Analysis of the Blockchain Protocol in Asynchronous Networks

Rafael Pass[1(✉)], Lior Seeman[2], and Abhi Shelat[3]

[1] Cornell Tech, New York City, USA
rafael@cornell.edu
[2] Uber, San Francisco, USA
lior.seeman@gmail.com
[3] Northeastern, Boston, USA
abhi@neu.edu

Abstract. Nakamoto's famous *blockchain* protocol enables achieving consensus in a so-called *permissionless setting*—anyone can join (or leave) the protocol execution, and the protocol instructions do not depend on the identities of the players. His ingenious protocol prevents "sybil attacks" (where an adversary spawns any number of new players) by relying on *computational puzzles* (a.k.a. "moderately hard functions") introduced by Dwork and Naor (Crypto'92).

The analysis of the blockchain consensus protocol (a.k.a. Nakamoto consensus) has been a notoriously difficult task. Prior works that analyze it either make the simplifying assumption that network channels are *fully synchronous* (i.e. messages are instantly delivered *without delays*) (Garay *et al.* Eurocrypt'15) or only consider specific attacks (Nakamoto'08; Sampolinsky and Zohar, FinancialCrypt'15); additionally, as far as we know, none of them deal with players joining or leaving the protocol.

In this work we prove that the blockchain consensus mechanism satisfies a strong forms of *consistency* and *liveness* in an *asynchronous network* with adversarial delays that are *a-priori* bounded, within a formal model allowing for adaptive corruption and spawning of new players, assuming that the computational puzzle is modeled as a random oracle. (We complement this result by showing a simple attack against the blockchain protocol in a fully asynchronous setting, showing that the "puzzle-hardness" needs to be appropriately set as a function of the maximum network delay; this attack applies even for static corruption.)

As an independent contribution, we define an abstract blockchain protocol and identify appropriate security properties of such protocols;

R. Pass—Supported in part by NSF Award CNS-1561209, NSF Award CNS-1217821, AFOSR Award FA9550-15-1-0262, a Microsoft Faculty Fellowship, and a Google Faculty Research Award.

L. Seeman—This work was done while Lior was at Cornell Tech and a postdoctoral fellow at the Harvard University Center for Research on Computation and Society, supported by Simons Foundation grant 315783.

A. Shelat—Supported in part by NSF grants 0845811, 0939718, 1565412, a Microsoft Faculty Fellowship, an SAIC Faculty Award, and a Google Faculty Research Award.

J.-S. Coron and J.B. Nielsen (Eds.): EUROCRYPT 2017, Part II, LNCS 10211, pp. 643–673, 2017.
DOI: 10.1007/978-3-319-56614-6_22

we prove that Nakamoto's blockchain protocol satisfies them and that these properties are sufficient for typical applications; we hope that this abstraction may simplify further applications of blockchains.

1 Introduction

Distributed systems have been historically analyzed in a *closed* setting in which both the number of participants in the system, as well as their identities, are common knowledge. A departure from this model started with the design of *peer-to-peer* systems, e.g. with systems such as *Napster* and *Gnutella* for file sharing. The success of those systems led to academically designed systems such as Freenet [CSWH00], CAN [RFH+00], Chord [SMK+01], and Pastry [DR01] which offered redundant file storage, distributed hashing, selection of nearby servers, and hierarchical naming.

A novel aspect of these 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 the protocol instructions do not depend on the identities of the players. As participants may continuously join and leave the system, successful permissionless systems require a fault-tolerant design. Unfortunately, the mentioned systems, while "robust" with respect to measures such as connectivity [DLN02], were not designed to tolerate against adversarial behavior. For example, there were no guarantee that one participant's experience with the system was *consistent* with another's: Two participants requesting the same file may end up receiving different versions *and never know that they did.* At first, one may think that using standard consensus/Byzantine agreement methods (e.g., [CL99, MA05, Lam10, Lam11]) could help overcome this issue. The problem is that such protocols require that a large fraction of the participating players are honest, but in the permissionless setting an attacker can trivially mount a "sybil attack"—it simply spawns players (that it controls) and can thus ensure that it controls a majority of all players. Indeed, Barak *et al.* [BCL+05] prove that this is a fundamental problem with the permissionless model.

Nakamoto's Blockchain. In 2008, Nakamoto [Nak08] proposed his celebrated "blockchain protocol" which overcomes the above-mentioned problems by relying on the idea of computational puzzles—a.k.a. *moderately hard functions* or *proofs of work*—put forth by Dwork and Naor [DN92]. Rather than attempting to provide robustness whenever the majority of the participants are honest (since participants can be easily spawned in the permissionless setting), it attempts to provide robustness as long as a *majority of the computing power* is held by honest participants. It explicitly claims *consistency* properties that are strong enough to support a financial transaction system; indeed, the first application of a blockchain is the Bitcoin digital currency which needs strong properties to prevent fraud and double-spending attacks. A number of follow-up digital currencies [Lit], micro-payment schemes [PS15, PD15], time-stamping [BTP], naming [Nam], fair secure computation [BK14] and secure messaging and PKI

applications [FVY14] are based on the blockchain idea. 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.

The core blockchain protocol (a.k.a. "Nakamoto consensus", or the "Bare-bones blockchain protocol"), roughly speaking, is a method for maintaining a *public*, *immutable* and *ordered* ledger of records (for instance, in the bitcoin application, these records are simply transactions); that is, records can be added to the *end* of the ledger at any time (but only to the end of it); additionally, we are guaranteed that records previously added cannot be removed or reordered and that all honest users have a *consistent view* of the ledger. While standard consensus/Byzantine agreement mechanisms could be used to achieve such an immutable ordered sequence of records, the amazing aspect of Nakamoto's consensus mechanism is that it functions in a fully permissionless setting.

Roughly speaking, in his protocol each participant maintains its own local "chain" of "blocks" of records/messages—called the *blockchain*. Each block consist of a triple $(h_{-1}, \eta, \mathsf{m})$ where h_{-1} is a pointer to the previous block in chain, m is the record component of the block, and η is a "proof-of-work"—a solution to a computational puzzle that is derived from the pair (h_{-1}, m). The proof of work can be thought of as a "key-less digital signature" on the whole blockchain up until this point.

Concretely, Nakamoto's protocol is parametrized by a parameter p—which we refer to as the *mining hardness parameter*, and a proof-of-work is deemed valid if η is a string such that $\mathsf{H}(h_{-1}, \eta, \mathsf{m}) < D_p$, where H is a hash function (modeled as a random oracle) and D_p is set so that the probability that an input satisfies the relation is less than p. In practice, the hardness parameter p is adaptively modified through some external process to incorporate an estimate of the number of participants in the system and the network delays; we shall return to the choice of p later. At any point of the protocol execution, each participant attempts to increase the length of its own chain by "mining" for a new block: upon receiving some record m, it picks a random η and checks whether η is a valid proof of work w.r.t. m and h_{-1}, where h_{-1} is a pointer to the last block of its current chain; if so, it extends is own local chain and broadcast it to the all the other participants (the broadcast takes places through some gossip protocol, which we do not discuss here). Whenever a participant receives a chain that is longer than its own local chain, it replaces its own chain with the longer one.

The fundamental question with such an approach is whether honest participants eventually end up with the same longest chain of blocks, and thus, the same ordered list of records, or whether the system devolves into a state where participants have *inconsistent* local chains.

1.1 Does Nakamoto's Protocol Achieve Consistency?

Requiring that all participants agree on the *whole* chain is a too strong consistency requirement if the protocol is executed on a network with message delays (as Nakamoto's protocol is intended to be)—for instance, some players may

have received the last block whereas other have not. Rather, as discussed by Nakamoto [Nak08], the appropriate notion of consistency for the blockchain—which we refer to as *T-consistency*—should require that honest players agree on the current chain, *except* for potentially a small number, T, of *unconfirmed* blocks at the end of the chain. If we can show this property holds except with exponentially small probability in T, honest parties are guaranteed that for a sufficiently large choice of T (except with tiny probability), *confirmed* blocks will never be lost from the chain (which is the property needed for all the above-mentioned applications; for instance, in bitcoin, it ensures that players cannot double-spend money).

Nakamoto provides an initial analysis of consistency assuming that the adversary only mounts a particular attack strategy (namely, an attacker tries to generate a chain faster than the honest players); for instance, his analysis does not consider more sophisticated attack strategies where the adversary may attempt to "split the players" and have them work on different chains.

A beautiful recent work by Garay, Kiayas and Leonardos [GKL15] provides a more formal model for studying Nakamoto's blockchain protocol; their analysis, however, only considers a *synchronous network* with a rushing adversary—that is, messages sent in a particular round arrive in the next round *without any delays*, but the adversary sees all messages sent by honest parties before having to send its own message. In this model, they demonstrate that the blockchain protocol satisfies consistency (under appropriate assumptions on the mining hardness and the relative computational power held by the attacker), in a setting with a fixed number of players (but the protocol is not aware of the exact number of players).

Assuming a synchronous network, however, is a very strong, possibly unrealistic assumption; indeed, Nakamoto's protocol is explicitly designed to work in a network *with message delays*, and indeed is executed on such a network (i.e., the Internet).

The Power of Network Delays. Consequently, we are interested in analyzing to what extent the blockchain protocol satisfies consistency in the more realistic setting of an *asynchronous network* in which an adversary controls the scheduling/delivery of messages between honest parties. As we observe (and formally prove in Theorem 10), in a *fully* asynchronous setting, where an adversary can arbitrarily delay messages, consistency cannot be satisfied: an adversary controlling a small percentage of the computational power can simply delay messages from honest parties for sufficiently long to ensure that the adversary can find its own chain (containing *any* set of records it desires) which is longer than the chain held by all honest players, and consequently it can make the honest players switch to the adversarial chain at any point. In fact, our attack works even in the setting of *partial synchrony* (see e.g. [DLS88]) where there is an *a-priori* bound Δ on the network latency (that is, the adversary may arbitrary delay messages as long as it delivers them within time Δ), as long as the mining

hardness parameter p exceeds[1] $\frac{1}{\rho n \Delta}$, where ρ is the fraction of the computational power held by the adversary and n is the number of players. Indeed, Decker and Wattenhofer [DW13] already experimentally observed that increasing the networks delays in Nakamoto's protocol leads to increased forks, and they noted (through heuristic calculations) that an attacker could use these delays to violate consistency with an attack that requires less than 50% of the mining power.

Motivated by the work by Decker and Wattenhofer, an elegant work by Sompolinsky and Zohar [SZ15] provides some initial analysis of the blockchain protocol even in a network with (bounded) delays. They show how to extend Nakamoto's analysis to deal with (bounded) delays, but again (just like Nakamoto) they only consider particular attack strategies—e.g., they do not consider "block-withholding or pre-mining attacks" where the attacker withholds blocks for later use [mtg10,ES14]; furthermore, their analysis only shows that consistency holds in the limit (when T goes to infinity), and consequently their bounds (even for the restricted attacker setting) are not useful for applications.

