Longest-chain Attacks: Difficulty Adjustment and Timestamp Verifiability

TZUO HANN LAW, Boston College, USA
SELMAN EROL, Carnegie Mellon University, USA
LEWIS TSENG, Boston College, USA

We study an adversary who attacks a Proof-of-Work (POW) blockchain by selfishly constructing an alternative longest chain. We characterize optimal strategies employed by the adversary when a difficulty adjustment rule alà Bitcoin applies. As time (namely the timestamp specified in each block) in most permissionless POW blockchains is somewhat subjective, we focus on two extreme scenarios - when time is completely verifiable, and when it is completely unverifiable. We conclude that an adversary who faces a difficulty adjustment rule will find a longest-chain attack very challenging when timestamps are verifiable. POW blockchains with frequent difficulty adjustments relative to time reporting flexibility will be substantially more vulnerable to longest-chain attacks. Our main fining provides guidance on the design of difficulty adjustment rules and demonstrates the importance of timestamp verifiability.

CCS Concepts: • **Theory of computation → Distributed algorithms**.

Additional Key Words and Phrases: Bitcoin, blockchain, longest chain attack, difficulty adjustment, cryptocurrency, Proof-of-Work

1 INTRODUCTION

Permissionless Proof-of-Work (POW) consensus systems feature a network of peer-to-peer nodes that add blocks containing information to a blockchain. Since every node prefers blocks containing information specific to their own benefit, there will be no consensus unless a mechanism that grants nodes permission to add blocks is in place. Nakamoto [12] consensus is one such mechanism. Nodes earn the privilege to propose a block by reporting a suitable nonce to accompany a block containing specific information of their own choice. Receiving nodes accept the proposed block if it is valid. A nonce is a number embedded in the block so that the output of some cryptographic hash function of the block fulfills some condition. Typically, this condition takes the form of a difficulty threshold. An acceptable nonce is one which yields a block hash that falls within a specified distance from zero. Additionally, the information within the blocks must conform to a set of rules referred to as the blockchain's protocol.

Due to the way that nonces are used, suitable ones can only be found by trial and error. The process of nonce finding is a race of who can test potential numbers most quickly and report their finding to the rest of the network. Nonce finding is literally work in the sense that the only practical way to do it is to pass electrical current through computer chips. A suitable nonce is quite simply, proof that work was done. Other nodes who receive this information then check if the newly received information contains the more *cumulative* and *valid* work than their own leading block as measured by the same metric.[1] If it is the case, the receiving node would accept the newly minted block and build on top of it. This process is referred to as *mining* and amounts to an arms race. Miners who control more (powerful) computer hardware and have access to cheap energy are able to test potential nonces more quickly. As a result, these miners add more blocks in their own favor.

Without additional safeguards, such a system would imply that an increase in the system mining capacity would result in a higher rate of token generation. To stabilize the token generation rate, blockchain protocols typically specify a *difficulty adjustment* protocol or rule. The difficulty is adjusted so that the token generation rate is steered towards some target rate as defined in the

[1]Usually, the longest chain in a blockchain also requires the most amount of cumulative work to create. However, this is not always the case. For example, see Ethereum prior to its transition to Proof-of-Stake.

protocol. There are many different difficulty adjustment rules being used with this same overarching mandate. In this paper, we consider one that is modeled after the protocol used in Bitcoin [12].

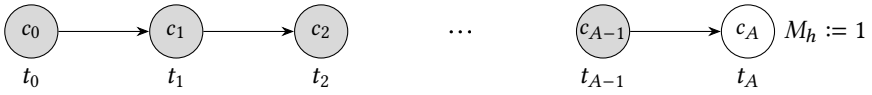

Fig. 1. Simplified POW blockchain. The block identifier, subscripted by its blockheight (e.g. c_i, c_A) refers to the block as well as the unique blockchain formed from tracing the block back to c_0. t_i is the reported timestamp for block c_i. The frame of reference is t_A where c_0 to c_{A-1} are mined (in grey) and c_A (in white) has been constructed, but not mined (unknown suitable nonce).

Ideally, a difficulty adjustment rule would adjust the difficulty level according to the mining capacity of the network since that is the primary determinant of the block-finding rate. However, the mining capacity of the network is unobservable and particularly so in permissionless systems. Instead, the difficulty adjustment algorithms utilize the time taken for successive blocks to be created as an estimate of mining capacity and this is in turn "**proxied**" by the timestamps reported in each block. We emphasize the word "proxied" because the system time of a different node is itself unobservable due to the nature of asynchronous networks. Nodes can report any timestamp they please so long as the reported timestamp conforms to some protocol which in turn ensures its acceptance by other nodes. In addition, nodes can successfully mine a block and not report their success until a later time. Since there are various protocols for accepting/rejecting timestamps, there is substantial variation in the flexibility nodes have for timestamp reporting across different POW blockchains.

