3319535.3363213
5. Bentov, I., Pass, R., Shi, E.: Snow white: Provably secure proofs of stake. IACR Cryptology ePrint Archive **2016**, 919 (2016), http://eprint.iacr.org/2016/919
6. Chen, J., Micali, S.: Algorand (2016)
7. Chen, J., Micali, S.: Algorand: a secure and efficient distributed ledger. Theoretical Computer Science **777**(nil), 155–183 (2019). https://doi.org/10.1016/j.tcs.2019.02.001, https://doi.org/10.1016/j.tcs.2019.02.001
8. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 66–98. Springer (2018)
9. Dolev, D., et al.: The byzantine generals strike again. J. Algorithms **3**(1), 14–30 (1982)
10. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) Peer-to-Peer Systems. pp. 251–260. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
11. Dwork, C., Peleg, D., Pippenger, N., Upfal, E.: Fault tolerance in networks of bounded degree. SIAM Journal on Computing **17**(5), 975–988 (1988)
12. Feldman, P., Micali, S.: An optimal probabilistic algorithm for synchronous byzantine agreement. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S.R. (eds.) Automata, Languages and Programming. pp. 341–378. Springer Berlin Heidelberg, Berlin, Heidelberg (1989)
13. Fitzi, M., Gazi, P., Kiayias, A., Russell, A.: Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. IACR Cryptol. ePrint Arch. **2018**, 1119 (2018), https://eprint.iacr.org/2018/1119
14. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. pp. 51–68 (2017)
15. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**(301), 13–30 (1963)
16. Katz, J., Koo, C.Y.: On expected constant-round protocols for byzantine agreement. In: Advances in Cryptology - CRYPTO. pp. 445–462. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2006). https://doi.org/10.1007/11818175_27, https://doi.org/10.1007/11818175_27
17. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Annual International Cryptology Conference. pp. 357–388. Springer (2017)
18. Micali, S.: Very simple and efficient byzantine agreement. In: Papadimitriou, C.H. (ed.) 8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA. LIPIcs, vol. 67, pp. 6:1–6:1. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ITCS.2017.6, https://doi.org/10.4230/LIPIcs.ITCS.2017.6
19. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th annual symposium on foundations of computer science (cat. No. 99CB37039). pp. 120–130. IEEE (1999)
20. Micali, S., Vaikuntanathan, V.: Optimal and player-replaceable consensus with an honest majority. Tech. Rep. MIT-CSAIL-TR-2017-004 (2017), http://hdl.handle.net/1721.1/107927

21. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system.(2008) (2008)
22. Pass, R., Shi, E.: Hybrid Consensus: Efficient Consensus in the Permissionless Model. In: Richa, A.W. (ed.) 31st International Symposium on Distributed Computing (DISC 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 91, pp. 39:1–39:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2017). https://doi.org/10.4230/LIPIcs.DISC.2017.39, http://drops.dagstuhl.de/opus/volltexte/2017/8004
23. Pass, R., Shi, E.: The sleepy model of consensus. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 380–409. Springer (2017)
24. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–33. Springer (2018)
25. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM (JACM) **27**(2), 228–234 (1980)
26. Turpin, R., Coan, B.A.: Extending binary byzantine agreement to multivalued byzantine agreement. Information Processing Letters **18**(2), 73–76 (1984)

A Committee Selection Strawman Counter-Example

Figure 1 provides a concrete example of the issues in the strawman committee selection with $n = 100$, where the adversary controls only 42% of the online stake[5], yet some honest parties end up with a majority-corrupt final committee! Consider 5 general groups whose VRFs are among the highest: A (29 honest parties), B (29 honest parties), C_1 (42 corrupt parties), D (25 honest parties), and C_2 (17 corrupted parties). By not delivering corrupted messages from C_1 to B, parties in B take their temporary view to contain the next $0.42 \cdot 100$ parties - those in D and C_2 (similarly, this can occur for D). In step 2, parties in A will see 71 votes (from parties in B and C_1) for each party in C_2, and so will add parties in C_2 to their final committee V_A. Additionally, parties in B and C_1 will remain in A's final committee, since parties in A will see 100 votes for each (from parties in A,B,C_1), but members of D will not appear in V_A, since A only sees 25 votes for them (from parties in B). This leaves V_A containing 59 corrupted parties, but only 58 honest parties!

B Safety Parameters

Figure 2 shows the committee size required to ensure these properties hold with probability $1 - 5x10^{-9}$, as a function of the online fraction of stake which is controlled by honest users. If one block were to be added every 15 seconds, we would expect it to take 95 years for a safety violation to occur with this safety parameter. Of course, if more confidence is desired, the parameters can be easily adjusted. Our approach requires committees ranging from sizes of less than 200

[5] This attack can be carried out with as little as 33.4% online stake, but requires a more complicated setup which does not lend itself well to small images.

