A Better Method to Analyze Blockchain Consistency

Lucianna Kiffer, Rajmohan Rajaraman and abhi shelat
Northeastern University
{lkiffer,rraj}@ccs.neu.edu,a.shelat@northeastern.edu

ABSTRACT

The celebrated Nakamoto consensus protocol [16] ushered in several new consensus applications including cryptocurrencies. A few recent works [7, 17] have analyzed important properties of blockchains, including most significantly, *consistency*, which is a guarantee that all honest parties output the same sequence of blocks throughout the execution of the protocol.

To establish consistency, the prior analysis of Pass, Seeman and Shelat [17] required a careful counting of certain combinatorial events that was difficult to apply to variations of Nakamoto. The work of Garay, Kiayas, and Leonardas [7] provides another method of analyzing the blockchain under the simplifying assumption that the network was synchronous.

The contribution of this paper is the development of a simple Markov-chain based method for analyzing consistency properties of blockchain protocols. The method includes a formal way of stating strong concentration bounds as well as easy ways to concretely compute the bounds. We use our new method to answer a number of basic questions about consistency of blockchains:

- Our new analysis provides a tighter guarantee on the consistency property of Nakamoto's protocol, including for parameter regimes which [17] could not consider;
- We analyze a family of delaying attacks first presented in [17], and extend them to other protocols;
- We analyze how long a participant should wait before considering a high-value transaction "confirmed";
- We analyze the consistency of CliqueChain, a variation of the Chainweb [14] system;
- We provide the first rigorous consistency analysis of GHOST [20] and also analyze a folklore "balancing"-attack.

In each case, we use our framework to experimentally analyze the consensus bounds for various network delay parameters and adversarial computing percentages.

We hope our techniques enable authors of future blockchain proposals to provide a more rigorous analysis of their schemes.

ACM Reference Format:
Lucianna Kiffer, Rajmohan Rajaraman and abhi shelat. 2018. A Better Method to Analyze Blockchain Consistency. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, Article 4, 16 pages. https://doi.org/10.1145/3243734.3243814

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243814

1 INTRODUCTION

In 2008, Nakamoto [16] proposed the celebrated blockchain protocol which uses *proofs of work* to implement a public, immutable and ordered ledger of records suitable for applications such as cryptocurrencies. While standard consensus/Byzantine agreement mechanisms could be used to achieve such an immutable ordered sequence of records, the amazing aspect of Nakamoto's protocol is that it functions in a fully permissionless setting *and* works as long as more than half of the computing power in the network follows the protocol. In contrast, prior work on Byzantine agreement showed strong lower-bounds in fixed-party settings when even just one-third of the participants were adversarial.

Thus, it is remarkable that the honest parties using Nakamoto can reach agreement on a sequence of blocks. This property has been *strongly* validated by the Bitcoin network over almost 10 years of operation during which the participation has grown by 12 *orders of magnitude* from millions of hashes/second to million trillions of hashes/second!

To understand this phenomena, the original Nakamoto paper provided the first intuitive analysis as to how the protocol achieved consensus. Specifically, the paper shows that if an honest participant adds a block B to the chain and then waits for k more blocks to be added, the probability that an attacker (with less than 50% of the computational power) can build an alternative chain that does not include B drops exponentially with k. While intuitive, this analysis unfortunately does not consider other attacks, and thus does not fully establish the consensus property for the protocol. For example, the analysis does not consider an adversary that attempts to introduce small disagreements between honest miners so as to split their computational power among several "forks".

Garay, Kiayas and Leonardos [7] provided the first formal modeling of Nakamoto consensus and proved that the protocol achieved a *common prefix-property.* More specifically, if μ denotes the fraction of honest parties, ρ denotes the fraction of adversarial power, and p represents the hardness of the proof of work, they show that if $\mu > \lambda\rho$ for some $\lambda > 1$ such that $\lambda^2 - p\lambda + 1 \geq 0$, then "the blockchains maintained by the honest players will possess a large common prefix." However, their analysis only considered a "static setting in which the participants operate in a synchronous communication network in the presence of an adversary that controls a subset of the players." In particular, players were either honest or adversarial throughout the protocol, and the network model ensured delivery of messages "in the next round." Assuming a synchronous network is a very strong, possibly unrealistic assumption; indeed, Nakamoto's protocol is explicitly designed to work in a network *with message delays* such as the public Internet. Furthermore, the notion of *common-prefix* is not strong enough because it does not preclude the chain from alternating between two different versions on even and odd rounds.

To address these issues, Pass, Seeman, and Shelat [17] provide a different formal model for studying the properties of Nakamoto's protocol. In particular, they introduce an idealized model for the protocol execution that can capture adaptive corruptions (and uncorruptions) of parties and a *partially synchronous* network in which the adversary can adaptively and individually delay messages up to some delay limit Δ. They also introduce a stronger notion of consistency and then proceed to show how Nakamoto can achieve this notion when the hardness of the proof of work p is set with an appropriate relation to Δ and the number of participants n. Their analysis is precise enough to make concrete claims about the relationship between p, Δ, and n for which consistency holds and also for which a simple delay attack can violate consistency.

In subsequent work in 2017, Garay, Kiayas and Leonardos [8] studied aspects of how the hardness p is adjusted as more players join the protocol during *epochs* in the Nakamoto consensus protocol and how this epoch must be sufficiently large to avoid certain attacks. Techniques from this paper were also used to update [7]; in particular, the updated version of the latter extends the analysis to the partially synchronous model but is not precise enough to make concrete claims about the parameters.

1.1 Contributions

In this paper we show how Markov chains can be used to model blockchain protocols to both simplify the analysis of blockchain protocols and attacks and to make precise claims about parameter relationships. To introduce our method, in §4.2 we show how to replicate the analysis of Nakamoto's protocol done by Pass et al. [17] using a Markov model. We validate our method by recovering essentially the same bound.

By inspecting this graphical Markov model, however, we discovered cases in which the counting of events in [17] under-counts a special event. Based on this insight, we show in §4.3 how our Markov model *exactly* counts so called "convergence opportunities" and thus leads to more accurate bounds on consistency (see Thm 4.4). To illustrate this new result, Fig. 1 replicates a graph from [17] showing a relationship between proof of work hardness and adversary control. Against the original analysis (- - -) and attack (—■—), our new result from the Markov model (——) shows higher resilience at lower parameters where the previous analyses provided no lower bounds.

To further illustrate our technique, in §5 we introduce Cliquechain, a specific example of the Chainweb protocol [14] for which we can use our techniques to show the same consistency lower-bound as Nakamoto's protocol. Chainweb proposes a blockchain protocol that creates a braid of various parallel chains. The main idea is that at each level the chains reference each other according to some base graph, and thus in order to replace one block in any chain, you must also replace the blocks in the parallel chains that reference it. The protocol claims to be able to handle 10K transactions per second over hundreds or thousands of parallel chains, but the analysis in the paper again only considers variations of the 50% attack. Our analysis on consistency does not support this 10k claim. Bitcoin, running Nakamoto's protocol with network delay of about 10 seconds, can handle approximately 7 txns/sec, we show that the Cliquechain protocol with any number of chains is bounded by the same throughput as Nakamoto's protocol for the same consistency guarantee.

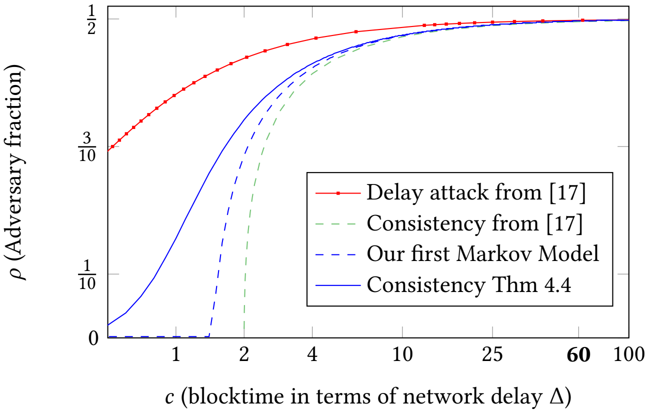

Figure 1: Replication of [17, Fig. 1], "Minimum percentage of computing power an adversary must hold in order to break consistency" using same parameters $n = 10^5$ and $\Delta = 10^{13}$, $p = \frac{1}{c\Delta}$, but illustrating the new bound from our Thm. 4.4.

In §6, we extend our techniques to establish that the protocol GHOST [20] also has the same consistency lower bounds. GHOST's main claim is also that it can handle higher transaction rates than Nakamoto's protocol while being resilient to 50% attacks.

For each protocol we also analyze variants of attacks using our technique and provide probability distributions for how long the attacks last. In §4.4 we first show a very simple Markov model that captures the '50% attack' on Nakamoto presented in [17]. As discussed so far, the notion of blockchain consistency considered in the literature is still an asymptotic notion that requires the "probability of a fork" to fall exponentially with some parameter T. We use this same attack model to answer very pragmatic questions about blockchains: For example "how long should one wait before confirming a transaction?" While folklore holds that one should ignore (i.e., wait for) the last $T = 6$ blocks, we provide a more precise answer to this question by modeling an attack in which the goal of the adversary is to "undo" a recently confirmed transaction. We show, for example, that $T = 6$ is a surprisingly low default for chains like Ethereum which use more aggressive network parameters.

Since this delay attack is not successful on GHOST, we present another attack, the 'Balancing Attack', with a Markov chain in §6.4 which captures a simplified version of the attack on GHOST. We use this model to capture a lower bound on transaction confirmation time for the GHOST protocol in Fig. 11

Our work is not the first to employ Markov-based analysis of blockchains. Indeed, Eyal and Sirer [6] develop a Markov model to analyze the success of the selfish mining attack on Nakamoto's protocol. Our work, however, is the first to use Markov-based models to analyze consistency against any adversary, and provides a general framework to analyze specific attacks on various blockchain protocols. Previous studies on consistency [7, 8, 17] have advocated using Markov methods, but have considered them too complicated to analyze: for instance, the authors of [17] write "the Markov chain that arises from this problem is too complicated to analyze using standard concentration bounds for Markov chains." We show that consistency of several blockchain protocols as well as the impact of specific attacks are well-captured by natural Markov chains, and can be analyzed using a judicious combination of (a) derivation of steady-state distributions; (b) concentration bounds for generalized Markov chains; (c) generating functions for deriving probability of significant events; and (d) simplification of the Markov chain, where appropriate. Even in cases, where we are unable to derive closed-form bounds in terms of the parameters involved, we are still able to numerically calculate the measures of interest.

We hope that subsequent papers proposing blockchain protocols can employ these techniques to frame and analyze properties about their protocols.

