PiLi: An Extremely Simple Synchronous Blockchain*

T-H. Hubert Chan Rafael Pass Elaine Shi

October 12, 2018

1 Introduction

We describe PiLi, an extremely simple synchronous blockchain that tolerates minority corruptions. The protocol description is the extremely natural and intuitive. Informally, every epoch, an eligible proposer proposes a block (tagged with the current epoch) extending the freshest notarized chain observed so far. Nodes vote on all valid proposals from eligible proposers as long as 1) the proposed block extends from a parent chain has been notarized in the node's view; and 2) this parent is "not too stale". When a block gains votes from the majority of nodes, it is considered *notarized* but not necessarily *final*. If a node observes a notarized chain ending with 6 blocks of consecutive epochs and no other notarized blocks of these 6 epochs have been seen, then this notarized chain except the trailing 5 blocks are considered *final*.

2 Preliminaries

2.1 Execution Model

We consider a set of n consensus nodes numbered $1, 2, \ldots, n$. The nodes have access to a public-key infrastructure (PKI), and we use $(\mathsf{pk}_i, \mathsf{sk}_i)$ to denote node i's public- and secret-key pair where $i \in [n]$.

We adopt a standard *synchronous* model of protocol execution. Within a synchronous round, the following takes place. First, nodes receive messages at the beginning of each round; they then perform computation and process these messages; finally, they send messages. Any honest message sent in round r will be received by honest recipients at the beginning of round $r + 1$. Henceforth in the paper, "the end of round r" and "the beginning of round $r + 1$" are used interchangably.

In our protocol, all messages are *multicast* to everyone. We assume the following about the communication medium:

> If an honest node receives a message m at the beginning of round r, then all honest nodes will have observed m by the beginning of round $r + 1$.

Assuming that messages are signed by the sender, this can be achieved, e.g., by having all nodes relay every fresh message they observe.

A probabilistic polynomial-time adversary is allowed to corrupt up to f number of nodes. Henceforth in the paper, we assume that $n \geq 2f + 1$, i.e., the adversary can control any minority coalition. The corrupt nodes form a single Byzantine coalition that is under the adversary's control,

*PiLi is a cute way of saying "thunder" in Chinese; it also means fast, furious, and streamlined.

i.e., all corrupt nodes forward all received messages to the adversary and send messages based on the adversary's instructions. Unless otherwise stated, we will assume static corruption for simplicity; however, later in Section A we will comment on how to extend our protocol to achieve adaptive security.

2.2 Blockchain Protocol

A blockchain protocol is a special case of a state machine replication protocol [13, 14, 18]. In state machine replication, a distributed system of nodes would like to agree on an ever-growing, linearly-ordered log of transactions. Two properties are desired: *consistency*, i.e., it must be that all honest nodes' logs are prefixes of each other and honest nodes' logs never shrink; *liveness*, i.e., outstanding transactions get confirmed in a polynomial amount of time. In a blockchain-style, transactions will be confirmed in batches called "blocks"; and in this paper we consider blockchains where each block contains a hash that explicitly refers back to the parent block.

3 The PiLi Protocol

3.1 Definitions

Blocks and blockchains are defined in the most natural manner.

Block. Each block is of the format $(e, \mathsf{TXs}, h_{-1})$ where $e \in \mathbb{N}$ denotes an epoch number, $\mathsf{TXs} \in \{0,1\}^*$ denotes the block's payload (e.g., a batch of transactions to confirm), and $h_{-1} \in \{0,1\}^\kappa$ denotes the parent hash, i.e., hash of the blockchain the block extends from.

Valid blockchain. A valid blockchain denoted chain is a sequence of blocks where for $1 \le i \le |\mathsf{chain}|$, $\mathsf{chain}[i]$ denotes the i-th block in chain. We often use the following useful blockchain notation:

- $\mathsf{chain}[-1]$ denotes the last block in chain and $\mathsf{chain}[:i]$ denotes the first i blocks of chain;
- assume that a block of epoch e exists in chain, we would then use $\mathsf{chain}\langle e\rangle$ to denote this block at epoch e in chain; and use $\mathsf{chain}\langle :e\rangle$ to denote the prefix of chain ending at the block $\mathsf{chain}\langle e\rangle$.