We investigate how this timestamp flexibility in relation to the difficulty adjustment rule influences the *optimal strategy* that an adversary employs when mounting a longest-chain attack. To make key ideas clearer and more transparent, we work with deterministic mining, which was also adopted in [9, 11]. The strategies that apply in a deterministic setting continue to apply in the appropriate probabilistic setting under the right assumptions about the adversary's preferences. Other than benign analytical oddities such as integer constraints, none of our key points and findings depends on the deterministic mining assumption that we make.

In this paper, we characterize the optimal strategies in a simplified POW blockchain with deterministic mining [9, 11] and comment on how the insights from our analysis would translate to real-world implications. Our main finding is that difficulty adjustment rules offer substantial protection against longest-chain attacks provided timestamps are accurate relative to the frequency of the difficulty adjustment.

2 RELATED WORK

We discuss the closely related works that investigate the effect of varying mining difficulty and attacks on manipulating timestamps or mining difficulty. Garay et al. [6] formalize and analyze the core of the Bitcoin protocol, namely the Bitcoin backbone, in the static setting (with fixed number of nodes and fixed difficulty). Subsequently, Garay et al. [7] extend and analyze Bitcoin backbone protocol with mining difficulty adjustment by formulating the target (re)calculation function in Bitcoin. Kraft [10] and Noda et al. [14] study the effect of mining difficulty adjustment and block arrival rate.

The notion of selfish mining is first proposed in [4], which demonstrates disobedient mining could be more profitable than being honest, i.e., following the Bitcoin specification. Subsequently, Davidson and Diamond [3] and Alarcón Negy et al. [13] investigate the profitability of selfish mining

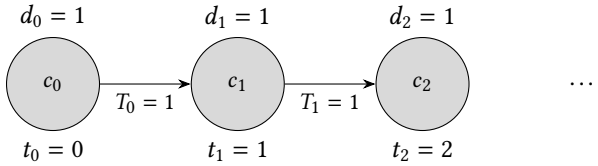

Fig. 2. With the assumptions made, the canonical chain grows constantly a block per unit of time.

[4] under different difficulty adjustment mechanism using simulation. In particular, it is shown that an intermittent selfish mining strategy [13] gives the attacker a higher profit by performing selfish mining intermittently. Per time-unit profitability of selfish mining under different difficulty adjustment mechanisms is also identified in [13].

Meshkove et al. [11] introduce the coin-hopping attack which allows attackers to gain profit compared to honest players by exploiting the mining difficulty adjustment. Boverman [2] describes an attack which forces honest players to accept a chain other than the canonical chain by tampering the "network time counter" at an honest player or even a majority of players. Fiat et al. [5] and Goren and Spiegelman [8] introduce the Energy Equilibria attack which minimizes operational (energy) costs and manipulate mining difficulty to increase mining rewards per unit of time on average.

Yaish et al. [16] introduce an approach, called stretching and squeezing, which can create and exploit interest-rate arbitrage between decentralized finance platforms by manipulating the mining difficulty. They also find two timestamp weaknesses in Geth's code. Later, Yaish et al. [15] identify an attack, called uncle maker, that allows attacker to gain higher profit by manipulating block timestamps at proper times.

To the best of our knowledge, we are the first to investigate the *joint effect* of difficulty adjustment and timestamp verifiability on the optimal strategy for mounting longest-chain attacks.

3 SIMPLIFIED PROOF-OF-WORK BLOCKCHAIN

3.1 Honest miners

We model a vastly simplified Proof-of-Work (POW) blockchain with genesis block c_0 as shown in Figure 1. Honest miners control $M_h := 1$ mining capacity and do *not* behave strategically. They naively extend the longest chain known and ignore all others. In our notation, block c_i is defined, but yet to be mined at time t_i, i.e., its header is fixed but a suitable nonce remains unknown. The only active miners prior to t_A are the honest miners. At that time, honest miners have mined c_0 to c_{A-1}, shaded grey, and are about to start mining block c_A. An adversary which we will later describe initiates an attack at time t_A.

3.2 Deterministic Mining

With deterministic mining [9, 11], if M_i mining power is dedicated to a block with difficulty d_i, then a suitable nonce will take

$$T_i = \frac{d_i}{M_i} \tag{1}$$

time units to be discovered.