This leaves open the question of analyzing Nakamoto's blockchain protocol—or in fact *any* consensus protocol in the permissionless setting—with respect to *arbitrary* attack strategies in networks with Δ-bounded delays.

> *Does Nakamoto's blockchain protocol satisfy consistency when executed in asynchronous networks with Δ-bounded delays?*

As mentioned above, Garay *et al.* provide a positive answer for the special case when $\Delta = 1$ (i.e., messages are delivered in the next time step[2]), and Sampolinsky and Zohar show that certain (natural, but restricted) strategies cannot be employed to break consistency of Nakamoto's protocol (in the limit) in Δ-bounded delay networks.

Let us highlight why dealing with network delays in the "proof-of-work" setting (where we assume that a majority of the computing power is honest) is significantly more challenging than in the standard permissioned setting. In the standard model, any synchronous protocol can be turned into a protocol that is secure also in Δ-delay networks by simply requiring that all honest players always *wait* (without doing anything) for Δ time steps before responding to any message, effectively emulating synchronous rounds. This approach completely fails in the proof-of-work setting—the adversary can now increase its computational resources by a factor Δ (since it can try to solve puzzles when the honest players are waiting).

1.2 Main Results

In this paper, we resolve the above-mentioned problem and demonstrate that (assuming puzzles are modeled as random oracles) Nakamoto's protocol satisfies

[1] Recall that a *larger* hardness parameter means that it is easier to find a block.

[2] Alternatively, one way to interpret the result of Garay *et al.*is that it shows consistency of Nakamoto's protocol also with Δ delays, but with a *particular delay structure* where time is divided into intervals of length Δ, and any message sent within an interval is delayed to the end of it.

consistency (under appropriate assumptions on the mining hardness and the relative computational power held by the attacker) also in networks with message delays. We emphasize that our analysis is not just a combination of the techniques/ideas from [GKL15] and [SZ15]—in fact, the bulk of our proof consists of dealing with the attack strategies which are omitted from the analysis in [SZ15], and dealing with them requires us to consider an altogether different proof technique. Additionally, our analysis considers adaptive corruption and spawning of new players (i.e., new players joining); as far as we know, it is the first analysis to formally deal with spawning of new players (which is a crucial desiderata of the blockchain protocol).

A Consistency Theorem with Delays. We provide a rough overview of our model and consistency theorem. Consider Nakamoto's protocol with mining-hardness p (that is, a single random oracle query is successful "in mining" with probability p), and consider an execution with n players, each of them with identical computing power—we assume the protocol proceeds in rounds (timesteps), and in each round each player gets a single random oracle query and the adversary controlling a ρ fraction of the players gets ρn random oracles queries (as in [GKL15], the honest players need to make their queries in parallel, but we allow the adversary to makes the queries sequentially). Let $\alpha = 1-(1-p)^{(1-\rho)n}$ be the probability that some honest player succeeds in solving a puzzle in one round, and let $\beta = \rho np$ be the expected number of blocks that an attacker can mine in a round. When $p \ll 1/n$ (which is the case considered in practice), we have that $\alpha \approx p(1-\rho)n$ and thus $\frac{\alpha}{\beta} \approx \frac{1-\rho}{\rho}$.

Theorem 1. *Assume there exists some $\delta > 0$ such that*

$$\alpha(1-(2\Delta+2)\alpha) \geq (1+\delta)\beta.$$

Then, except with exponentially small probability (in T), Nakamoto's protocol satisfies T-consistency in the random oracle model, assuming the network's latency is bounded by Δ.

As a consequence we have that as long as $\rho < \frac{1}{2}$ (i.e., the adversary controls less than half of the computational power), for every Δ there exists some (sufficiently small) p, such that Nakamoto's protocol satisfies consistency. (Note that as mentioned above, if $p > \frac{1}{\rho n \Delta}$, Nakamoto's protocol fails to satisfy consistency.)

1.3 What Is a Blockchain?

As an independent contribution, we formally define an *abstract* notion of a blockchain (as opposed to *the* blockchain protocol proposed by Nakamoto) and put forward desired security properties of such a blockchain. We believe that having such a notion will (a) simplify applications of blockchains (as we can ignore the implementation details of the blockchain protocol) and (b) enable formally studying to what extent the protocol can be improved. (As we explain below, both of these points have been illustrated in subsequent works

[PS16a,PS16b].) We mention that while abstract models for *higher-level applications* of the blockchain (e.g., a "smart contract" abstraction) were provided in the UC framework—see [KMS+15,BK14]—it is not clear to what extent those abstractions can be satisfied by Nakamoto's protocol; rather, we are here interested in having a simple notion of the blockchain itself that we can prove is satisfied by Nakamoto's protocol and yet is useful for applications.

Roughly speaking, a blockchain is an interactive protocol where each participant has a local variable `state` which contains a list of messages $\vec{m}$, called the "chain". Players receive inputs, called records/batches/messages, that they attempt to include in the chain of themselves and of others. We require the following properties from a secure blockchain:

- *consistency*: with overwhelming probability (in T), at any point, the chains of two honest players can differ only in the last T blocks;
- *future self-consistence*: with overwhelming probability (in T), at any two points r, s the chains of any honest player at r and s differ only within the last T blocks;
- *g-chain-growth*: with overwhelming probability (in T), at any point in the execution, the chain of honest players grows by at least T messages in the last $\frac{T}{g}$ rounds; g is called the chain-growth of the protocol.
- the *μ-chain quality* with overwhelming probability (in T), for any T consecutive messages in any chain held by some honest player, the fraction of messages that were "contributed by honest players" is at least μ.

The consistency property is just the plain one considered by Nakamoto [Nak08] (and formalized by Garay *et al.* [GKL15]). As we note, however, this consistency property is typically not sufficient for applications. In particular, it does not rule out a protocol that oscillates between two different chains $\vec{m}_1, \vec{m}_2$; on even rounds all players have $\vec{m}_1$ as their chain, and on odd rounds $\vec{m}_2$. Clearly such a protocol does not suffice for typical applications (e.g., bitcoin, or achieving a public ledger). Thus, to prevent it, we introduce the *future self-consistency* property.

The lower bound on chain-growth was explicitly considered by Sampolinsky and Zohar [SZ15] (but they only consider growth in expectation); Garay *et al.* [GKL15] implicitly show a lower bound on chain growth within one of their proofs, and [KP15] explicitly introduce it as a desideratum. In this paper, we additionally introduce an *upper-bound* on chain growth as a desirable property; as shown in subsequent work [PS16a,PS16b], this property is useful in applications.

Finally, the chain quality property was first discussed on the Bitcoin forum [mtg10] and made explicit in the selfish mining attacks by Eyal and Sirer [ES14] with respect to the bitcoin application of the blockchain.[3] The property was first formalized, and given the name "chain quality" by Garay *et al.* who also show new applications of it (as we discuss shortly).

[3] In the bitcoin application of the blockchain, each player receives a reward whenever if mines a block; the chain quality thus dictates a bound on how much more reward an adversary can get by deviating from the protocol.

We show the usefulness of these properties by demonstrating that *any* blockchain protocol satisfying them can be used to achieve a *public ledger (i.e., consensus)* satisfying (a) *persistence* (namely, if a message gets added to the public ledger, it never gets removed) and (b) *liveness* (that is, if all honest players want to add a some message to the ledger, the message should eventually appear on it). We mention that Garay *et al.* already noted that, *intuitively*, the chain quality property implies liveness (since, by chain quality the adversary cannot monopolize the chain), and consistency implies persistence. However, although they show how to use Nakamoto's protocol to obtain a public ledger (in the synchronous model), they use those two properties *and* additional properties of the concrete protocol to establish it. Kiayias and Panagiotakos [KP15] demonstrate that additionally requiring chain growth suffices to prove liveness in a black-box way, but proving persistence still requires an analysis of the concrete protocol. We highlight that it is our notion of future-self consistency that allows us to obtain also persistence in a black-box way. Subsequent works by Pass and Shi [PS16a,PS16b] give further evidence to the usefulness of our abstract notion of a blockchain (and its security properties).

Main theorem. Our main result demonstrates that Nakamoto's protocol achieves consistency as well as all of our other desiderata. Let $\gamma = \frac{\alpha}{1+\Delta\alpha}$; think of γ as a "discounted" version of α due to delays on the network. Intuitively, by delaying messages the adversary gets additional computation time.

Theorem 2. *Assume there exists some $\delta > 0$ such that*

$$\alpha(1 - 2(\Delta + 1)\alpha) \geq (1 + \delta)\beta.$$

Let $g = \frac{\gamma}{1+\delta}$ and $\mu = 1-(1+\delta)\frac{\beta}{\gamma}$. Then Nakamoto's protocol satisfies consistency, future self consistency, μ-chain quality and g-chain growth.

Note that when $p \ll 1/n\Delta$ (which is the case considered in practice), we have that $\gamma \approx \alpha \approx (1-\rho)np$ and thus $\frac{\gamma}{\beta} \approx \frac{1-\rho}{\rho}$. As a consequence, we have the following corollary:

Corollary 3. *Assume $\rho < \frac{1}{2}$. Then for every n, Δ, there exists some sufficiently small $p_0 = \Theta(\frac{1}{\Delta n})$ such that Nakamoto's protocol with mining parameter $p \leq p_0$ satisfies consistency, future self consistency, $1-\frac{\rho}{1-\rho}$-chain quality and $\frac{pn}{2}$-growth.*

Thus, as long as $\rho < \frac{1}{2}$, Nakamoto's protocol guarantees that messages contributed by honest players will eventually end up on the chain, and as long as $\rho < \frac{1}{3}$, we have that half of the messages on the chain will be contributed by honest players. We mention that our chain quality bound matches that established by Garay *et al.* assuming *no delays* (i.e., $\Delta = 1$), and is tight due to the selfish mining (a.k.a. "mining-cartel") attacks of [mtg10,ES14]).

A natural question left open by our main theorem is whether there exist protocols satisfying our abstract notion of a blockchain that improve upon the parameters achieved by Nakamoto's protocol (i.e., is Nakamoto's protocol "optimal"?). A subsequent result by Pass and Shi [PS16a] shows how we can "amplify"

the chain quality in Nakamoto's protocol to achieve a "close-to-optimal" chain quality of $1-(1-\delta)\rho$, where δ is an arbitrary small constant.[4] We highlight that the results in [PS16a] relies on the analysis from this paper in a blackbox way.