For a blockchain chain to be valid, the following must be respected:

1. For all $1 \le i < j \le |\mathsf{chain}|$, $\mathsf{chain}[i].e < \mathsf{chain}[j].e$; and
2. It must be that $\mathsf{chain}[1].h_{-1} = \perp$, and for every $2 \le i \le |\mathsf{chain}|$, $\mathsf{chain}[i].h_{-1} = H(\mathsf{chain}[:i-1])$ where H is randomly sampled from a collision-resistant hash family[1].

In other words, in a valid blockchain, all blocks' sequence numbers must strictly increase (but sequences numbers may jump); and moreover, every block must correctly contain the parent chain's hash.

Epoch-e blockchain and "fresher than" relation. We say that chain is an epoch-e chain iff $\mathsf{chain}[-1].e = e$. A chain is said to be *fresher* than another chain' iff $\mathsf{chain}[-1].e > \mathsf{chain}'[-1].e$.

[1] In practice, H should be incrementally evaluated using a base hash function H_0: let $H(\mathsf{chain}) := H_0(H(\mathsf{chain}[:-1])||\mathsf{chain}[-1])$ if $|\mathsf{chain}| > 1$; and let $H(\mathsf{chain}) := H_0(\mathsf{chain}[1])$ if $|\mathsf{chain}| = 1$.

Vote and notarization. A pair (h, σ) from a purported sender $i \in [n]$ is said to be a valid vote for some chain, iff $h = H(\mathsf{chain})$ and moreover $\Sigma.\mathsf{Verify}_{\mathsf{pk}_i}(h, \sigma) = 1$.

A collection of $f + 1$ valid votes for chain from distinct senders is said to be a *notarization* for chain.

Remark 1. Note that since our blockchain validity definition requires that each block specify the parent hash, assuming no hash collision, a block may be considered an alias for a chain. Thus, we often use the term "a vote/notarization for chain" and the term "a vote/notarization for the block $\mathsf{B} := \mathsf{chain}[-1]$" interchangeably.

3.2 Protocol

The protocol proceeds in *epochs*. Each epoch contains exactly two synchronous rounds called **Propose** and **Vote** respectively.

Our PiLi blockchain protocol proceeds in a natural manner: every epoch, an eligible proposer proposes a next block extending its current freshest notarized chain. Consensus nodes vote on all valid proposals from eligible proposers, as long as the proposed block extends from a chain that is sufficiently fresh. When a block collects $f + 1$ votes, it is considered *notarized*. Thus if an epoch has a single honest proposer, only a unique block of this epoch can be notarized. Not all notarized blocks are considered final. If, however, at the beginning of epoch e the freshest notarized chain (denoted chain) ends with 6 blocks at epochs $e-6, e-5, \ldots, e-1$ respectively, and moreover there are no other notarized blocks at these epochs have been observed, then $\mathsf{chain}\langle : e-6\rangle$ is considered *final* (i.e., chop off the trailing 5 blocks from chain).

A formal description of the protocol is presented in Figure 1. In this figure, we assume that each node can efficiently check if a node i is *eligible* for proposing some block $\mathsf{B} := (e, \_, \_)$. We will discuss how to design the eligibility determination function later in Section 5. For the time-being, if concreteness helps, the reader may assume a simple round-robin policy: node $(e \mod n) + 1$ is the only eligible proposer for proposing any epoch-e block. Later in Section 5 we will discuss other proposer election policies such as random and sticky policies.

In Section 4, we prove the following:

- *Consistency.* The protocol satisfies consistency as long as the majority of the nodes are honest (and even when proposers are corrupt); and
- *Liveness under good proposers.* Suppose that for each of $C \geq 6$ consecutive epochs $e + 1, e + 2, \ldots, e + C$, there is a single honest proposer per epoch and no corrupt node is elected proposer, then by the end each of the epochs $e+6, e+7, \ldots, e+C$, any honest node's finalized chain would grow one honest block per epoch. More intuitively, whenever sufficiently many consecutive epochs each has a single honest proposer and no corrupt proposer, then after 6 epochs of wait-time, afterwards, every epoch will have a new block finalized for every honest node.

