Securely Improving Stability and Performance of PoW Blockchains using Anchors

Ovia Seshadri*, Vinay J Ribeiro† and Shadab Zafar*
*Department of Computer Science and Engineering, Indian Institute of Technology Delhi
†Department of Computer Science and Engineering, Indian Institute of Technology Bombay
{ovia.seshadri@cse, shadab.zafar@alumni}.iitd.ac.in, vinayr@iitb.ac.in

2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 978-1-6654-2104-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/COMSNETS53615.2022.9668572

***Abstract*—Proof-of-work (PoW) consensus generates blocks at random time instants, and consequently, adds weight to the blockchain at these random instants. This unsteady increase in chain weight over time along with the large network delay of blocks is the root cause of many security and performance problems such as high fork resolution times, vulnerability to double-spend and selfish mining attacks, and also a high block confirmation time.**

In this paper, we propose a novel signaling scheme of PoW called *Anchors* that can be implemented on any PoW blockchain system to improve their stability, reduce fork resolution times, prevent forks, strengthen the system against double spend attacks and reduce block confirmation times by half. In a partially synchronous model with a general adversary, we prove that a system with anchors has faster, more stable chain growth as compared to PoW systems without anchors and show that consistency security bounds established by prior work are preserved as in the underlying blockchain. We incorporate anchors in a reference Bitcoin implementation and perform experiments to support our claims in a 210 node test-bed with real world latencies. Our results show that anchors propagate at least 6 times faster than blocks, help resolve forks much faster than Bitcoin (without anchors) does and are able to even prevent forks from occurring.

I. Introduction

A blockchain is a decentralized, distributed ledger containing data called transactions inside blocks. Blockchains are maintained by a P2P network of *miners* who periodically add blocks to the end of the blockchain using a consensus protocol. In this paper, we focus our attention on proof-of-work (PoW), permissionless blockchains like [1]–[9]. Each block effectively adds PoW "weight" to the chain. In case a miner observes more than one competing blockchain, he extends the chain with the most weight. The scenario when more than one chain has equal weight is known as a fork, and the miner picks the chain he saw first to extend.

Motivation: PoW consensus generates blocks randomly, modeled as a Poisson random process in [1], [10]. Thus weight gets added to the blockchain at random instants. The blocks generated have significant network delay owing to their large size due to which the expected block interval (difficulty of the PoW puzzle) is set high in order to avoid natural forks (forks formed via delays without adversary influence). This unsteady increase in chain weight over time combined with the large delay of blocks is the root cause of many security and performance problems.

Firstly, it leads to high fork resolution times (e.g. ten minutes in expectation in Bitcoin). A fork is resolved when one branch gets heavier than the others typically with the arrival of the next block which is a large time interval. Blocks on the lighter branch(es) are orphaned which reduces mining power utilization. The miners creating orphaned blocks lose their mining and transaction fees.

Secondly, when forks take a long time to get resolved, it opens the blockchain up to a host of attacks such as double-spend and selfish mining attacks [1], [11]–[20]. During a fork, honest mining power is split between the branches of the fork making the blockchain vulnerable where an attacker with less mining power than the honest miners can occasionally generate a longer private chain than the honest miners.

Thirdly, it leads to increased block confirmation times (CT). For example, to prevent natural forks and support its large block sizes, Bitcoin keeps the average time between block generation times large (10 minutes). For 1MB blocks, this translates to less than 10 transactions per second, which is orders of magnitude less than the VISA payment system [21]. To guard against double-spend attacks, users are required to wait till their transactions are at least 6 blocks deep (or 6 confirmations) [1], [10], due to which the CT becomes as large as 1 hour, which is impractical for typical fast-payments [22].

Our Contributions: In this paper, we propose *Anchors* (§II), which are small, frequent and fast structures having PoW weight that can be implemented on any new or existing PoW blockchain system without causing significant CPU, latency or bandwidth overheads to the underlying blockchain. Anchors help its underlying blockchain by (i) Maintaining chain stability by steadily adding PoW weight, aiding faster chain growth that helps mitigate double spend and selfish mining attacks. We derive in § III a rigorous lower bound for chain growth and show that it is higher than growth established in PoW blockchain systems from [23]; (ii) Securely improving the block confirmation time. Using the derived growth rate, we provide strong theoretical guarantees for the confirmation time reduction possible with Anchors (§ IV-A) and prove that consistency is preserved in a system with Anchors (§ IV-B); and (iii) Acting as a signaling mechanism to show mining power division in the network resolving forks faster and even preventing fork occurrences. We present in §V, a proof of concept emulation of anchors in Bitcoin consisting of 210 nodes

with real world latencies to study anchor and block delays. We analyze the impact of Anchors on forks by studying their resolution times and occurrence rate. Anchors significantly improve performance aspects in the form of confirmation time and fork resolution speeds without a security trade-off.

II. Anchors

In this section we define Anchors and discuss their working on a PoW blockchain. At times we relate with Bitcoin, the most familiar PoW blockchain, for ease of understanding, however, the idea can be translated to any PoW system.

Definition II.1. *An **Anchor** is a small bit string consisting of (i) Hash of some block in the main chain called its parent block, and (ii) a solution of a PoW puzzle that is different and easier than that of a block. It optionally contains information such as a coinbase transaction/an address of the entity creating it which can be used later to reward its creator. It contains no transactions other than the coinbase.*

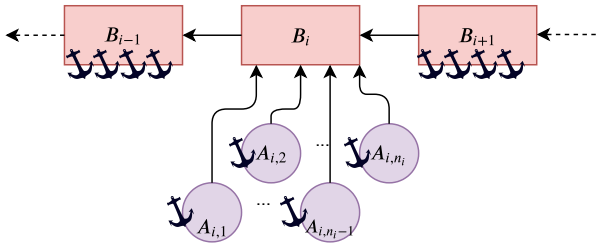

Fig. 1. Sample blockchain with anchors

Structure: An anchor is similar to the block of its system in terms of structure and its mining. Henceforth, we collectively refer to a block or an anchor as an *entity*. In a system like Bitcoin, anchors would be entities consisting of an 80 Bytes header exactly like the block header and an optional coinbase transaction in its body. The fields of relevance in the anchor header are: the hash of the previous block header which is a link to the parent block; the agreed difficulty of the PoW puzzle on which a block should be generated; a nonce field which can be tweaked in order to find the valid PoW solution for a block or an anchor; and a merkle root of transactions held in the body at the time of its generation. Unlike anchors, a block in the system has several other transactions in its body, which makes them much larger in size.