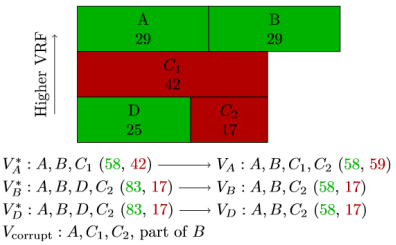

Fig. 1. Strawman with $n = 100$. All three honest groups (A,B,D) have a majority-honest temporary view, but A ends up with a majority-corrupt final view.

up to under 900 for 60-90% online stake being honest. In contrast, Algorand requires committees of size 2000 to 4000 for the same guarantees[6].

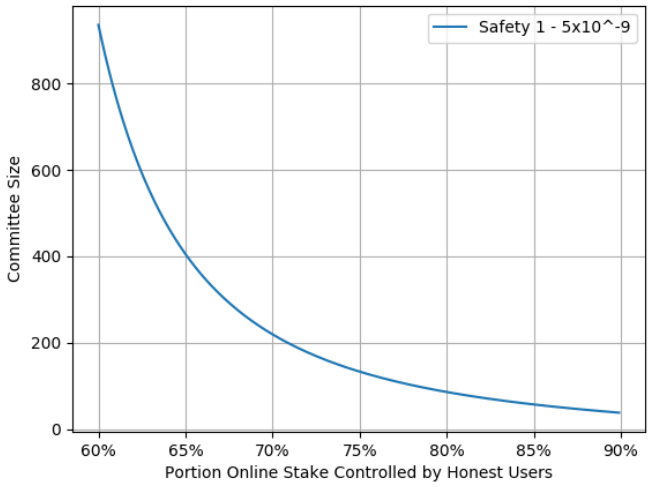

Fig. 2. Committee Size for Safety $1 - 5x10^{-9}$ vs Portion of Online Stake Controlled by Honest Users

[6] Algorand requires $> \frac{2}{3}$ fraction honest stake, but can work with $> \frac{1}{2}$ fraction honest stake with modifications. To be more favorable to Algorand in this comparison, we compare the committee size required for a particular *difference* in assumed fraction of honest stake from its lower bound.

C Proofs

Proof of Lemma 1

Proof. First, we will show property 1. If it were not true then there would exist a $j \in V_i$, for some honest i, such that $VRF(j,r)$ is not among the highest n VRFs in $\bigcup_{k \in H} V_k$. Since $V_i \subseteq V_i^*$, $j \in V_i^*$ and $VRF(j,r)$ is not among the highest n VRFs in $\bigcup_{k \in H} V_k^*$. Observe that all honest players see $VRF(j,r)$ for each $j \in \bigcup_{i \in H} V_i$ during step 2, since every honest player i broadcasts $VRF(j,r)$ for each $j \in V_i^*$. Therefore, $j \notin V_{U,k}$ for any honest k, since $VRF(j,r)$ is not even among the highest n VRFs in $\bigcup_{k \in H} V_k^*$. Thus, since i is an honest player, $j \notin V_i$, a contradiction.

For the second property, first consider the list of all VRFs of online parties for this round. Let H_C be the set of honest parties whose VRF appears in the highest n VRFs of this complete list. Observe that $H_C \subseteq V_i$ for every honest party i. If this were not the case, then $\exists j \in H_C$ such that $j \notin V_i$ for some honest party i, which can only occur if i either did not receive j's VRF, or if it saw n VRFs which were higher than j's. The former cannot occur in our model since all honest messages are delivered to all honest parties within Δ timesteps. The latter would violate the definition of H_C, since the adversary can only evaluate the VRFs of corrupt parties, which are always online.

It remains to argue that $|H_C| > \frac{n}{2}$ with high probability, which is equivalent to arguing that $n - |H_C| < \frac{n}{2}$ with high probability. Since we model VRFs as random oracles, this is precisely the problem of drawing n players from a bin containing $(\frac{1}{2} + \epsilon)N$ honest players and $(\frac{1}{2} - \epsilon)N$ corrupt players without replacement, and counting the number of *corrupt* players drawn. Since we consider N much larger than n, $n - |H_C|$'s distribution is well approximated by Binomial$(n, \frac{1}{2} - \epsilon)$. Let I_k be the indicator in the variable for this event that the k'th Bernoulli trial is successful (i.e. a corrupt player is drawn). A bound by Hoeffding [15] (see proposition 8 in the appendix for details) shows that

$$P\left(\frac{1}{n}\sum_{k=1}^{n} I_k \geq \frac{1}{2}\right) \leq \left(\left(\frac{\frac{1}{2}-\epsilon}{\frac{1}{2}-\epsilon+\epsilon}\right)^{\frac{1}{2}-\epsilon+\epsilon}\left(\frac{1-\left(\frac{1}{2}-\epsilon\right)}{1-\left(\frac{1}{2}-\epsilon+\epsilon\right)}\right)^{\frac{1}{2}-\epsilon+\epsilon}\right)^{n}$$
$$= \left(4\left(\frac{1}{4}-\epsilon^2\right)\right)^{n/2}$$

Since $n - |H_C|$'s distribution is well approximated by the distribution of $\sum_{k=1}^{n} I_k$, we have the result.

