Rethinking Large-Scale Consensus*

Rafael Pass
rafael@cs.cornell.edu

Elaine Shi
runting@gmail.com

November 12, 2020

Abstract

In this position paper, we initiate a systematic treatment of reaching consensus in a *permissionless* network. We prove several simple but hopefully insightful lower bounds that demonstrate exactly why reaching consensus in a *permissionless* setting is fundamentally more difficult than the classical, *permissioned* setting. We then present a simplified proof of Nakamoto's blockchain which we recommend for pedagogical purposes. Finally, we survey recent results including how to avoid well-known painpoints in permissionless consensus, and how to apply core ideas behind blockchains to solve consensus in the classical, permissioned setting and meanwhile achieve new properties that are not attained by classical approaches.

1 Introduction

Distributed systems have been historically analyzed in a *closed*, so-called *permissioned*, setting in which both the number of participants in the system, as well as their identities, are common knowledge, and communication among the participants take place over authenticated channels.

A departure from this model started with the design of *peer-to-peer* systems, such as e.g. *Napster* and *Gnutella* for file sharing. The success of those systems led to systems such as CAN [RFH+01], Chord [SMK+01], and Pastry [RD01] which offered redundant file storage, distributed hashing, selection of nearby servers, and hierarchical naming. A novel aspect of these peer-to-peer systems is that they are *permissionless*: anyone can join (or leave) the protocol execution (without getting permission from a centralized or distributed authority), and authentication mechanisms are not available. Additionally, participants may join and leave the system at will.

The permissionless setting. The permissionless setting differs from permissioned in the following ways:

- **Open participation and unauthenticated communication:** Anyone can join the protocol execution (without getting permission from a centralized or distributed authority), and communication among the parties is unauthenticated.
- **Late spawning:** Protocol participants can arbitrarily drop off, and new participants may arbitrarily join, and we expect the security properties to still hold for newly joined/spawned nodes.
- **Uncertainty in the number of players:** The protocol participants may be uncertain about the exact number of players that currently are participating in the protocol.

*Invited position paper at CSF, 2017.

The above-mentioned systems, however, provide no guarantee that one participant's experience with the system is *consistent* with another's: Two participants requesting the same file may end up receiving different versions *and never know that they did*. Resolving this issue requires using a *consensus* protocol.

Consensus. Consensus protocols are at the core of distributed computing. We here focus on consensus protocols for realizing a "linearly ordered log" abstraction, often referred to as *state machine replication* in the distributed systems literature. Roughly speaking, participants maintain a growing ordered log of transactions and any participant can add transactions to the end of the log. A bit more precisely, in a consensus protocol, each participant maintains is own log of transaction, and we require the protocol to satisfy two important properties, namely, *consistency* and *liveness*:

- **Consistency:** at any point in the execution, all honest participants have consistent logs—that is, either their logs are identical, or one participant's log is a prefix of the other's.
- **Liveness**: any honest protocol participant can propose to add a transaction; this transaction is then guaranteed to get incorporated into their logs within some fixed (small) amount of time; additionally, whenever a participant sees some transaction in their log, the same transaction will appear in every other participant's log within some fixed (small) amount of time.

In essence, these properties mean that from the view of the participants, they are essentially communicating with a trusted third party that maintains a *global*, *ordered*, and *immutable* ledger/log of transactions that anyone can simply add to (but not remove from), and the log that I see has either already been seen by others, or will shortly be seen by everyone.

At first sight one may think that we can simply use standard, off-the-shelf protocols (e.g., [CL99, Lam06]) to resolve permissionless consensus. The problem, however, is that traditional consensus protocols were designed only to work in a permissioned setting—in fact, it is easy to see that they completely break down in the permissionless setting. The key challenge with the permissionless setting is that an attacker can trivially mount a so-called "sybil attack"—it simply spawns lots of players (that it controls) and can thus easily ensure that it controls a majority of all the players.

Permissionless consensus: Nakamoto's blockchain. In 2008, Nakamoto [Nak08] proposed his celebrated "blockchain protocol" which claims to achieve consensus in a permissionless setting. An amazing aspect is that not only does Nakamoto's blockchain implement a "global ledger" (from which we can only add, but not remove, transactions), but this ledger is also *public*—since *anyone* can join the protocol!

Indeed, the first application of a blockchain is the Bitcoin digital currency [Nak08, BBSU12] which relies on the consistency property to prevent fraud and double-spending attacks. A number of follow-up digital currencies [Lit,Woo14], micro-payment schemes [Pas15,lig], time-stamping [btP], naming [Loi14], fair secure computation [BK14] and secure messaging and PKI applications [CF14] are based on blockchains as well. Additionally, financial firms have announced intentions of using the blockchain to lower transaction costs, remove geopolitical barriers to transferring assets, and reconcile differences between systems.

To overcome the aforementioned sybil attack, Nakamoto's blockchain protocol relies on "computational puzzles"—a.k.a. *moderately hard functions* or *proofs of work*—put forth by Dwork and Naor [DN92]: roughly speaking, the participants are required to solve the computational puzzle of some well-defined difficulty in order to confirm transactions—this is referred to as *mining*. Next, rather than attempting to provide security whenever the majority of the participants are honest

(since participants can be easily spawned in the permissionless setting), Nakamoto's goal was to provide security under the assumption that a *majority of the computing power* is held by honest participants.

Recent works by Garay et al. [GKL15] and Pass et al. [PSS17] formally proved that Nakamoto's blockchain satisfies the above-mentioned consistency and liveness properties under different network assumptions, as long as the puzzle difficulty (referred to as the *mining hardness*) is appropriately set as a function of the maximum delay in the network, and that the attacker controls a minority of the computing power in the network.

The price of permissionless consensus? Thus, permissionless consensus is possible, but comes at a price: in Nakamoto's protocol, we need to use proofs-of-work (and thus "waste computation"), the protocol needs to be properly parametrized by an a-priori upper bound on the network's delay, and transactions get confirmed slower than this network delay upper-bound (even if the actual delay may be smaller). Additionally, we need to assume that a majority of the participants (technically, the majority of the computing power) is under honest control.

In contrast, in the "classic" (permissioned) setting, assuming the existence of a public-key infrastructure, all of these issues can be overcome:

- Consensus tolerating any number of corrupt players can be achieved if the protocol can be parametrized with an upper bound on the network's delay [DS83].
- Consensus is possible in a fully asynchronous model of computation (where the message delay may be arbitrary, and confirmation times are only a function of the actual message delay), assuming that more than 2/3 of the players are honest [CL99, DLS88].

In this work, we systematically analyze consensus in the permissionless setting, attempting to answer the following question:

> *Is consensus in the permissionless setting inherently mode "difficult" than consensus in the permissioned setting, and if so, what properties of the permissionless model introduce these difficulties?*

We show that all the "deficiencies" Nakamoto's protocol turn out to be necessary.

- (Thm 1: Proofs-of-work are needed without authentication) Even without late spawning, even if all players know the exact number of protocol participants, and even if all messages get delivered in the next time step (i.e., no network delays), consensus is impossible without using proofs-of-work (unless communication is authenticated).
- (Thm 2: Proofs-of-work must be performed infinitely often in the presence of late spawning.) If we also want to support late spawning, we can never stop using proofs-of-work—the protocol needs to continue using proofs-of-work indefinitely—even if all players know the exact number of protocol participants, and even if all messages get delivered in the next time step.
- (Thm 3: Honest-majority of computing power is needed in the presence of late spawning.) Even if all players know the exact number of protocol participants, and even if all messages get delivered in the next time step, if nodes can spawn late, then honest majority (in terms of compute-power) is necessary for achieving consensus (even with proofs-of-work).
- (Thm 4: Protocol must know an upper bound of the network's delay if uncertain of the number of players.) Even without late spawning, the consensus protocol needs to be parametrized by

the maximum message transmission delay if how many players will show up is uncertain by a factor of 2.

We present the proof of all these relatively simple (but hopefully insightful) impossibility results. We additionally present a somewhat simplified proof of the analysis of Nakamoto's blockchain from Pass et al. [PSS17] (with slightly worse parameters) which we believe is useful for didactic purposes. Finally, we point out some widely criticized painpoints related to Nakamoto's blockchain (e.g., selfish mining, slow confirmation, waste of energy), and survey several recent works that aim to address these painpoints. We also discuss what lessons we can learn from blockchain-style consensus for the classical permissioned setting.

2 Preliminaries and Definitions

2.1 Modeling a Permissionless Network

A permissionless network is distinct from classical models considered in the distributed systems and cryptography literature in the following respects:

1. nodes can join and leave the protocol freely at any time, and participation is open, i.e., there is no access control mechanism that decides who can join and who cannot;
2. nodes are not aware of other protocol participants a-priori, and the network delivery mechanism does not provide sender authentication, i.e., there is no authenticated channels;
3. the protocol may not even be aware of the exact the number of nodes participating in the protocol; and
4. more generally, the number of nodes may vary over time.

In our paper, we will first focus on the former three properties (as mentioned above) for the permissionless model. In particular, we will show how each property of the permissionless model leads to several simple but interesting lower bounds (Section 3) — these lower bounds tell us that consensus and protocol design in the permissionless model is fundamentally different than in the classical, permissioned model. We will also prove how Nakamoto's blockchain protocol realizes consensus in a permissionless model satisfying the first 3 properties (Section 5) by leveraging a proof-of-work random oracle. Finally, in Section 6, we will survey recent results in this space and in particular mention results pertaining to the forth property (i.e., varying number of nodes over time).

In this section, we first formalize such a permissionless execution model which will set up a formal context for our discussions below.

Round-based protocol execution. A protocol refers to an algorithm for a set of interactive Turing Machines (also called nodes) to interact with each other. The execution of a protocol Π that is directed by an environment $\mathcal{Z}(1^\kappa)$ (where κ is a security parameter), which activates a number of nodes as either *honest* or *corrupt* nodes. Honest nodes faithfully follow the protocol's prescription, whereas corrupt nodes are controlled by an adversary $\mathcal{A}$ which reads all their inputs/message and sets their outputs/messages to be sent.

A protocol's execution proceeds in *rounds* that model atomic time steps. Henceforth we use the terms *round* and *time* interchangeably. At the beginning of every round, honest nodes receive inputs

from an environment $\mathcal{Z}$; at the end of every round, honest nodes send outputs to the environment $\mathcal{Z}$.

The environment $\mathcal{Z}$ is a terminology often used in the cryptography literature — one can regard the environment $\mathcal{Z}$ a catch-all term that encompasses everything that lives outside the "box" defined by the protocol. For example, as mentioned later, part of the environment $\mathcal{Z}$'s job is to provide inputs to honest nodes and receive outputs from them. This models the fact that the inputs to the protocol may originate from external applications and the protocol's outputs can be consumed by external applications where any external application or other protocols running in the system are viewed as part of $\mathcal{Z}$.

Spawning, corrupting, and killing nodes. We now describe how the environment can spawn, corrupt, and kill nodes during the execution.

- The environment $\mathcal{Z}$ can *spawn* new nodes that are either honest or corrupt any time during the protocol's execution.
- At any point, $\mathcal{Z}$ can *corrupt* an honest node j which means that $\mathcal{A}$ gets access to its local state and subsequently, $\mathcal{A}$ controls node j.
- At any point, $\mathcal{Z}$ can *kill* either an honest or a corrupt node — at this moment, the node is removed from protocol execution. The adversary $\mathcal{A}$ does not know the state of honest nodes prior to being killed, but it knows the state of corrupt nodes prior to being killed.

Henceforth in the paper, unless otherwise noted, an honest node refers to one that has been spawned to be honest, and has not been corrupted or killed; a corrupt node refers to a node that has been spawned to be corrupt, or one that was spawned to be honest but has been corrupt since, but has not been killed.