1.4 Is Nakamoto's Protocol Really Permissionless?

Our theorem only shows that for every n, Δ, there exists some mining-hardness parameter p that makes the protocol secure, so it might seem like the protocol needs to know n and therefore cannot be "permissionless"; see Sect. 1.5 for an experimental evaluation of how the level of security depends on the choice of p. As we pointed out above, this is not an anomaly of our analysis; when $p > \frac{1}{n\rho\Delta}$ the protocol is insecure. The point, however, is that the protocol only needs to know a *very rough upper-bound* on the number of players n (but the worse the upper-bound gets, the worse the efficiency of the protocol becomes.)

We additionally remark that our theorem regarding the lower bound on the chain growth actually does not make any assumption about p; this means that the honest players can use an initial set-up phase to estimate the chain growth and from this deduce a weak upper-bound on the number of players n, and then use this new upperbound to run the protocol. Indeed, as we hinted to before, the bitcoin protocol recalibrates the mining hardness parameter p every 2016 blocks (roughly 2 weeks) based on the time it took to find 2016 blocks. We leave a formal analysis of this update procedure for future work.

1.5 An Experimental Interpretation

In this section, we provide an experimental interpretation of our theorems by using estimates of parameters in a real world setting. Using estimates of hardware hashing rates (10^{12} h/s), we consider $n = 10^5$ participants and $\Delta = 10^{13}$, which corresponds to roughly $10s$ delay for the network at the given hashing rates. These numbers roughly coincide with estimates of the number of hash operations per second occurring in the Bitcoin network (7×10^{17}) at the beginning of 2016 [Blo16]. The 10 s estimation, under *network assumptions*, roughly aligns with the empirical measurements made by Decker and Wattenhofer [DW13] and their bitcoinstats.com website.[5]

The hardness parameter p in Nakamoto's protocol reflects the expected time between the discovery of blocks among all participants. Here, we explore how consistency is related to this parameter $p = \frac{1}{n\Delta \cdot c}$ by changing c. One can interpret c as the scale-free *expected block-time* in terms of the number of network delays.

[4] An "optimal" chain quality of $1-\rho$ means a ρ fraction attacker gets a ρ fraction of the blocks.

[5] However, in both cases, they measure connectivity *by number of nodes* instead of by computational resources; thus their "95$^{\text{th}}$ percentile" estimations are biased larger because they include many hobby nodes which are connected by slow network connections and do not contribute any noticeable computation to the system.

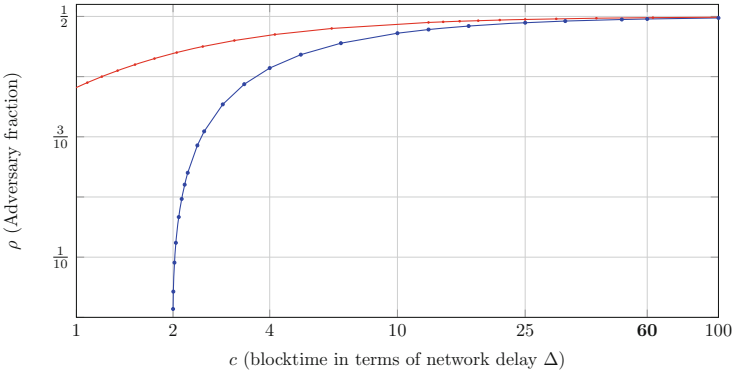

Fig. 1. For $n = 10^5$ and $\Delta = 10^{13}$ (i.e., 10 s delays at 1TH/s for commercially available mining hardware—these parameters roughly coincide with estimates of hashrate at the start of 2016), we set hardness parameter $p = \frac{1}{c \cdot n\Delta}$ where c varies along the x-axis. We can interpret c as the expected blocktime in terms of the network delay Δ. The blue graph depicts a numerically-computed maximum value of ρ for which $\alpha(1 - (2\Delta + 2)\alpha) > \beta$, i.e. parameters under which Theorem 6 shows consistency of the Nakamoto protocol. The red plot shows when our best attack succeeds in violating consistency. When $c = 60$, the hardness roughly corresponds to an expected 10-minute blocktime, and our theorem shows that Nakamoto tolerates a $\rho < 49.57\%$ attack, and our best attack succeeds when $\rho > 49.79\%$. (Color figure online)

For these choices, Fig. 1 depicts when our consistency theorem holds in Nakamoto's protocol by graphing c against the fraction (ρ) of computation controlled by the adversary. The blue graph depicts a numerically-computed maximum value of ρ for which $\alpha(1 - (2\Delta + 2)\alpha) > \beta$, i.e. parameters under which our Theorem 6 shows consistency of the Nakamoto protocol. The red plot shows when our best attack succeeds in violating consistency.

Nakamoto's protocol attempts to maintain a 10-minute blocktime by varying hardness p. For a delay $\Delta \sim 10$ s, this corresponds to a setting of $c = 60$. In this range, the Nakamoto protocol, as well as our attack give essentially the same result: Nakamoto tolerates an adversary with $\rho < 49.57\%$ and our best attack succeeds when $\rho > 49.79\%$. If we make a very conservative estimate of network delays being 1m, then $c = 10$, and Nakamoto remains consistent with respect to a 47.2% coalition.

Finally, our analysis is not tight when c is small because our attack only analyzes the probability that the adversary is able to completely control the chain. When c is small, there is also a large probability that honest players do not converge on a chain even without any adversarial messages.

1.6 Proof Highlights

Although our high-level approach follows similar intuitions as the analyses from Garay *et al.* [GKL15] and Sompolinsky and Zohar [SZ15], our actual proof uses

a quite different strategy. The bulk of our proof consists of dealing with the attack strategies which are omitted from the analysis in [SZ15], and dealing with them requires us to consider an altogether different proof technique: instead of *directly* analyzing the whole blockchain process, we consider a sequence of simplified processes which are "dominated" by the original one but are simpler to analyze. For instance, we aim to show that in the optimal attack, the adversary should always delay messages for as long as possible (so that messages are always delivered after Δ steps). An obstacle in performing such a stochastic domination analysis is that once we start delaying messages, honest parties start to "mine" different blocks and the executions of our two processes diverge and become hard to compare: Ideally, to perform the domination argument we would like to consider a *fixed* execution (where the randomness of all parties are fixed) and to show by induction that delaying messages *less than* Δ never helps the attacker in that *particular* execution. The problem is that such a domination claim is not true: in some *lucky* scenarios (where the randomness is fixed), delaying messages in fact *improves* the situation for the honest parties (they now start mining blocks that magically lead to more successes). Of course, the probability of this happening should be small, but formally showing this would require us to somehow couple the experiments with and without maximum delays which is non-trivial (due to dependencies created by the random oracle).

The $\mathcal{F}_{tree}$ model. To overcome this issue, we rely on "simulation techniques" from the cryptographic literature on secure computation [GMW87, Can00]: we first consider an idealized scenario where the players do not mine blocks but instead have access to an idealized "mining" functionality, which we call $\mathcal{F}_{tree}$. This functionality determines whether honest parties succeed in mining (at random) and the success probability is independent of the current chain an honest party is trying to extend. In this model, we can now perform a domination argument for every *fixed* randomness for the experiment. One of our main technical lemmas, which turns out to be quite subtle to prove, shows that any attack that succeeds in the "real-life" protocol in the random oracle model can be turned into (i.e., simulated by) an attack in the idealized $\mathcal{F}_{tree}$ model. The key technical issue here is to deal with the dependencies created by the random oracle. (As an independent contribution, we believe that our $\mathcal{F}_{tree}$ simulation lemma can be helpful in formalizing some steps left informal in e.g., [GKL15, KP15, SZ15].)

The chain growth lowerbound. Armed with the above-mentioned techniques, the next crucial step is demonstrating a lowerbound on the chain growth. Roughly speaking, we prove by induction that (in the $\mathcal{F}_{tree}$ model) the chain grows at least as fast in the real execution of the protocol, as in a "hybrid" experiment where (a) all messages are maximally delayed, (b) honest parties "freeze" and stop mining for Δ steps whenever some honest player mines a block and (c) all messages sent by the adversary are removed. The advantage of this hybrid experiment is that the chain growth process can now be described as a simple Markov chain—there are no longer any "adversarial transitions" and due to the "freezing", honest players never have any chain conflict. This process can

next be analyzed using standard Chernoff bounds. We emphasize that for the induction proof to work, we crucially rely on the fact that our analysis is in the $\mathcal{F}_{tree}$-model.

No "long" block withholding. We next use the chain growth lowerbound to demonstrate a central property of the blockchain protocol, which we refer to as the "no long block withholding" property: an adversary cannot withhold a block that it has mined for too long. Unless the adversary broadcasts the block to the honest players within some short amount of time, the block becomes "irrelevant" and will never be accepted by the honest players. Roughly speaking, we prove this by showing that, assuming that the adversary controls less than half of the computational power in the network, the chain of honest players will grow at a faster rate than any private chain the adversary can create, and thus unless it releases any block it finds quickly, the honest players' chain will be too long for the block to ever be relevant.

Proving consistency. Finally, proving consistency is the most challenging part of our proof. We start by first considering an execution *without adversarial messages*, and with deterministic delays, and identify a "pattern" which ensures that the chain of honest players converges: roughly, the pattern—which we refer to as a "convergence opportunity"—is that (1) there is a period of "silence" for Δ rounds where no honest player mines a block, (2) this is followed by a round where a *single* honest player mines a block, (3) which is followed by another Δ rounds of silence. (This notion of a convergence opportunity is closely related to a notion considered in [SZ15], and can be thought of a generalization of the notion of a "uniquely successful round" considered in the synchronous setting in [GKL15].) Whenever such a pattern occurs, all honest players converge on the chain (thus we call it a convergence opportunity): after the first period of silence, they all agree on the length of the chain (but may still have different chains), and thus the lone miner who finds a new block extends this longest chain by 1, and finally after the second period of silence this chain has propagated to all honest players (and since it is longer than all their current chains, they will switch to it). We are now interested in understanding how many times this patterns occurs within some specific period of time t. The crucial point here is that the process we now analyze is memoryless, and thus can be described by a (somewhat simple) Markov chain. On the negative side, the Markov chain that arises from this problem is too complicated to be analyzed with standard concentration bounds for Markov chains (see e.g., [CLLM12]); we instead, provide a direct analysis of a simplified experiment (which, roughly speaking, instead analyzes the times between successful mining of honest players.) and we then use this to provide a lower bound on the number of convergence opportunities.