Anchor Mining: Every miner participates in a PoW election to win a chance to extend the blockchain with his block. While competing in this election in a system with anchors, every header is first checked if it forms a valid block, if not it is then checked for a valid anchor. This ensures minimum mining effort for anchors. It is similar to the 2-for-1 PoW technique used in [8], [18], [24], [25]. Valid anchors are attached to the parent block on which the anchor was mined and mining continues on top of the parent block. Hence there may be many anchors mined by many peers on a block. Thus, the blockchain system with anchors consists of a chain of blocks where each block points to the previous block and may have anchors pointing to it as depicted in figure 1. For example in hash-based PoW blockchains like Bitcoin, if $H(header, nonce) \in [0, T)$ it is a valid block and if $H(header, nonce) \in [T, (a+1)T)$ it is a valid anchor. Here H represents a hash function and T is the size of the agreed target solution space for valid blocks (PoW difficulty). Parameter a is an integer fixed such that $a \geq 1$ and it represents the ratio of the frequencies of finding valid anchors to blocks. Note that anchor solutions have a larger, non intersecting target space. Before we discuss anchors further, let us define a chain selection protocol based on chain weight.

Chain Weight: Every honest miner picks the block on the *heaviest chain* known to him in terms of weight to mine on/extend. This rule is called the **Heaviest Chain Rule** or HCR. The weight of a block or anchor is proportional to the difficulty of their generation, meaning anchors weigh lesser than blocks. For simplicity of further discussions in this paper, we will assume each block contributes 1 unit of weight to the chain and every anchor contributes α units where $0 < \alpha \leq 1$. Thus, anchors contribute towards **steady chain weight gain**. The weight contributed by an anchor, α is also defined as the ratio of the target sizes of blocks to anchors i.e. $\alpha = \frac{1}{a}$.

We use the following notation, to formally define chain weight. Consider a miner who has received a blockchain of N blocks. Call the i^{th} block B_i $(i = 1, \ldots, N)$ and the anchors pointing to it $A_{i,j}$ for $j = 1, \ldots, n_i$, where n_i is the total number of anchors pointing to B_i as shown in figure 1. Let $w(.)$ denote the weight function for a block or anchor. In a chain without anchors, $\sum_{j=1}^{n_i} w(A_{i,j}) = 0$. The total weight of the chain can be written as $\sum_{i=1}^{N} \left[w(B_i) + \sum_{j=1}^{n_i} w(A_{i,j}) \right]$. Note that the chain weight is updated by miners *as soon as it receives a valid anchor* and not when an anchor is rewarded in a block. Miners are able to make decisions on the chain to extend in case of forks, based on their local weight.

Anchor Processing: In case a valid anchor is mined, an honest miner broadcasts that header and its coinbase along with the merkle path of the coinbase to its root. The weight of the local chain is updated with the weight of the new anchor. Mining is then continued on the parent block of the new anchor. Since anchors cannot be extended they cause no additional forks in the system. The merkle path that is additionally sent, is needed to prove to a receiver that the coinbase belongs to the merkle tree of transactions corresponding to the merkle root stored in the header [1]. The absence of transaction makes anchor's verification O(1) time.

Upon receiving an anchor, the honest peer confirms it's validity and compares his current weight to the weight of the chain(s) the new anchor belongs to. When a received anchor points to an older block on the chain and since then the chain has been forked, the anchor's weight can affected multiple chains. If the revised weight introduced by the new anchor causes a switch in case of a fork, as required by HCR, the peer shifts to the new chain tip and continues extending that chain. He then forwards the anchor to all neighboring peers.

Delay Model: Anchors being fixed small size structures, have **low broadcast latency**. To see why, consider the following

delay model, consistent with [8], for latency of a message of size M to travel from peer i to peer j. Suppose the bandwidth and speed of light delay between the peers are $C_{i,j}$ and $D_{i,j}$ respectively, then the total time to transfer and process a message M from peer i to peer j is: $D_{i,j} + \frac{M}{C_{i,j}} + \kappa M$. In this model the *first bit* of the message takes $D_{i,j}$ to travel across the network at the speed of light and then it takes a further $\frac{M}{C_{i,j}}$ time for the entire message to reach peer j. After this, peer j validates and verifies it. We assume that the verification time is proportional to the size of the message M with proportionality constant κ. Now the total time to broadcast will be the sum of such delays on the peer-level path from the origin to the last peer that receives the message. Due to their small size, anchor's broadcast latency consists mainly of speed of light delays, whereas for blocks the last two terms can dominate. Therefore, we can assume propagation delay of anchors is less than that of blocks. This is demonstrated in our emulation in section V.

Rewards: We suggest rewarding anchors proportional to their weight to the address provided in the coinbase i.e. *anchor_reward* $= \alpha *$ *block_reward*. Anchors can be rewarded by including them in the body of a block further down the chain. An advantage of rewarding anchors is that it can possibly obviate the need for smaller miners to join mining pools [20], [26]–[28] for small payments. A full discussion on rewards and a game theoretic analysis for a rational miner will be provided in an extended version of this paper.

A. System and Adversary Model

Let there be a PoW blockchain system comprised of n miners, connected in a P2P network similar to Bitcoin. We will assume the network is reliable i.e. honest miners see all entities created by each other. Without loss of generality, we will assume the miners control equal mining power. Let an adversary control at most a fraction q of the total mining power of the network where $q < 0.5$.

The analyses in this paper assumes an execution of the system in continuous time similar to [29]. Because of the network delay, different honest nodes may have different views of the blockchain. Let the maximum bounded network delay of blocks and anchors be Δ_b and Δ_a time units respectively i.e. if an honest node is aware of an entity at time t, it must be known to all other honest nodes by time $t + \Delta_b$ if it is a block and by $t + \Delta_a$ if it is an anchor. From our delay model supported by results in section V, we can assume $\Delta_b > \Delta_a$. We can assume the adversary nodes have zero delay between them and can always act in collusion.

Honest miners will always be working on the heaviest chain known to them. If a node is controlled by the adversary they can be working on any block in the tree of blocks from genesis known to them and choose when to broadcast their entities. We can assume the existence of an impartial oracle that knows which nodes are controlled by the adversary and the generations of all entities as soon as they are created.