4 Proofs

Good executions. Henceforth we ignore the negligible fraction of bad executions where honest nodes' signatures are forged or where a hash collision is found. For simplicity, in all theorems and lemmas below, we omit stating "except with negligible probability" — however, the reader should keep in mind that all theorems and lemmas below hold only for "good executions" where honest nodes' signatures are not forged and hash collisions do not exist in the union of honest nodes' views.

The PiLi blockchain protocol

In every epoch $e = 1, 2, \ldots$, every node performs the following. Henceforth, we assume that every message is tagged with the (purported) sender's identity.

1. **Propose.** Let chain be the freshest notarized chain observed so far. Let TXs be a set of outstanding transactions. and let $\mathsf{B} := (e, \mathsf{TXs}, H(\mathsf{chain}))$. If the node is an *eligible* proposer for proposing B, make a proposal as follows (we discuss how to determine proposer eligibility in Section 5):
   - sign $H(\mathsf{chain}||\mathsf{B})$, and let σ be the resulting signature;
   - multicast the proposal (B, σ).
2. **Vote.** If a node has received a proposal of the form $\mathsf{B} := (e, \mathsf{TXs}, h_{-1})$ with a valid signature from some node i that is an eligible proposer for proposer B, vote on B iff the following conditions hold:
   - the node has observed some chain such that $h_{-1} = H(\mathsf{chain})$ as well as a notarization for chain; and
   - unless $e = 1$, the following condition must hold: let chain' be the freshest notarized chain the node had observed at the beginning of the *previous* epoch $e - 1$, it must be that chain (whose hash matches h_{-1}) is at least as fresh as chain'.

   To vote on $\mathsf{B} := (e, \mathsf{TXs}, h_{-1})$, let chain be the blockchain matching h_{-1}: sign $h := H(\mathsf{chain}||\mathsf{B})$ to obtain the signature σ, and multicast (h, σ).

   A node is allowed to vote on multiple proposals if they all satisfy the above conditions.

Finalize. At any time, let chain be the freshest notarized chain ending with six blocks at consecutive epochs — and let $e, e+1, \ldots, e+5$ be these six epochs. If for every $e' \in \{e, e+1, \ldots, e+5\}$ the only epoch-e' notarized block observed so far belongs to chain, then output $\mathsf{chain}\langle : e\rangle$.

Figure 1: The PiLi blockchain.

Additional terminology. We introduce some additional useful terminology:

- We say that chain is *a notarized chain in honest view* in some execution if there exists some round in which some so-far honest node has observed chain and a notarization for it. Note that if chain is a notarized chain in honest view in some execution, then every prefix of chain is a notarized chain in honest view too.
- For convenience, we pretend that there is an imaginary genesis block denoted $\mathsf{chain}[0] := (0, \bot, \bot)$ at index 0 in every valid blockchain chain.
- $\mathsf{chain} \preceq \mathsf{chain}'$ means chain is a prefix of chain'; by convention, $\mathsf{chain} \preceq \mathsf{chain}$.

4.1 Consistency

Lemma 1. *Consider some good execution: suppose* chain *is some valid notarized chain in honest view and there are two consecutive blocks in* chain *at epochs* e *and* $e' > e$ *respectively. Let* $\mathsf{chain}\langle : e\rangle$ *denote the prefix of* chain *ending at the epoch-*e *block. It must be that* all *honest nodes have observed a notarization for* $\mathsf{chain}\langle : e\rangle$ *by the beginning of epoch* $e' + 1$*.*

Proof. Suppose for the sake of contradiction that some honest node has not observed a notarization for $\mathsf{chain}\langle : e\rangle$ by the beginning of epoch $e' + 1$. This means that no honest node has observed a notarization for $\mathsf{chain}\langle : e\rangle$ by the beginning of the **Vote** round of epoch e' (since otherwise all honest nodes would have observed it by the beginning of epoch $e' + 1$). This means that no honest node will vote on $\mathsf{chain}\langle : e\rangle$ in epoch e', and thus $\mathsf{chain}\langle : e\rangle$ cannot be followed by a notarized block at epoch e' in any valid chain in honest view — this contradicts the assumption of this lemma. □

We define the good event $\mathcal{G}(e)$ to be the following: some honest node has observed a notarized chain that contains both an epoch-$(e-1)$ block and an epoch-e block.