Finally, once we have established a strong concentration bound on the number of convergence opportunities, we argue that the only way that an attacker can *ruin* such a pattern is by itself mining a block that is accepted by the honest players during it. We here rely on the block-withholding lemma to argue that any block that the attacker can use to ruin a convergence opportunity must

have been mined by the adversary not long before the beginning of the period of time we are analyzing; we then show that the number of adversarial block mined during this (slightly extended) period of time is smaller than the number of convergence opportunities, and thus conclude that at least one convergence opportunity will remain even in the presence of the adversary, and thus honest parties still converge on their chain.

1.7 Related Work

The problem of reaching agreement in the presence of faulty participants, described first by Pease, Shostak, and Lamport [PSL80], and also known as distributed consensus has been very well studied over the past 40 years. The basic problem considers a set of n parties connected by reliable and authenticated pairwise network channels who wish to agree on a common output in the presence of an adversary who controls a fraction of the participants. Many aspects of the problem have been studied, with relaxations concerning the fraction of corrupted parties, the channels available to the participants, whether the protocols are deterministic or randomized and whether the participants are computationally bounded. Some protocols only consider fail-stop adversaries, while others consider a Byzantine setting in which some of the participants are malicious adversaries who attempt to disrupt the agreement. In the Byzantine agreement (BA) version of the problem, Castro and Liskov [CL99] implemented a replication library that was practical enough to use for a file system; subsequently, other works have considered *fast* or *simpler* versions of the Paxos protocol [MA05, Lam10, Lam11]. All of these works assume common knowledge of the number of participants n, as well as identities for the participants.

Okun [Oku05a, Oku05b, OB08] considers BA in an "anonymous [synchronous] model without port awareness" in which processors do not have identifiers and cannot correlate messages to their sources; Okun shows both an impossibility result for deterministic protocols, and a feasibility result for probabilistic ones. Aspnes *et al.* [AJK05] shows how to use a proof-of-work in a pre-processing step for this model to assign interim identities to parties so that the number of identities assigned is proportional to computational power. After the pre-processing, a standard authenticated BA protocol is used. Neither results, however, are in the peer-to-peer setting in which new users can join and leave during the execution.

Miller and LaViola [ML14] show that a variant of Nakamoto's protocol can be used to solve the single-shot Byzantine agreement problem in the presence of a minority of faults in an asynchronous setting. The single-shot setting is substantially easier, since the adversary is limited, and for example, cannot mount block-withholding attacks. Garay, Kiayias, and Leonardas [GKL15] provide a better analysis of Nakamoto's protocol, and also propose two protocols based on Nakamoto's protocol that satisfy all the properties of BA in the multiple-instance setting. They only consider synchronous networks (and no spawning of new honest players). As mentioned above, in synchronous networks, simpler solutions are possible.

2 Blockchain Protocols and Executions

In this section, we present an abstract model for blockchain protocols which aims to cover many variants of blockchain protocols.

2.1 Blockchain Protocols

A blockchain protocol is a pair of algorithms $(\Pi, \mathcal{C})$ where Π is a stateful algorithm that receives a security parameter κ as inputs and maintains a local state state. The algorithm $\mathcal{C}(\kappa, \mathsf{state})$ outputs an *ordered* sequence of "records", or "batches", $\vec{\mathsf{m}}$ (e.g., in the bitcoin protocol, each such record is an ordered sequence of transactions). We call $\mathcal{C}(\kappa, \mathsf{state})$ the "record chain" of a player with security parameter κ and local variable state; to simplify notation, whenever κ is clear from context we often write $\mathcal{C}(\mathsf{state})$ to denote $\mathcal{C}(\kappa, \mathsf{state})$.

Algorithm Π is parameterized by a *validity* predicate V (denoted by Π^V) that encapsulates the semantic properties (e.g., "no double spending") that a blockchain application aims to achieve. $V(\vec{\mathsf{m}})$ returns 1 if and only if the chain $\vec{\mathsf{m}}$ is *valid* for some notion of validity.

A Blockchain Execution. Following the framework for Universal Composability [Can00], we consider the execution of a blockchain protocol $(\Pi^V, \mathcal{C})$ that is directed by an environment $Z(1^\kappa)$ (where κ is a security parameter), which activates a number of parties $1, 2, \ldots, n$ as either "honest" or corrupted parties. Honest parties execute Π on input 1^κ with an empty local state state; corrupt parties are controlled by an attacker A which reads all their inputs/message and sets their outputs/messages to be sent.

- The execution proceeds in *rounds* that model time steps. In round r, each honest player i receives a message (a "record") m from Z (that it attempts to "add" to its chain) and potentially receives incoming network messages (delivered by A). It may then perform any computation, *broadcast* a message to all other players (which will be delivered by the adversary; see below) and update its local state state_i.
- A is responsible for delivering all messages sent by parties (honest or corrupted) to *all* other parties. 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. (Later, we shall consider restrictions on the delivery time.) The identity of the sender is not known to the recipient.[6]
- At any point, Z can communicate with adversary A or access $\mathcal{C}(\mathsf{state}_i)$ (i.e., the current record chain of the player) where state_i is the local state of player i.

[6] We could also consider a seemingly weaker model where messages sent by corrupted parties need not be delivered to all honest players. We can easily convert the weaker model to the stronger model by having honest parties "gossip" all messages they receive.

– At any point, Z can *corrupt* an honest party j which means that A gets access to its local state and subsequently, A controls party j. (In particular, this means we consider a model with "erasures"; random coin tosses that are no longer stored in the local state of j are not visible to A.)[7]
– At any point, Z can *uncorrupt* a corrupted player j, which means that A no longer controls j and instead player j starts executing $\Pi(1^\kappa)$ with a fresh state state_j. (This is also how we model Z spawning a "new" honest player.) A gets informed of all such uncorrupt messages and is required to deliver all messages previously sent by (currently alive) honest players.[8]

Let $\mathsf{EXEC}^{(\Pi^V,\mathcal{C})}(A,Z,\kappa)$ be 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.

Admissible Environments. We consider executions with restricted adversaries and environments; these restrictions will be specified by a predicate $\Gamma(\cdot,\cdot,\cdot)$.

Definition 1 (Admissible Environments). *We say that the tuple of parameters $(n(\cdot),\rho,\Delta(\cdot),A,Z)$ is* Γ-admissible w.r.t. $(\Pi^V,\mathcal{C})$ *if A and Z are non-uniform probabilistic polynomial-time algorithms, $\Gamma(n(\cdot),\rho,\Delta)=1$ and for every $\kappa\in N$, every view view in the support of $\mathsf{EXEC}^{(\Pi^V,\mathcal{C})}(A,Z,\kappa)$, the following holds:*

1. *Z activates $n=n(\kappa)$ parties in view;*
2. *A delays messages by at most $\Delta=\Delta(\kappa)$ rounds (and in the case of newly spawned players, instantly delivers messages that were sent more than Δ rounds ago);*
3. *at any round r in view, A controls at most $\rho\cdot n(\kappa)$ parties; and*
4. *in every round r in view, Z only sends local inputs m to an honest player i, if $V(\mathcal{C}(\mathsf{state}_i)||\mathsf{m})=1$, where state_i is player i's local state at round r in view.*

Whenever the protocol $(\Pi^V,\mathcal{C})$ is clear from context, we simply call (n,ρ,Δ,A,Z) Γ-admissible.

2.2 A Remark About the Communication Model

Our model assumed that any player can send a message to all other players in the network, and that those messages arrive within Δ rounds, no matter how long they are. This is clearly not a realistic model. In real-life, players communicate their messages through a gossip network, and thus we need to assume

[7] Our proof actually extends also to the model "without erasures".

[8] This models the fact that a player is not considered "honest" before it has joined the network and gotten "initialized". In the real-life execution of bitcoin, new players joining send out a message to the network, request to be initialized and download the longest chain known to the network. We only consider them honest once this process is over.

that this network is sufficiently connected and has sufficiently many honest players to ensure Δ delivery time. This model remains infeasible if messages can be arbitrary long. However, in the applications we consider—assuming that records m provided by the environment are of length $O(\kappa)$ (i.e., there is a "block-size limit"[9])—honest players only communicate messages that differ in the last $O(\kappa)$ bits from messages that they have previously received. For such cases it seems reasonable to assume that a sufficiently connected routing network has the desired property of ensuring delivery of all messages within Δ rounds.

2.3 Blockchain Protocols in the ROM

To study Nakamoto's blockchain protocol, we need to extend the model with a random oracle. In an execution with security parameter κ, we assume all parties 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 players (as well as A) may make *any* number of queries to $\mathsf{H.ver}$. On the other hand, in each round r, honest players can make only a *single* query to H, and an adversary 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 Z does not get direct access to the random oracle (but can instruct A to make queries).

2.4 Nakamoto's Protocol

We turn to describing Nakamoto's protocol [Nak08], which we refer to as $(\Pi^p_{Nak}, \mathcal{C}^p_{Nak})$. The local state state maintained by Π^p_{Nak} is a sequence of *(mined) blocks* $\vec{\mathsf{b}}$, where each mined block is a tuple $(h_{-1}, \eta, \mathsf{m}, h)$ that consists of a hash h_{-1} (a pointer to the previous record), a nonce η, a record m, and a hash h (a pointer to the current record[10]) and is initialized to a special "genesis" block: $(0, 0, \perp), \mathsf{H}(0, 0, \perp)$. Let $\mathcal{C}(\mathsf{state})$ be the sequence of records $\vec{\mathsf{m}}$ contained in the sequence of blocks state. The protocol is parameterized by a hardness function $p(\cdot)$ which defines a constant $D_p = p(\kappa) \cdot 2^\kappa$ such that for all (h, b), $\Pr_\eta[\mathsf{H}(h, \eta, b) < D_p] = p(\kappa)$. Whenever p is clear for context, we simply denote the protocol $(\Pi_{Nak}, \mathcal{C}_{Nak})$ (without the p superscript); additionally, whenever κ is clear from context, we let $p = p(\kappa)$.

[9] In Bitcoin's instantiation of the blockchain protocol, there is currently a severe restriction on the block-size. There is currently an active debate whether to raise the block-size limit or to leave it small.

[10] In reality (as well as in the description in the introduction), h is not included in the block (as it can be easily determined from the remaining elements); we include it to ensure that we can verify validity of a block using only $\mathsf{H.ver}$.

We say a block $\mathsf{b} = (h_{-1}, \eta, \mathsf{m}, h)$ is *valid with respect to (a predecessor block)* $\mathsf{b}_{-1} = (h'_{-1}, \eta', \mathsf{m}', h')$ if three conditions hold:

1. $h_{-1} = h'$,
2. $h = \mathsf{H}(h_{-1}, \eta, \mathsf{m})$,
3. and $h < D_p$.