Recall that the frequency of arrivals of blocks to anchors in the system is $\alpha = \frac{1}{a}$. Let λ_a be the total mining rate of anchors and λ_b be the total mining rate of blocks where $\lambda_a = \frac{\lambda_b}{\alpha}$. Let the Poisson arrival rates of honest blocks, honest anchors, adversary blocks and adversary anchors be $G = (1-q)\lambda_b$, $\frac{G}{\alpha}$, $q\lambda_b$ and $q\lambda_a$ respectively. It is assumed that, at any time, there can be a maximum of one entity mined. Let parameter p define the hardness for the proof of work and is set such that a block is expected to be mined in $cn\Delta_b$ attempts i.e. $p = \frac{1}{cn\Delta_b}$. Here, c is the expected number of network delays before the next block is mined (block interval in terms of Δ_b). Thus, we can say $\lambda_b = np$.

III. Chain Growth

The following lemma shows that if an honest player has a chain of weight w at time r, then all other honest players will have a chain of weight at least w by time $r + \Delta_b$. The proof is based on the fact that $\Delta_a < \Delta_b$ and all entities known to an honest node must reach all other honest nodes by Δ_b. Let C_i^r denote the heaviest chain at node i at time r and let $|C_i^r|$ denote its weight.

Lemma III.1. *In a blockchain with anchors, if nodes i, j are honest, then $|C_j^{r+\Delta_b}| \geq |C_i^r|$.*

Theorem III.2. *(Chain Growth) For any $0 < \delta < 1$ and any interval $[s, s+t]$ where $t > 2\Delta_b$, the system with anchors achieves an honest chain growth of at least $(1-\delta)\upsilon t$ in weight except with probability $e^{-\Omega(t)}$ which is negligible in t and honest growth rate parameter per time is,*

$$\upsilon = G\left[\frac{1}{G\Delta_b + 1} + \frac{\max\left(0, 1 - \frac{2G\Delta_b}{G\Delta_b+1}\right)\alpha}{G\Delta_b + \alpha}\right]$$

Proof. Consider the arrivals of blocks and anchors in the interval of interest, $[s, s+t]$, from the perspective of any honest node. In this interval we will mark certain blocks and anchors in such a way that the spacing between consecutive marked entities is larger than Δ_b. To ensure that only the blocks that have reached all the other honest nodes during the interval of interest is marked, we remove the first and last Δ_b time in this interval. Therefore, the interval of interest is now $[s + \Delta_b, s + t - \Delta_b]$.

The marking procedure for blocks is as follows. Mark the first honest block, skip Δ_b time and mark the next honest block and so on. Let N_b be the total number of marked blocks in $[s + \Delta_b, s + t - \Delta_b]$. We next mark honest anchors in between every pair of consecutive marked blocks in a similar fashion as we marked blocks in the interval $[s, s+t]$. Say marked block B_i arrived at t_i and the next marked block B_{i+1} arrived at t_{i+1}. To ensure that no previous marked entity is in transmission over the network before marking a new entity, we remove the first and last Δ_b times in the interval $[t_i, t_{i+1}]$. Let interval I_i, where $i = 1 \ldots N_b - 1$, denote the time available to mark anchors between marked blocks B_i and B_{i+1} after removing the first and last Δ_b times. We define I_i as

$$I_i = \begin{cases} [t_i + \Delta_b, t_{i+1} - \Delta_b], & \text{if } t_{i+1} - t_i > 2\Delta_b \\ \phi, & \text{otherwise} \end{cases}$$

Within I_i we mark the first honest anchor, skip ahead by Δ_b, mark the next honest anchor, and so on. Let N_a be the total number of marked anchors in the interval $[s+\Delta_b, s+t-\Delta_b]$. Note that this construction ensures a gap of at least Δ_b between any two consecutive marked entities (say E_{j-1} and E_j) even if E_{j-1} is an anchor. This guarantees that all ancestors of a marked entity are known to all honest miners before marking that entity. Let the marked entities be E_j created by miners M_j at time T_j where $j = 1, ..(N_b + N_a)$. Note that the same miner may generate multiple marked entities, hence we allow $M_j = M_k$ for $j \neq k$. Let $\theta_j \in \{\alpha, 1\}$ denote the weight of the entity E_j. Let the weight of the heaviest chain of miner M_j at time T_j after the addition of E_j be represented as $|C^{T_j}_{M_j}|$.

At $T_{j-1} + \Delta_b$ for any j (we skip Δ_b after every marked entity), miner M_j must have seen E_{j-1} and all its ancestors, before mining E_j. From lemma III.1, even if E_{j-1} is not on M_j's heaviest chain, he must have extended a chain at least as heavy as $|C^{T_{j-1}}_{M_{j-1}}|$ when creating E_j. E_j further added θ_j units to M_j's heaviest chain. Therefore, we can say,

$$|C^{T_j}_{M_j}| \geq \theta_j + |C^{T_{j-1}}_{M_{j-1}}| \tag{1}$$

At $T_j + \Delta_b$ for any j, any honest miner m would have seen E_j and must have chosen to extend the heaviest chain known to him. From lemma III.1 the chain he extends would be at least as heavy as M_j's heaviest chain at T_j i.e $|C^{T_j}_{M_j}| \leq |C^{T_j+\Delta_b}_{m}|$.

This can be written as $\theta_j + |C^{T_{j-1}}_{M_{j-1}}| \leq |C^{T_j+\Delta_b}_{m}|$ from eq 1.

The interval $[s, T_{N_b+N_a} + \Delta_b]$ is completely enclosed in the interval $[s, s+t]$ as we have a given a pad of Δ_b at the end of $[s, s+t]$. Therefore, the weight gained by m in $[s, T_{N_b+N_a} + \Delta_b]$ must be less than or equal to the weight gained by m in $[s, s+t]$. We take the difference in weight gain by miner m in $[s, T_{N_b+N_a} + \Delta_b]$ to find the lower bound weight gained in the interval $[s, s+t]$.

$$\begin{aligned} |C^{s+t}_m| - |C^s_m| &\geq |C^{T_{N_b+N_a}+\Delta_b}_m| - |C^s_m| \\ &\geq \theta_{N_b+N_a} + |C^{T_{N_b+N_a-1}}_{M_{N_b+N_L-1}}| - |C^s_m| \end{aligned}$$

Note that based on the reasoning given for equation 1, $|C^{T_1}_{M_1}| \geq \theta_1 + |C^{s+\Delta_b}_{M_1}|$. From Lemma III.1 we know $|C^{s+\Delta_b}_{M_1}| \geq |C^s_m|$. Recursively simplifying we get, $|C^{s+t}_m| - |C^s_m| \geq \sum_j \theta_j \geq N_b + \alpha N_a$.