Communication model. We assume that honest nodes can broadcast messages to each other. $\mathcal{A}$ is responsible for delivering all messages sent by honest nodes to *all* other nodes. $\mathcal{A}$ cannot modify the content of messages broadcast by honest players, *but it may delay or reorder the delivery of a message* as long as it eventually delivers all messages sent by honest nodes. Henceforth in this paper, we assume that each message sent by an honest node takes at least 1 round to deliver, but the adversary can possibly delay messages for longer — later, we shall consider restrictions on the delivery time. The identity of the sender is not known to the recipient. The adversary $\mathcal{A}$ can send messages to any subset of honest nodes.

2.2 Notations

We introduce some useful notations.

Conventions. Unless otherwise noted, all variables are functions of the security parameter κ; if a variable const is independent of κ, we will explicitly note it as a constant. When we say that $\mathsf{var}_1 \geq \mathsf{var}_2$ (or $\mathsf{var}_1 \geq 0$), we mean that for every $\kappa \in \mathbb{N}$, $\mathsf{var}_1(\kappa) \geq \mathsf{var}_2(\kappa)$ (or $\mathsf{var}_1 \geq 0$). A function $\mathsf{negl}(\cdot)$ is said to be *negligible* if for every polynomial function $p(\cdot)$, there exists some κ_0 such that $\mathsf{negl}(\kappa) \leq 1/p(\kappa_0)$ for all $\kappa \geq \kappa_0$.

Notations for randomized execution. A protocol's execution is randomized, where the randomness comes from honest players as well as the adversary denoted $\mathcal{A}$ that controls all corrupt nodes, and the environment $\mathcal{Z}$ that sends inputs to honest nodes during the protocol execution. We use the notation $\mathsf{view} \leftarrow_{\$} \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ to denote a randomly sampled execution trace, and $|\mathsf{view}|$ denotes the number of rounds in the execution trace view. More specifically, view is a random variable denoting the joint view of all parties (i.e., all their inputs, random coins and messages received, including those from the random oracle) in the above execution; note that this joint view fully determines the execution.

Compliant executions. Henceforth in the paper, we require that our protocols retain specific security properties for all but a negligible (in the security parameter κ) fraction of execution traces (i.e., views). A protocol may retain its security properties only if $(\mathcal{A}, \mathcal{Z})$ abides by certain constraints, e.g., corrupting not too many nodes, delivers messages not too slowly.

Let ρ, Δ, N^*, and χ be functions of the security parameter κ and possibly of other parameters. We say that N^* is a χ-approximate upper bound estimate of N iff $\frac{N^*}{\chi} \leq N \leq N^*$. We say that $(\mathcal{A}, \mathcal{Z})$ is $(\rho, \Delta, N^*, \chi)$-respecting iff for every view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, $\mathcal{Z}$ inputs parameters (ρ, Δ, N^*) to all honest nodes prior to the start of execution, and moreover there is some N such that N^* is a χ-approximate upper bound estimate of N and the following holds:

- *Number of nodes.* In every round in view, there are N nodes that have been spawned but not killed;
- *Resilience.* In every round in view, there are at most $\lceil \rho N \rceil$ corrupt nodes;
- Δ-*bounded delivery.* For any message sent by an honest node at time t, at time $t' \geq t + \Delta$, every node honest at time t' will have received the message (including nodes that that may have spawned after t).

2.3 State Machine Replication

State machine replication has been a central abstraction in the 30 years of distributed systems literature. In a state machine replication protocol, a set of nodes seek to agree on an ever-growing log over time. We require two critical security properties: 1) *consistency*, i.e., all honest nodes' logs agree with each other although some nodes may progress faster than others; 2) *liveness*, i.e., transactions received by honest nodes as input get confirmed in all honest nodes' logs quickly. We now define what it formally means for a protocol to realize a "state machine replication" abstraction. Henceforth in this paper, whenever we refer to "permissionless consensus" or "consensus", we specifically mean state machine replication (although the term consensus is also commonly used in the distributed systems literature to mean single-shot consensus).

Syntax. In a state machine replication protocol, in every round, an honest node receives as input a set of transactions txs from $\mathcal{Z}$ at the beginning of the round, and outputs a LOG collected thus far to $\mathcal{Z}$ at the end of the round.

Security. Let T_{confirm} be a polynomial function in the security parameter κ and possibly other parameters of the execution such as the number of nodes participating, the corrupt fraction, the network delay, etc.

Definition 1. We say that a state machine replication protocol Π satisfies consistency (or T_{confirm}-liveness resp.) w.r.t. some $(\mathcal{A}, \mathcal{Z})$, iff there exists a negligible function $\mathsf{negl}(\cdot)$, such that for any $\kappa \in \mathbb{N}$, except with $\mathsf{negl}(\kappa)$ probability over the choice of $\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, consistency (or T_{confirm}-liveness resp.) is satisfied:

- *Consistency*: A view satisfies consistency iff the following holds:
  - *Common prefix.* Suppose that in view, an honest node i outputs LOG to $\mathcal{Z}$ at time t, and an honest node j outputs LOG' to $\mathcal{Z}$ at time t' (i and j may be the same or different), it holds that either $\mathsf{LOG} \prec \mathsf{LOG}'$ or $\mathsf{LOG}' \prec \mathsf{LOG}$. Here the relation $\prec$ means "is a prefix of". By convention we assume that $\emptyset \prec x$ and $x \prec x$ for any x.
  - *Self-consistency.* Suppose that in view, a node i is honest during $[t, t']$, and outputs LOG and LOG' at times t and t' respectively, it holds that $\mathsf{LOG} \prec \mathsf{LOG}'$.
- *Liveness*: A view satisfies T_{confirm}-liveness iff the following holds: if in some round $t \leq |\mathsf{view}| - T_{\text{confirm}}$, some node honest in round t either received from $\mathcal{Z}$ an input set txs that contains some transaction tx or has tx in its output log to $\mathcal{Z}$ in round t, then, for any node i honest at any time $t' \geq t + T_{\text{confirm}}$, let LOG be the output of node i at time t', it holds that $\mathsf{tx} \in \mathsf{LOG}$.

  Intuitively, liveness says that transactions input to an honest node get included in honest nodes' LOGs within T_{confirm} time; and further, if a transaction appears in some honest node's LOG, it will appear in every honest node's LOG within T_{confirm} time.

We say that a state machine replication protocol Π is secure in (ρ, Δ, N, χ)-environments, and has transaction conformation time T_{confirm} iff for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ that is (ρ, Δ, N, χ)-respecting, Π satisfies consistency and T_{confirm}-liveness w.r.t. $(\mathcal{A}, \mathcal{Z})$.

2.4 Modeling Proofs-of-Work

Later, to model state machine replication protocols that leverage proofs-of-work, we need to extend the protocol execution model with a random oracle. In an execution with security parameter κ, we assume all nodes have access to a random function $H : \{0,1\}^* \to \{0,1\}^\kappa$ which they can access through two oracles: $\mathsf{H}(x)$ simply outputs $H(x)$ and $\mathsf{H.ver}(x, y)$ output 1 iff $H(x) = y$ and 0 otherwise. In any round r, the nodes (as well as $\mathcal{A}$) may make *any* number of queries to $\mathsf{H.ver}$. On the other hand, in each round r, an honest node can make only a *single* query to H, and an adversary $\mathcal{A}$ controlling q parties, can make q *sequential* queries to H. (This modeling is meant to capture the assumption that we only "charge" for the effort of finding a solution to a "proof of work" [DN92], but checking the validity of a solution is cheap. We discuss this further after introducing Nakamoto's protocol.) We emphasize that the environment $\mathcal{Z}$ does not get direct access to the random oracle (but can instruct $\mathcal{A}$ to make queries).

3 Limits of the Permissionless Model

In this section, we present some simple but interesting lower bounds which formally capture why the "permissionless" model fundamentally departs from the classical "permissioned" model adopted in the traditional distributed systems and cryptography literature; and why some of the intuitions and insights we have gained about the permissioned model are no longer applicable to the permissionless model. A subset of these lower bounds were proved in earlier works [PS17b]. Since the permissionless model differs from classical models in several respects (e.g., no authentication,

no exact knowledge of number of nodes, nodes dynamically joining), below in our description, we make it explicit exactly which permissionless assumption leads to each lower bound.

Although not explicitly noted, all of our lower bounds in this section apply even to a weaker execution model in which corruption must be *static*, i.e., $\mathcal{Z}$ cannot corrupt an honest node after it is spawned.

3.1 Proof-of-Work is Necessary Absent Authentication

If the underlying network model provides no authentication of the sender, then the adversary is free to impersonate any number of nodes. For example, in a voting based protocol, a corrupt node can launch a Sybil attack and cast many votes. In this setting, intuitively, consensus should not be possible unless there is a rate-limiting mechanism that limits the extent to which such Sybil attacks are successful. One possible rate-limiting mechanism is through proof-of-work. We formalize this intuition in the following lower bound which states that proof-of-work (assuming no other rate limiting mechanism in place) is indeed necessary for reaching consensus in a network without authentication, even when all nodes are honest (but the adversary can insert messages into the network). This lower bound holds even in a static corruption environment and when $\mathcal{Z}$ must spawn all nodes upfront prior to execution start.

Early spawning. We say that $(\mathcal{A}, \mathcal{Z})$ respects early spawning w.r.t. Π, iff for every view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, $\mathcal{Z}$ spawns all nodes prior to the protocol's execution and does not spawn more nodes after the execution starts. Henceforth, we say that a state machine replication protocol Π satisfies consistency and T_{confirm} liveness in (ρ, Δ, N, χ)-early-spawning environments if for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ that is (ρ, Δ, N, χ)-respecting and moreover respects early spawning w.r.t. Π, Π satisfies consistency and T_{confirm}-liveness w.r.t. $(\mathcal{A}, \mathcal{Z})$.

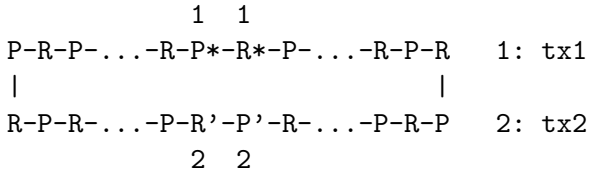
Theorem 1 (Proof-of-work is necessary absent authentication)**.** *Let Π denote a protocol whose protocol instructions never query the proof-of-work oracle. Then, for any $N \geq 3$, any polynomial $T_{\text{confirm}}(\kappa)$, Π cannot simultaneously satisfy consistency and T_{confirm}-liveness in $(\rho = 0, \Delta = 1, N, \chi = 1)$-early-spawning environments.*

Proof roadmap. The proof is reminiscent of the classical and elegant proof by Fischer et al. [FLM85] who showed that no deterministic protocol can realize weak byzantine agreement[1] under $\frac{1}{3}$ or more fraction of corrupt nodes (and their proof [FLM85] is a simplification of the original proof of the same statement by Lamport [Lam83]). Here, we use similar techniques, but we show a proof for an unauthenticated network — and since in such a network the adversary can impersonate any honest node, a lower bound of similar nature holds even when all nodes are honest but the adversary can insert messages into the network. We stress also that in contrast to the 1/3 lower bound for weak byzantine agreement, our lower bound proof rules out even *randomized* protocols.

The proof. We construct the following p.p.t. pair $(\mathcal{A}, \mathcal{Z})$ that is $(\rho = 0, \Delta = 1, N, \chi = 1)$-respecting and respects early spawning. Specifically, $\mathcal{Z}$ spawns N honest nodes upfront prior to the start of the execution. Without loss of generality we shall assume that N is even. Let P^* and R^* each denote a disjoint set of $N/2$ honest nodes. Henceforth we can consider P^* and R^* as super-nodes. We only care about messages transmitted in between super-nodes but not within.

[1] Weak byzantine agreement has a weaker validity requirement: validity is only required if all nodes remain honest and all nodes receive the same input bit b; in this case, every node is required to output b.

Let $M(\kappa)$ be an appropriate parameter to be determined later. Now, the adversary simulates in its head M super-nodes (each containing $N/2$ simulated nodes executing the honest protocol), and the execution is depicted in the following graph where the starred nodes denote the honest nodes, and every other super-node is simulated by $\mathcal{A}$ (note that $\mathcal{A}$ can do this without corrupting node since it can inject messages into the network).