1.2 Related Work

Chainweb and GHOST are two examples of a new class of blockchain protocols which consider DAG-based chains instead of a linear chain. Other such examples include the inclusive protocol of Lewenberg, Sompolinsky and Zohar [13] which use the DAG to reward miners for work but whose security is inherited by the security of GHOST or any other tree-based chain selection policy. SPECTRE, another protocol from the authors of the inclusive protocol [19], extends the DAG idea by having miners point to all recent child-free blocks they know in any newly mined block. They claim security by relaxing the restriction on linearity of transactions. PHANTOM [21] builds on SPECTRE and organizes the DAG-based chains in such a way that the blocks mined by honest players form a well-connected cluster of blocks; in particular, their aim is to ensure that the largest set of blocks with inter-connectivity exceeding a certain threshold is composed of honest mined blocks. However, none of these protocols give a formal argument for the consistency properties put forth in either [7, 17], but instead mostly rule out specific attacks.

In [11] and [12], Kiayias et al. analyze both Nakatomoto's protocol and GHOST in a synchronous setting where honest messages are delivered in the following round, but the adversary can reorder them. They show an attack on the growth rate of the main-chain to delay transaction confirmation time in this model of both GHOST and Nakamoto's protocol. Their results show that the attack produces greater delays in the GHOST protocol. In our analysis we consider a stronger adversary that can delay honest messages up to the network delay, and focus on consistency attacks (instead of chain growth attacks) which also delay confirmation time.

Sleepy [18] proposes a new blockchain in a model with a CRS and a PKI and participants who sometimes become inactive; they show how to replace a proof of work with another rate-limiting mechanism. We believe our techniques directly apply because they apply the same counting as [17] (see, e.g. Lemma 2 in [18]).

The Algorand schemes [2, 9] construct a blockchain from improved Byzantine agreement protocols; as far as we can tell, they require a 2/3 fraction of honest users and thus rely on different techniques for proving consistency.

The choice of network delay in this paper is supported by measurements of real delays in active blockchain systems [5]. Apostolaki et al. show how an ISP can partition the Bitcoin network and delay messages [1] thus justifying our choice to allow the adversary the power to rearrange and delay messages between players.

2 THE MODEL

We rely on the formalization of blockchain protocols introduced by Garay, Kiayas and Leonardos [7] and Pass, Seeman, and shelat [17].

A blockchain is a pair of algorithms (Π, ext); Π is a stateful algorithm that maintains a local state variable C—called the chain—which contains a set of abstract records called *blocks*, each of which contains a message m. The algorithm ext maps a set of blocks to a sequence of messages; e.g. $\mathrm{ext}(C)$ denotes the sequence of messages obtained by applying ext to C. The overall aim is for players to receive messages as inputs and then attempt to include their message in their own chain and those of others.

A blockchain protocol is executed in a partially asynchronous network model that involves the following components: (We use κ to denote the security parameter)

Environment: An environment, represented by $Z(1^{\kappa})$, is used to model all of the *external* factors related to an execution. It activates the n players, each either *honest* or *corrupt* and provides all of the inputs for the protocol.[1]

Honest players: The honest players run a given blockchain protocol specified by (Π, C); each honest player keeps a copy of their current view of the blockchain and tries to contribute to it by building blocks at the end of their chain.

Adversary: The corrupted players, who are at most a ρ fraction of the n players, are controlled by an adversary A. The adversary is given two advantages: (a) the adversary is able to delay and reorder all messages players receive up to a delay of Δ rounds; (b) the adversary can control the actions of each corrupt node; for instance, all corrupt nodes could work on the same block or different ones. Thus, the model gives *more power* to the adversary than might be realistic in an actual deployment, thus yielding conservative bounds on the performance of the system.

Random Oracle: 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 $\mathsf{H.ver}(x, y)$ outputs 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 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.

Protocols based on proof of work are parametrized by p—the *mining hardness parameter.* Informally, a proof-of-work for the block h_{-1} and message m is a string η such that $\mathsf{H}(h_{-1}, \eta, \mathrm{m}) < D_p$, where D_p is set so that the probability that an input satisfies the relation is less than p.

An execution of a blockchain protocol begins with Z which can instantiate n players, each of them with identical computing power. The protocol proceeds in rounds; at each round each player i receives some message from Z (e.g. transactions to be included in the blockchain), blocks created by other players, as well as the opportunity to make a query to oracle H. They include these blocks

[1] For technical reasons, the environment and Adversary must satisfy certain restrictions which we do not discuss here; see [17] for a description of *admissable* Z, A

in their chain based on the protocol Π, and include the message in the block they are trying to publish.

The adversary A controls a ρ fraction of the players and thus gets ρn random oracles queries in each round. A is responsible for delivering messages sent by the parties to all other parties. A cannot modify the content of messages broadcast by honest players, but it may delay or reorder their delivery as long is it eventually delivers all messages within some bound Δ. The environment Z can communicate with the adversary or access the local state variable $C(\mathsf{state}_i)$ of player i (i.e., player i's chain) at any point. At any given time, Z can either corrupt an honest party j which means that the adversary gets to control j's local state and messages, or uncorrupt a party j which means that A no longer controls j and instead player j starts executing the protocol $\Pi(1^\kappa)$ starting from an empty state.

We summarize the main parameters of a blockchain:

Δ	the network delay bound
$p = \frac{1}{c \cdot n\Delta}$	the mining hardness is expressed in terms of parameter c, roughly the expected number of network delays before some block is mined
ρ	the adversarial fraction of parties
$\mu = 1 - \rho$	the fraction of honest parties

A useful blockchain protocol offers three main properties that are parameterized by T: (a) *chain-growth*, i.e., at any point in the execution of the protocol, the chain of the honest players grows by T blocks in the last $O(T)$ rounds with very high probability (in T), (b) *chain-quality*, for any T consecutive blocks in any chain held by some honest player, $\Theta(T)$ blocks were contributed by honest players, and finally (c) *consistency*: for any T, with overwhelming probability (in T), at any two rounds r and s with $r < s$, all but the last T blocks in the chain of any honest player i at r must be a prefix of the chain of an honest player j at s.

If we establish consistency, honest parties are guaranteed that for sufficiently large T, *confirmed* blocks will never be lost from the chain except with tiny probability (which is the property needed for all the above-mentioned applications; for instance, in bitcoin, it ensures that players cannot double-spend money).

Although our techniques can apply to growth and quality, the main focus of this paper is to analyze consistency. Formally, we model an execution of the protocol through a random variable $\mathsf{EXEC}^{(\Pi, C)}(A, Z, \kappa)$ denoting the joint view of all players (i.e., all their inputs, random coins and messages received, including those from the random oracle) in an execution.

Let view be a joint view in the support of $\mathsf{EXEC}^{(\Pi, C)}(A, Z, \kappa)$, and let view^r denote the prefix of view up until round r. Let $C_i^r(\mathsf{view})$ denote the record chain in the local state of player i in the prefix of view until round r. Let $\mathsf{consistent}^T(\mathsf{view}) = 1$ iff for all rounds $r \le r'$, 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 obtain that the prefixes of $C_i^r(\mathsf{view})$ and $C_j^{r'}(\mathsf{view})$ consisting of the first $\ell = |C_i^r(\mathsf{view})| - T$ records are identical.

Definition 2.1. A blockchain protocol satisfies *consistency*, if 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:

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

$$\ge 1 - \epsilon_1(\kappa) - \epsilon_2(T)$$

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).

3 A SIMPLE MARKOV FRAMEWORK FOR ANALYZING BLOCKCHAIN PROTOCOLS

We present a simple analysis framework for blockchains that combines the approach of Pass et al. with natural *Markov chains* that capture protocol dynamics in the presence of adversaries.

At a high-level, each state of our Markov chain represents an initial state of the blockchain system or the state of the system following an event of interest. The events of interest include (a) a new block mined by an honest player; (b) a new block mined by the adversary; and (c) a sufficiently long quiet period. Each edge represents an event and has an associated length, which is a random variable denoting the time it takes for the event to occur, conditioned on its occurrence. The actual definition of the Markov chain (the states and the particular events of interest) depends on the protocol being analyzed, a model of the adversary or an attack being considered, and the performance measure of interest.

All of the Markov chains used in our analysis satisfy the following properties: (a) they are *time-homogeneous*; that is, the probability of transitioning from one state to another is only dependent on the states, and not on the time at which the transition occurs; (b) they are *irreducible*; that is, it is possible to get to any state from any other state; and (c) they are *ergodic*; that is, every state is aperiodic and has a positive mean recurrence time. We refer the reader to a standard text on probability theory or randomized algorithms (e.g., [15, Chapter 7]) for more information on Markov chains.

Stationary distribution: Every time-homogeneous, irreducible, ergodic Markov chain has a (unique) stationary distribution π. For any given state v, $\pi(v)$ represents the limit, as n tends to ∞, of the probability that the chain will be in state v after n transitions (independent of the starting state). Stationary probabilities can also be defined for the edges of the chain. Once we define a Markov chain to model a certain aspect of a blockchain protocol, we derive the stationary distributions for each state and edge of the Markov chain, through a set of difference equations. This yields, for each state v, the stationary probability $\pi(v)$ of being in state v. Thus, for any sequence of T rounds, the expected number of visits to v tends to $\pi(v)T$, as T grows. A similar calculation can be done for each edge of the Markov chain.

Concentration bounds: The stationary distribution of a Markov chain captures the expected number of occurrences of a particular state (or transition) over a long sequence of transitions. In any particular instance of the sequence (e.g., a particular run of the protocol), however, the exact number of occurrences of a state of interest could certainly deviate from the expectation. Many "well-behaved" Markov chains satisfy tight concentration bounds which indicate that the probability that a given measure of interest (e.g., the number of occurrences of a particular event) deviates from the expected value of the measure is exponentially small in the size of the deviation. Such concentration bounds enable us to establish that for a sufficiently long sequence of rounds, the measure of interest (e.g., number of convergence opportunities) is with high probability close to the expectation given by the stationary distribution.

More formally, we invoke the following theorem on Chernoff-Hoeffding bounds for "generalized" Markov chains to derive concentration bounds for random variables of interest [4].

Theorem 3.1 ([4]). *Let M be an ergodic Markov chain with state space $[n]$ and stationary distribution π. Let $\mathcal{T}$ be its ϵ-mixing time for $\epsilon \le 1/8$. Let $(V_1, \ldots, V_t)$ denote a t-step random walk on M starting from an initial distribution ϕ on $[n]$, i.e., $V_1 \leftarrow \phi$. For every $i \in [t]$, let $f_i : [n] \rightarrow [0,1]$ be a weight function at step i such that the expected weight $E_{v \leftarrow \pi}[f_i(v)] = \mu$ for all i. Define the total weight of the walk $(V_1, \ldots, V_t)$ by $X = \sum_{i=1}^{t} f_i(V_i)$. There exists some constant c (which is independent of μ, δ, and ϵ) such that*

1. *$Pr[X \ge (1+\delta)\mu t] \le c\|\phi\|_\pi e^{-\delta^2 \mu t/(72\mathcal{T})}$ for $0 \le \delta \le 1$,*
2. *$Pr[X \ge (1+\delta)\mu t] \le c\|\phi\|_\pi e^{-\delta \mu t/(72\mathcal{T})}$ for $\delta > 1$, and*
3. *$Pr[X \le (1-\delta)\mu t] \le c\|\phi\|_\pi e^{-\delta^2 \mu t/(72\mathcal{T})}$ for $0 \le \delta \le 1$,*

where $\|\phi\|_\pi$ is the π-norm of ϕ given by $\sum_{i \in [n]} \phi_i^2/\pi(i)$.