By estimating $N_b + \alpha N_a$ we obtain, with high probability in t, a lower bound on the honest chain growth in t.

N_b estimation: The marked blocks form an arrival process with i.i.d. inter-arrival times, each of which is equal to a Geometric random variable with mean $\frac{1}{G} + \Delta_b$. Define $t' = t - 2\Delta_b$ and $\overline{N_b} := \frac{Gt'}{G\Delta_b+1}$. Let e_1 denote the event that $N_b \in \left[(1-\delta_1)\overline{N_b}, (1+\delta_1)\overline{N_b}\right]$ for some $\delta_1 \in (0,1)$. Using propositions $C1 - C3$ from [29] we have $Pr[e_1] \geq 1 - e^{-\Omega_1(t')} = 1 - e^{-\Omega_2(t)}$.

N_a estimation conditioned on e_1: Suppose N_b blocks have been marked. Call the total time remaining for marking anchors t''. Then $t'' \geq t' - 2\Delta_b N_b$, since each marked block essentially reduces the time available for marking anchors by at most $2\Delta_b$. When e_1 occurs, we must have

$$t'' \geq \tau := \max\left(0, t' - (1+\delta_1)(2\Delta_b\overline{N_b})\right) \tag{2}$$

To obtain N_a, we first consider an arrival process in time τ whose inter-arrival times are i.i.d., each equal to a Geometric random variable with mean $\frac{\alpha}{G} + \Delta_b$. Then the number of arrivals of this process is dominated from above by the distribution of N_a since $t'' \geq \tau$. Let e_2 be the event that $N_a \geq \frac{(1-\delta_2)G\tau}{G\Delta_b+\alpha}$ for some $\delta_2 \in (0,1)$. Using propositions $C1 - C3$ from [29] we have $Pr[e_2|e_1] \geq 1 - e^{-\Omega_3(\tau)} = 1 - e^{\Omega_4(t)}$.

Applying Bayes' theorem, we have $Pr[e_1 \cap e_2] = Pr[e_2|e_1]Pr[e_1] \geq \left(1 - e^{-\Omega_4(t)}\right)\left(1 - e^{-\Omega_2(t)}\right) = 1 - e^{-\Omega(t)}$. This implies that w.h.p. in t, the event $e_1 \cap e_2$ occurs, that is, N_a and N_b are simultaneously greater than $\frac{(1-\delta_2)G\tau}{G\Delta_b+\alpha}$ and $\frac{(1-\delta_1)Gt'}{G\Delta_b+1}$ respectively. This gives us the desired growth rate $\upsilon = \frac{Gt'}{G\Delta_b+1} + \frac{G\tau\alpha}{G\Delta_b+\alpha}$. □

Note 1: Pass et. al. [23] identified $\upsilon_{\text{pow}} = \frac{G}{G\Delta_b+1}$ for PoW systems. We can notice $\upsilon_{\text{pow}} \leq \upsilon$ as the first term in υ is υ_{pow}.

Note 2: We can assume the expected adversary growth in unit time is $\beta = 2q\lambda_b$ or $\beta = 2\beta_{\text{pow}}$.

IV. Confirmation Time (CT) and Consistency

Confirmation Time (CT) is the time a seller must wait to believe, with enough confidence, that an adversary (holding a q fraction of power) cannot orphan a block containing a transaction of interest. CT is a practical measure of usability of a blockchain system. Lower CT allows the system to be used for fast payments and is a step towards making blockchains an alternative to centralized payment networks such as VISA. We find the CT reductions possible through anchors via an analysis of the double spend attack which has been traditionally accepted [1], [10] as the method to find CT.

Consistency is an important security property that tells us the maximum adversary power at which an honest entity, that is accepted by all honest peers and has stayed on their chain for at least till time t, will (with high confidence) continue to remain on honest chains forever even under influence of an attack. Later in this section we discuss the consistency threshold in a system with anchors.

CT is a measure of time defined for a fixed adversary power (q) and a probability of success of a specific double spend attack. Consistency, on the other hand, is a bound that tries to find the maximum adversary power under which the system is shielded from any type of adversary attack.

A. Double Spends & Confirmation Time

Nakamoto [1] and Rosenfeld [10] originally studied the double spend attack in a synchronous network model ($\Delta_a, \Delta_b = 0$). Their result stated that sellers should wait till there were at least 6 blocks, from the time their transaction enters a block, to confirm payment with 99.99% confidence

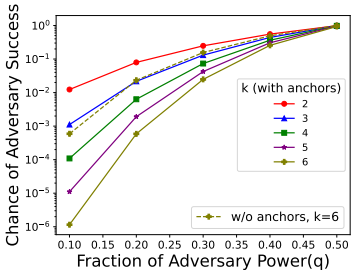

Fig. 2. CT comparison at anchor frequency $a = 2$ at various block confirmations (k) of systems with and without anchors

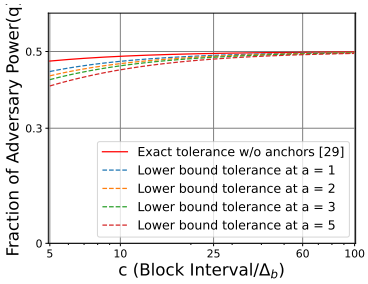

Fig. 3. Adversary tolerance at various block intervals between classic PoW systems and system with anchors

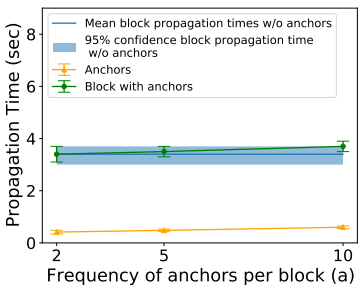

Fig. 4. Delay of ≈1MB blocks with and without the presence of anchors

in the presence of $q = 0.1$ adversary power. We call this 6 blocks the **number of confirmation blocks** denoted by k. In Bitcoin, $k = 6$ translates to a mean 60 minutes waiting time. We show that a Bitcoin system with anchors reduces this CT by half to 30 minutes for any adversary power. We start with an analysis of CT in a zero-delay model like that of [10] and factor in delay of blocks and anchors using the growth result from Theorem III.2.