Now $(\mathcal{A}, \mathcal{Z})$ samples $\mathsf{tx}_1, \mathsf{tx}_2$ from some high-entropy distribution $\mathcal{D}$ such that $\mathsf{tx}_1, \mathsf{tx}_2$ are different except with $\mathsf{negl}(\kappa)$ probability. $\mathcal{Z}$ input $\{\mathsf{tx}_1\}$ to every honest node in P^* and R^* in every round. For simulated super-nodes P' and R', all nodes within these super-nodes act as if they receive the input $\{\mathsf{tx}_2\}$ in every round. All other nodes (besides P^*, R^*, P' and R') act as if they receive the dummy input $\mathsf{tx} = \perp$ from $\mathcal{Z}$ in every round. We now consider such an execution $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ as depicted above.

Henceforth, if in some view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, a node outputs a log that contains tx_1 first before tx_2 appears in the log, then we say that the node outputs tx_1 first in view. Otherwise if a node outputs a log that contains tx_2 first before tx_1 appears in the log, then we say that the node outputs tx_2 first in view.

Claim 1. Suppose that the protocol Π satisfies T_{confirm}-liveness and consistency in $(\rho = 0, \Delta = 1, N, \chi = 1)$-early-spawning environments. Then, except with $\mathsf{negl}(\kappa)$ probability over the choice of $\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, if any real or simulated super-node outputs tx_b first for $b \in \{1, 2\}$, then all other super-nodes must output tx_b first.

The above claim holds regardless of the choice of M.

Proof. For every adjacent pair of super-nodes (either real or simulated) henceforth denoted P and R respectively, P and R's view in the above execution is identically distributed as an execution in which P and R are spawned to be honest, and every other super-node is simulated by the adversary $\mathcal{A}$, and where each node receives input in a way as said above. Then, by our assumption on Π, except with negligible probability over the choice of view, P and R should output the same tx_b first where $b \in \{1, 2\}$. The remainder of the proof follows by taking a union bound over the number of adjacent pairs in the figure. □

Claim 2. Suppose that the protocol Π satisfies T_{confirm}-liveness in $(\rho = 0, \Delta = 1, N, \chi = 1)$-early-spawning environments, and that $M > 3T_{\text{confirm}}(\kappa)$. Then it holds that except with negligible probability over the choice of view, P^* and R^* must output tx_1 first; and P' and R' must output tx_2 first.

Proof. Let X be a super-node; and let view_X^t be a random variable denoting the joint view of all nodes in the super node X's by the end of round t. Since it takes exactly 1 round to deliver each message, suppose that two super-nodes X and Y are r distance away from each other in the above graph. Then, it is not hard to see that view_X^t is independent of Y's input received from $\mathcal{Z}$ as long as $t < r$. This means that by time T_{confirm}, the views of P^* and R^*'s are independent of tx_2. Now, since tx_2 is chosen from a high-entropy distribution as mentioned earlier, it is not hard to see that tx_2 cannot be in P^* and R^*'s output log except with negligible probability over the choice of view. On the other hand, by the liveness assumption on Π, we know that except with negligible probability over the choice of view, tx_1 must be in the output logs of P^* and R^* by time T_{confirm}. In other words, P^* and R^* must output tx_1 first except with negligible probability.

The claim now follows by observing that a symmetric argument holds for P' and R': except with negligible probability, P' and R' must output tx_2 first. □

Proof of Theorem 1. Straightforward from Claims 1 and 2.

3.2 Limits Due to Late Spawning

When nodes may spawn dynamically, a few interesting lower bounds arise. First, we show that not only is proof-of-work necessary, any secure state machine replication protocol must in fact perform proof-of-work infinitely often. Second, we show that honest majority is necessary to achieve state machine replication in a network that allows dynamic spawning of nodes. We now formally state the following lower bounds — both of these lower bounds hold even when assuming the existence of a proof-of-work oracle, assuming static corruption, that all honest messages are delivered within a single round, and that the protocol knows the exact number of nodes N.

Let ρ, Δ, N, χ be polynomially bounded (or inverse polynomially bounded) in κ. We say that a protocol Π stops proof-of-work early in (ρ, Δ, N, χ)-environments iff for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ that is (ρ, Δ, N, χ)-respecting, there exists some polynomial $T(\cdot)$, such that in every view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, honest nodes stop querying the proof-of-work oracle after $T(\kappa)$ time. The following theorem was proved in our earlier work [PS17b].

Theorem 2 (Proof-of-work must be performed infinitely often [PS17b])**.** *Let $N \geq 3$ be polynomially bounded in κ and let Π be a protocol that stops proof-of-work early in $(\rho = 0.01, \Delta = 1, N, \chi = 1)$-environments. Then, for any polynomial $T_{\text{confirm}}(\kappa)$, Π cannot simultaneously satisfy consistency and T_{confirm}-liveness in $(\rho = 0.01, \Delta = 1, N, \chi = 1)$-environments.*

Proof. Suppose for the sake of contradiction that there is a protocol Π that stops proof-of-work early in $(0.01, 1, N, 1)$-environments; and moreover Π satisfies both consistency and T_{confirm}-liveness in $(0.01, 1, N, 1)$-environments for some polynomial T_{confirm}.

We now construct the following p.p.t. $(\mathcal{A}, \mathcal{Z})$ pair that is $(\rho = 0.01, \Delta = 1, N, \chi = 1)$-respecting: $\mathcal{Z}$ spawns N nodes upfront, $\lceil 0.01N \rceil$ fraction of which are corrupt and the remaining honest. Till time $T(\kappa)$ corrupt nodes participate honestly in the protocol. By assumption, at time $T(\kappa)$ and after honest nodes stop making queries to the proof-of-work oracle.

At time T, $\mathcal{A}$ relies on corrupt nodes to simulate an alternate execution in its head from scratch. This would require that corrupt nodes query the proof-of-work oracle; but otherwise the corrupt nodes still continue to play honestly in the real-world execution.

It is not difficult to see that at time $t^* = T + 100T + T_{\text{confirm}} + 1$, the simulated execution can simulate as many rounds as the real execution. $\mathcal{Z}$ generates two random transactions tx_0 and tx_1 from $\{0,1\}^\kappa$. For honest nodes in the real execution, $\mathcal{Z}$ inputs $\{\mathsf{tx}_0\}$ for every round. For the simulated execution, the simulated honest nodes act as if $\mathcal{Z}$ inputs $\{\mathsf{tx}_1\}$ in every round. Suppose also that at time t^*, $\mathcal{Z}$ kills an existing honest node and spawns a new honest node i^* (in the real execution). At this moment, $\mathcal{A}$ relays messages from both the real- and the simulated executions to i^*. Whenever i^* sends a message, the message is relayed to nodes (or simulated nodes) in both the real and simulated executions.

Now by the end of round $t^* - 1$, let LOG_0 be some honest node's log in the real execution. By liveness, it holds that $\mathsf{tx}_0 \in \mathsf{LOG}_0$ except for a negligible fraction of the views. By a symmetric argument, except for a negligible fraction of views, for the simulated execution, at simulated time $t^* - 1$, it must be that $\mathsf{tx}_1 \in \mathsf{LOG}_1$ where LOG_1 denotes the log of some simulated honest node at simulated time $t^* - 1$. Since tx_0 and tx_1 are generated from a high-entropy distribution, except with negligible probability over the choice of view, it must be that $\mathsf{tx}_0 \notin \mathsf{LOG}_1$ and $\mathsf{tx}_1 \notin \mathsf{LOG}_0$ since both LOG_0 and LOG_1 are of polynomial length.

Let LOG^* denote the log of node i^* by the end of round t^*. If we consider the real execution, except for a negligible fraction of views where either consistency or liveness fails, we have that

$\mathsf{tx}_0 \in \mathsf{LOG}^*$ due to liveness; and due to consistency, it must be that $\mathsf{LOG}^* \prec \mathsf{LOG}_0$ or $\mathsf{LOG}_0 \prec \mathsf{LOG}^*$. Taking a union bound, except with negligible probability over the choice of view, either tx_0 appears before tx_1 in LOG^* or tx_1 does not appear at all in LOG^*.

Since the view of node i^* in the real execution and the simulated execution are identically distributed, by a symmetric argument, except with negligible probability over the choice of view, either tx_1 appears before tx_0 in LOG^* or tx_0 does not appear in LOG^*. Thus we have reached a contradiction.

□

Theorem 3 (Honest majority is necessary for late spawning)**.** *For any polynomial $T_{\text{confirm}}(\kappa)$, and any $N \geq 3$, no protocol Π can simultaneously satisfy consistency and T_{confirm}-liveness in $(\rho = 0.5, \Delta = 1, N, \chi = 1)$-environments.*

Proof. Suppose for the sake of contradiction that there is a protocol Π that satisfies both consistency and T_{confirm}-liveness $(\rho = 0.5, \Delta = 1, N, \chi = 1)$-environments for some $N \geq 3$ and some polynomial T_{confirm}. We can construct the following p.p.t. $(\mathcal{A}, \mathcal{Z})$ pair that is $(0.5, 1, N, 1)$-respecting: $\mathcal{Z}$ spawns N nodes upfront among which exactly $\lceil 0.5N \rceil$ nodes are corrupt. Now, corrupt nodes do not participate in the real execution, but instead simulate an alternate execution among themselves. $\mathcal{Z}$ generates two random inputs tx_0 and tx_1 from $\{0,1\}^\kappa$, and inputs $\{\mathsf{tx}_0\}$ to every honest node in the real execution in every round. Every simulated honest node in the simulated execution acts as if $\{\mathsf{tx}_1\}$ is received from $\mathcal{Z}$ in every round.

At some time after T_{confirm}, suppose that $\mathcal{Z}$ kills an existing honest node from the real execution; and similarly a simulated node is killed from the simulated execution. At the same time $\mathcal{Z}$ spawns a new node i^*. $\mathcal{A}$ relays messages from both the real- and the simulated executions to i^*. Whenever i^* sends a message, the message is relayed to nodes (or simulated nodes) in both the real and simulated executions. The rest of the proof follows due to the same reasoning as the proof of Theorem 2 such that we can reach a contradiction. □

We note that late spawning is also necessary for the above lower bounds to hold. Interestingly, if we assume no late spawning of nodes, i.e., all nodes must be spawned upfront prior to protocol execution, state machine replication is known to be possible against an arbitrary corrupt fraction and without having to call proofs-of-work infinitely often (assuming full network synchrony). In particular, Miller et al. [KMS14] showed that assuming the existence of a proof-of-work oracle, there exists a protocol Π that realizes a "pseudonymous PKI" in $(\rho, \Delta = 1, N, \chi = 1)$-environments for any ρ and N. From a pseudonymous PKI, one can easily realize "pseudonymous broadcast" and in turn realize state machine replication simply by spawning one instance of pseudonymous broadcast per round where in the i-th instance, the $(i \mod N)$-th pseudonym acts as the sender. A similar result was shown by Andrychowicz et al. [AD15].

Remark 1. Finally, we remark that the lower bound stated in the above Theorem 3 (i.e., honest majority is necessary even if there is a single late-spawning node) and its proof still hold even in a permissioned model and even if assuming PKI — the permissioned version of this thereom was proven in our recent work [PS17c].

3.3 Limits due to Uncertainty in Number of Nodes

Another feature of the permissionless model is that the protocol may not know upfront the exact number of nodes that will show up. Our earlier paper [PS17b] showed that if this is the case, then state machine replication is not possible under partially synchronous or asynchronous network

models — even when the protocol is given an a-priori estimate of the number of nodes that is accurate to a factor of 2, and even when assuming that $\mathcal{Z}$ must spawn all nodes upfront and all nodes must be honest (but $\mathcal{A}$ controls network delivery). We capture this intuition in the following theorem, which, intuitively, states that if some protocol Π knowing an estimate N^* can achieve liveness when only $\frac{N^*}{2}$ honest nodes show up, then Π cannot guarantee consistency if N^* nodes actually show up.