Proof of Proposition 2

Proposition 7. *If at all times, $|V_i| \leq n$ for honest i and there exists a set of honest players H_C of size $> \frac{n}{2}$ common to all honest views, then algorithm3 achieves the following properties:*

1. *If $D \in H_{C,1}$ and sends m_D, then i accepts m_D from D.*
2. *If honest parties i and j accept m_D, m'_D, respectively, from a party D, then $m_D = m'_D$.*

Proof. For the first property, observe that i does not receive $sig_k(m'_D)$ for $m'_D \neq m_D$ if D is honest, since D never signs $m'_D \neq m_D$, and the adversary cannot forge digital signatures. Furthermore, all honest parties receive $sig_D(m_D)$ during step 1, since $D \in H_{C,1}$. Then, in step 2, j receives $sig_k(m_D)$ from all members of $H_{C,2}$, which accounts for $> \frac{n}{2}$ members of $V_{j,2}$. Since $D \in H_{C,1} \subseteq V_{i,1}$, i therefore accepts m_D.

Assume the second property was false. Then i must have received $sig_k(m_D)$ from $\frac{n}{2} + 1$ members of $V_{i,2}$. Since $|V_{i,2}| \leq n$ and the size of its honest core $|H_{C,2}| \geq \frac{n}{2} + 1$, at least one member of $H_{C,2}$ sent $sig_D(m_D)$ to i. Therefore that same member also sent $sig_D(m_D)$ to j, and so j would not have accepted any other message m'_D.

Proof of Propostion 3

Proof. Let i and j be honest parties. Since proposition 2 shows that i and j never accept different messages from any party k, between them, i and j accept at most $|V_i \bigcup V_j|$ votes. By assumption, $|V_i \bigcup V_j| \leq n$. For i to follow substep (a) while j follows substep (b), i must have seen $\frac{n}{2} + 1$ votes for 0 while j saw $\frac{n}{2} + 1$ votes for 1, a total of $n + 2$ votes, which is more than they saw combined.

Proof of Proposition 5

Proof. Say some honest party i follows substep a, setting $v_i = 0$. By proposition 3, every other honest party j either follows substep a, setting $v_j = 0$ or c, setting $v_j = mc$. By proposition 1, with probability $\frac{1}{2}$, all honest parties set mc to be the same bit, which was input by an honest party, so is a uniform random bit. When this occurs, with probability $\frac{1}{2}$, that bit is 0, and all honest parties are in agreement on 0.

The proof is symmetric for the case where i follows substep b.

Proof of Proposition 6

Proof. For the first property, we wish to show that if $g_i = 2$ for some honest player i, then if $g_j \neq 2$ for some honest player j, $g_j = 1$. Since $g_i = 2$, i receives $\geq \frac{n}{2} + 1$ consistent sets of signatures in step 3. Recall that $|V_{i,3}| \leq n$, and that $|H_{C,3}| \geq \frac{n}{2} + 1$. Therefore, at least one member, ℓ, of $H_{C,3}$ sent i a consistent set of signatures, and so ℓ also sent j a set of $> \frac{n}{2}$ signatures. These signatures belong to distinct members of $V_{\ell,2}$, and in order for j to output $g_j = 1$, at least $|V_{j,2}| - \frac{n}{2} + 1$ of these must also be members of V_j. Symbolically, this is the case

if $\frac{n}{2}+1-|V_{\ell,2}\backslash V_{j,2}| \geq |V_{j,2}|-\frac{n}{2}+1$. The following arithmetic shows this is true.

$$\begin{aligned}\frac{n}{2}+1-|V_{\ell,2}\backslash V_{j,2}| &= \frac{n}{2}+1-(|V_{\ell,2}\cup V_{j,2}|-|V_{j,2}|)\\ &\geq \frac{n}{2}+1-n+|V_{j,2}|\\ &= |V_{j,2}|-\frac{n}{2}+1\end{aligned}$$

The proof of the second property will require two steps. First, we will observe that if no honest parties signs $(v,2)$ in step 2, then no honest party receives a consistent signature set for $(v,2)$ in step 4. This would require $|V_{2,i}|-\frac{n}{2}+1 > |V_{2,i}|-|H_C|$ total signatures from members of $V_{2,i}$, which is more than the number of corrupted parties in $V_{2,i}$. Accordingly, if this is the case, no honest party outputs v with a grade other than 0. Second, we show that in step 2, two honest parties cannot trigger their thresholds for two values v_1, v_2. So, if some honest party signs v_1 in step 2, no honest party signs any v_2 in step 2, and therefore no honest party can output anything other than v_1 with a grade other than 0. This part requires us to recall that algorithm 3 prevents any player from sending signatures for two different messages which are both accepted in step 1. Therefore, honest players i and j see at most $|V_i \bigcup V_j|$ signatures between them in step 2, too few to trigger both their thresholds for different values, since $|V_i \bigcup V_j| < n+2 = 2(\frac{n}{2}+1)$.