In a *Double Spend Attack*, miners are divided into honest and Byzantine. Consider a blockchain without anchors formed of blocks B_i where $i = 0, 1, 2 \ldots$. A Byzantine adversary with q fraction of total hashing power wants to double-spend some money already spent in a transaction in block B_{i+1} in the publicly accepted heaviest chain. Assume that the recipient of that money waits for k blocks (the CT), that is till B_{i+k}, to be confident that the transaction will be in the chain with high probability. To do a double spend, the attacker includes a double-spend in a privately mined chain forked from B_i and releases it any time after B_{i+k} is generated, provided it is at least as long as the public chain at the time of release. Note that the adversary has focused all their mining power on the growth of only their private chain from B_i in this attack.

Factoring Delay: Recall our system and adversary model from section II. To factor in block and anchor delays and to allow a most direct and impactful comparison of results with [10], we adjust the honest mining power in this analysis such that it approximates the partially synchronous network honest growth rate derived in theorem III.2. This way the adversary gains an advantage over his actual mining power q. For the rest of our analysis we will assume the adversary holds a fraction $q' > q$ such that $\frac{1-q'}{q'} = \frac{\upsilon}{\beta}$. We compare the CT results of using anchors against the Bitcoin blockchain by substituting the following values to calculate honest (υ) and adversary (β) growth rates: $c = 60$, $\Delta_b = 10^{13}$ to represent the Bitcoin system. Using the new higher fraction of mining power for the adversary, q', we perform the analysis similar to [10]. By changing the parameters c, Δ_b we can compare any other PoW blockchain system.

Assumptions and Notations: We fix the number of confirmation blocks as $k = 6$ given the acceptable $k \geq 6$. Note that any $k > 6$ will only improve the security guarantee. Although we discuss results for $k = 6$ confirmation blocks, this analysis can be easily extended to other values of k. Suppose B_{i+1} is the block on the honest chain to be confirmed. B_i is in both honest and adversary chains. Let there be a total of H honest anchors mined between B_i and B_{i+k}, and let the adversary have L blocks mined in his private chain from B_i with a total of R anchors on it by the time of arrival of B_{i+k}. The process of anchor and block generation are modeled as Poisson distributions with different intensities depending on q' and α. We have the probability of success of the double spend attack by an adversary holding q fraction of power as, $\mathbb{P}_q(\text{success}_{\text{adv}}) = \sum_z P(z)Q(z)$ where $P(z)$ is the probability that the honest chain is ahead by z in weight over the attacker's chain at the time of confirmation i.e at the arrival of B_{i+k} and $Q(z)$ is the probability of the attacker catching up to the honest chain given that the attacker's chain differs by z in weight. We study the $\mathbb{P}_q(\text{success}_{\text{adv}})$ after $k = 6$ honest confirmations in 2 systems - Bitcoin and Bitcoin with anchors.

Computation of $P(z)$: Computation of $P(z)$ can be modeled as a sequencing problem of ordering the arrivals (from the perspective of the oracle) of $k = 6$ honest blocks, H honest anchors, L adversary blocks and R adversary anchors at an adversary node. Note that all attacker anchors that arrive before the 1st block of the attacker's chain or the honest anchors before B_{i+1} must not be considered for this analysis as they point to B_i and the weight contributed by them is common to both the honest and adversary chain and will not affect the outcome of the double spend attack. These anchors are not counted as part of H and R. Therefore this analysis differs from those of [1], [10] in that it includes anchors which have a different weight than blocks, and takes care of issues such as common anchors for both honest and attacker chains (on block B_i) which do not occur when there are no anchors. Hence we start our sequencing from the arrival of the first block after B_i. Let us call the case when the first block after B_i is honest as case(i) and the case where the first block after B_i is from an adversary as case(ii). The number of positions is $N = 6 + H + L + R$, over a wide range of H, L, R such that $z = (6 + H\alpha) - (L + R\alpha)$ and $\sum_z P(z) \approx 1$. $P(z)$ is the sum of the probabilities from case(i) and case(ii) over all z. We provide the outline for the calculation below. The exact

calculation will be provided in the extended version of this paper.

Outline to calculate the probability of Case(i): In the sequencing of $1 \dots N-1$ arrivals (where N^{th} arrival is the k^{th} honest block), we know that an honest block arrives before the first adversary block. Let position j be the arrival of the first adversary block such that $2 \leq j \leq N-1$. The $j-2$ positions between the first honest block and the first adversary block can only be filled by two possibilities - more honest blocks (which has an upper limit of 4, excluding the first and last honest blocks) or honest anchors. The remaining $N-j-1$ positions can be filled by all 4 possibilities - honest and adversary blocks and anchors. This is followed by an honest block at the N^{th} position. Multiplying all these probabilities over all possible values of j gives us the probability of Case(i). A special scenario in case(i) is when $j > N$ i.e. $L = 0$ or no adversary block arrives by the time of confirmation. Hence adversary anchors cannot be counted. Therefore, we only consider the probability of choosing 4 blocks in N positions and honest anchors fill the remaining positions.

Outline to calculate the probability of Case(ii): In the sequencing of $1 \dots N-1$ arrivals, we know that an adversary block arrives before the first honest block. Let position j be the arrival of the first honest block such that $2 \leq j \leq N-1$. The $j-2$ positions between the first adversary block and the first honest block can only be filled by two possibilities - more adversary blocks or adversary anchors. The remaining $N-j-1$ positions can be filled by all 4 possibilities - honest and adversary blocks and anchors. This is followed by an honest block at the N^{th} position. Multiplying all these probabilities over all possible values of j gives us the probability of Case(ii).