Formally, we say that a protocol Π is partially synchronous if Π's instructions do not depend on the Δ value received from $\mathcal{Z}$ prior to protocol start. The following theorem was proved in our earlier work [PS17b].

Theorem 4 (Impossibility of partial synchrony and asynchrony due to uncertainty in number [PS17b])**.** *Let $N(\cdot)$ and $\rho(\cdot)$ be polynomial functions in κ and let $T_{\text{confirm}}(\cdot,\cdot)$ be a polynomial function in κ and the actual maximum delay of a* view*. For any $N \geq 2$, any ρ and any T_{confirm}, there exists a polynomial function Δ such that no partially synchronous protocol Π can simultaneously achieve consistency and T_{confirm}-liveness in $(\rho, \Delta, 2N, 2)$-early-spawning environments.*

Proof. For any $N \geq 2$ and any polynomial T_{confirm}, we construct a p.p.t. $(\mathcal{A}, \mathcal{Z})$ pair that is $(\rho = 0, \Delta, 2N, \chi = 2)$-respecting for some polynomial $\Delta > T_{\text{confirm}}(\kappa, 1)$ and moreover respects early spawning. We show that for any partially synchronous protocol Π, if Π satisfies T_{confirm}-liveness w.r.t. $(\mathcal{A}, \mathcal{Z})$, then Π cannot satisfy consistency w.r.t. $(\mathcal{A}, \mathcal{Z})$.

Upfront $\mathcal{Z}$ spawns $2N$ nodes all of which honest, and informs honest nodes of the parameters $(0, \Delta, 2N)$. These $2N$ nodes are divided into two partitions each containing N honest nodes, and $\mathcal{A}$ delivers messages within each partition within 1 round; however, for messages in between partitions, they are delayed for $\Delta > T_{\text{confirm}}(\kappa, 1)$. $\mathcal{Z}$ samples two random transactions tx_1 and tx_2 from $\{0,1\}^\kappa$. For honest nodes in the first partition, $\mathcal{Z}$ inputs $\{\mathsf{tx}_1\}$ in every round; and for honest nodes in the second partition, $\mathcal{Z}$ inputs $\{\mathsf{tx}_2\}$ in every round.

Clearly by time $T_{\text{confirm}}(\kappa, 1)$, the view of honest nodes in the first partition is identically distributed as an alternate execution $\mathsf{EXEC}^{\Pi}(\mathcal{A}', \mathcal{Z}', \kappa)$ where $(\mathcal{A}', \mathcal{Z}')$ is also $(\rho = 0, \Delta, 2N, \chi = 2)$-respecting and moreover respects early spawning, but now 1) $\mathcal{Z}'$ spawns only N honest nodes upfront; 2) $\mathcal{A}'$ always delivers honest messages immediately in the next round; and 3) $\mathcal{Z}'$ inputs transactions in the same way $\mathcal{Z}$ inputs transactions to the first partition. Intuitively, this is saying honest nodes cannot tell whether the other partition has not been spawned or whether delay is large. Let LOG_1 denote the log of some honest node in the first partition by the end of round $T_{\text{confirm}}(\kappa, 1)$. By liveness of the alternate execution $\mathsf{EXEC}^{\Pi}(\mathcal{A}', \mathcal{Z}', \kappa)$, except with negligible probability over the choice of view, $\mathsf{tx}_1 \in \mathsf{LOG}_1$. By a symmetric argument, let LOG_2 denote the log of some honest node in the second partition by the end of round $T_{\text{confirm}}(\kappa, 1)$ — then except with negligible probability over the choice of view, $\mathsf{tx}_2 \in \mathsf{LOG}_2$. Since tx_1 and tx_2 are sampled at random from a high entropy distribution, it must hold that except with negligible probability over the choice of view, $\mathsf{tx}_2 \notin \mathsf{LOG}_1$ and $\mathsf{tx}_1 \notin \mathsf{LOG}_2$. Thus, we conclude that Π cannot satisfy consistency w.r.t. $(\mathcal{A}, \mathcal{Z})$. □

Remark 2. Finally, we remark that the above partial synchrony lower bound stated in Theorem 4 and its proof still hold even in a permissioned model and even if assuming PKI, as long as the number of players is uncertain — the permissioned version of this theorem was originally proven in our recent work [PS17c].

4 Abstract Blockchain Protocols

A blockchain protocol can be regarded as a way to realize state machine replication. We now formally define what it means for a protocol to realize to a blockchain abstraction.

Blockchain notations. Henceforth we often use the notation chain to denote an abstract blockchain. The notation $\mathsf{chain}[:-\ell]$ denotes the entire chain except the trailing ℓ blocks; $\mathsf{chain}[:\ell]$ denotes the entire chain upto the block at length ℓ; $\mathsf{chain}[-\ell:]$ denotes the trailing ℓ blocks of chain; and $\mathsf{chain}[\ell:]$ denotes all blocks at length ℓ or greater.

Syntax. In a blockchain protocol, in every round, an honest node receives an input txs from $\mathcal{Z}$ at the beginning of the round, and outputs an abstract blockchain chain to $\mathcal{Z}$ at the end of the round. Each $\mathsf{chain}[i]$ where $i \in [|\mathsf{chain}|]$ is referred to as an abstract block.

Security. A blockchain protocol should satisfy chain growth, chain quality, and consistency. Intuitively, chain growth requires that honest nodes' blockchains grow steadily, neither too fast nor too slow. Chain quality requires that in any honest node's chain, any sufficiently long window of consecutive blocks contains a certain fraction of blocks that are mined by honest nodes. Consistency requires that all honest nodes' chains agree with each other except for the trailing few blocks.

We now define these security properties formally. Henceforth if an honest node outputs some chain to $\mathcal{Z}$ in some round r in view, we say that chain is an "honest chain" in round r in view.

Definition 2. We say that a blockchain protocol Π satisfies (T, g_0, g_1)-chain growth, (T, μ)-chain quality, and T-consistency in $(\rho, \Delta, N^*, \chi)$-environments iff for any $(\rho, \Delta, N^*, \chi)$-respecting p.p.t. $(\mathcal{A}, \mathcal{Z})$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\kappa \in \mathbb{N}$, except with $\mathsf{negl}(\kappa)$ probability over the choice of $\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, the following hold for view:

- (T, g_0, g_1)-*chain growth.* A view satisfies (T, g_0, g_1)-chain growth iff the following hold:
  - *Consistent length:* If in round r some honest chain is of length ℓ, then in round $r + \Delta$, all honest chains must be of length at least ℓ.
  - *Growth lower bound:* For any r and t such that $g_0(t - r) \geq T$, let chain^r and chain^t denote two honest chains in round r and t respectively, it holds that
  $$|\mathsf{chain}^t| - |\mathsf{chain}^r| \geq g_0(t - r)$$
  - *Growth upper bound:* For any r and t such that $g_1(t - r) \geq T$, let chain^r and chain^t denote two honest chains in round r and t respectively, it holds that
  $$|\mathsf{chain}^t| - |\mathsf{chain}^r| \leq g_1(t - r)$$
- (T, L, μ)-*chain quality.* A view satisfies (T, L, μ)-chain quality iff the following holds: for any honest chain denoted chain in view, for any T consecutive blocks $\mathsf{chain}[j + 1..j + T]$, more than μ fraction of the blocks in $\mathsf{chain}[j + 1..j + T]$ are mined by honest nodes at most L blocks ago — here we say that a block $\mathsf{chain}[i]$ is "mined by an honest node at most L blocks ago" iff $\mathcal{Z}$ input the contents of $\mathsf{chain}[i]$ to some honest node when its last output to $\mathcal{Z}$ contains the prefix $\mathsf{chain}[: i - L]$ (here if $i - L < 0$, we round it to 0).
- T-*consistency.* A view satisfies T-consistency iff the following hold: for any two honest chains chain^r and chain^t in round r and $t \geq r$ respectively, it holds that
  $$\mathsf{chain}^r[:-T] \prec \mathsf{chain}^t$$
  We stress that since chain^r and chain^t can possibly belong to the same node, the above definition also implies "future self consistency" except for the trailing T blocks.

Blockchain implies state machine replication. Given any blockchain protocol $\Pi_{\text{blockchain}}$, it is easy to construct a state machine replication protocol where 1) nodes run an instance of $\Pi_{\text{blockchain}}$; 2) an honest node broadcasts all newly seen transactions to each other; and 3) in every round, nodes remove the trailing κ blocks from the present chain and output the truncated chain to the environment $\mathcal{Z}$ [PS17c, PS17b]. It is not difficult to see that consistency (of the resulting state machine replication protocol) follows directly from consistency of the blockchain; and liveness follows from chain quality and chain growth. The above intuition has been formalized in our earlier works [PS17c, PS17b].

5 Nakamoto's Blockchain

5.1 Nakamoto's Blockchain Protocol

We describe (a simplified variant of) Nakamoto's blockchain [Nak08] referred to as $\Pi_{\text{nak}}(p)$. $\Pi_{\text{nak}}(p)$ takes in a puzzle difficulty parameter p that denotes the probability that each player mines a block in a single round. As we mention later, for the protocol to retain consistency, this difficulty parameter p must be chosen based on the total mining power and the maximum network delay, such that on average, it takes significantly more time for the entire network to mine a block than the maximum network delay.

Protocol. In Π_{nak}, each honest node maintains a blockchain denoted *chain* at any point of time. Each $\mathit{chain}[i]$ is referred to as a (mined) block and is of the format $\mathit{chain}[i] := (h_{-1}, \eta, \mathsf{txs}, h)$, containing the hash of the previous block denoted h_{-1}, a nonce η, a record txs, and a hash h.

We use $\mathsf{chain} := \mathsf{extract}(\mathit{chain})$ to denote the sequence of records contained in the sequence of blocks *chain*. Honest nodes then output the sequence of extracted records chain to the environment $\mathcal{Z}$ — in other words, chain is an abstract blockchain (i.e., a linearly ordered log) stripped of all protocol-specific metadata.

The Π_{nak} protocol works as follows:

- Nodes that are newly spawned start with initial chain containing only a special genesis block: $\mathit{chain} := (0, 0, \perp, \mathsf{H}(0, 0, \perp))$.
- In every round: a node reads all incoming messages (delivered by $\mathcal{A}$). If any incoming message chain' is a valid sequence of blocks that is longer than its local blockchain *chain*, replace *chain* by chain'. We define what it means for a chain to be valid later. Checking the validity of chain' can be done using only $\mathsf{H.ver}$ queries.
- Read an input record txs from the environment $\mathcal{Z}$. Now parse $\mathit{chain}[-1] := (\_, \_, \_, h_{-1})$, pick a random nonce $\eta \in \{0,1\}^\kappa$, and issue query $h = \mathsf{H}(h_{-1}, \eta, \mathsf{txs})$. If $h < D_p$, then append the *newly mined* block (h_{-1}, η, b, h) to *chain* and broadcasts the updated *chain*. More specifically, $D_p = p(\kappa) \cdot 2^\kappa$ such that for all (h, txs), $\Pr_\eta[\mathsf{H}(h, \eta, \mathsf{txs}) < D_p] = p$. In other words, D_p is appropriately parametrized such that the probability that any player mines a block in a round is p. We will describe shortly how the Π_{nak} protocol sets the parameter p based on the input parameters ρ, Δ, N^* it receives from $\mathcal{Z}$.
- Output $\mathsf{chain} := \mathsf{extract}(\mathit{chain})$ to the environment $\mathcal{Z}$. Note that the notation chain extracts only the sequence of records from *chain* removing all other metadata that are not needed by external applications.