The target block finding rate is 1 block per-unit time and each difficulty epoch contains one block. In the spirit of Bitcoin's difficulty adjustment rule, the difficulty adjusts with every block/epoch

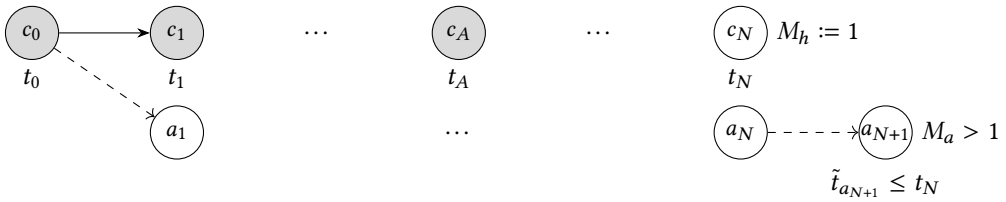

Fig. 3. Attack starts at time t_A. Adversary must construct some chain a_{N+1} that is at least a block longer than the honest miner's chain. In this figure, we align both chains by block height. Note that $\tilde{t}_{N+1}$'s timestamp must be no more than t_N.

following

$$d_{i+1} = \frac{d_i}{\tilde{T}_i} = \frac{d_i}{(\tilde{t}_{i+1} - \tilde{t}_i)} \qquad d_0 = 1 \tag{2}$$

where $\tilde{t}$ refers to the timestamps reported in the blocks.

Combining Equations (1) and (2) yields

$$d_{i+1} = \frac{d_i}{\tilde{T}_i} = \frac{d_{i-1}}{\tilde{T}_{i-1}\tilde{T}_i} = \frac{d_0}{\prod_{j=0}^{i} \tilde{T}_{i-j}}$$

As in [11, 12], we assume no propagation latency effects and no block transition delays. Honest miners propagate a block as soon as its nonce is discovered and immediately start mining on the next block. On the honest chain,

$$d_{i+1} = \frac{d_i}{T_i} = M_i$$

and consequently, in our setup, $d_i = T_i = M_h = 1$ and that $t_i = i$ as shown in Figure 2.

3.3 Adversary

Our adversary seeks to replace c with a longer chain without any concern for the value of c or for the energy costs consumed. More concretely, the adversary could be a hostile government or sophisticated hacker, a large short financial position on c's tokens, or a competing blockchain. Given recent developments in quantum computing, it would not be a stretch to think of the adversary as a miner armed with quantum computing capabilities as described in Bard et al. [1].

The adversary initiates the longest-chain attack at time t_A which coincides with the time honest miners start mining block c_A. The adversary controls mining capacity $M_a > M_h$ and seeks to replace c by constructing an alternative chain starting at c_0. This analysis also applies if the adversary attacks a blockchain A blocks behind its most recent block. We name this adversarial chain a. We use $\tilde{t}$ to denote timestamps as reported by the adversary for chain a.

While honest miners are extending c_A, the adversary must construct a_1, a_2, ..., a_{N+1} where a_{N+1} is the first block higher than c_N which is the terminal block on the chain c. Revealing a_{N+1} to the honest miners ends the game in favor of the adversary and invalidates all transactions between c_N and c_1 due to the longest-chain rule. The adversary also seeks to end the game as *quickly* as possible as measured by the time it spends on the attack. Since chain a is revealed at t_N, $\tilde{t}_{a_{N+1}}$ must be less than or equal to t_N. The timestamp associated with a_1 is t_1 because c_1 and a_1 share a common ancestor and we assume no network propagation delay. Finally, we assume in this paper that the adversary does not dedicate any of its mining capacity to the canonical chain and operates stealthily until it reveals chain a.

With all that we have described, what is the best that the adversary can do? As was suggested earlier in this paper, the answer will depend on how much flexibility the adversary has in reporting timestamps which is unobservable by honest miners.

4 WHAT IS THE TIME?

Time is a subjectively defined object in Bitcoin Core's protocol. As of January 2023, a node accepts a block if the timestamp reported is within lower and upper bounds.[2] The lower bound is the median timestamp of the node's previous 11 blocks. The upper bound is the median time reported by other connected nodes plus two hours. In other words, "time" specified in a block is valid so long it is not from the distant past or future.

This rather fluid notion of time directly affects the difficulty adjustment rule. Specifically, difficulty adjustment ensures that fluctuations in mining capacity changes will result in temporary deviations from the target token production schedule. Since this rule applies to all chains, an adversary who chooses to mount a longest-chain attack must also adhere to this rule while constructing chain a. In order to determine how the difficulty adjustment rule affects the adversary's attack, we must now be precise about the nature of time.

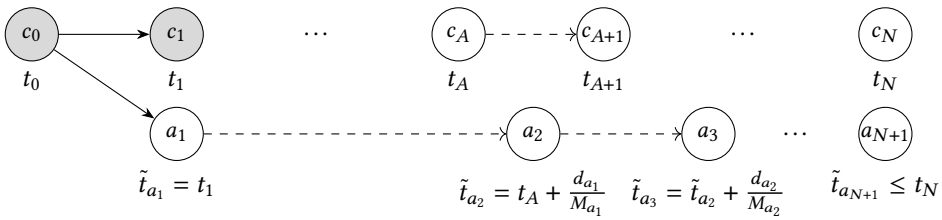

Fig. 4. We now align both chains by time. The adversary reports all timestamps truthfully (because it has to). Notice that a_2 is created after t_A because the adversary started mining at time t_A. a_{N+1} must be timestamped no later than t_N.