Computation of $Q(z)$: $Q(z)$ is the probability that the attacker will catch up to the honest chain given the honest chain is ahead by z. It follows that $Q(z) = 1$ when the attackers chain is longer that the honest chain. An upper bound to $Q(z)$ can be estimated by modeling this problem as a random walk starting from z to 0, over all z with 4 jump possibilities (± 1, and $\pm\alpha$) due to occurrence of two types of arrivals - anchors and blocks. This is a well known result from [30]. Calculation for the Bitcoin blockchain is similar to Rosenfeld [10] i.e. $Q(z) = q^z$ for $z > 0$ and 1 otherwise.

We hence obtain an upper bound on $\mathbb{P}_q(\text{success}_{\text{adv}})$ using the upper bound on $Q(z)$ and the exact computation of $P(z)$. Since what we plot is an upper bound on $\mathbb{P}_q(\text{success}_{\text{adv}})$, the actual improvements with anchors exceed those observed in the result.

Observations: We compare the chance of adversary successfully performing a double spend attack in a system with anchors to a system without anchors by [10] where it was shown that at $q = 10\%$ the probability of a successful attack with $k = 6$ is $0.59 * 10^{-3}$. By the inclusion of anchors at $a = 2$, the probability of adversary success plunges down to an insignificant $0.11 * 10^{-5}$ at the same $q = 10\%$ and $k = 6$. This is illustrated in figure 2 where solid line plots represent Bitcoin with anchors at various CT ($2 \leq k \leq 6$) and the dashed line plot is Bitcoin at CT $k = 6$. The Bitcoin hash-rate distribution [31] shows the largest mining pool size varying between 17 to 23%. Therefore, at a more realistic $q = 25\%$, we still see an order of magnitude difference between with anchors and without anchors. Intuitively we see that confirmation time can reduce significantly with the help of anchors. To know by how much, we simulate the probability of success for $k = \{2, 3, 4, 5\}$ in the discussed method with anchors and compare it to $k = 6$ without anchors. Figure 2 shows the results for $a = 2$. We can see that with anchors at just $a = 2$ the line $k = 3$ overlaps the plot without anchors at $k = 6$. Our double spend analysis and comparison to that of [10] is conservative in many ways: (1) We use the adversary model from section II which describes a general adversary. In this attack we know the adversary only extends his private chain, hence υ can be much higher; and (2) We have compared Rosenfeld's [10] result for Bitcoin in a synchronous model with anchors studied in a partially synchronous model for network delays. This means Rosenfeld's probability for adversary success is higher than the actual while our results are not. Therefore we conclude, anchors are capable of reducing the number of confirmation blocks to at least $k = 3$ at the current level of security. Alternatively, they can significantly increase the security threshold at the current number of confirmations.

B. Consistency

Consistency bound for PoW systems was first defined by Garay et.al. [18] in a synchorous model and was extended to partially synchrounous model by Pass et.al. [23]. It was later improved by [32], [33]. Recent works by Dembo et. al. [29] and Gazi et. al. [34] made a breakthrough for tight consistency bounds in PoW blockchains that follow the Nakamoto consensus. They showed that all attacks on blockchain system can be viewed as a race between an honest chain and an adversary chain and that the 51% attack is the worst possible attack. A 51% attack is when the adversary gains control of more than 50% of a network's mining power. Attackers with majority control of the network can interrupt the recording of new blocks. They would also be able to reverse transactions that were confirmed, meaning they could double-spend coins. The 51% attack scenario is rare, largely because of the logistics, hardware and costs required to carry one out.

This implies that as long as the honest growth rate is higher than adversary growth rate in unit time i.e. $\upsilon > \beta$, consistency is always maintained. We found that the approach followed by [29] can be extended to our work on a PoW system with anchors to derive the same result in terms of our growth in weight from Theorem III.2. To summarize, the technical concepts such as blocktree partitioning, fictitious honest tree, nakamoto block and loner blocks that are used in the arguments of [29] can be defined with slight modifications to suit a system with anchors to achieve the same result that consistency is achieved when $\upsilon > \beta$. Due to space constraints we defer the detailed conversion of Dembo et. al. for anchors in the extended version of this paper.

With the help of the growth result in general adversary model defined in theorem III.2, we show the lower bound consistency threshold in systems with anchors in Figure 3. We numerically compute the maximum threshold of q at typical block intervals c, used in practice under which consistency holds i.e. $\upsilon > \beta$. For instance Bitcoin (10 min block interval) can be studied at $c = 60$ [23], [32] and Ethereum (20 sec block interval) is represented at $c = 5$ [35]. We evaluate this for $n = 10^5$, $\Delta_b = 10^{13}$ and $a = 1$ to 5 shown in Figure 3. The comparison made in figure 3 is between tight consistency threshold of classic PoW chains [29], [34] and a lower bound consistency in systems with anchors. This is because the growth used for anchors is conservative. For a system with anchors, at $c = 60$, adversary tolerance is at most $q = 49.5\%$ in all values of a tested. In the state of the art analysis by [29], [34] for classic PoW systems, adversary tolerance is at most $q = 49.57\%$ at $c = 60$. For a system with anchors, at $c = 5$, tolerance is at most $q = 45.7\%$ and $q = 43\%$ at $a = 1, 5$ respectively. For classic PoW systems, adversary tolerance is at most $q = 47.5\%$ at $c = 5$. With a tightened growth we expect anchors to close the threshold difference as in classic systems for $c < 60$. This study is left as future work.

This result shows that PoW blockchains with anchors has similar consistency lower-bounds as classic PoW blockchains at higher $c \geq 60$ and differs by at most a factor of 8% at $c < 60$ from the analysis of [29], [34]. Thus, we can conclude that anchors preserve the consistency of the underlying classic blockchains they are implemented on.

V. Experimental Evaluation

In this section, we study the feasibility of implementation and performance of anchors in a PoW blockchain system. We present our emulated results using a reference implementation of a Bitcoin system that incorporates anchors. We benchmark it against unmodified Bitcoin in a set of experiments which test whether anchors propagate faster than blocks, if the presence of anchors impacts block propagation times, the fork resolution times and number of forks. We do not consider an anchor reward structure as it is inconsequential to our experimental hypotheses. Security against a general adversary is proved in section IV-B and hence adversary presence is not considered in our emulations.