Valid chain. We say a block $chain[i] = (h_{-1}, \eta, \mathsf{txs}, h)$ is *valid with respect to (a predecessor block)* $chain[i-1] = (h'_{-1}, n', \mathsf{txs}', h')$ if two conditions hold: $h_{-1} = h'$, $h = \mathsf{H}(h_{-1}, \eta, \mathsf{txs})$, and $h < D_p$. A chain of blocks *chain* is *valid* if a) $chain[0] = (0, 0, \perp, \mathsf{H}(0, 0, \perp))$ is the genesis block, and b) for all $i \in [\ell]$, $chain[i]$ is valid with respect to $chain[i-1]$.

5.2 Parameters and Notations

Setting the protocol's difficulty parameter. Henceforth, we assume that the protocol Π_{nak} is parametrized by a constant ϕ whose meaning will become obvious later. As mentioned earlier, the protocol Π^{ϕ}_{nak} receives as input the parameters (ρ, Δ, N^*). Based on these parameters, we assume that the protocol Π^{ϕ}_{nak} will choose an appropriate difficulty parameter p (i.e., the probability that each mining attempt is successful) satisfying the following relations:

a) $\nu := 2pN^*\Delta < 1$; and

b) Honest nodes must outnumber corrupt nodes by an appropriate constant margin, or formally,

$$\frac{1-\rho}{\rho} \geq \frac{1+\phi}{1-\nu} \tag{1}$$

It is not hard to see that for any positive constants $\rho < \frac{1}{2}$ and ϕ, for any Δ, N^*, such a p can be efficiently determined.

Additional notations. Henceforth without loss of generality, we use the notation N to denote the actual number of nodes during an execution. If $(\mathcal{A}, \mathcal{Z})$ is $(\rho, \Delta, N^*, \chi)$-respecting, it must hold that $\frac{N^*}{\chi} \leq N \leq N^*$. We may assume that every round has exactly $N_H := (1-\rho)N$ number of honest nodes and exactly $N_C := \rho N$ corrupt nodes — since if there are more honest nodes in a round, we can always imagine that some of the honest nodes are corrupt nodes that simply follow the honest protocol (and our proof works for any corrupt behavior).

Let $\alpha = p \cdot N_H$ denote the expected number of honest nodes that mine a block in each round. Let $\beta = p \cdot N_C$ denote the expected number of corrupt nodes that mine a block in each round.

5.3 Main Theorem for Nakamoto's Blockchain

Theorem 5 (Nakamoto's blockchain)**.** *For any T that is super-logarithmic in κ, any constant $\rho < \frac{1}{2}$, any Δ, N^*, χ, any positive constant $\phi > 0$, Π^{ϕ}_{nak} satisfies the following properties in $(\rho, \Delta, N^*, \chi)$-environments where p is chosen according to the rules defined in Section 5.2 based on ρ, Δ, N^*, and ϕ:*

- *(T, g_0, g_1)-chain growth where $g_0 = (1-\epsilon)(1-2pN)\alpha$, and $g_1 = (1+\epsilon)Np$ where N denotes the actual number of nodes in each round;*
- *$(T, 1, \mu)$-chain quality where $\mu = 1 - \frac{1+\epsilon}{1+\phi}$; and*
- *T-consistency.*

5.4 Ideal-World Protocol

To prove the security properties of Nakamoto's blockchain, it helps to think of an ideal protocol denoted $\Pi^{\phi}_{\text{ideal}}$ that captures the nature of the randomized process behind Π^{ϕ}_{nak}. In this ideal protocol, nodes mine blocks by interacting with an ideal functionality $\mathcal{F}_{\text{tree}}$ which keeps track of all the blockchains mined thus far. Below we first describe the ideal protocol Π_{ideal}, and then we explain to prove the security properties of Π_{nak}, it is sufficient to prove the same properties for Π_{ideal}.

Ideal functionality $\mathcal{F}_{\text{tree}}$. One may think of the ideal functionality $\mathcal{F}_{\text{tree}}(p)$ as a trusted party in the ideal protocol Π_{ideal}. The ideal functionality $\mathcal{F}_{\text{tree}}$ is parametrized with an appropriate difficulty parameter p such that each mining attempt is successful with probability p — choice of p is made in the same manner as for the real-world protocol Π_{nak} (see Section 5.2).

$\mathcal{F}_{\text{tree}}$ keeps track of the set (denoted tree) of all abstract blockchains mined thus far. Initially, the only blockchain in the set tree is $\mathsf{genesis}$. $\mathcal{F}_{\text{tree}}$ allows honest and corrupt nodes to mine blocks and verify the validity of (abstract) blockchains through the following interfaces:

- Upon receiving $\texttt{extend}(\mathsf{chain}, \mathsf{txs})$: $\mathcal{F}_{\text{tree}}$ checks if chain is a valid blockchain in tree. If so, $\mathcal{F}_{\text{tree}}$ flips a coin that comes up heads with probability p. If the coin flip is successful, $\mathcal{F}_{\text{tree}}$ records $\mathsf{chain}||\mathsf{txs}$ in the set tree, and returns success.
- Upon receiving $\texttt{verify}(\mathsf{chain})$: $\mathcal{F}_{\text{tree}}$ checks if chain is a valid blockchain in tree; if so, return true; else return false.

Ideal protocol Π_{ideal}. The ideal protocol $\Pi^{\phi}_{\text{ideal}}$ chooses a difficulty parameter p in the same fashion as Π^{ϕ}_{nak} based on input parameter (ρ, Δ, N^*) received from $\mathcal{Z}$, and then the protocol works as follows where $\mathcal{F}_{\text{tree}} := \mathcal{F}_{\text{tree}}(p)$:

- Every node maintains a longest abstract blockchain seen thus far denoted chain.
- In every round, every honest node first receives all incoming messages on the network. For any received message chain': If $\mathcal{F}_{\text{tree}}.\texttt{verify}(\mathsf{chain}') = 1$ and chain' is longer than the current local chain, then let $\mathsf{chain} := \mathsf{chain}'$ and broadcast chain'.
- In every round, every honest node receives a record txs from the environment $\mathcal{Z}$, and then queries $\mathcal{F}_{\text{tree}}.\texttt{extend}(\mathsf{chain}, \mathsf{txs})$. If this mining query is successful, the node broadcasts $\mathsf{chain}||\mathsf{txs}$ and replaces its chain with $\mathsf{chain} := \mathsf{chain}||\mathsf{txs}$.

Real emulates ideal. The main difference between the real-world protocol Π_{nak} and the ideal-world protocol Π_{ideal} is that in the real world, the possibility of hash collisions may allow the adversary to perform attacks that are not relevant in the ideal world — however, intuitively, assuming that the proof-of-work function H is a random oracle, the probability of having a hash collision is negligible. Pass et al. [PSS17] has formally proved that the real-world protocol securely emulates the ideal-world:

Proposition 1 (Real emulates ideal [PSS17])**.** *For any p.p.t. adversary $\mathcal{A}$, there exists a p.p.t. simulator $\mathcal{S}$, such that for any p.p.t. $\mathcal{Z}$,*

$$\{\mathit{view}_{\mathcal{Z}}(\mathit{EXEC}^{\Pi^{\phi}_{ideal}}(\mathcal{S}, \mathcal{Z}, \kappa))\}_{\kappa \in \mathbb{N}} \stackrel{c}{\equiv} \{\mathit{view}_{\mathcal{Z}}(\mathit{EXEC}^{\Pi^{\phi}_{nak}}(\mathcal{A}, \mathcal{Z}, \kappa))\}_{\kappa \in \mathbb{N}}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability; and moreover, if $(\mathcal{A}, \mathcal{Z})$ is (ρ, Δ, N, χ)-respecting w.r.t. $\Pi_{nak}(p)$ for any $\rho, \Delta, N, \chi)$, then $(\mathcal{S}, \mathcal{Z})$ is (ρ, Δ, N, χ)-respecting w.r.t. Π_{ideal}.

The implication of the above proposition is that if a real-world adversary can break the security properties of Π_{nak} with probability ϵ, then there is an ideal-world adversary that can break the security properties of Π_{ideal} with probability $\epsilon - \mathsf{negl}(\kappa)$ where $\mathsf{negl}(\cdot)$ is some negligible function. We refer the reader to Pass et al. [PSS17] for the proof of this proposition — in essence the proof reasons that 1) the probability of hash collision is negligible in the real world; and 2) any real-world attack that does not exploit hash collisions would translate to an ideal-world attack.

Due to the above proposition, to prove Theorem 5, it suffices to prove the relevant security properties (i.e., consistency, chain quality, and chain growth) for the ideal protocol Π_{ideal} — this will be our focus in the remainder of this section.

5.5 Proof Roadmap

Related works. Several prior works analyzed the security of Nakamoto's blockchain protocol under the assumption that the adversary controls only minority of the total mining power. An elegant work by Sompolinsky and Zohar shows that Nakamoto's blockchain retains consistency against certain restricted attacks [SZ15]. Garay, Kiayas and Leonardos [GKL15] were the first to show that Nakamoto's blockchain retains consistency against an arbitrarily malicious adversary that controls a minority of the mining power, as long as the adversary must deliver messages immediately. Pass et al. [PSS17] were the first to prove the security of Nakamoto's blockchain when the adversary can reorder and delay messages arbitrarily, as long as any message sent by honest nodes is delivered with a maximum of Δ rounds.

Our proof. We present a simplified version of the proof by Pass et al. [PSS17] suitable for pedagogical purposes.

Shorthand notation. Henceforth in our proofs, we adopt the following shorthand notation: whenever we say *"Except with negligible probability over the choice of* views *some event* $\mathsf{ev}(\mathsf{view})$ *holds"*, we formally mean that for any positive constants ϕ and $\rho < \frac{1}{2}$, any Δ, N^*, χ, for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ pair that is $(\rho, \Delta, N^*, \chi)$-respecting, there exists some positive constant η such that for any $\kappa \in \mathbb{N}$,

$$\Pr\left[\mathsf{view} \leftarrow_{\$} \mathsf{EXEC}^{\Pi^{\phi}_{\text{nak}}}(\mathcal{A}, \mathcal{Z}, \kappa) : \mathsf{ev}(\mathsf{view})\right] \geq 1 - \exp(-\eta\kappa)$$

In particular, this means that in our proofs, *whenever we refer to a negligible function, we mean a strongly negligible function that is exponentially small in the security parameter.*

5.6 Convergence Opportunities

We now define a useful pattern called convergence opportunities [PSS17], which we shall later use in both our chain growth lower bound proof as well as consistency proof. Intuitively, a convergence opportunity is a Δ-period of silence in which no honest node mines a block, followed by a time step in which a single honest node mines a block, followed by another Δ-period of silence in which no honest node mines a block. We formalize this notion below.

Convergence opportunity. Given a view, suppose $T \leq |\mathsf{view}| - \Delta$, we say that $[T - \Delta, T + \Delta]$ is a convergence opportunity iff

- For any $t \in [\max(0, T - \Delta), T)$, no honest node mines a block in round t;
- A single node honest at T mines a block in round T;

- For any $t \in (T, T+\Delta]$, no node honest in round t mines a block.

Let T denote the time in which a single honest node mines a block during a convergence opportunity. For convenience, we often use T to refer to the convergence opportunity. We say that a convergence opportunity T is contained within a window $[t' : t]$ if $T \in [t' : t]$.

Henceforth, we use the notation $\mathbf{C}(\mathsf{view})[t' : t]$ to denote the number of convergence opportunities contained within the window $[t' : t]$ in view.

Convergence opportunities are common. We now show that convergence opportunities happen sufficiently often.

Lemma 1 (Number of convergence opportunities)**.** *For any positive constants η and ϵ, except with negligible probability over the choice of view, the following holds: for any $t_0, t_1 \geq 0$ such that $t := t_1 - t_0 > \frac{\epsilon\kappa}{\alpha}$, we have that*

$$\mathbf{C}(\mathit{view})[t_0 : t_1] > (1-\eta)(1-2pN\Delta)\alpha t$$

where $\alpha := p \cdot N_H$ denotes the expected number of honest nodes that mine a block in each round.