4.1 Two Extreme Scenarios

We now consider two extremes of this flexible notion of time. In the first, time is verifiable and the adversarial miner must behave like an honest miner when it comes to reporting time. In the second, time is not verifiable and the adversary is able to report timestamps with a lot more flexibility.

4.1.1 Verifiable timestamps. When timestamps are **verifiable**, the timing of the adversary's actions are observable by the honest miners which essentially forces the adversary to be honest as well. By this, we do not mean that honest miners can see everything the adversary does. We have in mind a situation where it is essentially impossible to falsify time due to the presence of a well-designed accept/reject protocol. It is also possible for a blockchain protocol to feature some sort of hardware technology like Intel SGX which provides trusted time. Figure 4 shows what these timestamps look like when the adversary is reporting honestly.

4.1.2 Unverifiable timestamps. When timestamps are ***unverifiable***, and the honest miners are naive and do not use any additional safeguards, nothing prevents the adversary from choosing a timestamp for block a_i that is different from the time he found the nonce for block a_{i-1}. We allow for any timestamp $\tilde{t}_i$ so long it is after the timestamp $\tilde{t}_{i-1}$. We also assume that blocks cannot contain a timestamp greater than the honest miner's time when the alternative chain is revealed.

[2]Block timestamp https://en.bitcoin.it/wiki/Block_timestamp

Therefore, an adversarial and possibly dishonest miner is free to report any strictly increasing sequence of $\tilde{t}_i$ subject to those caveats. Of course, actual time taken to mine a block continue to obey Equation (1) and the difficulty level continues to evolve following Equation (2) for the chain the adversary is mining on. To highlight the flexibility that the adversary possesses, Figure 5 displays two possibilities for chain a_i.

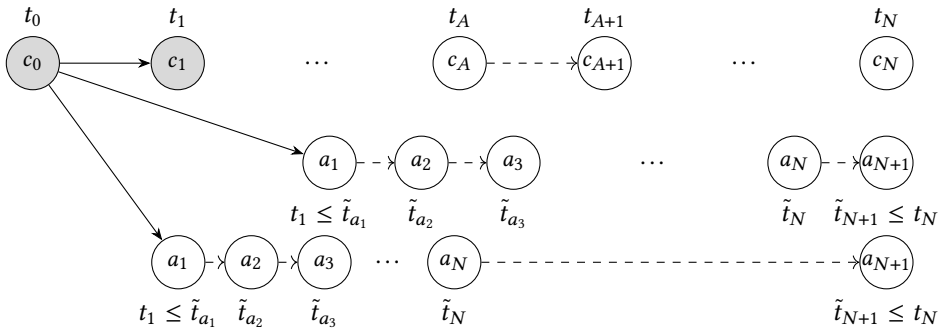

Fig. 5. When time is not verifiable, the adversary needs to report that block a_1 was created after block c_0 and timestamp (and reveal) a_{N+1} on or before t_N. In between, the adversary can choose any increasing sequence of timestamps. Because we have assumed deterministic mining and the honest miners have mined c_0, we assume that the earlier time possible for $\tilde{t}_{a_1}$ is t_1.

5 ADVERSARY'S OPTIMAL STRATEGIES

5.1 Verifiable Timestamps

In this situation, the adversary's optimally chooses $M_i < M_a$ for $i \in \{1, \ldots, N\}$ to construct a.

5.1.1 The naive approach. It is easy to see why choosing $M_i = M_a$ for all i will not work. a_1 has a difficulty level of 1 and is mined by the adversary in $1/M_a < 1$. Therefore,

$$T_1 = t_A + \frac{1}{M_a} - t_1.$$

t_A is the time the adversary commences the attack. $1/M_A$ is the time it took for a block with difficulty 1 to be mined, and t_1 is the time block c_0 was found and hence is the timestamp recorded in block a_1.

Consequently,

$$d_2 = \frac{d_1}{T_1} = \frac{M_a}{(1 + M_a(A-1))}$$

.

$$T_2 = \frac{d_2}{M_a} = \frac{1}{(1 + M_a(A-1))}$$

and

$$d_3 = M_a$$

In other words, the adversary will gain from applying $M_a > 1$ to a single block with difficulty 1. In addition, the adversary will enjoy the delay of $A - 1$ for one single block and thereafter, extend chain a at the target block growth rate of 1 just like c. In total, a_1 and a_2 would have taken the adversary

$$\frac{1}{M_a} + \frac{1}{(1 + M_a(A-1))}$$

which clearly approaches zero for large M_a. However, the difficulty adjustment rule ensures that after two blocks the adversary will end up never catching up with c because both chain will grow at rate 1 per unit time. In a probabilistic setting, the best that the adversary can do is to reach a random walk with an initial deficit of $A-2$.