A sequence of blocks $\mathsf{state} = (\mathsf{b}_0, \ldots, \mathsf{b}_\ell)$ is *valid* if (a) $\mathsf{b}_0 = (0, 0, \perp, \mathsf{H}(0, 0, \perp))$ is the genesis block, (b) for all $i \in [\ell]$, b_i is valid with respect to b_{i-1}, and (c) $V(\mathcal{C}(\mathsf{state})) = 1$.

Each round of Π^V_{Nak} proceeds as follows:

- Read all incoming messages (delivered by A). If any incoming message state' is a valid sequence of blocks that is longer than its local state state, replace state by state'. (Note that checking the validity of state' can be done using only $\mathsf{H.ver}$ queries)
- Read local message m (from Z). If m is such that $V(\mathcal{C}(\mathsf{state})||\mathsf{m}) \neq 1$, proceed to the next round. Otherwise, pick a random nonce $\eta \in \{0,1\}^\kappa$ and issue query $h = \mathsf{H}(h_{-1}, \eta, \mathsf{m})$ where h_{-1} is the 4'th element in the last block in state. If $h < D_p$, then Π adds the *newly mined* block (h_{-1}, η, b, h) to state and broadcasts the updated state.

Depending on the definition of V, one can instantiate either Bitcoin, e.g., by having V enforce that m can be parsed into a sequence of well-formed *transactions* each of which is *authorized* and spends money from a source account to a destination account at most once without deficit, etc., as well as other cryptocurrencies with different semantics such as Namecoin. We may also consider a simpler predicate $V_{\mathcal{L}}$ that simply accepts all messages; that is $V_{\mathcal{L}}(\vec{\mathsf{m}}) = 1$; such a predicate is useful, for instance, to use a blockchain to provide a public ledger.

A Remark on our use of the Random Oracle. Recall that in our model, we restrict players to a single evaluation query H per round, but allow them any number of verification queries $\mathsf{H.ver}$ in the same round. We do this to model the fact that checking the validity of mined blocks is "cheap" whereas the mining process is expensive. (To enable this, we have included a pointer h to the current record in every mined block in the description of Nakamoto; thus a player need not spend an H query to compute the pointer to the previous record.)

In practice, the cost of evaluating a hash function (which is used to instantiate the random oracle) is the same as verifying its outputs, but our modeling attempts to capture the phenomena that a miner typically use various heuristics (such as black lists of IP addresses that have sent invalid blocks) and different hardware to check the validity of a mined block versus to mine a new block.

3 Formal Definitions of the Desiderata

In this section, we provide formal definitions of the desiderata mentioned in the introduction. We start with some notation and preliminaries.

Notation. For some A, Z, consider some view in the support of $\mathsf{EXEC}^{(\Pi^V, \mathcal{C})}(A, Z, \kappa)$. We use the notation $|\mathsf{view}|$ to denote the number of rounds in the execution, view^r to denote the prefix of view up until round r, $\mathsf{state}_i(\mathsf{view})$ denotes the local state of player i in view, $\mathcal{C}_i(\mathsf{view}) = \mathcal{C}(\mathsf{state}_i(\mathsf{view}))$ and $\mathcal{C}_i^r(\mathsf{view}) = \mathcal{C}_i(\mathsf{view}^r)$.

(Strongly) Negligible Functions. A function $\epsilon(\cdot)$ is said to be *negligible* if for every polynomial $p(\cdot)$, there exists some κ_0 such that $\epsilon(\kappa) \leq \frac{1}{p(\kappa)}$ for all $\kappa \geq \kappa_0$. Our bounds will actually also apply to an *exponentially-strong* interpretation of what it means for a function to be negligible. A function $\epsilon(\cdot)$ is said to be *(strongly) negligible* if there exists constants $c_0 > 0, c_1$ such that for all κ, $\epsilon(\kappa) \leq e^{-c_0\kappa + c_1}$. In the rest of the paper, we simply use the term "negligible", but all uses of it can be replaced by strongly negligible. We often use the shorthand $\mathsf{neg}(\kappa)$ to denote a function that is negligible as a function of κ.

3.1 Chain Growth

Our first desiderata is that the chain grows proportionally with the number of rounds of the protocol. This intuitive property was explicitly considered by Sompolinsky and Zohar [SZ15] but only *in expectation*; it was also implicitly considered in Garay *et al.* within one of their proofs (but was not highlighted as a desideratum), and it was explicitly highlighted as a desideratum by Kiayias and Panagiotakos [KP15]. We here generalize these definitions to abstract blockchain protocols, and add a useful length-consistency property.[11] (Looking forward, in Sect. 3.4, we also consider an *upper-bound* on chain growth.) Let,

$$\mathsf{min\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) = \min_{i,j} |\mathcal{C}_j^{r+t}(\mathsf{view})| - |\mathcal{C}_i^r(\mathsf{view})|$$

where we quantify over players i, j such that i is honest at view^r and j is honest at view^{r+t}.

Let $\mathsf{growth}^t(\mathsf{view}, \Delta, T) = 1$ iff the following two properties hold:

- **(consistent length)** for all rounds r, r' such that $r \leq |\mathsf{view}| - \Delta$ and $r + \Delta \leq r' \leq |\mathsf{view}|$, for every two players i, j such that in view, i is honest at r and j is honest at r', we have that

$$|\mathcal{C}_j^{r'}(\mathsf{view})| \geq |\mathcal{C}_i^r(\mathsf{view})|$$

[11] The length-consistency requirement is actually not needed for any of our applications, but having it enables achieving sharper bounds, and this property is trivially satisfied by Nakamoto's protocol.

– **(chain growth)** for every round $r \leq |\mathsf{view}| - t$, we have

$$\mathsf{min\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) \geq T.$$

In other words, growth^t is a predicate which tests that (a) honest parties have chains of roughly the same length, and (b) during any t rounds in the execution, all honest parties' chains increase by at least T.

Definition 2. *A blockchain protocol* $(\Pi, \mathcal{C})$ *has* chain growth rate $g(\cdot,\cdot,\cdot,\cdot)$ *in* Γ*-environments if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some constant* c *and negligible functions* ϵ_1, ϵ_2 *such that for every* $\kappa \in \mathbb{N}, T \geq c\log(\kappa)$*, and* $t \geq \frac{T}{g(n(\kappa),\rho,\Delta(\kappa))}$*, the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi^V,\mathcal{C})}(A, Z, \kappa) : \mathit{growth}^t(\mathit{view}, \Delta(\kappa), T) = 1\right] \geq 1 - \epsilon_1(\kappa) - \epsilon_2(T)$$

If $\epsilon_1 = 0$*, we say that* $(\Pi, \mathcal{C})$ *has* error-less chain growth rate g in Γ*-environments.*

3.2 Chain Quality

Our second desideratum is that the number of records contributed by the adversary is proportional to its relative power. This property was first discussed on the Bitcoin forum [mtg10] and made explicit in the selfish mining attacks by Eyal and Sirer [ES14] w.r.t. the bitcoin application of the blockchain.[12] The property was first formalized, and given the name "chain quality" by Garay *et al.* [GKL15]. We generalize their definition to abstract blockchain protocols. Doing so is somewhat non-trivial in that it is not directly clear what it means for a record to be adversarial (Garay *et al.* only provide a definition of an adversarial block for the particular protocol of Nakamoto, and their definition only applies in the random oracle model).

We say that a *record* m *is non-adversarial (or honest) w.r.t.* view *and prefix* $\vec{\mathsf{m}}$ if there exists a player j and some round r' such that in $\mathsf{view}^{r'}$, j is honest, the environment provided m as input to j, and $\vec{\mathsf{m}}$ is a prefix of $\mathcal{C}_i(\mathsf{view}^{r'})$. (That is, there exists some honest player that received m as an input when their chain contained $\vec{\mathsf{m}}$).

Let $\mathsf{quality}^T(\mathsf{view}, \mu) = 1$ iff for every round r and every player i such that i is honest in view^r, among any consecutive sequence of T records M in $\mathcal{C}_i^r(\mathsf{view})$, the fraction of records m that are honest w.r.t. view^r and $\vec{\mathsf{m}}$, where $\vec{\mathsf{m}}$ is the prefix of $\mathcal{C}_i^r(\mathsf{view})$ preceeding M, is at least μ.

[12] In the bitcoin application of the blockchain, each player receives a reward whenever if mines a block; the chain quality thus dictates a bound on how much more reward an adversary can get by deviating from the protocol.

Definition 3. *A blockchain protocol* $(\Pi, \mathcal{C})$ *has* chain quality $\mu(\cdot,\cdot,\cdot,\cdot)$ *in* Γ *environments, if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some constant* c *and negligible functions* ϵ_1, ϵ_2 *such that for every* $\kappa \in \mathbb{N}, T > c\log(\kappa)$ *the following holds:*

$$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi^V,\mathcal{C})}(A, Z, \kappa) : \mathsf{quality}^T(\mathsf{view}, \mu(\kappa, n(\kappa), \rho, \Delta(\kappa))) = 1\right] \geq 1 - \epsilon_1(\kappa) - \epsilon_2(T)$$

If $\epsilon_1 = 0$*, we say that* $(\Pi, \mathcal{C})$ *has* errorless chain quality μ *in* Γ*-environments.*

3.3 Consistency

The *common-prefix* property by Garay *et al.* [GKL15], which was already considered and studied by Nakamoto [Nak08], requires that in any round r, the record chains of any two honest players i, j agree on all, but potentially the last T, records. We note that this property (even in combination with the other two desiderata) provides quite weak guarantees: even if any two honest parties perfectly agree on the chains, the chain could be completely different on, say, even rounds and odd rounds. We here consider a stronger notion of consistency which additionally stipulates players should be consistent with their "future selves".[13]

Let $\mathsf{consistent}^T(\mathsf{view}) = 1$ iff for all rounds $r \leq r'$, and all players i, j (potentially the same) such that i is honest at view^r and j is honest at $\mathsf{view}^{r'}$, we have that the prefixes of $\mathcal{C}_i^r(\mathsf{view})$ and $\mathcal{C}_j^{r'}(\mathsf{view})$ consisting of the first $\ell = |\mathcal{C}_i^r(\mathsf{view})| - T$ records are identical.[14]

Definition 4. *A blockchain protocol* $(\Pi, \mathcal{C})$ *satisfies* consistency in Γ *environments, if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some constant* c *and negligible functions* ϵ_1, ϵ_2 *such that for every* $\kappa \in \mathbb{N}, T > c\log(\kappa)$ *the following holds:*

$$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi^V,\mathcal{C})}(A, Z, \kappa) : \mathsf{consistent}^T(\mathsf{view}) = 1\right] \geq 1 - \epsilon_1(\kappa) - \epsilon_2(T)$$

If $\epsilon_1 = 0$*, we say that* $(\Pi, \mathcal{C})$ *has* errorless consistency in Γ*-environments.*

Note that a direct consequence of consistency is that the chain *length* of any two honest players can differ by at most T (except with negligible probability in T).