Proof. It suffices to prove the lemma for any fixed choice of t_0 and t — if we can do so, the lemma then follows by taking a union bound over polynomially many choices of t_0 and t. We now focus on the coins $\mathcal{F}_{\text{tree}}$ flips for honest nodes upon their `extend` queries — henceforth we refer to these coins as honest coins for short.

- Let $\mathbf{X}$ denote the total number of heads in all the honest coins during $[t_0, t_1]$. Due to the Chernoff bound (see Appendix A), for any constant $0 < \epsilon < 1$, it holds that

  $$\Pr[\mathbf{X} < (1-\epsilon) \cdot \alpha t] \leq \exp(-\Omega(\alpha t))$$

  Henceforth, let $L := (1-\epsilon) \cdot \alpha t$ for a sufficiently small constant ϵ.

- Let $\mathbf{Y}_i = 1$ iff after the i-th heads in the honest coin sequence during $[t_0, t_1]$, there exists a heads in the next $N_H\Delta$ coin flips. Notice that all of the $\mathbf{Y}_i$'s are independent — to see this, another way to think of $\mathbf{Y}_i$ is that $\mathbf{Y}_i = 0$ iff the i-th coin flip and the $(i+1)$-th coin flip are at least $N_H\Delta$ apart from each other.

  Let $\mathbf{Y} := \sum_{i=1}^{L} \mathbf{Y}_i$. We have that

  $$\mathbf{E}[\mathbf{Y}] \leq (1-(1-p)^{N_H\Delta}) \cdot L \leq pN_H\Delta \cdot L = \alpha\Delta L$$

  By Chernoff bound, it holds that for any $\epsilon_0 > 0$,

  $$\Pr[\mathbf{Y} > \alpha\Delta L + \epsilon_0 L] \leq \exp(-\Omega(L)) = \exp(-\Omega(\alpha t))$$

  More concretely, the inequality above arises from the Chernoff bound (see Appendix A); there are 2 cases:

  - If $\delta := \frac{\epsilon_0}{\alpha\Delta} < 1$, we have that $\Pr[\mathbf{Y} > \alpha\Delta L + \epsilon_0 L] \leq \exp(-\delta^2\alpha\Delta L/3) = \exp(\frac{\epsilon_0^2 L}{3\alpha\Delta})$. Thus the above inequality follows from the fact that $\epsilon_0 < \alpha\Delta < 1$.
  - If $\delta := \frac{\epsilon_0}{\alpha\Delta} \geq 1$, we have that $\Pr[\mathbf{Y} > \alpha\Delta L + \epsilon_0 L] \leq \exp(-\delta\alpha\Delta L/3) = \exp(-\epsilon_0 L/3)$.

- Let $\mathbf{Z}_i = 1$ iff before the i-th heads in the honest coin sequence during $[t_0, t_1]$, there exists a heads in the previous $N_H\Delta$ coin flips. Similarly as before, all of the $\mathbf{Z}_i$'s are independent. Let $\mathbf{Z} := \sum_{i=1}^{L} \mathbf{Z}_i$. We have that

$$\mathbf{E}[\mathbf{Z}] \leq (1-(1-p)^{N_H\Delta}) \cdot L \leq pN_H\Delta \cdot L = \alpha\Delta L$$

  By the Chernoff bound, it holds that for any $\epsilon_0 > 0$,

$$\Pr[\mathbf{Z} > \alpha\Delta L + \epsilon_0 L] \leq \exp(-\Omega(L)) = \exp(-\Omega(\alpha t))$$

- Observe that for any view,

$$\mathbf{C}(\mathsf{view})[t_0 : t_1] \geq \mathbf{X}(\mathsf{view}) - \mathbf{Y}(\mathsf{view}) - \mathbf{Z}(\mathsf{view})$$

  Recall that our compliance rule implies that $\alpha\Delta \leq pN\Delta < \frac{1}{2}$. For any view where the aforementioned relevant bad events do not happen, we have that for any $\eta > 0$, there exist sufficiently small positive constants ϵ_0 and ϵ such that the following holds:

$$\begin{aligned}
&\mathbf{X} - \mathbf{Y} - \mathbf{Z} \geq (1 - 2\alpha\Delta - 2\epsilon_0)L \\
=&(1 - 2\alpha\Delta - 2\epsilon_0) \cdot (1-\epsilon) \cdot \alpha t \geq (1-\eta)(1-2\alpha\Delta) \cdot \alpha t \\
\geq&(1-\eta)(1-2pN\Delta) \cdot \alpha t
\end{aligned}$$

  The proof concludes by observing that there are at most $\exp(-\Omega(\alpha t))$ fraction of bad views that we could have ignored in the above.

□

The above lemma bounds the number of convergence opportunities for any fixed window. By taking a union bound, we can conclude that except for a negligible fraction of bad views, in all good views, it must hold that any sufficiently long window has many convergence opportunities.

5.7 Chain Growth Lower Bound

To prove the chain growth lower bound, we observe that for any view, whenever there is a convergence opportunity, the shortest honest chain must grow by at least 1 (see Fact 1). Since earlier, we proved that except with negligible probability over the choice of view, there are many convergence opportunities, it naturally follows that honest chains must grow rapidly. We now formalize this intuition.

Fact 1. For any view, any t_0, any $t_1 \geq t_0$, and any honest chains chain^{t_0} and chain^{t_1} in rounds t_0 and t_1 respectively, it holds that

$$\mathbf{C}(\mathsf{view})[t_0 + \Delta : t_1 - \Delta] \leq |\mathsf{chain}^{t_1}| - |\mathsf{chain}^{t_0}|$$

Proof. By definition, if t is a convergence opportunity in view, then the shortest honest chain at the end of round $t + \Delta$ must be longer than the longest honest chain at the beginning of round $t - \Delta$. The remainder of the proof is straightforward. □

Lemma 2 (Chain growth lower bound)**.** *For any positive constants ϵ_0, ϵ', except with negligible probability over the choice of* view*, the following holds: for any t_0 and any $t \geq \frac{\epsilon_0 \kappa}{\alpha}$, let* chain^{t_0} *be any honest chain at time t_0 and let* chain^{t_0+t} *be any honest chain at time $t_0 + t$, then*

$$|\mathsf{chain}^{t_0+t}| - |\mathsf{chain}^{t_0}| \geq (1-\epsilon')(1-2pN\Delta)\alpha t$$

Proof. It suffices to prove the above lemma for any fixed t_0 and t since if so, we can take a union bound over the polynomially many choices of t_0 and t. Ignore the negligible fraction of views where bad events pertaining to Lemma 1 take place. For every remaining good view, due to Fact 1 and Lemma 1, it holds that for every positive constant ϵ', there is a sufficiently small positive constant ϵ such that for sufficiently large κ,

$$\begin{aligned}
&|\mathsf{chain}^{t_0+t}| - |\mathsf{chain}^{t_0}| \\
>&(1-\epsilon)(1-2pN\Delta)\alpha(t-2\Delta) \\
=&(1-\epsilon)(1-2pN\Delta)\alpha t - 2(1-\epsilon)(1-2pN\Delta)\alpha\Delta \\
\geq&(1-\epsilon')(1-2pN\Delta)\alpha t
\end{aligned}$$

where the last inequality is due to the fact $\alpha\Delta < 2pN\Delta < 1$, and moreover $\alpha t = \Theta(\kappa)$. □

5.8 Chain Quality

Intuitively, we will prove chain quality by comparing the number of adversarially mined blocks with the honest chain growth lower bound. If corrupt nodes mine fewer blocks than the minimum honest chain growth, we can thus conclude that there cannot be too many corrupt blocks in an honest node's chain. We formalize this intuition below. Below, if an honest node called $\mathcal{F}_{\text{tree}}.\texttt{extend}(\mathsf{chain}[:-2], \mathsf{chain}[-1])$ and the query was successful, we say that $\mathsf{chain}[-1]$ is mined by an honest node (or an honest block). Otherwise, we say that $\mathsf{chain}[-1]$ is mined by the adversary (or is an adversarial block).

Fact 2 (Total block upper bound)**.** For any positive constants ϵ, ϵ', except with negligible probability over the choice of view, the following holds: for any r and t such that $Np(t-r) \geq \epsilon'\kappa$, the total number of blocks successfully mined during $(r, t]$ by all nodes (honest and corrupt alike) is upper bounded by $(1+\epsilon)Np(t-r)$.

Proof. For any fixed choice of r and t, as long as $Np(t-r) \geq \epsilon'\kappa$, the above statement holds by a straightforward application of the Chernoff bound. The fact then holds by applying a union bound over all possible choices of r and t. □

Upper bound on adversarial blocks. Given a view, let $\mathbf{A}(\mathsf{view})[t_0 : t_1]$ denote the number of blocks mined by corrupt nodes during the window $[t_0 : t_1]$. Let $\mathbf{A}^t(\mathsf{view})$ denote the *maximum* number of adversarially mined blocks in any t-sized window in view.

Fact 3 (Upper bound on adversarially mined blocks)**.** For any positive constant ϵ_0, for any constant $0 < \epsilon < 1$, except with negligible probability over the choice of view, the following holds: for any $t \geq \frac{\epsilon_0 \kappa}{\beta}$, $\mathbf{A}^t(\mathsf{view}) \leq (1+\epsilon)\beta t$.

Proof. It suffices to prove that for any fixed t_0, for any positive constant ϵ, except with negligible probability, it holds that $\mathbf{A}(\mathsf{view})[t_0 : t_0 + t] \leq (1+\epsilon)\beta t$ — if we can show this, the rest of the proof follows by taking a union bound over the choice of t_0. To prove the above for any fixed t_0, it suffices to apply the Chernoff bound in a straightforward manner. □

Lemma 3 (Chain quality)**.** *For any positive constant ϵ_0, ϵ, except with negligible probability over the choice of* view*, the following holds for $\mu := 1 - \frac{1+\epsilon}{1+\phi}$: for any honest* chain *in* view *and any consecutive $T \geq \epsilon_0 \kappa$ blocks* $\mathsf{chain}[j+1..j+T]$ *blocks in* chain*, at least μ fraction of these T blocks are mined by honest nodes.*

Proof. Let r be any round, let i be any node honest at $r \leq |\mathsf{view}|$. Consider an arbitrary honest chain $\mathsf{chain} := \mathsf{chain}_i^r(\mathsf{view})$, and an arbitrary sequence of T blocks $\mathsf{chain}[j+1..j+T] \subset \mathsf{chain}_i^r$, such that $\mathsf{chain}[j]$ is not adversarial (either an honest block or genesis); and $\mathsf{chain}[j+T+1]$ is not adversarial either (either an honest block or $\mathsf{chain}[j+T]$ is end of chain_i^r). Note that if a sequence of blocks is not sandwiched between two honest blocks (including genesis or end of chain), we can always expand the sequence to the left and right to find a maximal sequence sandwiched by honest blocks (including genesis or end of chain). Such an expansion will only worsen chain quality.

As we argued above, without loss of generality we may assume that $\mathsf{chain}[j+1..j+T]$ is sandwiched between two honest blocks (or genesis/end-of-chain). By definition of the ideal-world protocol, all blocks in $\mathsf{chain}[j+1..j+T]$ must be mined between r' and $r'+t$, where r' denotes the round in which the honest (or genesis) block $\mathsf{chain}[j]$ was mined, and $r'+t$ denotes the round in which $\mathsf{chain}[j+T+1]$ is mined (or let $r'+t := r$ if $\mathsf{chain}[j+T]$ is end of chain_i^r).

We ignore the negligible fraction of bad views where bad events related to chain growth lower bound, adversarial block upper bound, or total block upper bound take place.