Lemma 2. *Consider some good execution such that for some* e*,* $\mathcal{G}(e)$ *is true. Then, for every notarized* chain *in honest view, if for some* i $\mathsf{chain}[i].e \geq e + 2$*, it holds that* $\mathsf{chain}[i-1].e \geq e - 1$*.*

Proof. By Lemma 1, if $\mathcal{G}(e)$ is true, then all honest nodes will have observed an epoch-$(e-1)$ notarized block by the beginning of epoch $e+1$. Thus, by our honest protocol definition, in epochs $e+2$ or higher, no honest node will vote to extend any chain that ends at epoch $e-2$ or smaller. Thus, any notarized block at epoch $e+2$ or higher cannot extend from a block at epoch $e-2$ or smaller. □

Lemma 3. *Consider some good execution in which* $\mathcal{G}(e+1)$ *is true: then for any valid* chain *ever in honest view such that* $\mathsf{chain}[-1].e > e + 2$*, it must be that* chain *contains at least one block at epoch* e*,* $e+1$*, or* $e+2$*, i.e., it cannot be that* chain *skips all three epochs* e*,* $e+1$*, and* $e+2$*.*

Proof. For the sake of contradiction, suppose that there is some valid chain in honest view such that $\mathsf{chain}[-1].e > e + 2$ and moreover chain skips epochs e, $e+1$, and $e+2$. Consider the earliest (i.e., least fresh) block in chain whose epoch is greater than $e+2$ and let $\mathsf{chain}[i]$ be this block. It must be that $\mathsf{chain}[i-1].e < e$ but this violates Lemma 2. □

Theorem 1 (Consistency)**.** *Consider some good execution: suppose that* ch *is output by honest node* i *at some time* t *and* ch' *is output by honest node* j *at some time* t' *(where the nodes* i *and* j *and the times* t *and* t' *need not be distinct). Then, it holds that* $\mathsf{ch} \preceq \mathsf{ch}'$ *or* $\mathsf{ch}' \preceq \mathsf{ch}$*.*

Proof. Let chain be the freshest notarized chain of node i at time t, and let chain' be the freshest notarized chain of node j at time t'. Further, suppose that chain and chain' ends at epochs e and e' respectively. Without loss of generality, assume $e' \geq e$. By definition, $\mathsf{ch} := \mathsf{chain}\langle : e-6\rangle$ and $\mathsf{ch}' := \mathsf{chain}'\langle : e'-6\rangle$. It suffices to show that $\mathsf{ch} \preceq \mathsf{chain}'$. Suppose this is not true for the sake of reaching a contradiction. It must hold that ch and chain' diverge before the block $\mathsf{ch}\langle e-5\rangle$.

We next show that chain' has some block with epoch in $\{e-2, e-1, e\}$. If $e' = e$, then this is satisfied. Otherwise, we apply Lemma 3. Since the truth of $\mathcal{G}(e-1)$ is witnessed by chain, and the last block of chain' is of epoch $e' \geq e+1$, it follows that chain' must contain some block with epoch in $\{e-2, e-1, e\}$, as required.

Let $\mathsf{chain}'\langle e^*\rangle$ be the earliest block in chain' where $e^* \in \{e-2, e-1, e\}$. We apply Lemma 2. Since the truth of $\mathcal{G}(e-4)$ is witnessed by chain, and chain' contains a block with epoch $e^* \geq e-2$, the parent block B of $\mathsf{chain}'\langle e^*\rangle$ has epoch at least $e-5$.

By our assumption that ch and chain' diverge before the block $\mathsf{ch}\langle e-5\rangle$, B is not in ch. Now, by Lemma 1, by the end of epoch $e^* \leq e$ (which is the beginning of epoch e^*+1), all honest nodes would have observed a notarization for B. Note that no honest node can have seen any epoch-e block notarized before the end of epoch e. Thus, t must be at least the end of epoch e or larger. However, since node i output $\mathsf{chain}\langle : e-5\rangle$ at time t, by the definition of the honest finalization procedure (see Figure 1), it must be that node i has not seen a notorization for B at time t. We have thus reached a contradiction.