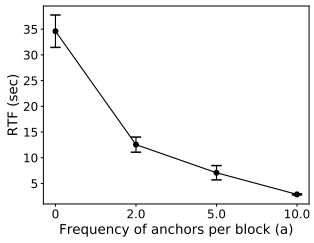

Fig. 5. Mean fork resolution time with 95% confidence intervals

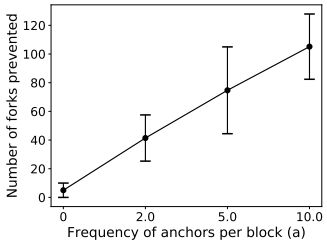

Fig. 6. Mean number of forks prevented with 95% confidence intervals

Modifications made to Bitcoin: We use Bitcoin Core client v0.16 [36] as a base and add support for anchors. This includes a new anchor message type and a single new RPC - generateanchor, which creates a valid anchor on the current tip of the chain before broadcasting it. This RPC is modeled closely after the existing generate call, which provides similar functionality for blocks. The only other change was adding methods to process anchors upon receiving them i.e. updating the chain weight and invoking HCR.

Network overlay structure: Our experiments were conducted on a 210 Bitcoin node testbed, running on a cluster of 40 machines (8 GB RAM, 4-8 CPU cores). Similar to other Bitcoin emulations [37], we construct a topology where each node is connected to 4-8 other nodes, chosen uniformly at random. The network set-up phase also includes adding additional delays between every pair of connected nodes that mimic real world latencies between nodes. A group of nodes represents a major city in the world, and each connection between two groups has corresponding inter city latency taken from [38].

A. Observations on Propagation Delays

Propagation time (or delay) of any message is the difference between the time of its broadcast at the source node and the time of its arrival at the destination. To remove extraneous outliers, we calculate the 95th percentile of propagation times as in [39]. All experiments were conducted with the default compact block relay in Bitcoin. We set the standard average block interval as 600 secs as in Bitcoin. To have near-full blocks, there must exist sufficient transactions to fill them, which requires miners to have bitcoins to spend. Therefore, we generate several blocks with just the coinbase at the start of the experiment, which gives miners bitcoins via block rewards. Nodes are also given the responsibility of generating their own transactions every few seconds. This is to ensure healthy traffic during the emulation and non-empty mempools.

For the anchor structure implemented, we see that all anchors have a fixed size of 264 Bytes. The mean block size observed across all runs of our experiments was 1.12MB. Comparing average propagation times of blocks vs anchors, we observed that anchor propagation times are at least 6 times faster than blocks as shown in figure 4. Our hypothesis that an anchor will propagate faster than a block is verified. The average propagation times of blocks were found to be $3.46, 3.52$, and 3.7 secs under modes $a = 2, 5, 10$ respectively, whereas, it was 0.45 secs for all values of a for anchors. In the system without anchors, the average block delay was 3.45 secs. We can thus say that anchors work with the system without creating drastic bandwidth or latency overheads.

B. Observations on Forks

In Bitcoin, the block interval time is set high to make forks rare events and so studying fork resolution times is difficult. We could not observe enough forks in the 600 secs block interval setting for any value of a in our short experimental runs to make good inferences. However, other

PoW systems [2], [6], [7] have small block interval that make forks frequent events. Therefore, in order to get good estimates of forking behavior in short experiments, we scale down the block interval from 600 seconds to 30 seconds, so that natural forks occur in every run. The anchor generation is set in an exponential fashion with mean interval as 30α secs. This scaling did not affect any entity propagation delays.

Resolution Time of forks (RTF): RTF is calculated by every miner as the difference between the time a fork was realized on its chain and the time at which the next block or anchor increased the chain weight on one of the branches, effectively resolving it. We observe that anchors reduce the mean fork resolution time by 64, 80 and 91 percent, with a = 2, 5 and 10 respectively. Figure 5 shows the results of average resolution time of forks observed in our network while varying anchor frequency (a) to 2, 5, and 10 when compared to the Bitcoin network without anchors($a = 0$). Forks are resolved by each miner, based on the chain he observes locally. We found that, for every fork, the chosen resolution branch is the same across the network. Thus anchors succeed in reuniting mining power (divided across the branches of a fork) to the same chain.

Prevention of Forks: We consider fork prevention as a scenario, occurring at each node, where an anchor arrives on a block earlier than a sibling block that would otherwise create a fork in the chain, preventing the fork before it could even occur. For this experiment, we calculate the number of forks prevented as the number of times a potential fork is averted due to the arrival of an anchor at each node. Figure 6 shows the average forks prevented when varying a and comparing it to a system without anchors. Therefore, anchors help reduce the number of forks formed over time.

VI. Related Work

We designed anchors as a solution to securely improve their chain stability, fork resolution and confirmation speeds on any new or existing PoW blockchain system. We now discuss some related attempts to improve security and performance in PoW blockchains in the literature and if anchors can benefit them.

Strong Chain [25] adopts a hierarchical structure of weaker blocks (created with lesser PoW than full blocks similar to anchors) without transactions aiming to reduce mining variance, selfish mining and centralization by mining pools. However, the weak-blocks of Strongchain differ from anchors in the following important ways. Firstly, the weak-blocks affect chain weight (calculated according to their chainPoW() function) at a later point in the chain when the weak-blocks are added inside normal blocks. This is unlike anchors as miners update anchor weight as soon as it is received and this can influence the chain to continue mining on. This detail is critical to all analyses done on this system. Secondly, Strongchain does not consider the propagation delays or network overheads imposed by weaker-blocks. Consequently, they propose an impractically high frequency of weak-blocks per-block (1024) which will cause significant overheads in terms of latency and bandwidth and reduce security. In sections III and IV, we find that the anchor frequency(a) inversely affects chain growth and consistency. Therefore, choosing an appropriate a is important. Additionally, including 1024 weak-blocks inside blocks will also increase block size by $\approx$10% which will affect its propagation time that may further delay consistency as studied in section IV-B.

Many other works that aim to reduce CT by reducing average block size and intervals [2], [5]–[7], [40], [41], introduce forks. They later use the orphaned blocks to resolve later forks via the GHOST [42] protocol. Anchors cause no additional forks in the system. Further they help reduce fork resolution times and prevent forks from occurring. Anchors is a solution orthogonal to GHOST and can be implemented on system using GHOST for better results. Some works such as [8], [9], aiming to improve CT as a partial solution to the scalability problem, modify the single-chain, linear structure of the blockchain. Therefore, implementing them to benefit existing, widely-used, single-chain PoW blockchains like [1], [3], [4] is complex. They maintain multiple parallel blockchains where each chain may be susceptible to the instabilities of PoW chains. Assigning rewards, transaction duplication and ordering are other complex problems that need to be addressed. Anchors offer a simple implementation with minimal modifications on top of popular, available PoW blockchains to help securely improve CT and fork resolution times. Anchors can even be implemented on the multiple PoW chains inside systems like [8], [9] to reduce the instabilities and improve security on each chain.