5.1.2 Optimal mining of k blocks. We now solve the adversary's problem of optimally allocating mining power if it desires to construct k blocks as quickly as possible. Later, we will relate k to the problem of overtaking the canonical chain c. Also, we ignore the fact that there is an added speed bonus from the delay $A-1$ since this is a one-time bonus that only serves to complicate the math with no additional insights.

The adversary's optimization problem is to

$$\begin{aligned} \min_{M_i} \quad & \textstyle\sum_{i=1}^{k} T_i \\ \text{s.t.} \quad & M_i \le M_a \\ & T_i = \tfrac{d_i}{M_i} \\ & d_{i+1} = \tfrac{d_i}{T_i} \\ & d_1 = 1 \end{aligned}$$

This can be restated as

$$\begin{aligned} \min_{M_i} \quad & \tfrac{1}{M_1} + \tfrac{M_1}{M_2} + \ldots + \tfrac{M_{k-1}}{M_k} \\ \text{s.t.} \quad & M_i \le M_a \end{aligned}$$

Clearly $M_k = M_a$ is optimal. Then by Arithmetic-Geometric mean inequality, the smallest value is $kM_a^{-\frac{1}{k}}$, which is attained uniquely when all terms in the summation are equal. Hence the solution is

$$M_i = M_a^{\frac{i}{k}} \tag{3}$$

The adversary initiates the attack with $M_1 = M_a^{1/k}$ and increases the mining capacity by a factor of $M_1 > 1$ for k blocks. Consequently, the adversarial chain adds a block every $1/M_1 < 1$ time units and is able to gain on the canonical chain. What we have left unanswered is the amount of mining capability M_a required to overcome the adversary's deficit of A blocks which we address next.

5.1.3 Overtaking the canonical chain. While the adversary is constructing chain a, the canonical chain c is still growing at rate 1. Therefore, the adversary must maintain this scaling up of mining capacity to construct as many blocks at it takes to *gain* A blocks on the canonical chain so that the latest block it mines is ahead of c_N by a single block.

$$k = A + \frac{k}{M_1} \tag{4}$$

Equation (4) relates number of blocks mined to conduct the attack k and the duration of the attack $\frac{k}{M_1}$ to the initial deficit of A blocks. The honest miners have a head-start of A and continue to mine $\frac{k}{M_1}$ in the time it takes for the adversary to mine k blocks. Combining Equation (4) with (3) yields

$$A = k\left(1 - \frac{1}{\sqrt[k]{M_a}}\right) \tag{5}$$

which links the three fundamental quantities of the initial deficit A to the duration of the optimal attack k and the adversary's mining capacity M_a. For a fixed A, note that the mining capacity

required grows the faster one intends to attack. As M_a approaches infinity, k approaches A. Indeed, instantaneously overcoming a deficit of A would require an infinite amount of M_a.

For example with $M_a = 16$, dedicating $M_a = \{2, 4, 8, 16\}$ would allow for a gain of 4 blocks in 2 units of time since each block takes half a unit of time. Therefore, the adversary can overtake on $A = 2$ if it possessed at least $M_a = 16$ times more mining power than the honest miners. One quickly realizes that the mining power needed to overcome larger values of A blows up the required M_a due to the k-th root since $k > A$ for finite M_a.

M_a also decreases in the number of blocks taken k. For $A = 2, k = 3; M_a = 27$. For $A = 2, k = 4; M_a = 16$ and for large n, M_a is about 7.5. For $A = 3, k = 4; M_a = 256$. M_a approaches about 20 for large n. In other words, to overtake the canonical chain from a deficit of two blocks over an arbitrarily long attack, the adversary would need at least 7.5 times more mining power than honest miners. This corresponds to an adversarial miner who controls about 90pct of the hash rate. To accomplish the same feat in 3 blocks which is the fastest possible, the adversarial miner would need to be in control of about 95pct of the mining power. In short, the difficulty adjustment protocol coupled with a protocol for validating timestamps provides vast amounts of protection against longest-chain attacks particularly those originating from a long range.

We illustrate this optimal strategy in Figure 6.

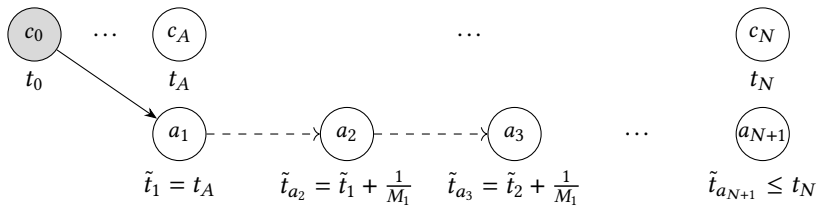

Fig. 6. The adversary initiates the attack with $M_1 = M_a^{1/N}$ and increases it by factor of M_1 each block so that $M_i = M_1^i$. Therefore, every block takes $1/M_1 < 1$ to mine. We ignore the initial $A - 1$ delay bonus since that only helps the adversary, and only for two blocks.