□

4.2 Liveness

Theorem 2 (Liveness)**.** *Consider some good execution: suppose that in each of 6 consecutive epochs $e, e+1, e+2, \ldots, e+5$, a single honest node is elected proposer and and no corrupt node is elected proposer. Then, at the beginning of epoch $e+6$, all honest nodes must output a finalized chain ending at epoch e.*

Note that if the above theorem holds, at the end of epoch $e+6$, honest nodes' finalized chain ends at an epoch-e block proposed by an honest node, and this honest node must include all outstanding transactions into this block.

Proof. If some epoch e, a single honest is elected proposer and no corrupt node is proposer, then a *unique* epoch-e block will be proposed in the **Propose** round of this epoch. To prove the above theorem, it suffices to argue the following:

1. *Progress:* all honest nodes will have oberved a notarization for the proposed epoch-e block by the end of epoch e;
2. *Uniqueness:* there cannot be two different epoch-e blocks notarized in honest view.

Uniqueness is obvious since the single honest proposer of epoch e makes a unique proposal and thus honest nodes will not vote for any other epoch-e block other than the proposed. We now argue about progress. If $e > 1$ then for any notarized chain ch seen by any honest node at the beginning of the **Vote** round of epoch $e-1$, all honest nodes must have seen ch by the beginning of epoch e; thus an honest proposer of epoch e must propose to extend some chain that is at least as fresh as ch. Thus all honest nodes will vote on the proposal by this honest proposer in epoch e, and all honest nodes will have observed a notarization for the proposal by the end of epoch e.

Applying the same argument to epoch $e+i$, for $i \in \{1, \ldots, 5\}$, we can conclude that by the beginning of epoch $e+6$, all honest nodes must have observed the notarized chain containing the 6

blocks proposed by the unique honest node from epochs e to $e+5$, and observed no other notarized blocks in these epochs.

Therefore, according to the finalization procedure specified in Figure 1, by the beginning of epoch $e+6$, all honest nodes must output the same finalized chain ending at epoch e. □

5 Proposer Eligibility Determination

As we mentioned earlier (and proven later in Theorem 2), honest nodes' finalized chains grow whenever there are 6 consecutive epochs each with a single honest proposer and no corrupt proposer. We now discuss how to determine proposer eligibility. Specifically, we give a few example policies and discuss their implications on the transaction confirmation time. In this section for simplicity we assume static corruption. Tolerating adaptive corruption will be discussed later in Section A.

5.1 The Democracy-Favoring Approach

In the democracy-favoring approach (exemplified in earlier systems such as Bitcoin [12], Algorand [7], Tendermint [10] and Dfinity [9]), we would like to give everyone an opportunity to be the proposer:

1. *Round-robin.* In a simple round-robin strategy, in epoch i, node $(i \mod n)+1$ is the proposer. This strategy has the drawback that if the adversary corrupts $\Theta(n)$ consecutive nodes, there can be no liveness for $\Theta(n)$ epochs, i.e., the transaction confirmation time can be linear in n in the worst case.

2. *Random.* Let H^* denote a random oracle that is chosen after the adversary makes corruption choices[2]. The random strategy elects a random proposer every epoch. Specifically, we can elect node $(H^*(e) \mod n)+1$ to be an eligible proposer. Under static corruption, there is $\Theta(1)$ probability that the next 6 epochs all have honest proposers. Thus transaction confirmation takes expected $O(1)$ rounds.

   We can also modify this idea to improve the confirmation time by a constant factor under static corruption: once elected, each proposer is responsible for proposing at least $\mu \geq 6$ blocks. More specifically, the proposer for epoch e is evaluated as $(H^*(\mathsf{rnd}_\mu(e)) \mod n) + 1$ where $\mathsf{rnd}_\mu(e)$ means round e down to the nearest multiple of μ.

5.2 The Stability-Favoring Approach

The stability-favoring approach advocates an almost opposite philosophy that gives emphasis to stability and performance. The idea is to try to stick to the same proposer as long as it is performing its job well. However, when the proposer crashes or behaves maliciously such that progress is hampered, nodes invoke proposer re-election. Earlier systems such as PBFT [5] and Thunderella [16] adopt this approach.

We suggest a few modifications to our basic PiLi protocol to implement the stability-favoring policy, i.e., re-elect proposer only when the old proposer stops working.