Using the above theorem, we are able to establish our bounds with high probability; that is, the bound fails with probability that decreases exponentially in the number of rounds executed by the protocol. In order to apply the theorem, we show how to transform our original Markov chain and set the weight functions so that X captures the particular measure of interest. For instance, to measure the number of visits to a particular state v, we set $f_i(v)$ to 1 and set $f_i(u)$ to 0 for $u \neq v$, for all i. To measure the number of traversals of a particular edge, we add an auxiliary vertex in the middle of the edge, and set the measure to be number of visits to the auxiliary vertex. In all of our applications of the theorem, the transformed Markov chains are such that both the number of states and the mixing time are constant, independent of the number of rounds T, but possibly dependent on the model parameters, such as p and Δ. Thus, for every v, we are able to show that in T rounds, the number of visits to state v is $(1 \pm \delta)\pi(v)T$ with probability $1 - e^{-\delta\pi(v)\Omega(T)}$. Such concentration bounds enable us to focus our attention on analyzing the stationary distributions of the Markov chain.

In the following sections, we apply our approach to analyze three different blockchain protocols—Nakamoto, Cliquechain, and GHOST— and derive a range of analytical results: (a) consistency proofs for the protocols via bounds on convergence opportunities; (b) analysis of resilience against delaying and balancing attacks; (c) bounds on new performance measures (e.g., length of forks).

4 NAKAMOTO ANALYSIS

In this section, we analyze the Nakamoto protocol using our Markov framework. We begin by reviewing, in §4.1, Pass et al.'s analysis of Nakamoto using bounds on chain growth, block expiry, and the important notion of convergence opportunities they introduce for establishing consistency. In §4.2, we reconsider [17]'s analysis of convergence opportunities using our Markov framework, and show how their analysis yields an underestimate. In §4.3, we present a new lower bound for achieving consistency in Nakamoto's protocol by an improved analysis of convergence opportunities using our Markov chain. Finally, in §4.4, we present a detailed analysis of Nakamoto's protocol under a consensus attack, deriving bounds on the probability that the attack can force forks of a specific length.

4.1 Chain Growth, Block Expiry, Consistency

We begin with the lower bound of [17] on chain growth. Recall that the maximum number of rounds any message can be delayed is Δ. Let $\mu = 1 - \rho$ denote the fraction of honest players.

Lemma 4.1 (**Nakamoto Chain Growth**). *For any $\delta > 0$, the growth of the main chain of any honest player in Nakamoto's protocol in T rounds is at least $T(1-\delta)\frac{\mu}{\Delta(c+\mu)}$, except with probability that drops exponentially in T.*

Proof. For any $i \ge 1$, let T_i denote the number of rounds it takes for the main chain to grow from i to $i+1$. If an honest player mines a block for a chain length l at time r, by time $r + \Delta$ all honest players know about this block and will now mine a block for a chain of length at least $l+1$. The expected number of rounds for an honest player to mine a block is $\frac{c\Delta}{\mu}$; therefore, $E[T_i] \le \frac{c\Delta}{\mu} + \Delta$. The expected number of rounds for a chain growth of 1 is at most $\frac{c\Delta}{\mu} + \Delta$; using standard Chernoff-Hoeffding bounds [3, 10], the number of rounds for a chain growth of g is at most $(1+\delta)(\frac{c\Delta}{\mu} + \Delta)g$ with probability $1 - e^{-\Omega(g)}$. Thus, in T rounds, Nakamoto achieves chain growth of at least $(1-\delta)T\frac{\mu}{\Delta(c+\mu)}$ with probability $1 - e^{\Omega(T)}$. □

A key part of the consistency proof of Pass, Seeman, and Shelat relies on their "no long block withholding" lemma [17, Lemma 6.10], which states that if an adversary withholds a block for *too long*, it will not end up in the chain of any honest player. This lemma allows us to make statements about what an adversary is able to do in a given window of rounds without having to consider more than a constant number of blocks the adversary mined previously and didn't announce, which they may still use in an attack. In this section we restate that lemma within our framework. This lemma is useful when we redefine the consistency bounds of Nakamoto's protocol, and also for evaluating other protocols in later sections.

Using Lemma 4.1, we get an altered version of the block withholding lemma which we refer to as block expiry. Let b be a block mined by the adversary at time r, and let $r + t$ be the first time any honest player hears of b; we say b expires if there exists a negligible function $\epsilon(.)$ such that the probability b ends up in the mainchain of any honest player anytime after $r + t$ is $\le \epsilon(t)$.

Lemma 4.2 (**Nakamoto Block Expiry**). *There exists a $\delta \in (0,1)$ such that if $\mu \ge \delta\rho$, then every adversarial block expires.*

Proof. Let b be any adversarial block. We set δ such that the expected growth of any adversarial chain is smaller than the expected growth of any honest chain. For any T, the expected growth of any adversarial chain is at most $T\frac{\rho}{c\Delta}$. By a standard Chernoff-Hoeffding bound [3, 10], for any $\delta' > 0$, the probability that the adversarial chain grows by at least $(1+\delta')T\frac{\rho}{c\Delta}$ is at most inverse exponential in T. So, from Lemma 4.1, we set the parameters such that

$$\frac{\mu}{(c+\mu)} > \frac{\rho}{c}$$

. Thus, the probability that at any time after $r + t$, the adversary mines a chain longer than any honest player's chain at that time is $\le \epsilon(t\rho)$ where ϵ is inverse exponential in its argument. □

In the consistency analysis of Nakamoto, Pass, Seeman, and shelat consider any window of T rounds and count special events, called *Convergence Opportunities*, which are events after which all honest players agree on a single chain; we define them formally in the following subsection. If an adversary wants to break consistency, they must combat all convergence opportunities. To analyze what an adversary can do in a given window of T rounds, we must also argue that there are only a constant number of blocks the adversary mined before the window, which the adversary can use in an attack during the window. We now state our version of the consistency theorem of [17] for *any blockchain protocol* which states that if those two properties hold, then the protocol satisfies consistency.

Theorem 4.3 (**Blockchain Consistency**). *A blockchain protocol satisfies consistency if* $\exists \delta \in (0,1)$ *satisfying* $\mu \geq \delta\alpha$ *such that for any integer* T *and in any window of* T *rounds, with probability* $1-\epsilon(T)$ *for a negligible function* $\epsilon(\cdot)$*, the number of convergence opportunities* C *is greater than the number of adversarial blocks needed to break all convergence opportunities* A*, and the number of blocks mined before* T *which the adversary can use in* T *to break convergence opportunities is less than* $C-A$*.*

Using the above theorem, Pass et al derive the following condition for Nakamoto's protocol to achieve consistency, where $\alpha = 1-(1-p)^{(1-\rho)n}$ and $\beta = \rho np$.

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

4.2 Counting Convergence Opportunities Using Markov Chains

We reconsider the analysis of convergence opportunities in [17] using our Markov approach. A convergence opportunity is an event after which all honest players agree on a single block as the latest block and therefore agree on a single longest chain. The convergence opportunity is made up of 3 sequences of rounds, each characterized by the outcome of mining by the honest players.

- First, Δ rounds pass in which no honest player mines a block. Thus, by the model, at the end of the Δ rounds, all honest players know all honest blocks, and therefore agree on what is the maximum length of the chain (though not necessarily the same chain).
- Second, a single honest player mines, thus extending a chain by one more block than the previous longest chain.
- Third, another Δ rounds pass in which no honest player mines. Thus, at the end, all honest players know the new block and therefore agree on the single longest chain.

To prove that a given protocol achieves consistency, the analysis first argues that to prevent consensus, it is necessary for the adversary to "break" all convergence opportunities. An adversary can break a convergence opportunity by disrupting either of the quiet periods of step one and three by announcing one of their own blocks during that time. Thus, the analysis attempts to bound both the number of convergence opportunities the honest players have and the number of blocks the adversary must mine to break those. To obtain this count, the analysis in [17] sums over all honest blocks mined (*hits*) in any time interval and tracks whether the "quiet" period between honest hits is less than Δ. In a given period of L honest hits, let q denote the number of quiet periods between two honest hits that are less than Δ rounds, and let Q denote the same for quiet periods longer than Δ.

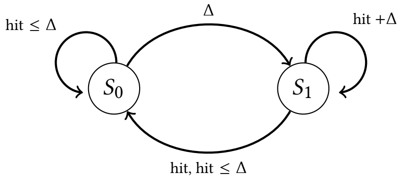

Figure 2: A simple Markov model for counting convergence opportunities

To arrive at their consistency proof, the consistency lemma [17, Lemma 6.11] derives a lower bound of $2Q-L$ on the number of convergence opportunities. Specifically, they show that except with probability $1-e^{-\Omega(\beta t)}$, there are at least $(1-\delta''')(1-2\alpha(\Delta+1))\alpha t$ convergence opportunities between any two rounds r and $r+t$, and moreover, an adversary only mined at most $(1+w'')(t+1)\beta$ blocks, for arbitrary small constants $\delta''', w'' \geq 0$.

Using a simple Markov chain, we show below that the above lower bound is not accurate; it may underestimate the true count of convergence opportunities.

Figure 2 presents a Markov model which precisely captures the count from Pass et al. It has 2 states: S_0 represents a "messy" state where honest mined blocks occur in less than Δ rounds from one another, while S_1 is the state where quiet periods between honest mined blocks is at least Δ rounds. As long as quiet periods are shorter than Δ rounds the system stays in S_0; otherwise we move to state S_1. Once in S_1, the system stays in S_1 as long as quiet periods between honest mined blocks are at least Δ rounds, otherwise the state changes to S_0. Let e_{ij} represent the edge from state S_i to state S_j. Below are the events that happen when each edge is traversed:

e_{00} = one quiet period of less than Δ rounds followed by a single honest mined block
e_{01} = one quiet period that is at least Δ rounds
e_{11} = a single honest mined block followed by a quiet period of at least Δ rounds
e_{10} = an honest mined block followed by one quiet period of less than Δ rounds followed by an honest mined block.

Consider a random walk on this Markov chain. We can compute the number of honest mined blocks by counting one block every time e_{00} or e_{11} is traversed, and 2 every time e_{10} is traversed. To calculate Q, we count the number of times e_{01} is traversed plus the number of times e_{11} is traversed. Letting E_{ij} represent the expected number of times e_{ij} is traversed, we have:

$$\begin{aligned} 2Q - L &= 2(E_{01}+E_{11}) - (E_{00}+E_{11}+2E_{10}) \\ &= 2E_{01}+E_{11}-2E_{10}-E_{00} \end{aligned}$$

Our analysis plan is to compare the expected fraction of events that are convergence opportunities with the expected fraction of events that are blocks mined by the adversary, and then invoke concentration bounds from Theorem 3.1. To calculate the expectations, we solve for the probability of crossing each edge, and the stationary probabilities. Let $P_\Delta = (1-\mu p)^\Delta$ be the probability of Δ

silent rounds.

$$\begin{aligned}\Pr[e_{00}] &= \Pr[e_{10}] = 1 - P_\Delta\\ \Pr[e_{01}] &= \Pr[e_{11}] = P_\Delta\\ \pi_0 &= \Pr[S_0] = (1-P_\Delta)\pi_0 + (1-P_\Delta)\pi_1\\ \pi_1 &= \Pr[S_1] = P_\Delta \pi_1 + P_\Delta \pi_0\end{aligned}$$

Since $\pi_0 + \pi_1 = 1$ we get that $\pi_0 = 1 - P_\Delta$ and $\pi_1 = P_\Delta$.