5.2 Unverifiable Timestamps

While verifiable timestamps are useful, they are not easily attainable without reliance on more frequent communication (e.g., gossip, clock synchronization, etc.) and subjectivity. One may desire a blockchain design with more objectivity, or perhaps, reduce the bandwidth taken up by frequent communication to obtain a network time. We now study such a scenario to shed light on these questions.

While the adversary has flexibility in reporting $\tilde{T}_i$, the following conditions must still be respected.

$$\begin{aligned} \tilde{T}_i &> 0 \\ \sum_{i=1}^{N} \tilde{T}_i &\leq N - 1 \\ \tilde{t}_{a_1} &\geq t_1 = 1 \\ \tilde{t}_{N+1} &\leq t_N = N \end{aligned}$$

The first equation is equivalent to an assertion that the time reported in a block must be ahead of the time reported in its parent. Also, a block discovery time of zero implies infinite mining power which is impossible. The second condition simply states that the reported block times must occur

in the past at the time of revelation to the honest miners which occurs when the honest miners are on block-height N. The third condition states that both chains share a common ancestor at height 0. The final condition which follows from the previous two simply states that the adversary must be a block ahead of the honest miners at the time the alternative chain is revealed.

5.2.1 Optimal reporting of $\tilde{t}_i$. It is obvious that $M_i = M_a$ irrespective of d_i. The adversary's optimization problem is to minimize the actual time taken to mine while deploying full mining power by choosing the times to report which influences d_i subject to the conditions above. Leaving the blockheight of overtaking as an unknown parameter, N, the adversary's problem is

$$\begin{aligned} \min_{\tilde{T}_i} \quad & \textstyle\sum_{i=1}^{N} T_i \\ \text{s.t.} \quad & d_{i+1} = \tfrac{d_i}{\tilde{T}_i} \\ & T_i = \tfrac{d_i}{M_a} \\ & \tilde{T}_i > 0 \\ & \textstyle\sum_{i=1}^{N} \tilde{T}_i \leq N-1 \\ & \tilde{t}_{a_1} \geq t_1 \\ & d_1 = \tfrac{d_0}{T_0} = 1 \end{aligned}$$

Rewriting, we get

$$\begin{aligned} \min_{\tilde{T}_i} \quad & \tfrac{d_1}{M_a} + \tfrac{d_1/\tilde{T}_1}{M_a} + \tfrac{d_1/(\tilde{T}_1\tilde{T}_2)}{M_a} + \cdots + \tfrac{d_1/\prod_{i=1}^{N-1}\tilde{T}_i}{M_a} \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{N} \tilde{T}_i \leq N-1 \end{aligned} \tag{6}$$

Noting that $d_1 = 1$ since $d_1 = d_0/T_0 = d_0 = 1$ and that M_a is some constant, we get

$$\begin{aligned} \min_{\tilde{T}_i} \quad & 1 + \tfrac{1}{\tilde{T}_1} + \tfrac{1}{\tilde{T}_1\tilde{T}_2} + \cdots + \tfrac{1}{\prod_{i=1}^{N-1}\tilde{T}_i} \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{N-1} \tilde{T}_i + \tilde{T}_N \leq N-1 \end{aligned}$$

The objective is to minimize the actual time spent on the attack by optimally reporting fake timestamps which affect the difficulty of the next block. For example, the time that a_1 takes is fixed at $1/M_a$ but a_2 will depend on how long a_1 was reported to have taken. Finally, the difficulty of the last block a_N will depend on all the reported times taken in the previous blocks. Since the objective is to report as quickly as possible, the adversary will claim that the last block was found approximately instantaneously even though it took the adversary $\frac{d_1/\prod_{i=1}^{N-1}\tilde{T}_i}{M_a}$ time to mine. Likewise, $\tilde{t}_1 = t_1$ because reporting any later time only makes subsequent blocks more difficult to mine.

As a result of this setup, the adversary will claim that block N was done in zero time (effectively claiming it has M_a arbitrarily large) making the difficulty for block $N+1$ infinitely difficult after the honest miners naively move to chain a. This rather ridiculous solution is optimal because our game ends and there is no additional reward from continuing the chain. To avoid this arguably absurd result, we would need to additionally specify what the honest miners believe about M_a. For instance, if the honest miners believe that M_a lies in some lower and upper bound, $M_a \in [M_l, M_u]$, the adversary would have to report timestamps that imply $M_N = M_u$. This doesn't change any of the key ideas we take away from our analysis but adds an analytical burden.