[13] This stronger notion of consistency combines what we called "plain" consistency and "future-self" consistency in the introduction.

[14] Pedantically, the "first ℓ records of $\mathcal{C}_j^{r'}(\mathsf{view})$ is not defined if $\mathcal{C}_j^{r'}(\mathsf{view}) < \ell$; to formalize it, we may represent the chains as infinite sequences of records, where all records after the end of the chain is a special "nil" symbol. In particular, this ensures that $\mathsf{consistent}^T(\mathsf{view}) = 0$ if $\mathcal{C}_j^{r'}(\mathsf{view}) < \ell$.

3.4 Chain Growth Upperbound

Our final desiderata is the existence of an *upperbound on the chain growth*. While we do not present any applications of this property in the current paper, it is an intuitively useful property—for instance, combined with the chain growth lower bound, it implies we can use a blockchain as a "partially-synchronized clock". (Additionally, subsequent work by Pass and Shi [PS16a,PS16b] demonstrate the usefulness of this property.)

Let,

$$\mathsf{max\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) = \max_{i,j} |\mathcal{C}_j^{r+t}(\mathsf{view})| - |\mathcal{C}_i^{r}(\mathsf{view})|$$

where we quantify over players i, j such that i is honest at view^r and j is honest at view^{r+t}. Let $\mathsf{upper\text{-}growth}^t(\mathsf{view}, \Delta, T) = 1$ iff for every round $r \leq |\mathsf{view}| - t$, we have

$$\mathsf{max\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) \leq T.$$

Definition 5. *A blockchain protocol $(\Pi, \mathcal{C})$ has* upper-bound on chain growth rate $g'(\cdot,\cdot,\cdot,\cdot)$ *in Γ-environments if for all Γ-admissible $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, there exists some constant c and negligible functions ϵ_1, ϵ_2 such that for every $\kappa \in \mathbb{N}, T \geq c\log(\kappa)$, and $t = \frac{T}{g'(n(\kappa),\rho,\Delta(\kappa))}$, the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi^V,\mathcal{C})}(A, Z, \kappa) : \mathit{upper\text{-}growth}^t(\mathit{view}, \Delta(\kappa), T) = 1\right] \geq 1 - \epsilon_1(\kappa) - \epsilon_2(T)$$

If $\epsilon_1 = 0$, we say that $(\Pi, \mathcal{C})$ has error-less upper-bound on chain growth rate g' *in Γ-environments.*

4 Main Theorem Statements

Our main results will be most convenient to parameterize in the following two quantities (which are defined for some fixed mining hardness function $p(\cdot)$; recall that Nakamoto's protocol is parametrized by p):

- let $\alpha(\kappa, n, \rho, \Delta) = 1 - (1 - p(\kappa))^{(1-\rho)n}$. That is, α is the probability that *some* honest player succeeds in mining a block in a round;
- let $\beta(\kappa, n, \rho, \Delta) = \rho np(\kappa)$. That is, β is the expected number of blocks that an attacker can mine in a round.

Whenever κ, n, ρ, Δ are clear from the context, we simply write α, β. In essence, the quantities capture the per-round *expected increase* in chain length by the honest parties and the adversary; the reason the quantities are defined differently is that we assume that the adversary can sequentialize its queries in a round, whereas honest players make a single parallel query (they each act independently), and thus even if they manage to mine several blocks, the longest chain held by honest players can increase by at most 1. Note, however, that

when p is small (in comparison to $1/n$), which is case for the Bitcoin protocol, α is well-approximated by $(1-\rho)np$ and thus $\frac{\alpha}{\beta} \approx \frac{1-\rho}{\rho}$, so this difference is minor.

We also consider the following quantity:

– let $\gamma(\kappa, n, \rho, \Delta) = \frac{\alpha}{1+\Delta\alpha}$ (When clear from context, we simply write γ.)

Roughly speaking, γ should be thought of a *discounted* version of α due to the fact that messages sent by honest parties can be delayed by Δ rounds and this may lead to honest players redoing work; γ corresponds to their *effective* mining power. Note that if p is sufficiently small then $\gamma \approx \alpha$ and thus $\frac{\gamma}{\beta} \approx \frac{1-\rho}{\rho}$.

We are now ready to state our main theorems. The proof of these theorems are all given in the Appendix (see Sect. 1.6 for a high-level overview of key ideas). We will consider two environments:

– In the least restrictive environment, Γ_0, we make *no restrictions* on the parameters (more than them being "valid"). Namely, let $\Gamma_0(n(\cdot), \rho, \Delta(\cdot)) = 1$ iff $n(\cdot), \Delta(\cdot)$ are functions $\mathcal{N} \to \mathcal{N}^+$ and $0 \leq \rho \leq 1$.
– In the more restrictive environment, we additionally assume that the adversary controls a sufficiently small fraction of the computational power. Let $\Gamma^p_\lambda(n(\cdot), \rho, \Delta(\cdot)) = 1$ iff $\Gamma_0(n(\cdot), \rho, \Delta(\cdot))) = 1$ and for all $\kappa, n = n(\kappa), \Delta = \Delta(k)$,

$$\alpha(1 - 2(\Delta + 1)\alpha) \geq \lambda\beta$$

The following three theorems formalize Theorem 2 from the introduction (which in turn implies Theorem 1). We first prove a lower bound on the chain growth.

Theorem 4 (Chain growth). *For any $\delta > 0$, any $p(\cdot)$, $(\Pi^p_{Nak}, \mathcal{C}^p_{nak})$ has chain growth rate*

$$g^p_\delta(\kappa, n, \rho, \Delta) = (1-\delta)\gamma$$

in Γ_0 environments.

We next prove a lower bound on the chain quality.

Theorem 5 (Chain quality). *For all $\delta > 0$, any $p(\cdot)$, $(\Pi^p_{Nak}, \mathcal{C}^p_{nak})$ has chain quality*

$$\mu^p_\delta(\kappa, n, \rho, \Delta) = 1 - (1+\delta)\frac{\beta}{\gamma}$$

in Γ_0 environments.

We finally show consistency.

Theorem 6 (Consistency). *For any $\lambda > 1$, any $p(\cdot)$, $(\Pi^p_{nak}, \mathcal{C}^p_{nak})$ satisfies consistency in Γ^p_λ environments.*

Chain growth upperbound. We additionally present an upperbound on the chain growth. (As mentioned before, this property is not needed for any of the applications that we present in the current paper, nor for the statement of the main result in the introduction, but may be useful in other contexts such as [PS16a,PS16b].)

Theorem 7 (Upper-bound on Chain growth). *For any $\delta > 0$, any $p(\cdot)$, $(\Pi^p_{Nak}, \mathcal{C}^p_{nak})$ has the upper-bound on chain growth rate*

$$\hat{g}^p_\delta(\kappa, n, \rho, \Delta) = (1+\delta)np$$

in Γ^p_λ environments.

5 Application: Public Ledger

In this section, we demonstrate how to use *any* blockchain satisfying the growth, quality, and consistency properties defined in Sect. 3 to construct a secure *public ledger* system. Garay *et al.* [GKL15] show a similar theorem, in the synchronous setting, for the *specific* blockchain of Nakamoto.

Informally, a *public ledger* serves as an immutable "bulletin board" to which anyone can post a message, and everyone can read all messages posted. As described by Garay *et al.* [GKL15], such a bulletin board ought to satisfy two properties, *liveness* and *persistence*:[15]

- *Liveness:* The liveness property stipulates that from any given round r, if a sufficiently long period of time t elapses—we refer to this time as the *wait-time* of the ledger—*every* honest player will output a message m as part of their (local) ledger, where m was provided as an input to some honest player between rounds r and $r+t$. (In particular, this implies the liveness condition of [GKL15] which requires that if the same message was provided to all honest players between rounds r and $r+t$, this messages will be output in the ledger.)
- *Persistence:* The persistence property stipulates that if some honest player i outputs a message m at position i in its local ledger, then (1) m is the only message that can ever be output at position i of any other honest player's ledger and (2) every honest player will eventually output m at position i.

Let us turn to a formal definition.

5.1 Definition of a Public Ledger

Just like the blockchain protocol, a *public ledger* is pair of algorithms $(\Pi, \mathcal{L})$ where Π is a stateful algorithm that maintains a local state state. The algorithm $\mathcal{L}(\kappa, \mathsf{state})$ outputs *ordered* sequence of messages $\vec{\mathsf{m}}$. We call $\mathcal{L}(\kappa, \mathsf{state})$

[15] The notion of Garay *et al.* [GKL15] is actually somewhat different and weaker: for instance, (1) they only require these properties to hold for records that are sufficiently "deep" in the ledger (we feel it is more natural/simpler to require it for *all* records in the ledger), and (2) they only require the liveness property to hold if all players received the *same* message.

the (local) ledger of a player with security parameter κ and local variable state. We define the execution of a public ledger protocol in exactly the same way as the execution of a blockchain protocol (see Sect. 2.1), and define the random variable $\mathsf{EXEC}^{(\Pi,\mathcal{L})}(A, Z, \kappa)$ in exactly the same way. Let $\mathcal{L}_i(\mathsf{view})$ denote the ledger of player i in the view view and let $\mathcal{L}_i^r(\mathsf{view}) = \mathcal{L}_i(\mathsf{view}^r)$.

Liveness. Let $\mathsf{live}(\mathsf{view}, t) = 1$ iff for any t consecutive rounds $r, \ldots, r+t$ in view there exists some round $r' \in [r, r+t]$ and players i such that in view, (1) i is honest at r', (2) i received a message m as input at round r', and (3) for every player j that is honest at $r+t$ in view, $\mathsf{m} \in \mathcal{L}_j^{r+t}(\mathsf{view})$.

Definition 6 (Liveness). *We say that public ledger $(\Pi, \mathcal{L})$ is* live with wait-time $w(\cdot,\cdot,\cdot,\cdot)$ *in Γ environments if for all Γ-admissible $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, there exists a negligible function ϵ in the security parameter $\kappa \in \mathbb{N}$, such that*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathcal{L})}(A, Z, \kappa) : \mathit{live}(\mathit{view}, w(\kappa, n(\kappa), \rho, \Delta(\kappa)) = 1\right] \geq 1 - \epsilon(\kappa)$$

Persistence. Let $\mathsf{persist}_\Delta(\mathsf{view}) = 1$ iff for every round $r \leq |\mathsf{view}| - \Delta$, every player i that is honest at view^r and every position $\mathsf{pos} \leq |\mathcal{L}_i^r(\mathsf{view})|$, if $\mathcal{L}_i^r(\mathsf{view})$ contains the message m at position pos, then for every round r' such that $r + \Delta \leq r'$ and every honest player j (possibly the same as i) we have that m is also at position pos in $\mathcal{L}_j^{r'}(\mathsf{view})$.