To calculate the expected number of times we hit each edge e_{ij}, we divide $\pi_i p_{ij}$ by the total weighted time spent on all edges, which in turn requires the expected time spent on each edge l_{ij}. Letting $p_{i|\leq\Delta} = \Pr[\text{hit at time } i| \text{ silence lasted } \leq \Delta] = \frac{p_i}{p_{\leq\Delta}}$, we get:

$$p_{i|\leq\Delta} = \frac{(1-\mu p)^{i-1}\mu p}{\sum_{j=1}^{\Delta}(1-\mu p)^{j-1}\mu p}; \qquad l_{00} = \sum_{i=1}^{\Delta} i p_{i|\leq\Delta}$$

$$l_{01} = \Delta; \quad l_{11} = \frac{1}{\mu p} + \Delta; \quad l_{10} = \frac{1}{\mu p} + \sum_{i=1}^{\Delta} i p_{i|\leq\Delta}$$

The total weighted time spent on all edges is $\sum_{i,j}\Pr[e_{ij}]\pi_i l_{ij}$. Thus $2Q - L$ is equal to $2(e_{01}\pi_0 + e_{11}\pi_1) - (e_{00}\pi_0 + e_{11}\pi_1 + 2e_{01}\pi_0)$ divided by the total weighted time spent on all edges. We simplify this to

$$\begin{aligned}&2(e_{01}\pi_0 + e_{11}\pi_1) - (e_{00}\pi_0 + e_{11}\pi_1 + 2e_{01}\pi_0)\\ &\quad= 2\cdot(P_\Delta(1-P_\Delta) + P_\Delta^2)\\ &\qquad - ((1-P_\Delta)^2 + P_\Delta^2 + 2 * P_\Delta(1-P_\Delta))\\ &\quad= P_\Delta^2 - (1-P_\Delta)^2\end{aligned}$$

We then calculate the total weighted time spent on the edges and plot the bound for convergence opportunities as

$$\frac{P_\Delta^2 - (1-P_\Delta)^2}{\sum_{i,j}\Pr[e_{ij}]\pi_i l_{ij}}$$

in Figure 1 as (- - -). Note this bound is slightly stronger than the same count from [17] because we use a more accurate probability for μ (while Pass et al. use a conservative approximation); these two calculations are equivalent when we use the same approximation.

In order to establish a concentration bound for the convergence opportunities count, we show the following. For each state v, the number of visits to v in T rounds is concentrated around the expected number of visits in T rounds with high probability; for each edge e, the number of visits to e as well as the time spent on e are concentrated around their respective expectations with high probability. Since the count we are measuring is a linear combination of the number of visits, we obtain the desired high probability concentration bound. We obtain these concentration bounds by an application of Theorem 3.1. Before we can apply the theorem, we transform the Markov chain to another equivalent Markov chain, presented in Figure 3, in which traversing each edge takes one step of the chain. Now, the number of visits to a vertex v in T rounds can be captured by the random variable X by setting $f_i(v)$ to be 1, and $f_i(u)$ to be 0 for all $u \neq v$, for all i. Theorem 3.1 immediately yields a bound that the number of visits to v in T rounds is within $(1 \pm \delta)$ of its expectation with probability $1 - e^{-\Omega(T)}$, where the hidden constant depends on Δ and p, factors that determine the mixing time of the transformed Markov chain.

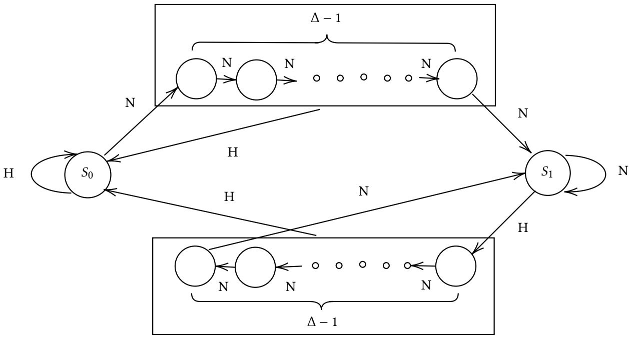

Figure 3: Markov chain equivalent to that in Figure 2. The label H (resp., N) on an edge marks event that a block (resp., no block) was mined by an honest player in the round. The edge labeled H from the two rectangular blocks of states represents an edge from each state in the blocks.

Problems with this counting. The analysis of Pass et al. lower bounded the number of convergence opportunities by counting the number of (honest) hits and by counting the number of "quiet" periods that were longer than Δ, and comparing this with an upper bound on the expected number of blocks the adversary can mine. We now show when this analysis underestimates the number of convergence opportunities, even getting a negative count.

Consider the following sequence of events where, slightly abusing notation, H represents a round with a "hit", Q represents at least Δ rounds with no mined blocks, and q represents a quiet period of fewer than Δ rounds,

$$H, q, H, Q, H, Q, H, q, H, q, H, q, \ldots,$$

the Pass et al. method underestimates the number of convergence opportunities as -2, when there should be 1. We see that when $c < 2$, multiple honest blocks are being mined in each Δ in expectation, so we are mostly looping in state S_0. In this setting, $2Q$ will be less than all honest blocks mined, so this analysis gives a negative count, i.e., an obvious underestimate!

In the following section we use our same Markov model to do an exact count of all convergence opportunities, so, at low c, i.e. high mining probability, we still get meaningful results.

4.3 An Improved Analysis of Convergence Opportunities

We present an improved analysis of convergence opportunities and bound the number of blocks the adversary would need to kill all convergence opportunities, i.e. break consistency. In the Markov model we created in the previous analysis to reproduce the count of Pass, Seeman and shelat, we notice that the edge $e_{1,1}$ looping on state S_1 exactly captures convergence opportunities. When we transition from S_0 to S_1 we get a big quiet period, a 'Q',and whenever we loop in S_1 we get a hit and big quiet period, i.e. 'HQ'. Thus looping in S_1 gives us a sequence of convergence opportunities. To *exactly* count the expected number of honest convergence opportunities we thus only need to count the expected number of times the edge $e_{1,1}$ is traversed. This is $\Pr[e_{1,1}]\pi_1 = P_\Delta^2$

divided by the total weighted time spent on all edges. We get the following new theorem of our new consistency lower bound plotted in figure 2.

Theorem 4.4 (**Nakamoto Consistency**). *Nakamoto's protocol satisfies consistency if there exists $\delta > 0$ such that*

$$\frac{P_\Delta^2}{\sum\limits_{i,j} P_{i,j}\pi_i l_{i,j}} \geq (1+\delta)\beta \tag{1}$$

4.4 Nakamoto Consensus Attack

Pass et al. [17] introduce the delay attack on the consistency of Nakamoto's protocol in which the adversary simply *delays* all honest blocks the maximum amount allowed by the model. Through this delay, the adversary is able to thwart the growth rate of the honest chain, while mining efficiently their own private chain of length at least the size of the honest chain. Figure 7 (1-chain) shows a simple Markov model which captures this attack, where once an honest block is mined, any honest blocks mined in the Δ steps after are wasted work since those honest players don't know about the initial block. Figure 1 'example attack', taken from [17] and recalculated using our Markov model, shows the minimum fraction of mining power the adversary needs for each c in order for the attack to succeed with high probability.

We now present the Markov model of Figure 4 for this attack, and calculate the probability the adversary can generate a private chain of length k, for each μ, Δ and c. The states are as follows:

- S_x: the state where the attack fails
- S_{-1}: state where the honest chain is ahead by one block
- S_0: state with honest and attacker's chains of equal length
- S_i for $i \geq 1$: state where the adversary chain is longer than the honest chain by i blocks

For this analysis we introduce the following variables:

$$\mu' = \mu(1-\rho p)^\Delta \; \rho' = \mu(1-(1-\rho p)^\Delta)$$
$$\rho'' = 1-(1-\rho p)^\Delta + (1-\rho p)^\Delta \rho$$

Let $P_i(k)$ be the probability that starting from state S_i, we visit ρ, ρ', or ρ'' edges $\geq k$ times before hiting state S_x. We calculate:

$$\begin{aligned}
P_i(0) &= 1 \quad i \geq 0 \\
P_{-1}(k) &= \rho'' P_0(k-1) \\
P_0(k) &= \rho P_1(k-1) + \mu P_{-1}(k) \\
P_i(k) &= \rho P_{i+1}(k-1) + \rho' P_i(k-1) + \mu' P_{i-1}(k) \quad i > 0
\end{aligned}$$

Using generating functions, we show how to derive closed form expressions for $P_i(k)$ for fixed i and k. For all $k \geq 0$, define

$$f_k(x) = \sum_{i \geq 0} P_i(k) x^i.$$

We show that $f_k(x)$ satisfies the following equation.

$$\begin{aligned}
f_k(x) \;\; &= \;\; \frac{\rho}{x}(f_{k-1}(x) - f_{k-1}(0)) + \rho'(f_{k-1}(x) - f_{k-1}(0)) \\
&\quad\;\; \mu' x f_k(x) + \mu\rho'' f_{k-1}(0)
\end{aligned} \tag{2}$$

To establish Equation 2, we show that for every $i \geq 0$, the coefficient of x^i in the right-hand side equals $P_i(k)$. For $i = 0$, we observe that the constant term in the right-hand side is the sum of two terms: the constant term in $(\rho/x)f_{k-1}(x)$ and $\mu\rho'' f_{k-1}(0)$. This equals

$$\rho P_1(k-1) + \mu\rho'' P_0(k-1) = P_0(k),$$

as desired. For $i > 0$, the term x^i appears in the right-hand side of Equation 2 in three summands: $(\rho/x)f_{k-1}(x)$, $\rho' f_{k-1}(x)$, and $\mu' x f_k(x)$. Adding these up, we obtain

$$\rho P_{i+1}(k-1) + \rho' P_i(k-1) + \mu' P_{i-1}(k) = P_i(k).$$

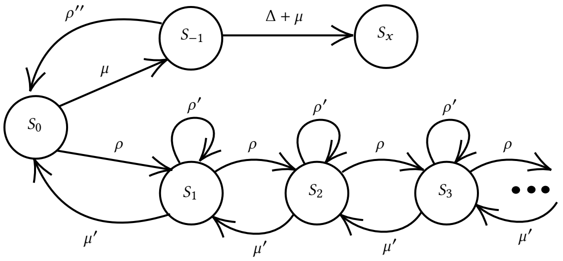

Figure 4: Our Markov chain model which we use to capture the probability that the delay attack lasts for some k blocks.