Additionally, the constraint must bind since reporting any longer time to find the first block makes all subsequent blocks easier to work on. Omitting the first term gives us.

$$\begin{aligned} \min_{\tilde{T}_i} \quad & \tfrac{1}{\tilde{T}_1} + \tfrac{1}{\tilde{T}_1\tilde{T}_2} + \cdots + \tfrac{1}{\prod_{i=1}^{N-1}\tilde{T}_i} \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{N-1} \tilde{T}_i = N - 1 = N^* \end{aligned}$$

Now let's factor out $1/\tilde{T}_1$ and replace $\tilde{T}_1 = N^* - \sum_{i=2}^{N-1} \tilde{T}_i$ to obtain

$$\min_{\tilde{T}_2,\ldots,\tilde{T}_{N-1}} \frac{1}{N^* - \sum_{i=2}^{N-1} \tilde{T}_i} \left(1 + \frac{1}{\tilde{T}_2} + \frac{1}{\tilde{T}_2\tilde{T}_3} + \cdots + \frac{1}{\prod_{i=2}^{N-1}\tilde{T}_i}\right)$$

Table 1. Time taken to mount an attack of N blocks and the corresponding maximum lead A the adversary can overcome for $M_a = 3$ and $M_a = 99$.

N	$M_a = 3$ **(75pct of total capacity)**				$M_a = 99$ **(99pct of total capacity)**			
	Verifiable Time		**Unverifiable Time**		**Verifiable Time**		**Unverifiable Time**	
	$T^*(N)$	A_{max}	$T^*(N)$	A_{max}	$T^*(N)$	A_{max}	$T^*(N)$	A_{max}
3	2.08	0.92	0.96	2.04	0.65	2.35	0.03	2.97
5	4.01	0.99	1.43	3.57	1.99	3.01	0.04	4.96
10	8.96	1.04	2.21	7.79	6.32	3.68	0.07	9.93
20	18.93	1.07	3.04	16.96	15.89	4.11	0.09	19.91
100	98.91	1.09	4.37	95.63	95.51	4.49	0.13	99.87

Clearly, each choice variable is bounded away from zero as otherwise, the objective would diverge. So there is some $\epsilon > 0$ such that $\tilde{T}_i^* > \epsilon$. Consider the same minimization problem with constraints $\tilde{T}_i \geq \epsilon$. Then we know the solution to the new problem is interior. Then the solution is given by First Order Conditions (FOCs). But the FOCs of the new problem are the same as the FOCs of the original problem. If the FOCs yield a unique solution, it is the unique minimizer of the original problem. Next, we take the FOCs of the original problem.

So, zero gradient for any $\tilde{T}_x$

$$\frac{1}{\left(N^* - \sum_{i=2}^{N-1} \tilde{T}_i\right)^2} \left(1 + \frac{1}{\tilde{T}_2} + \frac{1}{\tilde{T}_2\tilde{T}_3} + \cdots + \frac{1}{\prod_{i=2}^{N-1}\tilde{T}_i}\right) = \frac{1}{\left(N^* - \sum_{i=2}^{N-1} \tilde{T}_i\right)} \left(\frac{1}{\tilde{T}_x} \sum_{k=x}^{N-1} \frac{1}{\prod_{j=2}^{k}\tilde{T}_j}\right)$$

gives

$$\frac{1}{\left(N^* - \sum_{i=2}^{N-1} \tilde{T}_i\right)} \left(1 + \frac{1}{\tilde{T}_2} + \frac{1}{\tilde{T}_2\tilde{T}_3} + \cdots + \frac{1}{\prod_{i=2}^{N-1}\tilde{T}_i}\right) = \frac{1}{\tilde{T}_x} \sum_{k=x}^{N-1} \frac{1}{\prod_{j=2}^{k}\tilde{T}_j}$$

Note that the LHS is the objective which is a constant when evaluated at its optimal. Immediately, note as well that the higher x is, the fewer terms there are in the sum on the RHS. A lower x also

contains all the terms in summation from $x+1$ plus one more. Hence, T_x has to be decreasing in x which is of course, again, consistent with the idea that a smaller T_x increases the difficulty on the next block and must be avoided.