Definition 7 (Persistence). *We say that $(\Pi, \mathcal{L})$ is persistent in Γ environments if for all Γ-admissible $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, there exists a negligible function ϵ such that for every security parameter $\kappa \in \mathbb{N}$,*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathcal{L})}(A, Z, \kappa) : \mathit{persist}_{\Delta(\kappa)}(\mathit{view}) = 1\right] \geq 1 - \epsilon(\kappa)$$

5.2 Constructing a Public Ledger from a Blockchain

We turn to constructing a public ledger from any blockchain protocol. Let TRUE be the predicate that always outputs 1 (on any input).

Definition 8. *Given a blockchain protocol $(\Pi, \mathcal{C})$, we call $(\Pi', \mathcal{L})$ the* public ledger $T(\kappa)$-induced by $(\Pi, \mathcal{C})$, *where $\Pi' = \Pi^{\mathsf{TRUE}}$ and $\mathcal{L}(\kappa, \mathsf{state})$ computes $\mathcal{C}(\kappa, \mathsf{state})$, truncates the last $T(\kappa)$ records of it, and outputs the results.*

Theorem 8. *Let $T(\cdot)$ be a strictly positive, super-constant, polynomial, $(\Pi, \mathcal{C})$ a blockchain protocol satisfying chain growth g, chain quality μ and chain consistency in Γ-environments, where μ and g are strictly positive. Then, for every $\delta > 0$, the public ledger $(\Pi', \mathcal{L})$ $T(\cdot)$-induced by $(\Pi, \mathcal{C})$ is persistent and live with wait-time $w(\kappa, n, \rho, \Delta) = (1+\delta)\frac{T(\kappa)}{g(\kappa,n,\rho,\Delta)}$ in Γ-environments.*[16]

Proof. Consider Γ-admissible $n(\cdot), \rho, \Delta(\cdot), A, Z$, some $\delta > 0$, some κ, and some view $\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi,\mathcal{L})}(A, Z, \kappa)$. Let $n = n(\kappa), \Delta = \Delta(\kappa), g = g(\kappa, n, \rho, \Delta)$ and $\mu = \mu(\kappa, n, \rho, \Delta)$, $T = T(\kappa)$. We now separately show liveness and persistence.

[16] We are grateful to Elaine Shi for pointing out that a variant of our proof for the liveness property works with a sharper wait-time bound. Our original theorem and

Liveness. Let $T' = (1+\delta)T$ and let $t = \frac{T'}{g}$. Pick δ' such that $0 < \delta' < \delta$. Condition on the events that $\mathsf{growth}^t(\mathsf{view}, \Delta(\kappa), T') = 1$, $\mathsf{consistent}^{\delta' T - 1}(\mathsf{view}) = 1$, and $\mathsf{quality}^{T'-T}(\mathsf{view}) = 1$; by our assumptions and the union bound, these events occur with probability $1 - \mathsf{neg}(T)$; since T is polynomial in κ, these events occur except with probability $\mathsf{neg}(\kappa)$. Let j, j' be players such that in view, j' is honest at r and j is honest at $r + t$ such that $r + t \leq |\mathsf{view}|$.

By the conditioning, we have that:

- By chain growth, $|\mathcal{C}_j^{r+t}(\mathsf{view})| - |\mathcal{C}_{j'}^{r}(\mathsf{view})| \geq T + \delta T$; thus $|\mathcal{C}_j^{r+t}(\mathsf{view})| \geq T + \delta T$
- By "truncation", at least δT records that were not part of the chain of j' at r are thus output as part of j's ledger.
- By consistency, before round r, no honest player has *ever* had a chain whose length exceeds $|\mathcal{C}_{j'}^{r}(\mathsf{view})| + \delta' T - 1$.
- Thus, we have a segment of length at least $(\delta - \delta')T$ of records in $\mathcal{C}_j^{r+t}(\mathsf{view})$ which is output as part of j's ledger such that each record appears at a position which exceeds $|\mathcal{C}_{j'}^{r}(\mathsf{view})| + \delta' T$. By (strictly positive) chain quality, at least $(\delta - \delta')T\mu > 0$ records at a position exceeding $|\mathcal{C}_{j'}^{r}(\mathsf{view})| + \delta' T$ are "non-adversarial"; since no honest player ever had a chain of length $|\mathcal{C}_{j'}^{r}(\mathsf{view})| + \delta' T$ before round r, these non-adversarial records must have been provided by the environment at or after round r.

Persistence. Let $t = \frac{T}{g}$. Condition on the events that $\mathsf{growth}^t(\mathsf{view}, \Delta(\kappa), T) = 1$ and $\mathsf{consistent}^T(\mathsf{view}) = 1$; by our assumptions and the union bound, these events occur with probability $1 - \mathsf{neg}(T)$; since T is polynomial in κ, these events occur except with probability $\mathsf{neg}(\kappa)$.

Consider players i, j such that in view, i is honest at round r, and j is honest at round r' such that $r' \geq r + \Delta$. By the conditioning, we have that:

- Because $\mathsf{consistent}^T(\mathsf{view}) = 1$, prefixes of $\mathcal{C}_i^r(\mathsf{view})$ and $\mathcal{C}_j^{r'}(\mathsf{view})$ consisting of the first $|\mathcal{C}_i^r(\mathsf{view})| - T$ records are identical.
- By the consistent-length property of the chain-growth property, it also follows that $|\mathcal{C}_j^{r'}(\mathsf{view})| \geq |\mathcal{C}_i^r(\mathsf{view})|$.

By the above two statements, and the fact that $\mathcal{L}$ simply truncates the last T records of the chain, it follows that $\mathcal{L}_i^r(\mathsf{view})$ is a *prefix* of $\mathcal{L}_j^{r'}(\mathsf{view})$. Therefore, if $\mathcal{L}_i^r(\mathsf{view})$ contains a message m at position p, then so does $\mathcal{L}_j^{r'}(\mathsf{view})$. Because this holds for all such $r, r' > r + \Delta, i, j$, it follows that $\mathsf{persist}_{\Delta(\kappa)}(\mathsf{view}) = 1$.

---

proof (which set parameters in a non-optimal way) only claimed $w(\kappa, n, \rho, \Delta) = \frac{(1+\delta)T(\kappa)}{\mu(\kappa,n,\rho,\Delta)\cdot g(\kappa,n,\rho,\Delta)}$. The reason we do not need a dependency on μ is that by our definition of chain quality, it suffices for the fraction of non-adversarial blocks to be *positive* (as opposed to greater than $\frac{1}{T}$) to conclude the existence of at least one non-adversarial block.

Corollary 9. *For any $\lambda > 1$, any $\delta > 0$, any $p(\cdot)$, and any strictly positive, super-constant, polynomial $T(\cdot)$, the public ledger $(\Pi_{\mathsf{Nak}}, \mathcal{L}_{\mathsf{Nak}})$ that is $T(\cdot)$-induced by the blockchain protocol $(\Pi^p_{\mathsf{Nak}}, \mathcal{C}^p_{\mathsf{Nak}})$ is persistent and live with wait-time*

$$w(n, \kappa, \rho, \Delta) = (1 + \delta)\,\frac{T(\kappa)}{\gamma}$$

in Γ^p_λ environments.

Proof. From Theorems 4, 5 and 6, for every δ', δ'', $(\Pi^p_{\mathsf{Nak}}, \mathcal{C}^p_{\mathsf{Nak}})$ has growth $(1 - \delta')\gamma$, chain quality $1 - (1 + \delta'')\frac{\beta}{\gamma}$, and satisfies consistency. It can be shown that the chain quality is thus strictly positive. From Theorem 8, for every δ'', $(\Pi_{\mathsf{Nak}}, \mathcal{L}_{\mathsf{Nak}})$ thus has rate

$$w(n, \kappa, \rho, \Delta) = (1 + \delta''')\frac{T(\kappa)}{(1 - \delta')\gamma} < (1 + \delta)\,\frac{T(\kappa)}{\gamma}$$

where the last inequality follows by picking sufficiently small δ', δ'''.

6 An Attack on Nakamoto with Long Delays

In this section, we formally demonstrate that Nakamoto's protocol satisfies neither consistency nor positive chain quality in a fully asynchronous network without an upperbound Δ on the network delay, even if the adversary controls just a tiny fraction of computational power. More specifically, we show that for every hardness parameter p, $\Pi^p_{Nak}, \mathcal{C}^p_{Nak}$, satisfies neither consistency nor chain quality when $\Delta = \frac{1+\delta}{\rho np}$ for some $\delta > 0$. This demonstrates why our consistency theorem needs to rely on the assumption that $p \leq \frac{\Theta(1)}{\Delta n}$, and why the chain quality is $1 - \frac{\beta}{\gamma}$ as opposed to just $1 - \frac{\beta}{\alpha}$ (recall that $\gamma = \frac{\alpha}{1+\Delta\alpha}$ is a discounted version of α that takes delays into account.) In particular, we present a "51%" attack a la Nakamoto in which the attacker at some point in the future replaces the *whole chain* with a chain of its choice, even if it only controls a small fraction of the computational power.

Intuitively, in every segment of Δ rounds, if we delay all messages between honest players until the end of the segment, honest players are effectively "mining on their own" and thus are unlikely to extend their chain by more than 1. The adversary, on the other hand, coordinates its mining and thus in expectation extends its chain by $\Delta \cdot \rho np$; if we set $\Delta > \rho np$ the adversary can mine its own *longer* chain (without sending it to the honest player).

Theorem 10 (Inconsistency of Nakamoto with Unbounded Delays). *Let $\widehat{\Gamma}^p_{\rho',\delta}(\kappa, n, \rho, \Delta) = 1$ iff (1) $n = \frac{2}{\rho^2} \cdot \kappa$, (2) $\rho = \rho'$ and (3) $\Delta = \frac{1+\delta}{\rho np}$. For every $0 < \delta < \frac{1}{2}, 0 < \rho' < 1$, and every inverse polynomial $p(\cdot)$, $(\Pi^p_{Nak}, \mathcal{C}^p_{Nak})$ does not satisfy neither consistency nor chain quality q in $\widehat{\Gamma}^p_{\rho',\delta}$-valid environments, where $q > 0$.*

Proof. Consider an environment Z that invokes $n = \frac{2}{\rho^2} \cdot \kappa$ players, a fraction ρ of them being adversarial, and sends messages $m_1, m_2 \ldots$ to the honest players; for simplicity, assume $m_i = 0$ for all i. The environment runs for $\kappa\Delta + 1$ steps.