- Now, due to chain growth lower bound, for any positive constant ϵ_0, we have that

$$t < \frac{T}{(1-\epsilon_0)(1-2pN\Delta)\alpha}$$

- Due to total block upper bound (Fact 2), it holds that $t \geq \Theta(\frac{\kappa}{Np})$. Due to the adversarial block upper bound (Fact 3), for any positive constant $\epsilon'' > 0$, there exist sufficiently small positive constants ϵ' and ϵ_0, such that

$$\begin{aligned}
&\mathbf{A}[r' : r'+t] \leq \mathbf{A}\left[r' : r' + \frac{T}{(1-\epsilon_0)(1-2pN\Delta)\alpha}\right] \\
\leq &\frac{(1+\epsilon')\beta T}{(1-\epsilon_0)(1-2pN\Delta)\alpha} \leq \frac{(1+\epsilon')(1-2pN\Delta)T}{(1-\epsilon_0)(1-2pN\Delta)(1+\phi)} \\
\leq &\frac{(1+\epsilon'')T}{1+\phi}
\end{aligned}$$

- Therefore, the fraction of honest blocks in this length T sequence is lower bounded by

$$1 - \frac{1+\epsilon''}{1+\phi}$$

□

5.9 Consistency

Fact 4 (Adversary must expend work to deny a convergence opportunity)**.** Let $t \leq |\mathsf{view}| - \Delta$ denote a convergence opportunity in view in which a single honest node mines a block denoted B^* at length ℓ. It holds that for any honest chain chain at time $t' \geq t + \Delta$ in view, chain must be at least ℓ in length; moreover, if $\mathsf{chain}[\ell]$ is either mined by a corrupt node, or $\mathsf{chain}[\ell] = \mathsf{B}^*$.

Proof. By the definition of a convergence opportunity, no honest node will mine blocks at length ℓ after $t+\Delta$, and no honest node could have mined a block at length ℓ before $t-\Delta$ since otherwise the honest block mined during the convergence opportunity must be at length at least $\ell+1$. Finally, since B^* is the only honest block mined during $[t-\Delta, t+\Delta]$, it holds that there is no other honest block at length ℓ in view. □

Lemma 4 (Consistency[2])**.** *For any positive constant ϵ, except with negligible probability over the choice of view, the following holds: for any round r and any round $t \geq r$ in view, let chain^r be any honest chain in round r and let chain^t be any honest chain in round t; then it must hold that*

$$\mathsf{chain}^r[:-\epsilon\kappa] \prec \mathsf{chain}^t$$

Proof. Suppose for the sake of reaching a contradiction that $\mathsf{chain}^r[:-\kappa]$ is not a prefix of chain^t. Let $\mathsf{chain}^r[:-\ell]$ be the longest common prefix of chain^r and chain^t where $\ell > \kappa$. Let $\mathsf{chain}^r[:i] \preceq \mathsf{chain}^r[:-\ell]$ be the longest prefix that ends at an honestly mined block, i.e., $\mathsf{chain}^r[i]$ is the first honest block to the left of $\mathsf{chain}^r[-\ell]$ (and including $\mathsf{chain}^r[-\ell]$); and let $s-1$ be the round in which $\mathsf{chain}^r[i]$ was mined. It holds that all blocks in $\mathsf{chain}^r[i+1:]$ and $\mathsf{chain}^t[i+1:]$ must be mined in or after round s.

Henceforth, let $\tau := r - s$. By total block upper bound, it must be that $\tau > \frac{\kappa}{2pn}$, which is large enough to make sure that our failure probabilities later will be negligibly small.

Case 1: when $t-r \leq \frac{\phi}{2}\tau$. Observe that all convergence opportunities that come in or after round s must be at length greater than i and moreover they must be at different lengths. Combining this observation and Fact 4, it must be the case that $\mathbf{C}[s : r-\Delta] \leq \mathbf{A}[s : t]$, since otherwise, there must be an honest block B mined during a convergence opportunity between $[s, r-\Delta]$, and B must appear in both chain^r and chain^t. Below we prove that except with negligible probability over the choice of the execution, it must be that $\mathbf{C}[s : r-\Delta] > \mathbf{A}[s : t]$ — if we can do so, then we reach a contradiction, and thus we can conclude the proof.

Below we ignore the negligible probability mass of executions where relevant bad events take place. By Lemma 1, for any positive constant ϵ_c, it holds that[3]

$$\mathbf{C}[s : r-\Delta] > (1-\epsilon_c)(1-2pn\Delta)\alpha(\tau-\Delta)$$

By Fact 3, for any positive constant ϵ_a, it holds that

$$\mathbf{A}[s : t] < (1+\epsilon_a)\beta \cdot (1+\frac{\phi}{2})\tau$$

Thus for any positive constants ϕ, and as long as $0 < 2pn\Delta < 0.5$, there exist sufficiently small constants $\epsilon_c, \epsilon_a, \epsilon_1$ such that the following holds for sufficiently large κ:

$$\mathbf{C}[s : r-\Delta] > (1-\epsilon_c)(1-\nu)\alpha(\tau-\Delta) \tag{2}$$

$$> (1-\epsilon_1)(1-\nu)\alpha\tau \tag{3}$$

$$> (1-\epsilon_1)(1+\phi)\beta\tau \tag{4}$$

$$> (1+\epsilon_a)\beta \cdot (1+\frac{\phi}{2})\tau > \mathbf{A}[s : t] \tag{5}$$

where (3) stems from the fact that $\alpha\tau = \Theta(\kappa)$ and $\alpha\Delta = O(1)$; and (4) stems from our parameter choices.

[2]Elaine Shi gratefully acknowledges the outstanding students of CMU 15827-E / 18847-D (Fall 2020) for helping correct bugs in the proof.

[3]The choice of ϵ_c affects the choice of the negligible function.

Case 2: when $t - r > \frac{\phi}{2}\tau$. So far we proved that except with negligible probability,

$$\mathbf{C}[s : r - \Delta] \geq \mathbf{A}[s : s + (1 + \frac{\phi}{2})\tau]$$

Therefore, there must be some length ℓ^* corresponding to the length of a convergence opportunity during $[s, r - \Delta]$, such that if $\mathsf{chain}^r[\ell^*]$ or $\mathsf{chain}^t[\ell^*]$ is mined by a corrupt node, it must be mined after $r + (1 + \frac{\phi}{2})\tau$. This means that $\mathsf{chain}^r[\ell^*]$ cannot be mined by a corrupt node. By Fact 4, $\mathsf{chain}^t[\ell^*]$ must be mined by a corrupt node after $r + \frac{\phi}{2}\tau$, since otherwise it must be that $\mathsf{chain}^t[\ell^*] = \mathsf{chain}^r[\ell^*]$.

However, by chain growth lower bound, for an arbitrarily small consant ϵ, except with negligible probability, in round $r + \frac{\phi}{2}\tau$, even the shortest honest chain must have length more than

$$\widetilde{\ell} := \ell^* + (1 - \epsilon) \cdot \frac{1}{2}\phi\tau \cdot (1 - 2pn\Delta)\alpha$$

This means that all blocks in $\mathsf{chain}^t[\ell^* : \widetilde{\ell}]$ must be mined by corrupt nodes. However, due to chain quality, this cannot happen except with negligible probability.

□

5.10 Chain Growth Upper Bound

Lemma 5 (Chain growth upper bound)**.** *For any positive constants ϵ_0, ϵ except with negligible probability over the choice of* view*, the following holds: for any t_0 and any $t \geq \frac{\epsilon_0\kappa}{\alpha}$, let chain^{t_0} be any honest chain at time t_0 and let chain^{t_0+t} be any honest chain at time $t_0 + t$, then*

$$|\mathsf{chain}^{t_0+t}| - |\mathsf{chain}^{t_0}| \leq (1 + \epsilon)Npt$$

Proof. Suppose for the sake of contradiction that there exist positive constants ϵ and ϵ', such that for a polynomial fraction of views, we can find t_0 and $t \geq \frac{\epsilon'\kappa}{\alpha}$ such that for some honest chains chain^{t_0} and chain^{t_0+t} at times t_0 and $t_0 + t$ respectively, $|\mathsf{chain}^{t_0+t}| - |\mathsf{chain}^{t_0}| > (1 + \epsilon)Npt$. Among these polynomial fraction of views, for any sufficiently small constant η' except for a negligible fraction of views where relevant bad events happen, the following hold for the remaining views.

Let chain_0 denote the shortest honest chain at time t_0 and let chain_1 denote the longest honest chain at time $t_1 = t_0 + t$. By our assumption, it holds that $|\mathsf{chain}_1| - |\mathsf{chain}_0| > (1 + \epsilon)Npt$. Let $\ell_0 := |\mathsf{chain}_0|$. Suppose that $\mathsf{chain}_1[\ell_0]$ and $\mathsf{chain}_1[-1]$ are mined in rounds r_0 and r_1 respectively. By definition of the honest protocol, it must hold that $r_1 \leq t_1$.

- Notice that all the $(1 + \epsilon)Npt$ or more blocks in $\mathsf{chain}_1[\ell_0 :]$ must be mined during $[r_0, r_1]$. By total block upper bound, there must exist a positive constant $\eta \geq \epsilon$ such that $r_1 - r_0 \geq (1 + \eta)t$, since otherwise, all these $(1 + \epsilon)Npt$ or more blocks in $\mathsf{chain}_1[\ell_0 :]$ cannot all be mined during $[r_0, r_1]$.
- Since $r_1 \leq t_0 + t$, it must hold that $r_0 \leq t_0 + t - (1 + \eta)t \leq t_0 - \eta t \leq t_0 - \epsilon t$.
- By chain quality, there must be an honest block in $\mathsf{chain}_1[\ell_0 - \eta'\kappa : \ell_0]$.
- The above means that there exists an honest node whose chain length is at least $\ell_0 - \eta'\kappa$ at some time $r' < r_0$. We also know that there is an honest node whose chain length is ℓ_0 at t_0 — this means that the minimum honest chain growth between $[r_0, t_0]$ is at most $\eta'\kappa$. But recall that at least ϵt time has elapsed between r_0 and t_0 — by chain growth lower bound, for a sufficiently small constant η' (w.r.t. ϵ and ϵ'), this cannot happen.

□

6 Survey of Recent Results

Despite its enormous success, Nakamoto's blockchain has several well-known and widely criticized drawbacks [CDE+16]. First, due to a selfish mining attack [ES14], it is well-known that the selfish miners can harm the fairness of the eco-system. Second, the transaction confirmation is very slow, and requires waiting for multiple blocks. Third, proofs-of-work are enormously wasteful. We now survey recent results that aim to address these well-known drawbacks. We also discuss the feasibility of permissionless consensus when the number of players can vary over time.

6.1 Thwarting Selfish Miners and Achieving Fairness

As is well-known, since participants need to expend costly computation in Bitcoin mining, it is important to create incentives for participation. Therefore Bitcoin issues rewards to those who mine blocks where the rewards come from two sources: a miner collects a *block reward* for having mined a block; additionally it also collects *transaction fees* paid by transactions contained in the block.

As stated in Theorem 5, for sufficiently small choices p and ϵ, Nakamoto's blockchain achieves approximately $1 - \frac{\rho}{1-\rho}$ chain quality where ρ denotes the fraction of corrupt mining power. For example, suppose that the adversary wields roughly $\rho = \frac{1}{3}$ mining power (and additionally controls network delivery), then the adversary controls roughly $\frac{1}{2}$ of the blocks! Moreover, if the adversary wields close to $\frac{1}{2}$ mining power, then it can own almost all of the blocks. This is alarming: it says that a minority coalition can earn an unfair proportion of rewards and thus harm the remaining honest participants.

Selfish mining and degraded chain quality. What leads to such unfairness is a well-known *selfish mining* attack [ES14]: when a corrupt node mines a block — let chain^* denote the adversary's private chain at this point: the adversary need not release chain^* immediately as honest nodes would have. Instead, the adversary withholds chain^* until it observes that some honest node has mined a chain denoted chain_H of equal length — at this moment, the adversary performs a rushing attack and delivers chain^* ahead of chain_H to all honest nodes. Now, all honest nodes will extend the adversary's fork chain^*, and thus the honest work expended in mining chain_H has been wasted. Since the adversary can perform this attack for every block it mines, effectively out of the $1 - \rho$ fraction of honest mining power, the adversary can erase a ρ fraction. This explains why Nakamoto's chain quality is only $1 - \frac{\rho}{1-\rho}$ whereas "ideal chain quality" would have been $1 - \rho$.