[2] As pointed out in the earlier work [15], we can remove the random oracle with a common reference string (CRS) that is chosen after the adversary makes corruption choices, and instead use a $(\mathsf{PRF}_{\mathrm{crs}}(e) \mod n) + 1$ to determine the eligible proposer for epoch e.

Normal blocks and timeout blocks. In a blockchain, we allow two types of blocks called *normal* blocks and *timeout* blocks respectively. A normal block's epoch must increment the parent block's epoch number; whereas a timeout block's epoch number must be sufficiently far apart from the parent block's epoch number and moreover a timeout block's epoch number must be a multiple of an integer $T > 6$. More formally, a blockchain chain is said to be valid iff all the blockchain validity conditions defined in Section 3.1 hold, and moreover for each $1 \leq i \leq |\mathsf{chain}|$:

- either $\mathsf{chain}[i].e = \mathsf{chain}[i-1].e + 1$ — in this case $\mathsf{chain}[i]$ is said to be a normal block; or
- $\mathsf{chain}[i].e > \mathsf{chain}[i-1].e + T$ and moreover $\mathsf{chain}[i].e$ is a multiple of T — in this case $\mathsf{chain}[i]$ is said to be a timeout block.

For convenience, we may assume that for any valid blockchain, there is an imaginary genesis block denoted $\mathsf{chain}[0] := (0, \bot, \bot)$ associated with epoch 0, and this imaginary genesis block is henceforth considered a timeout block.

Slight modification to proposer algorithm. We make a small modification to the earlier proposer algorithm. Let e be the current epoch. Before proposing a block $\mathsf{B} := (e, \mathsf{TXs}, H(\mathsf{chain}))$ extending the current freshest notarized chain chain, a node first checks if $\mathsf{chain}||\mathsf{B}$ actually forms a valid blockchain (i.e., check that B is a either a valid normal block or timeout block extending chain). If not, the node does nothing in the **Propose** round of epoch e.

Sticky eligibility determination. Let $\mathsf{B} := (e, \mathsf{TXs}, h_{-1})$ and let chain be a valid blockchain in a node's view such that $H(\mathsf{chain}) = h_{-1}$;

- if B is a normal block extending from chain: then i is considered an eligible proposer for B iff i is an eligible proposer for $\mathsf{chain}[-1]$;
- if B is a timeout block extending from chain: then i is considered an eligible proposer for B iff $i = H^*(e) \mod n + 1$.

In Appendix B, we argue why this stability-favoring variant preserves our consistency and liveness proofs.

6 Related Work

PiLi is an extremely simple instantiaion of the "pipelined-BFT" paradigm [4, 6, 19] in the synchronous model. In classical synchronous and partially synchronous consensus, to confirm every batch of transactions, nodes must vote perform two or more rounds of voting. For example, in PBFT, the voting rounds are commonly referred to as "prepare" and "commit" (and for many synchronous protocols [11,17], a common pattern occurs). The idea of pipelined-BFT is the following: why don't we piggyback the present block's commit-round on the next block's notarization? This beautiful idea was implicitly described in the elegant Casper-FFG work by Buterin and Griffith [19] in the form of a "finality gadget" for a proof-of-work blockchain. Forums and blog-posts online [2,3]indicate that DPoS (i.e., the consensus protocol adopted by EOS, a top-5 cryptocurrency by market cap [1]) might be considering a similar approach, although no formal description of the DPoS protocol and proofs have been released at the time of the writing. A beautiful work called HotStuff by Abhraham et al. [17], and the Pala consensus protocol by Chan, Pass, and Shi [6] illustrated how to instantiate this idea in a partially synchronous network. All of these instantiations mentioned above tolerate less than $\frac{1}{3}$ corruptions (which is optimal for partial synchrony). In comparison, PiLi works in a synchronous model and tolerates upto minority corruptions.

Acknowledgments

We acknowledge helpful technical discussions with Yue Guo, Alexis Gauba, Aparna Krishnan, and Zubin Koticha. We are also grateful to Robbert van Renesse and Lorenzo Alvisi for being supportive.

References

[1] http://coinmarketcap.com/.

[2] Dpos consensus algorithm - the missing white paper. https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper.

[3] Fix dpos loss of consensus due to conflicting last irreversible block #2718. https://github.com/EOSIO/eos/issues/2718.