We now express the generating function $f_k(x)$ of Equation 2 as the following recurrence in k.

$$\begin{aligned}
f_0(x) &= 1 + x + x^2 + \ldots = \frac{1}{1-x} \\
f_k(x) &= \frac{(\rho + \rho' x)(f_{k-1}(x) - f_{k-1}(0)) + \mu\rho'' x f_{k-1}(0)}{x(1-\mu' x)}, k > 0.
\end{aligned}$$

Note that $P_0(k)$ is $f_k(0)$; so by unravelling the above recurrence, we can derive a closed form expression for $P_0(k)$ for any given k.

In Figure 5 we plot $P_0(k)$ for $c = 1, 4$ and 60 for a 49% and 25% adversary. We see that for the Bitcoin hardness parameter ($c = 60$), forks of length 6 (the suggested confirmation time) are possible with roughly 5% probability for an adversary controlling 25% of the mining power and are even more than 1% for length 9. For Ethereum whose c parameter is set to less than 4, waiting 15 confirmations corresponds to roughly 1% probability, which perhaps justifies the aggressive choice of c.

5 CLIQUECHAIN ANALYSIS

Informally, Nakamoto consensus relies on a simple "longest chain" rule to pick between different forks when the network is not in agreement. Sompolinsky and Zohar and later Sompolinsky, Lewenberg and Zohar began to study a more general class of rules for picking between forks that apply to directed acyclic graphs. The first idea in this framework was the GHOST [20] protocol which considered trees of blocks instead of linear chains, they provide an analysis which we extend in the next section. They extended this idea to general DAG protocols where blocks can point to more than one parent block, they call these *inclusive* protocols [13]. Inclusive protocols have a voting mechanism for which transactions to accept, but inherit security from GHOST or any other tree-based selection policy underlying it. In subsequent ongoing work, they consider ideas of how blocks can reference multiple parent blocks and how to reason about linearity of transactions [19, 21].

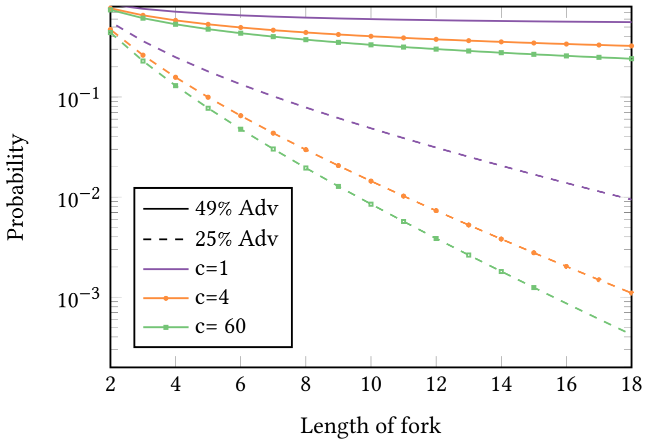

Figure 5: This graph depicts the probability for an execution of Nakamoto to sustain a fork of a particular length. The three regions correspond to this probability at settings of $c = 1, c = 4, c = 60$ where the hardness for the proof of work is set such that a block is expected to be mined in $c\Delta$ attempts. In each case, the top solid line of a shaded region represents the probability for a 49% adversary, whereas the bottom dashed line represents the same for a 25% adversary.

In this section we explore another class of DAG protocols inspired by Chainweb [14]. In Chainweb, the blockchain is a *blockbraid* made of multiple parallel chains in which each block must refer to blocks in specific braids according to a reference base graph. Chainweb security analysis is base-graph dependent and the authors of Chainweb have attempted to analyze the general graph case and provide a '50%'-attack type analysis. We specifically choose a clique as the base graph (resulting in a proposal we call Cliquechain[2]) to facilitate a rigorous analysis. As far as we know, we provide the first consensus lower bound for any variant of a non-trivial DAG-style protocol. Our consensus analysis applies to any number of chains, while in our attack analysis we focus on the 2-chain and 3-chain examples and we see that as we add chains the protocol becomes more resilient to these attacks. Our analysis, however, does not support all the performance claims made in the Chainweb paper as we provide provable consistency for any-chain Cliquechain only up to the same throughput as Nakamoto's protocol.

5.1 The Model

In Cliquechain we have m parallel blockchains and, at any layer l, each block points to a layer $l-1$ block on its chain, as well as a layer $l-1$ block on each chain. Thus in total a block at layer l references m blocks of layer $l-1$, one on each chain.

Blocks in a layer must be *compatible* with one another, meaning they must all point to the same blocks in the previous layer. Figure 6 shows an example of this with 1, 2, and 3-chain versions of Cliquechain. Note that the 1-chain version is simply Nakamoto's protocol. When choosing which block to mine while running an m-chain Cliquechain, a miner runs the following protocol.

(1) Let C be the set of all individual Cliquechains possible from the graph of all blocks mined.
(2) Let L be the longest length of any chain in C, where the length of the chain is the highest level of any block.
(3) Let S be the set of all compatible sets of L level blocks, where a compatible set is a set where all blocks point to all the same blocks in level $L-1$.
(4) Let s be the maximum sized set in S or a random set from one of the maximum sized sets.
(5) If $|s| = m$, level L is complete, choose a random chain to mine a $L+1$ level block pointing to all blocks in s.
(6) Otherwise, mine a block on a chain not in s which is compatible with the blocks in s.

Note that a new layer cannot start being mined until a compatible previous layer has been mined. Thus, all chains grow synchronously. As blocks are mined on a chain in a layer, honest miners move to the remaining chains. If all players act honestly, each layer grows in expected $c\Delta m$ rounds since the probability a block is mined in a given round is $\frac{1}{c\Delta}$.

5.2 Block Expiry

The block expiry argument for Cliquechain works similarly to the block expiry argument for Nakamoto's protocol. The argument is two part: the first is that Cliquechain's honest chain growth is lower bounded by the same bound as Nakamoto's protocol, thus this allows us to use the same lowerbound for block expiry.

Lemma 5.1 (**Cliquechain Chain Growth**). *For any $\delta > 0$, the growth of the main chain of any honest player in an m-chain Cliquechain protocol in T rounds is at least $T(1-\delta)\frac{\mu}{\Delta(c+\mu)}$ blocks over all chains, except with probability that drops exponentially in T.*

Proof. In the worst case, honest players in Cliquechain all work on the same chain at all times, i.e. sequentially. Thus the chain growth is similar to that of a single chain, where all players must learn of a block in the previous chain before moving on to the next chain. Thus Cliquechain's growth is lowerbounded by the same bound as Nakamoto's protocol which is the 1-chain version of Cliquechain. □

For a block to be included in the main Cliquechain at level L, all blocks in layers $> L$ must have a path to this block. Thus if a block is not included in the main Cliquechain at the time it is created, then as time goes on all future blocks, starting on the next layer, point to another block an honest player has of the same chain and level. Thus for Cliquechian we also get the same block expiry lemma as Nakamoto's protocol.

Lemma 5.2 (**Cliquechain Block Expiry**). *There exists a $\delta \in (0,1)$ such that if $\mu \geq \delta\rho$, then every adversarial block expires.*

Proof. Let b be a block mined by the adversary at time r, and let $r+t$ be the first time any honest player hears of b. The adversary is mining efficiently so it's expected number of blocks in T rounds $T\frac{\rho}{c\Delta}$, while as proven above, any honest web has at least $T\frac{\mu}{\Delta(c+\mu)}$ added to it. Starting at the next layer from b, all blocks on b's web

[2] Cliquechain is the block consensus protocol. For reasoning about validity of messages, we refer to Chainweb's SPV protocol for creating inter-chain transactions[14].

must point to it, and all blocks on an honest chain's web with another block in $b's$ place, call it b', must point to b'. So in order for b to not be able to replace b' in the honest chain's web, the adversary must not have mined more blocks, and created a heavier web, than any honest web. By a standard Chernoff-Hoeffding bound, for any $\delta' > 0$, the probability that the adversarial chain grows by at least $(1+\delta')T\frac{\rho}{c\Delta}$ is at most inverse exponential in T. So, from Lemma 4.1, we set the parameters such that

$$\frac{\mu}{(c+\mu)} > \frac{\rho}{c}$$

. □

5.3 Convergence Opportunities

Recall that "convergence opportunities" are events at the end of which all honest players agree on a single chain. A convergence opportunity has 3 parts: Δ rounds where no honest player mines a block, a single honest block mined (termed a 'hit'), then another Δ rounds where no honest player mines a block. After a convergence opportunity in Cliquechain, all honest players agree that the convergence opportunity block is in the longest Cliquechain, and any blocks they now mine on must be compatible with this block.

Lemma 5.3. *At the end of a convergence opportunity (Δ silence + single honest hit +Δ silence), all honest players in Cliquechain start working on blocks compatible with the honest hit block.*

Proof. Let C be the set of all m-Cliquechains from the blocks mined which all honest players see after the first Δ silent rounds, let h be the player who mines the block in step 2, and C' the new set of chains created by the addition of this new block.

1. After the first Δ silence, the longest layer of any chain in C is of length L. Player h chooses the largest compatible subset in the set of all sets of blocks of layer L. If there is not one unique largest set, h chooses one of the largest sets at random.

2. If the set is of size m, player h mines a block on level L on a random chain and points it to all blocks in the set. This new block is now the only $L+1$ block, and after the second Δ silence, all honest players hear about this block and start working on blocks compatible with this block.

3. Otherwise, h chooses a chain not in the set and mines a block which is compatible with the set. This new set is now the unique largest set and after Δ silence all honest players hear of this block and mine blocks compatible with this set. □

Note that after a convergence opportunity, honest miners now all agree on level $L-1$, and which chains still need to be mined blocks compatible with level $L-1$. A convergence opportunity however does not differentiate between the case where there are two layer L blocks on the same chain which point to all the same blocks. Since these two blocks agree on the previous layer, we still say we've converged on the previous layer. From the above we get that any convergence opportunity in Nakamoto is also a convergence opportunity in Cliquechain. Thus we can use the convergence opportunity count we derived in §4.3. We have already shown that block expiry in Cliquechain is satisfied under the same conditions as Nakamoto. Thus, we can extend the consistency theorem of Nakamoto's protocol to Cliquechain.

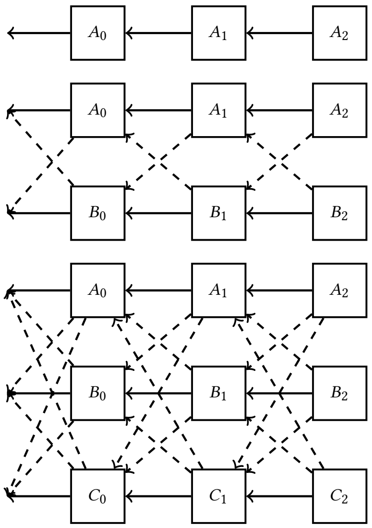

Figure 6: 1, 2, and 3-chain Cliquechain examples where the solid line represents a block pointing to a parent block in its chain, and a dotted line represents a block cross-referencing another chain. Notice 1-chain is simply Nakamoto's protocol. In an m-chain Cliquechain protocol each block at layer ℓ points directly their parent (the $\ell - 1$ block of their chain) plus references to a block in each $m-1$ of the other chains.