For the third property, observe that during step 1, every honest player i receives at least $\frac{n}{2}+1$ signatures for v from members of $H_{C,1}$, and so they all sign and send v during step 2. Therefore all honest parties have received at least $\frac{n}{2}+1$ signatures for v from members of $V_{i,2}$ during step 2, and so send a set of signatures for $(v,2)$ in step 3. Each honest party then receives a set of signatures from each member of $H_{C,3}$. As shown during the proof of property 1, any honest player will consider these sets to be consistent, so since $|H_{C,3}| > \frac{n}{2}$, every honest player i outputs $(g_i = 2, v_i = v)$.

Proof of Theorem 1

Proof. For the first property, first note that if the Byzantine Agreement assumptions hold, then with probability $> 1-\frac{3}{4}^k$, all honest players output the same bit from the execution of binary Byzantine Agreement (algorithm 4, lemma 2). By lemma 1, each committee satisfies the assumptions with probability $\approx 1-\left(1-4\epsilon^2\right)^{n/2}$. The committee formation algorithm is executed 2k times during the Byzantine Agreement protocol. Therefore by the union bound, all players output the same bit from the execution of binary Byzantine Agreement with probability at least $1-\left(\frac{3}{4}^k+2k\left(1-4\epsilon^2\right)^{n/2}\right)$, which is overwhelmingly close to 1 for appropriate choices of k and n.

If that bit was 0, then B_i is the empty block for all honest players i. Otherwise, some honest player i input $v_i =' 1$ to the Byzantine Agreement protocol, which only occurs if i output $g_i = 2$ from the Graded Consensus execution in

step 2. Therefore, every other honest player j outputs B'_i with grade > 0 as a result of the Graded Consensus execution (proposition 6), and $B_i = B'_i = B_j$. Since graded consensus uses committee selection 6 times, by union bound the requirements of proposition 6 are satisfied with probability $\geq 1 - 6\left(1 - 4\epsilon^2\right)^{n/2}$.

For the second property, observe that if the leader (i.e. the owner of the highest VRF) for a round is honest, then it proposes a valid non-empty block associated with that VRF, which every honest party inputs to Graded Consensus. Thus, that valid, non-empty block is subsequently output with grade 2 in the Graded Consensus execution (proposition 6), and then becomes the final output block for every honest party, since all honest parties input 1 to the Byzantine Agreement algorithm (lemma 2). By proposition 1, this occurs with probability $> \frac{1}{2}$.

D Performance Analysis

Confirmation Times We express times in terms of message propogation periods (multiples of Δ). Leader Election (figure 1) takes Δ time, as in Algorand. Committee Selection (figure 2) requires 2Δ. Graded Broadcast (figure 3) requires two committee selections, for a total of 4Δ. This is where the approximation discussed in the related work section comes from. Byzantine Agreement (figure 4) takes k rounds, each of which requires a Leader Election and a Graded Broadcast, for a total of $5k\Delta$. Graded Consensus (figure 5) requires 3 graded broadcasts, for a total of 12Δ. All together, adding the next block requires a Leader Election, a Graded Consensus, and a Byzantine Agreement, for a total of $(13+5k)\Delta$. If the block proposer was malicious, which occurs with probability $\leq \frac{1}{2}$, however, this block could be empty. Therefore the expected time to add the next block is bounded by $(26 + 10k)\Delta$.

Communication Complexity During Committee Selection, all-to-all communication is required to decide the committee, since the adversary may reorder honest messages such that the top n are also the last n received. This gives a quadratic communication complexity.

E Hoeffding's Bound

The following probability bound is due to Hoeffding [15].

Proposition 8. *If $X_1, \ldots X_n$ are independent and $0 \leq X_i \leq 1$ for $i = 1 \ldots n$, then for $0 < t < 1 - \mu$, where $\mu = E[\frac{1}{n}\sum_{i=1}^{n} X_i]$, the following holds:*

$$P\left(\frac{1}{2}\sum_{i=1}^{n} X_i - \mu \geq t\right) \leq \left(\left(\frac{\mu}{\mu + t}\right)^{\mu+t}\left(\frac{1-\mu}{1-\mu-t}\right)^{1-\mu-t}\right)^{n}$$