The attacker A proceeds as follows:

- A divides the rounds into κ segments of Δ rounds and delays all messages sent by honest players within such a segment to the end of it (note that this means no messages are delayed more than Δ);
- A ignores the content of the messages sent by honest players and tries to independently build its own chain $\hat{C}$ with messages $m'_1, m'_2, \ldots$ such that $m_i \neq m'_i$ for $\kappa\Delta$ rounds (for simplicity, assume $m'_i = 1$ for all i);
- In the next to last round $r = \kappa\Delta$, it sends $\hat{C}$ to any (strict) subset of the honest players (and delivers it instantly).

Note that in any view $\mathsf{view} \in \mathsf{EXEC}^{(\Pi^V, \mathcal{C})}(A, Z, \kappa)$ where (1) $|\hat{C}| > \kappa$ and (2) $\hat{C}$ is longer than the longest chain known to the honest players, we have that $\mathsf{consistent}^{\kappa}(\mathsf{view}) = 0$ and $\mathsf{quality}^{\kappa}(\mathsf{view}, 1) = 0$. We show that the probability that both events happen is constant, which proves the theorem.

The following two claims bound the probability that either event does not happen; by a union bound we can then conclude that the probability that both happen is constant.

Claim. Let $\hat{C}(\mathsf{view})$ denote the length of the adversary's chain in the next to last round (i.e., round $\kappa\Delta$) of view. Then,

$$\Pr[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi^V, \mathcal{C})}(A, Z, \kappa) : |\hat{C}(\mathsf{view})| < (1 + \frac{\delta}{2})\kappa] \leq e^{-\Omega(\kappa)}.$$

Proof. In the $\kappa\Delta$ rounds, the adversary has $\rho np \cdot \kappa\Delta$ chances to mine a block; each chance succeeds with probability p; since $\Delta = \frac{(1+\delta)}{\rho np}$, the expected number of mined blocks is thus $(1 + \delta)\kappa$. The desired bound thus follows directly from the Chernoff bound.

Claim. Let $\ell(\mathsf{view})$ denote the length of the longest chain known to the honest players in the last round of view. Then,

$$\Pr[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi^V, \mathcal{C})}(A, Z, \kappa) : \ell(\mathsf{view}) \geq \kappa] \leq \frac{3}{4}.$$

Proof. In every fixed segment of Δ rounds, the number of blocks mined by a *single* honest player is distributed as a binomial distribution with parameters Δ (trials) and p (success probability). Let X be such a random variable. The probability that some *fixed* single honest player mines more than 1 block in any *fixed* segment is

$$\begin{aligned} Pr[X > 1] &= 1 - Pr[X \leq 1] = 1 - Pr[X = 0] - Pr[X = 1] \\ &= 1 - (1 - p)^{\Delta} - \Delta p(1 - p)^{\Delta - 1} \\ &= 1 - (1 - p)^{\Delta - 1}(1 + (\Delta - 1)p) \end{aligned}$$

$$\leq 1 - (1 - (\Delta - 1)p)(1 + (\Delta - 1)p)$$
$$= (\Delta - 1)^2 p^2 \leq \frac{(1+\delta)}{\rho^2 n^2} \leq \frac{(1+\delta)}{2n\kappa}$$

By a union bound over the number of players n and the number of segments κ, we have that except with probability $\frac{1+\delta}{2} \leq \frac{3}{4}$, no honest player mines more than one block in any segment, and whenever that happens, the length of the longest chain grows by at most 1 for each segment and thus becomes of length at most κ after κ segments.

Remark 11. *We note that our proof applies even in the setting of static corruptions, and already to a weaker notion of consistency which ignores "future-self consistency". In addition, the attacker never looks at the messages sent by honest players.*

Remark 12. *We additionally point out that at the cost of complicating the proof (and increasing the number of players), we can obtain an even stronger attack—which works also when* $\Delta > \frac{1}{c \cdot np}$ *where* $\frac{1}{c} > \frac{1}{\rho} - \frac{1}{1-\rho}$ *(as opposed to just* $\frac{1}{\rho}$ *as in our previous proof)—as follows: instead or partitioning the rounds into segments, simply always delay messages between honest players by* Δ. Intuitively *(but significantly over-simplifying), when we delay the messages between honest parties by* Δ*, the expected time they need to wait until finding* and propagating *a block is roughly* $\frac{1}{(1-\rho)np} + \Delta$*, whereas the adversary only needs to wait* $\frac{1}{\rho np}$ *in expectation; thus, the attacker succeeds whenever it mines faster (i.e., when* $\frac{1}{\rho} < \frac{1}{1-\rho} + \Delta np$*), and since* $\Delta np = \frac{1}{c}$*, the attack succeeds when* $\frac{1}{c} > \frac{1}{\rho} - \frac{1}{1-\rho}$.

We turn to describe how to formalize this attack (following the proof of the second claim above). We, in fact, show an attack that works as long as $\beta > \gamma$ (i.e., the adversary mining rate is higher than the "discounted" honest player mining rate), and then use this to deduce that the attack applies when $\frac{1}{c} > \frac{1}{\rho} - \frac{1}{1-\rho}$.

It follows using exactly the same proof as the *lowerbound* on chain growth in the "hybrid" model (see the full version of this paper) that we can get $(1+\delta)\gamma$ as an *upperbound* on the chain growth of the honest players in a modified game where all honest players "freeze" for Δ rounds whenever some honest player mines a block. Since successes in each round are independent, it follows that *conditioned* on no single player ever mining two blocks within Δ rounds, the chain growth of honest players is upperbounded by $(1+\delta)\gamma$, whereas the chain growth of the adversary is lowerbounded by $(1-\delta)\beta$. Thus when $\beta > (1+\delta')\gamma$, if we run the experiment for t steps (and condition on no single player ever mining two blocks within Δ rounds), we get an attack except with probability $e^{-\Omega(\gamma t)}$. Since γ is monotonically increasing in α and $\alpha \leq (1-\rho)np$, it follows that the above also holds when[17]

$$\beta = \rho np > \frac{(1-\rho)np}{1 + \Delta(1-\rho)np}$$

[17] For readability, we ignore the $(1+\delta')$ term.

and thus when

$$\Delta np > \frac{1}{\rho} - \frac{1}{1-\rho}$$

So if we set $\Delta = \frac{1}{cnp}$, we get an attack (conditioned on no single player ever mining two blocks within Δ rounds) when $\frac{1}{c} > \frac{1}{\rho} - \frac{1}{1-\rho}$.

Finally, as in the proof of the second Claim above we have that at any given round r, for any fixed player j, the probability of j mining more that 1 block within the next Δ rounds is upperbounded by $(\Delta - 1)^2 p^2 \leq \frac{1}{c^2 n^2}$. Thus, if we set $n > 2t$, it follows that no player every mines more than 1 block within Δ rounds, except with probability 1/2 (by the union bound).

Acknowledgements. We are extremely grateful to Elaine Shi for many helpful comments on an earlier draft of this paper, and in particular for suggestion of how to sharpen the parameters in the construction of a public ledger from a blockchain.

References

[AJK05] Aspnes, J., Jackson, C., Krishnamurthy, A.: Exposing computationally-challenged byzantine impostors (2005)

[BCL+05] Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005). doi:10.1007/11535218_22

[BK14] Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_24

[Blo16] Blockchain.info. Hash rate for blockchain, February 2016. https://blockchain.info/charts/hash-rate

[BTP] BTProof. https://www.btproof.com

[Can00] Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000). http://eprint.iacr.org/2000/067

[CL99] Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: OSDI 1999 (1999)

[CLLM12] Chung, K.-M., Lam, H., Liu, Z., Mitzenmacher, M.: Chernoff-hoeffding bounds for markov chains: generalized and simplified. In: 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, 29th February – 3rd March 2012, Paris, France, pp. 124–135 (2012)

[CSWH00] Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: a distributed anonymous information storage and retrieval system. In: Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability (2000)

[DLN02] Karger, D., Liben-Nowell, D., Balakrishnan, H.: Analysis of the evolution of peer-to-peer systems. In: PODC 2002 (2002)

[DW13] Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE International Conference on Peer-to-Peer Computing, pp. 1–10 (2013)

[DR01] Druschel, P., Rowstron, A.: Past: persistent and anonymous storage in a peer-to-peer networking environment. In: HotOS 2001, pp. 65–70 (2001)

[DLS88] Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. ACM (JACM) **35**(2), 288–323 (1988)
[DN92] Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). doi:10.1007/3-540-48071-4_10
[ES14] Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 436–454. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45472-5_28
[FVY14] Fromknecht, C., Velicanu, D., Yakoubov, S.: A decentralized public key infrastructure with identity retention. IACR Cryptology ePrint Archive 2014, 803 (2014)
[GKL15] Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_10
[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
[KP15] Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols (2015)
[KMS+15] Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. Technical report, Cryptology ePrint Archive, Report 2015/675 (2015). http://eprint.iacr.org
[Lam10] Lamport, L.: Byzantizing paxos by refinement (2010)
[Lam11] Lamport, L.: Leaderless Byzantine Paxos. In: DISC 2011 (2011)
[Lit] Litecoin. https://litecoin.org
[MA05] Martin, J.-P., Alvisi, L.: Fast Byzantine consensus. In: DSN 2005 (2005)
[ML14] Miller, A., LaViola, J.J.: Anonymous Byzantine consensus from moderately-hard puzzles: a model for bitcoin (2014)
[mtg10] mtgox (2010). https://bitcointalk.org/index.php?topic=2227.msg29606#msg29606
[Nak08] Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
[Nam] Namecoin. https://www.namecoin.org
[Oku05a] Okun, M.: Agreement among unacquainted Byzantine generals. In: Fraigniaud, P. (ed.) DISC 2005. LNCS, vol. 3724, pp. 499–500. Springer, Heidelberg (2005). doi:10.1007/11561927_40
[Oku05b] Okun, M.: Distributed computing among unacquainted processors in the presence of byzantine failures (2005)
[OB08] Okun, M., Barak, A.: Efficient algorithms for anonymous Byzantine agreement. Theor. Comp. Sys. **42**, 222–238 (2008)
[PS15] Pass, R., Shelat, A.: Micropayments for decentralized currencies. In: CCS 2015 (2015)
[PS16a] Pass, R., Shi, E.: Fruitchains: an (almost) optimally fair blockchain (2016)
[PS16b] Pass, R., Shi, E.: Hybrid consensus (2016)
[PSL80] Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**, 228–234 (1980)
[PD15] Poon, J., Dryja, T.: The bitcoin lightning network: scalable off-chain instant payments draft 0.5.9.1 (2015). https://lightning.network/lightning-network-paper.pdf

[RFH+00] Ratanasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: SIGCOMM 2000 (2000)

[SZ15] Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 507–527. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47854-7_32

[SMK+01] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. In: SIGCOMM 2001 (2001)