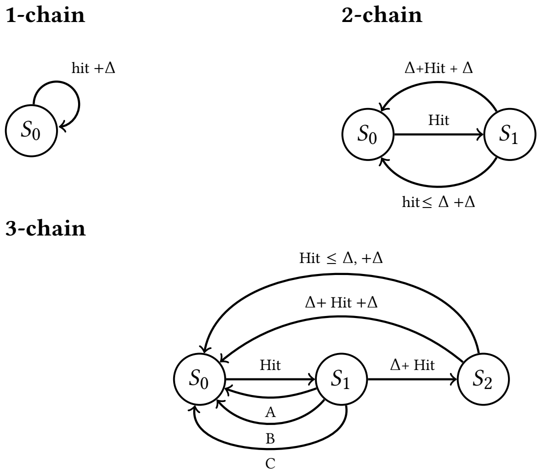

Figure 7: Markov chain models capturing the delay attack on 1, 2 and 3-chain Cliquechain protocols.

Corollary 5.4. *Cliquechain satisfies consistency under the same conditions as stated in Theorem 4.4.*

5.4 Cliquechain Consensus Attacks

We evaluate how Cliquechain preforms under a version of the delay attack of [17]. This attack works on Cliquechain similarly to how it works on Nakamoto's protocol. The goal of the adversary is to delay

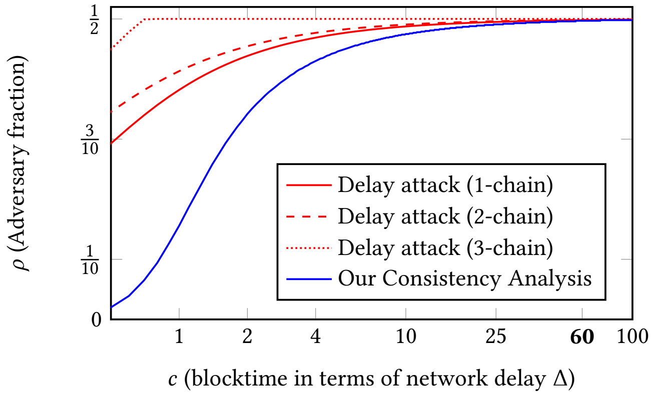

Figure 8: The minimum percentage of computing power an adversary must hold in order to break consistency for $n = 10^5$, $\Delta = 10^{13}$, $p = \frac{1}{c\Delta}$. We compare the delay attacks for Cliquechain's 1-chain, 2-chain and 3-chain models

all honest messages the maximum amount Δ. The adversary's strategy is to maximize wasted honest work by having honest miners work on blocks they don't know have already been mined, therefore delaying the growth of the honest chain(s) while the adversary mines efficiently on their own secret chain(s). With Cliquechain, this attack is thwarted by the fact that the honest players split their mining power among all chains, so if a block is mined and delayed on one chain, the honest miners on the remaining chains that don't yet have a block on that level, are not wasting work during that Δ delay.

To evaluate these attacks we construct Markov models which represent all possible scenarios of how honest blocks are mined in a layer of Cliquechain. Crucial to this analysis is the fact that in Cliquechain no blocks in a new layer can be mined until the previous full layer is mined. Thus all variations of how a layer is mined can restart once the full layer has been mined. Figure 7 shows our Markov models for the 1,2 and 3-chain Cliquechain protocols, with 1-chain being just the Nakamoto delay attack.

For all models, state S_0 represents the state where miners are mining a fresh new layer, and S_i is the state where i chains have a block at that layer. We say an attack succeeds if the expected time for the honest players to mine a block in this model is more than the expected time for an adversary to mine a block efficiently. The expected time for an honest player to mine a block is the expected time to leave state S_0 and get back to state S_0 divided by the number of chains (i.e. how many blocks were added to the honest full braid).

The 1-chain analysis is just the Nakamoto analysis, we analyze the 2-chain and 3-chain attack below:

Theorem 5.5. *For any $\delta > 0$, the delay attack on the 2-chain Cliquechain protocol succeeds when*

$$\frac{(1+\delta)}{2}(l_{01} + \Pr[e_{10A}]l_{10A} + \Pr[e_{10A}]l_{10A}) < \frac{1}{(1-\mu)p}$$

except with exponentially small probability in the length of the attack.

Proof.

$$\begin{aligned} \Pr[e_{01}] &= 1 & l_{01} &= \frac{1}{\mu p} \\ \Pr[e_{10A}] &= (1-\mu p)^{\Delta} & l_{10A} &= \Delta + \frac{1}{\mu p} + \Delta \\ \Pr[e_{10B}] &= 1 - \Pr[e_{10A}] & l_{10B} &= [\sum_{i=1}^{\Delta} \frac{i(1-\mu p)^{\Delta}\mu p}{\Pr[e_{10B}]}] + \Delta \end{aligned}$$

Recall that an attack succeeds if the time for the honest players to grow the chain in this model is more than the time taken for an adversary to mine a block. Comparing the expectations of the random variables representing these two measures, we obtain the following condition for the success of the attack.

$$\frac{1}{2}(l_{01} + \Pr[e_{10A}]l_{10A} + \Pr[e_{10A}]l_{10A}) < \frac{1}{(1-\mu)p}$$

As we did for Nakamoto's analysis, we establish strong concentration bounds (within $(1 \pm \delta)$ factors for any $\delta > 0$) for both measures using Theorem 3.1 in conjunction with a larger expanded Markov chain equivalent to the 2-chain of Figure 7, and a Chernoff-Hoeffding bound, respectively. This yields the desired condition of the theorem. □

Theorem 5.6. *For any $\delta > 0$, the delay attack on the 3-chain Cliquechain protocol succeeds when*

$$\frac{1+\delta}{3}(l_{01} + T_1) < \frac{1}{(1-\mu)p}$$

except with exponentially small probability in the length of the attack.

Proof.

$$\begin{aligned} \Pr[e_{01}] &= 1 & l_{01} &= \frac{1}{\mu p} \\ \Pr[e_{10A}] &= \Pr[\text{two honest hits in} \leq \Delta \text{ time steps}] \\ P_{Ai} &= \text{Probability the second hit happens at time } i \\ &= \sum_{j=1}^{i-1}(1 - \frac{2}{3}\mu p)^{j-1}\frac{2}{3}\mu p(1 - \frac{1}{3}\mu p)^{i-j-1}\frac{1}{3}\mu p \\ \Pr[e_{10A}] &= \sum_{i=2}^{\Delta} P_{Ai} & l_{10A} &= \sum_{i=1}^{\Delta} i\frac{P_{Ai}}{\Pr[e_{10A}]} \\ \Pr[e_{10B}] &= \Pr[\text{one honest hit in} \leq \Delta_1 \text{ time,} \\ &\qquad \text{a second honest hit after } \Delta_1 \text{ before } \Delta_2\ ] \\ P_{Bj} &= \text{Probability first hit happens at time } j \end{aligned}$$

$$= \textstyle\sum_{i=1}^{j}(1 - \frac{2}{3}\mu p)^{j-1}\frac{2}{3}\mu p(1 - \frac{1}{3}\mu p)^{\Delta-j}(1 - \frac{1}{2}\mu p)^{i-1}\frac{1}{2}\mu p$$

$$P_{Bi} = \text{Probability second hit happens at time } i$$

$= \sum_{j=1}^{\Delta}(1 - \frac{2}{3}\mu p)^{j-1}\frac{2}{3}\mu p(1 - \frac{1}{3}\mu p)^{\Delta-j}(1 - \frac{1}{2}\mu p)^{i-1}\frac{1}{2}\mu p$

$$\Pr[e_{10B}] = \sum_{j=1}^{\Delta} P_{Bj} \qquad l_{10B} = \sum_{i=1}^{\Delta} i\frac{P_{Bi}}{\Pr[e_{10B}]}$$

$$\begin{aligned}\Pr[e_{10C}] &= \Pr[\text{one honest hit in} \leq \Delta_1 \text{ time,}\\ &\qquad \text{a second honest hit after } \Delta_2\ ]\\ P_{Cj} &= \text{Probability first hit happens at time } j\\ &= (1 - \frac{2}{3}\mu p)^{j-1}\frac{2}{3}\mu p\\ \Pr[e_{10C}] &= \sum_{j=1}^{\Delta} P_{Cj}(1 - \frac{1}{3}\mu p)^{\Delta-j}(1 - \frac{1}{2}\mu p)^{j}\\ l_{10C} &= [\sum_{j=1}^{\Delta} j\frac{P_{Cj}}{\Pr[e_{10C}]}] + \Delta + \frac{1}{\mu p} + \Delta\end{aligned}$$

$$\Pr[e_{12}] = (1 - \frac{2}{3}\mu p)^{\Delta} \qquad l_{12} = \Delta + \frac{1}{\mu p}$$

$$\begin{aligned}P_i &= \text{Probability a hit happens at time } i\\ &= (1 - \frac{1}{2}\mu p)^{i-1}\frac{1}{2}\mu p\end{aligned}$$

$$\Pr[e_{20A}] = \sum_{i=1}^{\Delta} P_i \qquad l_{21A} = [\sum_{i=1}^{\Delta} i\frac{P_i}{\Pr[e_{20A}]}] + \Delta$$

$$\Pr[e_{20B}] = (1 - \frac{1}{2}\mu p)^{\Delta} \qquad l_{20B} = \Delta + \frac{1}{\mu p} + \Delta$$

Let T_i be the expected time to get from state S_i to state S_0, we have:

$$\begin{aligned}T_2 &= \Pr[e_{20A}]l_{20A} + \Pr[e_{20B}]l_{20B}\\ T_1 &= \Pr[e_{10A}]l_{10A} + \Pr[e_{10B}]l_{10B} + \Pr[e_{10C}]l_{10C}\\ &\quad + \Pr[e_{12}]l_{12}T_2\end{aligned}$$

An attack succeeds if the time for the honest players to grow the chain in this model is more than the time taken for an adversary to mine a block. Comparing the expectations of the random variables representing these two measures, we obtain the following condition for the success of the attack.

$$\frac{1}{3}(l_{01} + T_1) < \frac{1}{(1-\mu)p}$$

As we stated for the 2-chain analysis, we establish strong concentration bounds (within $(1 \pm \delta)$ factors for any $\delta > 0$) for both measures using Theorem 3.1 in conjunction with a larger expanded Markov chain equivalent to the 3-chain of Figure 7, and a Chernoff-Hoeffding bound, respectively. This yields the desired condition of the theorem. □

Figure 8 shows the minimum adversarial percentage needed for the attacks to succeed for each value of c (where the probability any block is mined in a round is $\frac{1}{c\Delta}$). We compare this with the lower bound for any m-chain Cliquechain protocol, which is the same for Nakamoto's protocol. We can see that as the number of chains goes up in the Cliquechain protocol, so does the resilience of the protocol to the delay attack. At 3-chain, the protocol is essentially resilient to the attack except for very small c. Note however that the consistency lower bound remains the same, so there may exist another attack to which these protocols are susceptible.