[4] Ittai Abraham, Guy Gueta, and Dahlia Malkhi. Hot-stuff the linear, optimal-resilience, one-message BFT devil. *CoRR*, abs/1803.05069, 2018.

[5] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, 1999.

[6] T-H. Hubert Chan, Rafael Pass, and Elaine Shi. Pala: A simple partially synchronous blockchain. Manuscript, 2018.

[7] Jing Chen and Silvio Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

[8] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Cryptology ePrint Archive, Report 2017/573, 2017. http://eprint.iacr.org/2017/573.

[9] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series: Consensus system. https://dfinity.org/tech.

[10] Jae Kwon. Tendermint: Consensus without mining. http://tendermint.com/docs/tendermint.pdf, 2014.

[11] Silvio Micali and Vinod Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. https://dspace.mit.edu/bitstream/handle/1721.1/107927/MIT-CSAIL-TR-2017-004.pdf?sequence=1, 2017.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[13] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.

[14] Rafael Pass and Elaine Shi. Rethinking large-scale consensus. In *CSF*, 2017.

[15] Rafael Pass and Elaine Shi. The sleepy model of consensus. In *Asiacrypt*, 2017.

[16] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Eurocrypt*, 2018.

[17] Ling Ren, Kartik Nayak, Ittai Abraham, and Srinivas Devadas. Practical synchronous byzantine consensus. Cryptology ePrint Archive, Report 2017/307, 2017. http://eprint.iacr.org/2017/307.

[18] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990.

[19] Virgil Griffith Vitalik Buterin. Casper the friendly finality gadget. https://arxiv.org/abs/1710.09437.

A Extensions

So far in the paper, we considered the static corruption model where the adversary announces the corrupt nodes upfront.

For the stability-favoring policy, since an honest proposer is responsible for proposing multiple blocks in a row, an adaptive adversary can always corrupt the proposer, and it can continue doing so for $\Theta(n)$ proposers. Therefore, in the worse case, liveness may require $\Theta(n)$ minutes even during periods of synchrony if consecutive $\Theta(n)$ proposers are adaptively corrupt.

For the democracy-favoring random leader election policy, we can extend our guarantees to adaptive security while preserving expected $O(1)$ transaction confirmation time, using ideas from several recent works [6–9, 15]. Instead of adopting a hash function H to determine eligibility as a proposer, we rely on a Verifiable Random Function (VRF) instead. Further, a proposer would use a *forward-secure* signing scheme to sign the proposal and evolve the signing key (erasing the old key in the process) immediately after the proposal is sent out.

We refer the reader to the appendices of the recent Pala work for more details [6] – the exact approach described there would work in our case too.

B Analysis of the Stability-Favoring Variant

In this section, we analyze the security of the stability-favoring variant described in Section 5.2. In this variant, proposer eligibility depends not only on the epoch number, but also the chain the proposed block extends from. Thus, it may appear that depending on which parent chain is chosen, the eligible proposer for proposing the next block may not be unique (even for the same epoch). It is not difficult to see that this does not affect our consistency proof (see Section 4.1) since having multiple proposers can be thought of as having a corrupt proposer; further, our consistency proof does not depend on whether the proposers are honest or corrupt. If the proposers are corrupt the protocol may get stuck but consistency cannot be broken. Besides the ambiguity in proposer definition, another change in the stability-favoring variant is that we have a couple new constraints on blockchain validity (due to normal and timeout blocks) — but it is easy to verify that our consistency proof is not affected by the new blockchain validity constraints either.

Thus it suffices to argue that the variant in Section 5.2 preserves liveness. Note that in every epoch $e \in \mathbb{N}$ that is a multiple of T, if an honest node i votes on a normal block of epoch e, then i must observe a notarized chain ending at epochs $e - T, \ldots, e - 1$ due to the validity requirement for normal and timeout blocks.

We thus conclude that for every epoch $e > T$ that is a multiple of T, either honest nodes' finalized chains have grown since T epochs ago, or else only a timeout block of epoch e can possibly be considered as a valid proposal by honest nodes. In the latter case by the honest algorithm the

proposer will be re-elected based on the epoch number e; and if the new proposer is honest then the protocol will make progress from that point on due to Theorem 2.