Achieving fairness. A natural question arises: *can we achieve fairness in permissionless consensus and how do we define fairness?* Our recent work Fruitchains [PS17a] answered this question. Specifically, we show that there is a blockchain protocol (called Fruitchains) that achieves $1-(1-\epsilon)\rho$ chain quality for an arbitrarily small constant ϵ (i.e., almost ideal chain quality) where ρ denotes the corrupt fraction. Interestingly, Fruitchains makes use of Nakamoto's blockchain in a blackbox manner. Conceptually, Fruitchains has two separate mining processes — in reality the two mining processes are piggybacked on top of each other such that no additional computation is required — one for mining *blocks*, and one for mining *fruits*. In Fruitchains, the blocks contain fruits and the fruits in turn contain transactions. We leverage the underlying blockchain's liveness to ensure

that no honest work in mining fruits can be erased by the adversary, thus attaining almost perfect "fruit quality" — thus in Fruitchains, one should think of the "fruits" as the new "blocks". In Fruitchains, we also define a new notion of fairness and describe a mechanism for distributing rewards and transaction fees, such that the resulting protocol achieves a coalition-resilient ϵ-Nash equilibrium. We refer the reader to our paper [PS17a] for details. The ideas proposed in Fruitchains were later adopted in the context of proof-of-stake applications [KRDO17, DPS16] to achieve incentive compatibility.

6.2 Responsiveness in Permissionless Consensus

Blockchain protocols are often criticized for being slow in transaction confirmation. Our analysis shows that for Nakamoto's blockchain to retain security, the block interval must be set to be a constant factor larger than the network's maximum delay Δ; and moreover, to obtain sufficient security one must wait for multiple blocks for confirmation. A natural question arises: *can we achieve permissionless consensus with low response time?* To understand this question, we must first more precisely define what "low response time" means. A natural definition is due to the elegant work of Attiya et al. [ADLS94] who defined a notion of *responsiveness*. In our context of state machine replication, informally speaking, responsiveness requires transactions get confirmed as fast as the network delivers messages. More precisely, a state machine replication protocol satisfying T_{confirm}-liveness is said to be *responsive* if T_{confirm} is a function only of the actual maximum network delay, and not of the upper bound parameter Δ that the protocol receives as input. By this definition and our reasoning above, Nakamoto's blockchain does not satisfy responsiveness.

In the classical distributed consensus literature, protocols in the partial synchronous or asynchronous models naturally achieve the aforementioned notion of responsiveness since a partially synchronous (or asynchronous) protocol does not have a-priori knowledge of an upper bound Δ of the network's delay. Unfortunately, recall that in Section 3, we showed that in a permissionless model where the number of nodes is uncertain, it is not possible to have a partially synchronous (or asynchronous) permissionless consensus protocol (even when all nodes are honest).

We stress, however, that interestingly, the infeasibility of partial synchrony (i.e., when the protocol does not know any delay upper bound Δ) in the permissionless setting does not rule out the possibility of achieving responsiveness when the protocol knows a network delay upper bound Δ. Specifically, in a couple of recent works [PS17b, PS18], we show the following results (informally stated below):

- Assuming that it takes a short while for an adversary to corrupt and kill honest nodes, then for any $\rho = \frac{1}{3} - \epsilon$ where ϵ is an arbitrarily small constant, there exists a state machine replication protocol in the proof-of-work model (referred to as Hybrid Consensus [PS17b]) that achieves responsive transaction confirmation (after a non-responsive warmup period) and tolerates up to ρ fraction of corrupt nodes.

- Moreover, hybrid consensus is almost tight in terms of resilience: we also show that even assuming the existence of a proof-of-work oracle, there is no responsive state machine replication protocol that can tolerate $\frac{1}{3}$ or more fraction of corrupt nodes — even when $(\mathcal{A}, \mathcal{Z})$ must respect early spawning and static corruptions [PS17b].

- Finally, in a more recent work called Thunderella [PS18], we show that protocols that aim to achieve responsiveness only in the "optimistic" case need not be subject to the aforementioned $\frac{1}{3}$ lower bound. In particular, we show that there exists a permissionless consensus protocol Π that provides consistency and "slower blockchain performance" as long as the adversary controls

only minority and it takes a while to corrupt nodes; however, when a larger fraction of nodes are honest and stick around for some time before dropping offline, the protocol achieves responsive transaction confirmation.

6.3 What Can We Learn for Permissioned Consensus?

From a mathematical perspective [GKL15, PSS17, PS17c], the way blockchains reach consensus fundamentally departs from classical consensus. The following questions naturally arise: *Can we apply the ideas behind blockchains to solve the classical, permissioned consensus problem? Can we remove the wasteful proof-of-work in a permissioned setting?* Blockchain protocols are commonly believed to be more robust than classical consensus. *Exactly what robustness properties does a blockchain-style protocol offer that classical consensus does not?*

In a recent work [PS17c], we were the first to phrase and explore these questions. We propose the Sleepy consensus protocol [PS17c], and show that one can apply core ideas behind blockchains to solve consensus in a classical, permissioned setting while dispensing with expensive proofs-of-work. In the permissioned setting, we also observe that blockchain-style protocols (without proofs-of-work) can reach consensus even when 1) the number nodes that will show up is unknown a-priori; 2) the number of nodes in each round can change rapidly; and 3) the nodes that show up in adjacent rounds can be completely disjoint — as long as among the nodes who do show up in each round, majority are honest. We refer to such a model with sporadic participation as the "sleepy" model — interestingly, no classical consensus protocol can reach consensus in such a "sleepy" model, even when we are guaranteed that 99% of those who show up are honest. Note that in the "sleepy" model, nodes that are honest but have crashed are not treated as faulty — unlike the modeling approach in classical distributed computing where crashes typically count towards the corruption budget.

Several recent works including Algorand [CM16], Snow White [DPS16], and Ouroboros [KRDO17] showed how to leverage a permissioned, synchronous state machine replication protocol to build a proof-of-stake application. In a proof-of-stake consensus protocol, roughly speaking, users have voting power proportional to their amount of stake in the system. Among various differences in protocol details, these works differ primarily in the choice of the underlying synchronous state machine replication protocol. Algorand [CM16] constructs a new synchronous state machine replication protocol; Ourosboros [KRDO17] constructs a blockchain-style protocol without proof-of-work but their protocol achieves only classical properties similar to those of Algorand [CM16]; and finally Snow White adopts a blockchain-style protocol like Sleepy consensus [PS17c] that is tolerant of "sleepiness".

6.4 Permissionless Consensus with Varying Number of Players

For conceptual simplicity, most of our paper has focused on the case that the number of players N stays fixed throughout the execution. However, as we point out at the beginning of the paper, in a truly permissionless model the number of nodes can also vary over time. Thus, a natural question arises: *can we achieve permissionless consensus when the number of nodes varies over time?*

A couple recent works [GKL,CEM+17] have (partially) answered this question in the affirmative: these works show that state machine replication is indeed possible in a permissionless model where the number of nodes may vary over time, as long as 1) the protocol knows an upper bound on the initial number of players; and 2) the number of nodes does not "abruptly" increase (we refer the reader to Chan et al. [CEM+17] for a more precise technical characterization of "abruptly"). In particular, Garay et al. [GKL] show a feasibility result assuming a constrained adversary that

does not delay messages and must commit to the number of nodes in each round upfront — these restrictions were later removed in the work by Chan et al. [CEM+17] who also presented a proof with tighter parameters.

Acknowledgments

We gratefully thank Hubert Chan and the students of CS6432, especially Abhinav Agarwal, Yue Guo, and Andrew Morgan for helping debug and improve the proofs. This work is supported in part by NSF grants CNS-1217821, CNS-1314857, CNS-1514261, CNS-1544613, CNS-1561209, CNS-1601879, CNS-1617676, AFOSR Award FA9550-15-1-0262, an Office of Naval Research Young Investigator Program Award, a Microsoft Faculty Fellowship, a Packard Fellowship, a Sloan Fellowship, Google Faculty Research Awards, a VMWare Research Award, and a Baidu Research Award.

References

[AD15] Marcin Andrychowicz and Stefan Dziembowski. Pow-based distributed cryptography with no trusted setup. In *CRYPTO*, pages 379–399, 2015.

[ADLS94] Hagit Attiya, Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. *J. ACM*, 41(1):122–152, 1994.

[BBSU12] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better—how to make bitcoin a better currency. In *Financial cryptography and data security*, pages 399–414. Springer, 2012.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to Use Bitcoin to Design Fair Protocols. In *CRYPTO*, 2014.

[btP] `http://www.btproof.site/`.

[CDE+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gun Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains (a position paper). In *Bitcoin Workshop*, 2016.

[CEM+17] T-H. Hubert Chan, Naomi Ephraim, Antonio Marcedone, Andrew Morgan, Rafael Pass, and Elaine Shi. Blockchain with varying number of players. Manuscript, 2017.

[CF14] Sophia Yakoubov Conner Fromknecht, Dragos Velicanu. A decentralized public key infrastructure with identity retention. Cryptology ePrint Archive, Report 2014/803, 2014. `http://eprint.iacr.org/2014/803`.

[CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, 1999.

[CM16] Jing Chen and Silvio Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 1988.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, 1992.

[DPS16] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *Siam Journal on Computing - SIAMCOMP*, 12(4):656–666, 1983.

[ES14] Ittay Eyal and Emin Gun Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *FC*, 2014.

[FLM85] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In *PODC*, 1985.

[GKL] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. Cryptology ePrint Archive, 2016/1048.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Eurocrypt*, 2015.

[KMS14] Jonathan Katz, Andrew Miller, and Elaine Shi. Pseudonymous secure computation from time-lock puzzles. *IACR Cryptology ePrint Archive*, 2014:857, 2014.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto*, 2017.

[Lam83] Leslie Lamport. The weak byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.

[Lam06] Leslie Lamport. Fast paxos. *Distributed Computing*, 19(2):79–103, 2006.

[lig] Lightning network. `https://lightning.network/`.

[Lit] Litecoin - Open source P2P digital currency. `http://litecoin.org/`.

[Loi14] Andreas Loibl. Namecoin. `namecoin.info`, 2014.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[Pas15] Rafael Pass and abhi shelat. Micropayments for peer-to-peer currencies. In *ACM CCS*, 2015.

[PS17a] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *PODC*, 2017.

[PS17b] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.

[PS17c] Rafael Pass and Elaine Shi. The sleepy model of consensus. In *Asiacrypt*, 2017.

[PS18] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Eurocrypt*, 2018.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Eurocrypt*, 2017.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, 31(4):161–172, August 2001.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *SIGCOMM*, 2001.

[SZ15] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography*, 2015.

[Woo14] Gavin Wood. Ethereum: A secure decentralized transaction ledger. `http://gavwood.com/paper.pdf`, 2014.

A Additional Preliminary: Chernoff Bound

We use the following version of Chernoff bound.

Theorem 6. *Let* $\mathbf{X} := \sum_{i=1}^{n} \mathbf{X}_i$ *where each* $\mathbf{X}_i = 1$ *with probability* p_i *and* $\mathbf{X}_i = 0$ *with probability* $1 - p_i$*. Further, all* $\mathbf{X}_i$*'s are independent. Let* $\mu := \sum_{i=1}^{n} p_i$*. Then, we have the following:*

- **Upper tail:** $\Pr[\mathbf{X} \geq (1+\delta)\mu] \leq \exp(-\frac{\delta \cdot \min\{\delta,1\} \cdot \mu}{3})$ *for all* $\delta > 0$*;*
- **Lower tail:** $\Pr[\mathbf{X} \leq (1-\delta)\mu] \leq \exp(-\frac{\delta^2 \mu}{2})$ *for all* $0 < \delta < 1$*.*