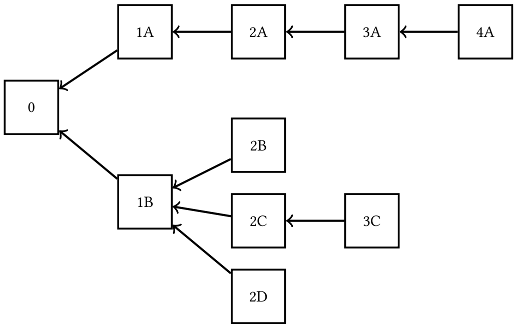

Figure 9: An example of a block tree where a miner following Bitcoin's longest chain rule would mine on 4A, but a miner following the GHOST rule would mine on 3C.

6 GHOST ANALYSIS

In this section we extend our method to analyze the GHOST protocol by Sompolinsky et al. [20]. Section 6.1 provides a summary of GHOST. We extend our analysis of Nakamoto to GHOST and introduce a new consensus attack on GHOST and a Markov model which captures the attack. We note that in their analysis of GHOST, Sompolinsky et al. define a *fork collapse* similar to Pass et al.'s *convergence opportunity* which we use in this paper. We note however that a crucial point of the analysis we do in this paper is that, under *any* adversarial strategy, blocks *expire*, meaning any block has a limited time interval in which it can effect the mainchain. This is not the same as the proof provided in [20] which only accounts for a 50% attack, and not other adversaries.

In §6.4 we show this with an attack of GHOST which utilizes the concept of the adversary saving blocks they have mined as *bank* to be utilized as needed in the attack. In the following section we introduce the notion of a 'subtree expiry' to replace the requirement of 'block expiry'.

6.1 Review of GHOST protocol

The main claim of the GHOST protocol is to be able to handle higher transaction rates through higher block creation rates and/or larger block sizes which increase the network delay (i.e. time it takes for blocks to propagate through the network). The protocol works by miners keeping track of a *tree* of blocks instead of a *chain* and choosing to mine on the block tree which is *heaviest*, rather than the chain which is *longest.* A block's weight is calculated by summing the number of blocks in it's subtree (i.e. the number of blocks who directly point to it or who point to a chain which eventually points to it). Thus a miner starts at the root block and successively picks the heaviest subtree until it arrives at a childless block to build on. Figure 9 illustrates this where a miner following the GHOST rule would mine on block 3C, while a miner following Nakamoto's longest chain protocol would mine on 4A. The idea behind this new rule is that even if two honest nodes mine competing blocks which point to the same parent block, both blocks still increase the weight of the parent block and therefore the probability at least the parent block will be on the mainchain.

6.2 Subtree Expiring

For our analysis of GHOST we extend the idea of block expiry to what we call *subtree expiry.* In short, if an adversary wants a path to beat the current heaviest announced path of any honest player, both paths share a last common block where their subtrees diverged. We argue that in order for the adversary to make an honest player choose the other subtree in the future that is not their current heaviest announced subtree, blocks in that subtree must be announced.

In GHOST each block not only has a length in a chain path which corresponds to it's depth in the tree, but it has a weight equal to the sum of all blocks in the subtrees pointing to it. We reason that for the GHOST protocol, all blocks on any honest player's path have a weight increase of at least that of the Nakamoto chain growth. Below we state the subtree growth and expiry lemmas for GHOST.

Lemma 6.1. *For any $\delta > 0$, and for any honest player's chain at time r, there is some block b in the chain at some length l with weight w, where at time $r + T$ for some T, the block the player now has at length l and all blocks it points to have an expected weight increase $\geq T(1-\delta)\frac{\mu}{\Delta(c+\mu)}$.*

Proof. In this proof we use the same reasoning as the Nakamoto growth lemma of this paper [lemma 4.1]. Consider the path P any honest player takes in the tree to find the heaviest path. Whenever a new honest block b' is announced, either this block is now part of P and all blocks in P have a weight of at least 1 added to it. Or the path to b' diverges from P at some block b, where P is in some subtree pointing to b and b' is in another. The subtree of P must have at least the weight of the subtree of b if the honest player did not add b to it's path. Thus for each Δ period surrounding an honest hit, the weight of some subtree in all honest paths increases by at least 1, and thus all blocks the subtree points to also increase in weight by at least 1. We count this using the same Markov model of lemma 4.1. The expected number of rounds needed for a weight increase of one is at most $\frac{c\Delta}{\mu} + \Delta$; using standard Chernoff-Hoeffding bounds, the number of rounds for an increase of g is at most $(1+\delta)(\frac{c\Delta}{\mu} + \Delta)g$ with probability $1 - e^{-\Omega(g)}$. That is, in T rounds, GHOST achieves a subtree weight increase of at least $(1-\delta)T\frac{\mu}{\Delta(c+\mu)}$ with probability $1 - e^{\Omega(T)}$. We get that for any honest player's chain at time r, there is some block at time $\geq r + T$ whose weight (and therefore the weight of all blocks it points to) increased in time T by $\geq (1-\delta)T\frac{\mu}{\Delta(c+\mu)}$ blocks. □

We now use the subtree growth to prove that if the adversary withholds blocks in a subtree for *too long*, then that subtree will not become part of any honest path in the future except with negligible probability.

Lemma 6.2. *Let C be some subtree where the adversary is mining which no honest player is mining on, but which some honest player is mining on another subtree which points to C's parent. Let r be the point when only the adversary is mining on C and $r + t$ be when the first honest player hears of any block in C after r. There exists a negligible function $\epsilon(.)$ and some $\delta \in (0,1)$ s.t. $\mu \geq \delta\rho$ and the probability C becomes part of any honest path is $\leq \epsilon(t)$.*

Proof. Consider the block b which C points to. At time r, there is some other subtree which points to b which an honest player is mining on. If at some time $T \geq r+t$, some honest player was mining on this other subtree before hearing of a block in C, then from the previous section we know that this subtree grew in expectation by $\geq T\frac{\mu}{\Delta(c+\mu)}$ blocks. Thus if the following inequality holds, then except with negligible probability, the adversary was not able to mine enough blocks in C to make C the heavier choice from the honest subtree:

$$\frac{\mu}{(c+\mu)} > \frac{\rho}{c}$$

If at time $r + t$ no honest player was mining on any subtree that b points to, then that means there was some time after r which the path pointing to C diverged from all honest paths. We consider the latest such point in the path, i.e. the last block the adversary's path has in common with any honest path and the point where this divergence occurred. For any honest player to now be mining on another subtree pointing to this block, this subtree must satisfy the previous lemma's subtree growth since time r. Thus if this subtree's growth is more than the adversary's, then at time T, the probability that the adversary mined a heavier subtree is $\leq \epsilon(t)$. This holds for the following inequality:

$$\frac{\mu}{(c+\mu)} > \frac{\rho}{c}$$

□

Thus if an adversary keeps a subtree silent, i.e. any block in that subtree silent, then no honest players will contribute to it, and if the adversary mines less blocks then any honest subtree growth, then the adversary's subtree will not be the heaviest choice on any honest player's path in the future.

6.3 Convergence Opportunity

Recall from the previous section that the analytical analysis of Nakamoto's protocol done by Pass et al. for consensus relies on the idea of "convergence opportunities" which are events at the end of which all honest players agree on a single chain. The convergence opportunities are made up of 3 steps where we consider only what happens with the honest players and in order for an adversary to be able to break consensus, they must at the very least be able to break all convergence opportunities.

With GHOST, we can't use the same "longest path" or "heaviest block" argument, so we instead use a "heaviest path" argument.

Lemma 6.3. *At the end of a convergence opportunity (Δ silence + single honest hit +Δ silence), all honest players in GHOST will follow the same "heaviest path" down the block tree.*

Proof. Let T be the tree all honest players see after the first Δ silent rounds, let h be the player who mines the block in step 2, and T' the new tree created by the addition of this new block. Now consider the path h took when deciding which block to mine on. Starting at the genesis block, at each block h chooses the next heaviest subtree in T and two things can happen to this same choice in T' :

1. There is a single heaviest sub-tree, thus, in T' this subtree will have an additional block, while the other subtree won't change, and the heaviest subtree will continue to be the heaviest.

2. There is more than one subtree with the max weight, so h chooses an arbitrary one to go down. Thus in T', the subtree h chooses will have one more block while all other subtrees won't change, and therefore in T' this subtree will be the single heaviest subtree. □

Again, for the adversary to be able to break consensus in GHOST, they must at the very least be able to break all convergence opportunities. Since the convergence opportunities in GHOST are the same as that of Nakamoto's protocol, we can use our Markov model from section 4.3 to also count the number of convergence opportunities an adversary would need to match in order to break consensus in GHOST. In the previous section we also proved that subtree expiry in GHOST is also satisfied under the same bounds as Nakamoto's protocol. Thus, we can extend the consistency theorem of Nakamoto's protocol to GHOST.

Corollary 6.4. *GHOST satisfies consistency under the same bounds of theorem 4.4.*

6.4 GHOST Consensus Attacks

The original GHOST paper analyzes GHOST under the 50% attack introduced by Nakamoto [16] where the adversary silently mines their own chain in an attempt to overtake the mainchain (i.e. the honest chain). In §4.4 we saw that with Nakamoto's protocol, as block size increases (i.e. Δ increases) or block time decreases (i.e. c decreases), more honest forks take place meaning honest nodes divide their computing power among more blocks while the adversary continues to mine efficiently and can therefore overtake the honest mainchain with less than 50% of the computational power of the network. In contrast, with this attack, GHOST remains resilient for any value of Δ or c since all honest nodes contribute to the overall weight of the honest subtree. We now present a new attack on GHOST, the balance attack, in which the adversary leverages honest computing power to maintain two subtrees of equal weight.

The point of our attack is for the adversary to maintain a fork in the block tree persisting for as long as they can, thereby delaying consensus and the time to confirm that a transaction has made it onto the main chain. The adversary does this by splitting the honest computing power among the two subtrees and mining on both subtrees and using their blocks to balance the two subtrees whenever they become uneven. The adversary's strategy begins once a fork takes place (i.e. two blocks are mined within Δ of each other). We define the attack in Algorithm 1.