Other works [8], [24], [37], [43]–[45] use weaker blocks that store transactions aiming to partly solve transaction throughput problem or improve chain quality. Anchors solve an orthogonal problem compared to these systems. Our delay model (§II) shows how transactions make the weak blocks heavy and increase their propagation times. Anchors are light and hence can be used as a fast signaling mechanism to regularly identify the heaviest chain. Anchors can also be used to overcome vulnerabilities like frequent forks on such systems.

VII. Conclusion and Future Work

Anchors can offer better confirmation times, faster fork resolutions, fewer forks and better growth rate with security guarantees to any PoW blockchain they are implemented on. They signal mining power division and help miners make a more rewarding choice in case of forks also improving mining power utilization. They have the potential to mitigate and reduce profitability of double spends and a broad genre of selfish mining attacks. The concept behind anchors is not limited to single chain PoW blockchains. It can be extended to DAG-based or multi-chain blockchains as long as they can apply the HCR rule and support weaker PoW solutions. Analysis of the benefits of such implementations are left as future work.

Acknowledgment

This work was supported by the Visvesvaraya Scheme by Meity, GOI. The authors would like to thank Aditya Kumar for

his valuable contributions in the early stages of this work. We also thank Subodh Sharma, Himanshu Gandhi, Aditya Ahuja, Dilpreet Kaur and Sandeep Kumar for their insightful reviews on the early draft of the paper.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] V. Buterin and Others, "Ethereum white paper, 2014," *URL https://github.com/ethereum/wiki/wiki/White-Paper*, 2013.

[3] "Litecoin - open source p2p digital currency," https://litecoin.org/, (Accessed on 05/16/2019).

[4] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," *July 7th*, 2013.

[5] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[6] Y. Sompolinsky and A. Zohar, "Phantom: A scalable blockdag protocol," 2018.

[7] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 515–528.

[8] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the Blockchain to Approach Physical Limits," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: ACM, 2019, pp. 585–602. [Online]. Available: http://doi.acm.org/10.1145/3319535.3363213

[9] H. Yu, I. Nikolic, R. Hou, and P. Saxena, "OHIE: blockchain scaling made simple," *arXiv preprint arXiv:1811.12628*, 2018.

[10] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, 2014.

[11] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Čapkun, "Misbehavior in bitcoin: A study of double-spending and accountability," *ACM Transactions on Information and System Security (TISSEC)*, vol. 18, no. 1, p. 2, 2015.

[12] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.

[13] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 515–532.

[14] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 305–320.

[15] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 154–167.

[16] C. Natoli and V. Gramoli, "The balance attack against proof-of-work blockchains: The R3 testbed as an example," *arXiv preprint arXiv:1612.09426*, 2016.

[17] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network." in *USENIX Security Symposium*, 2015, pp. 129–144.

[18] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[19] K. Liao and J. Katz, "Incentivizing double-spend collusion in bitcoin," in *Financial Cryptography Bitcoin Workshop*, 2017.

[20] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 104–121.

[21] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, and Others, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 106–125.

[22] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM CCS, 2012, pp. 906–917.

[23] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 643–673.

[24] R. Pass and E. Shi, "FruitChains: A Fair Blockchain." in *Acmccs*, vol. 2016, 2017, pp. 1–26.

[25] P. Szalachowski, I. Homoliak, and S. Sun, "StrongChain: Transparent and Collaborative Proof-of-Work Consensus," *arXiv preprint arXiv:1905.09655*, 2019.

[26] I. Eyal, "The miner's dilemma," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 89–103.

[27] "Bitcointalk: Mining cartel attack," 2010. [Online]. Available: https://bitcointalk.org/index.php?topic=2227.msg29606{\#}msg29606

[28] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, "Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '15. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 919–927.

[29] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Everything is a race and Nakamoto always wins," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 859–878.

[30] R. Gallager, "6.262 Discrete stochastic processes, Spring 2011 Chapter 7: Random Walks, Large Deviations, and Martingales," *Massachusetts Institute of Technology: MIT OpenCourseWare) http://ocw. mit. edu (Accessed 23 Mar, 2013). License: Creative Commons BY-NC-SA*, 2011.

[31] "Bitcoin hashrate distribution - blockchain.info," https://www.blockchain.com/pools?, (Accessed on 11/12/2019).

[32] L. Kiffer, R. Rajaraman, and Others, "A better method to analyze blockchain consistency," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM CCS18, 2018, pp. 729–744.

[33] L. Ren, "Analysis of nakamoto consensus." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 943, 2019.

[34] P. Gaži, A. Kiayias, and A. Russell, "Tight Consistency Bounds for Bitcoin," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 819–838.

[35] N. Awathare, S. Das, V. J. Ribeiro, and U. Bellur, "Renoir: Accelerating blockchain validation using state caching," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, 2021, pp. 9–20.

[36] "bitcoin project," https://github.com/bitcoin/tree/v0.16.0, 2018.

[37] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A Scalable Blockchain Protocol." in *NSDI*, 2016, pp. 45–59.

[38] "Global Ping Statistics - WonderNetwork," https://wondernetwork.com/pings, 2019.

[39] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[40] A. Kiayias and G. Panagiotakos, "Speed-Security Tradeoffs in Blockchain Protocols." *IACR Cryptology ePrint Archive*, vol. 2015, p. 1019, 2015.

[41] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.

[42] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[43] G. Andresen, "[bitcoin-dev] Weak block thoughts..." https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-September/011157.html, 2015.

[44] P. R. Rizun, "Subchains: A technique to scale bitcoin and improve the user experience," *Ledger*, vol. 1, pp. 38–52, 2016.

[45] A. Zamyatin, N. Stifter, P. Schindler, E. R. Weippl, and W. J. Knottenbelt, "Flux: Revisiting Near Blocks for Proof-of-Work Blockchains." *IACR Cryptology ePrint Archive*, vol. 2018, p. 415, 2018.