We model this attack with the Markov chain in figure 10. In the model, we have 3 layers of states: the *Parity, delta left*, and *delta right* layers. Each parity state $P(l, r)$ represent the state where both subtrees are of equal length and the adversary's banks have amounts l and r. Each delta left state $\Delta^{left}(l, r)$ represents the state where the left subtree is up by one honest block and there is a delta race until the adversary has to reveal the honest block to all players. In this state, either an honest node on the right subtree wins a block in the delta race, the adversary uses a right bank to rebalance the subtree, or an honest player on the left wins again and the adversary uses bank to pay off the last left honest block and begins another delta race. The delta right states, $\Delta^{right}(l, r)$, are equivalent to the delta left states, but with the right subtree dominating. Figure 10 represents a sample of states reachable from $P(l, r)$.[3]

Algorithm 1 GHOST_attack(n, ρ, c, Δ)

```
Once a fork takes place (i.e. two blocks (adversarial or honest)
are created within $\Delta$ rounds) The adversary sends one block to
half the honest nodes, and the other block to the other half.
while the fork persists do
    Adversary mines on the subtree with least bank
    if honest node mines a block then
        $\Delta$ rounds count down begins
        if honest node mines on the opposite subtree then
            fork is rebalanced
        if no honest miner mines in $\Delta$ rounds then
            if adversary had bank to use then
                adversary uses bank to rebalance the fork
            else adversary loses
        if the side that is winning mines again then
            use bank to balance the previous win
            $\Delta$ counter restarts
    if there is no $\Delta$ counter then
        adversary mines on side with least bank
```

In this model, let a be the adversarial mining probability, where a_l, a_r denotes which subtree the adversary was mining on, and $a_{l\Delta}, a_{r\Delta}$ are the probabilities the adversary mines within Δ steps. Let h_l and h_r be the honest mining probabilities on either subtree, and $h_{l\Delta}, h_{r\Delta}$ the probabilities an honest player mines a block within Δ rounds on a given subtree. Lastly, let Δ_{free} be the probability that Δ rounds pass without anyone (adversary or honest player) mining a block. Every transition crossing a "Δ" edge (i.e $h_{l\Delta}, h_{r\Delta}, a_{l\Delta}, a_{r\Delta}$ and Δ_{free}) causes an increase of the fork length by one block. Thus to determine the probability of reaching a fork of a specific length k, we calculate the probability we cross k edges before reaching a state where the attack fails. We say the attack fails when the adversary needs to balance one side of the fork but does not have any stored bank on that side, i.e. a state where $l = -1$ or $r = -1$. We now calculate the probability that the attack lasts for k blocks. We introduce the following variables:

$$h_r = h_l = 0.5\mu \qquad a_r = a_l = \rho \qquad \Delta_{\text{free}} = (1-p)^{\Delta}$$

$$(h_{r\Delta} + a_{r\Delta}) = (h_{l\Delta} + a_{l\Delta}) = (1-p)^{\Delta}(0.5\mu + \rho)$$

Let $S_{l,r}(k)$ be the probability that starting from state $S_{l,r}$, we visit a Δ edge $\geq k$ times before the attack fails, i.e. before we visit a state where l or r equals -1. We get the following probabilities:

$$\begin{array}{rccc}
P_{l,r}(k) = 1 & \text{for} & l \geq k, & r \geq k \\
\Delta^{right}_{l,r}(k) = 1 & \text{for} & l \geq k, & r - 1 \geq k \\
\Delta^{left}_{l,r}(k) = 1 & \text{for} & l - 1 \geq k, & r \geq k \\
S_{l,r} = 0 & \text{for} & l = -1 & \text{or} \quad r = -1
\end{array}$$

[3] Variations on the attack can handle *uncle* limits, where we limit the bank the adversary can use to be within u blocks of the current block.

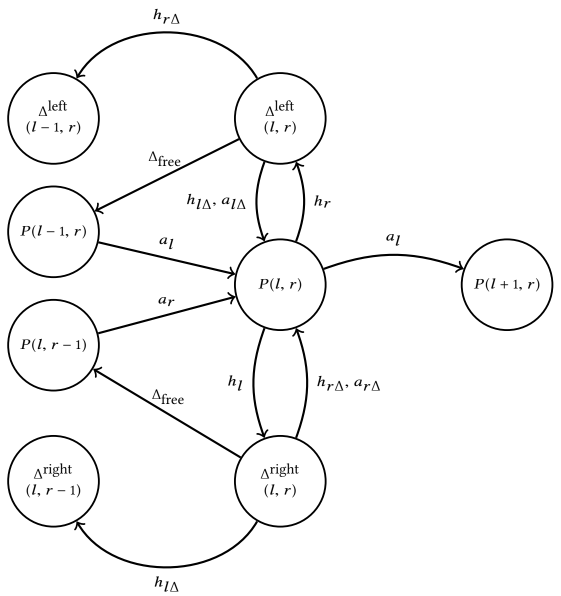
$h_{r\Delta}$
Δ^{left} $(l-1, r)$
Δ^{left} (l, r)
Δ_{free}
$h_{l\Delta}, a_{l\Delta}$
h_r
$P(l-1, r)$
a_l
$P(l, r)$
a_l
$P(l+1, r)$
a_r
$P(l, r-1)$
Δ_{free}
h_l
$h_{r\Delta}, a_{r\Delta}$
Δ^{right} $(l, r-1)$
Δ^{right} (l, r)
$h_{l\Delta}$

Figure 10: GHOST attack markov chain snapshot.

$$\Delta^{right}_{l,r}(k) = (h_{r\Delta} + a_{r\Delta})P_{l,r}(k-1) + h_{l\Delta}\Delta^{right}_{l,r-1}(k-1) + \Delta_{\text{free}}P_{l,r-1}(k-1)$$

$$\Delta^{left}_{l,r}(k) = (h_{l\Delta} + a_{l\Delta})P_{l,r}(k-1) + h_{r\Delta}\Delta^{right}_{l-1,r}(k-1) + \Delta_{\text{free}}P_{l-1,r}(k-1)$$

$$P_{l,r}(k) = h_r\Delta^{right}_{l,r}(k) + h_l\Delta^{left}_{l,r}(k) + a_lP_{l+1,r}(k) \quad r \geq l$$

$$P_{l,r}(k) = h_r\Delta^{right}_{l,r}(k) + h_l\Delta^{left}_{l,r}(k) + a_rP_{l,r+1}(k) \quad r < l$$

We are interested in $P_{0,0}(k)$, the probability of a fork of at least k. We can then use this to calculate for a certain protocol parameter, how many blocks should you wait for confirmation of a transaction, for a given confidence. In Figure 11 we plot $P_{0,0}(k)$ for $c = 1, 4, 60$ with $\rho = .49, .25$. When compared with the plots for Nakamoto's protocol, GHOST is more resilient to this attack than Nakamoto's protocol is to the delay attack. However, for low c, forks can last for more than 10 blocks with non-negligible probabilities.

7 LIMITATIONS AND FUTURE WORK

We make conservative choices in our analysis. For example, for GHOST we define the minimum subtree growth as the same as the minimum chain growth of Nakamoto's protocol. This ignores the case when honest players all work on the same subtree and thus all honest blocks contribute to the growth of the weight or when multiple blocks are mined in Δ rounds which contribute to a single subtree. These cases suggest that a tighter bound of GHOST's subtree weight growth is possible. Similarly in Cliquechain's growth, we assumed honest players were mining sequentially and not taking advantage of the parallel work possible with the protocol. The delay attacks for Cliquechain give a better lower bound for the growth of any honest chain; we leave the generalization of our Markov model to any n-chain Cliquechain as future work.

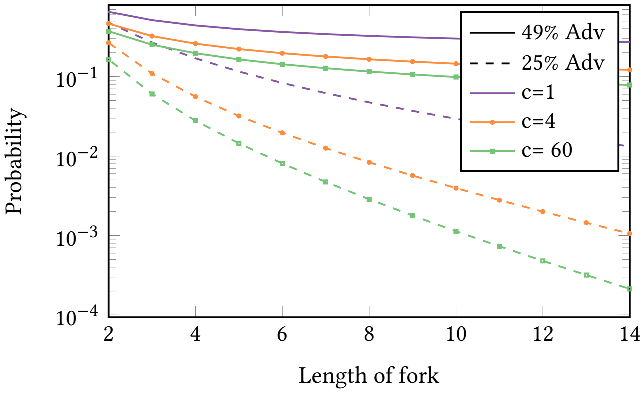

Figure 11: This graph depicts the probability for an execution of GHOST to sustain a fork of a particular length for various values of c and ρ. The three regions correspond to this probability at settings of $c = 1, c = 4, c = 60$ where the hardness for the proof of work is set such that a block is expected to be mined in $c\Delta$ attempts. In each case, the top solid line of a shaded region represents the probability for a 49% adversary, whereas the bottom dashed line represents the same for a 25% adversary.

It has been our experience that the simplest or most intuitive model for an attack which we consider may not be solvable for all analyses we are interested in. In Nakamoto's protocol, the simple one state model of the attack yields an asymptotic upper bound for the effectiveness of the attack, but we need a more complex Markov model in order to analyze the attack over short time periods. With the Cliquechain attacks, the Markov models become more complex as chains are added. We used our Markov models to provide an asymptotic upper bound for the attack, but it is not yet clear what model we need to perform a short-term analysis of the attack like we do for Nakamoto's protocol. With GHOST we see the opposite. We have derived a recurrence relation for short-term fork length distributions under the balance attack, but obtaining asymptotic bounds under the attack is still open. We are pursuing ways to unify our techniques and make our Markov-based method for analyzing blockchain consistency more comprehensive.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. This research was supported in part by NSF grants CCF-1422715, CCF-1535929, TWC-1646671, TWC-1664445, CNS-1616234 and CNS-1409191.

REFERENCES

[1] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2016. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. *arXiv preprint arXiv:1605.07524* (2016).
[2] Jing Chen and Silvio Micali. 2016. Algorand. https://arxiv.org/abs/1607.01341.
[3] H. Chernoff. 1952. A Measure of the Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Mathematical Statistics* 23 (1952), 493–509.
[4] Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. 2012. Chernoff-Hoeffding Bounds for Markov Chains: Generalized and Simplified. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*. 124–135.

[5] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 1–10.

[6] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*. Springer, 436–454.

[7] Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications.. In *EUROCRYPT (2)*. 281–310.

[8] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2017. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*. 291–323.

[9] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *SOSP'17*.

[10] W. Hoeffding. 1956. On the Distribution of the Number of Successes in Independent Trials. *Annals of Mathematical Statistics* 27 (1956), 713–721.

[11] Aggelos Kiayias and Giorgos Panagiotakos. 2015. Speed-Security Tradeoffs in Blockchain Protocols. *IACR Cryptology ePrint Archive* 2015 (2015), 1019.

[12] Aggelos Kiayias and Giorgos Panagiotakos. 2016. On Trees, Chains and Fast Transactions in the Blockchain. *IACR Cryptology ePrint Archive* 2016 (2016), 545.

[13] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. 2015. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*. Springer, 528–547.

[14] Quaintance Martino and Popejoy. 2018. Chainweb: A Proof-of-Work Parallel-Chain Architecture for Massive Throughput.

[15] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA.

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

[17] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 643–673.

[18] Rafael Pass and Elaine Shi. 2017. The Sleepy Model of Consensus. In *ASIACRYPT'2017*.

[19] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. 2016. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *IACR Cryptology ePrint Archive* 2016 (2016), 1159.

[20] Yonatan Sompolinsky and Aviv Zohar. 2015. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 507–527.

[21] Yonatan Sompolinsky and Aviv Zohar. 2018. PHANTOM: A Scalable BlockDAG Protocol. Cryptology ePrint Archive, Report 2018/104. https://eprint.iacr.org/2018/104.