Next, note that we can break up the sum so that:

$$\frac{1}{\tilde{T}_x}\sum_{k=x}^{N-1}\frac{1}{\prod_{j=2}^{k}\tilde{T}_j} = \frac{1}{\tilde{T}_{x+1}}\sum_{k=x+1}^{N-1}\frac{1}{\prod_{j=2}^{k}\tilde{T}_j}$$

$$\frac{1}{\tilde{T}_x}\left(\sum_{k=x}^{x}\frac{1}{\prod_{j=2}^{k}\tilde{T}_j} + \sum_{k=x+1}^{N-1}\frac{1}{\prod_{j=2}^{k}\tilde{T}_j}\right) = \frac{1}{\tilde{T}_{x+1}}\sum_{k=x+1}^{N-1}\frac{1}{\prod_{j=2}^{k}\tilde{T}_j}$$

$$\frac{\frac{1}{\prod_{j=2}^{x}\tilde{T}_j}}{\sum_{k=x+1}^{N-1}\frac{1}{\prod_{j=2}^{k}\tilde{T}_j}} = \frac{\tilde{T}_x}{\tilde{T}_{x+1}} - 1$$

And then we can multiply the top and bottom of the LHS by $\prod_{j=2}^{x}\tilde{T}_j$ to get

$$\frac{1}{\sum_{k=x+1}^{N-1}\frac{1}{\prod_{j=x+1}^{k}\tilde{T}_j}} = \frac{\tilde{T}_x}{\tilde{T}_{x+1}} - 1$$

$$\sum_{k=x+1}^{N-1}\frac{1}{\prod_{j=x+1}^{k}\tilde{T}_j} = \frac{\tilde{T}_{x+1}}{\tilde{T}_x - \tilde{T}_{x+1}}$$

Some manipulation yields

$$\begin{aligned}\tilde{T}_x &= \frac{\tilde{T}_{x-1} - \tilde{T}_x}{\tilde{T}_x - \tilde{T}_{x+1}} \text{ for } x = 2\ldots N-2\\ \frac{\tilde{T}_{N-1}}{\tilde{T}_{N-2} - \tilde{T}_{N-1}} &= \frac{1}{\tilde{T}_{N-1}}\\ \sum_{i=1}^{N-1}\tilde{T}_i &= N^*\end{aligned} \tag{7}$$

which fully characterizes the solution.

6 HOW IMPORTANT IS TIMESTAMP VERIFIABILITY?

We now compare the adversary's optimal strategy between the two regimes. As we discussed earlier, a longest-chain attack does not make sense for adversaries who possess a small majority of mining power. We consider two adversaries, one with 75pct of the mining capacity and the other with 99pct of the mining capacity. While it may be comical to think about a conventional miner with such capabilities, this risk is a lot more tangible when we allow for quantum computing possibilities [1]. Such a risk may also be a lot more conceivable for POW blockchains where the overall hash rate is much lower than of Bitcoin's for example.

6.1 Results

We now report what values of initial deficits A an adversary can overcome if it mounts an attack where it selfishly mines $N = 3, 5, 10, 20, 100$ blocks. We also report the time taken $T^*(N) = t_{a_{N+1}} - t_{a_1}$ to mine the alternative chain. As the adversary mines N blocks in such a fashion, the honest miners would have extended chain c_A by $T^*(N)$ blocks. Therefore, the largest A the adversary could have overcome would be given by $N - T^*(N)$. Integer constraints are ignored.

6.2 Discussion: Practical Implications

What we call a block should not be taken literally. In reality, our block represents an epoch. Secondly, our assumptions on the extremes of timestamp verifiability also should not be taken literally. What matters is how flexible timestamps can be relative to epoch length. For instance, Bitcoin's timestamps can be any time in a 3-hour window and be accepted. Its epoch length is 2016 blocks which will take about 2 weeks to mine. As a ratio, the relative flexibility approaches zero. This means that Bitcoin is probably very close to our setting with verifiable time and it suggests that an adversary controlling 75pct of the mining capacity will be able to start an entire epoch behind the canonical chain and overtake it after 4 epochs have elapsed on the canonical chain. Monero and Bitcoin Cash, recalculate the block adjustment every block using the previous day's worth of blocks. The degree of time reporting flexibility is similar to Bitcoin's. As a ratio, these two blockchains would be further away from perfect time verifiability compared to Bitcoin. We do not attempt to extrapolate the effects of the moving average or comment on the various approaches in determining which reported timestamp is valid. Our analysis is deliberately kept very simple so that the forces at work are transparently characterized.

It is also important to note that our results are silent on how vulnerable blockchains are to the best possible strategy an adversary can mount. We are only commenting on this particular strategy which is the selfish mining of an alternative longest chain.

6.2.1 Takeaway 1: Verifiable timestamps diminish the efficacy of M_a. Observe that with both cases for M_a, the time it takes to construct more blocks increases with the number of blocks constructed. Since the canonical chain is growing at the same time, the adversary will need to control huge amounts of mining power in order to overcome small leads. In other words, an adversary will find it very difficult to start an alternative chain that is more than a few blocks behind the leading block and overtake it. The crucial insight here is that the *best* strategy the adversary can employ is to scale up its mining efforts following a power law and power law progressions ramp up very quickly. It also sets limits on how far ahead the target chain can be for an adversary with capacity M_a. For example, an adversary with $M_a = 3$ cannot attempt the longest-chain attack starting from 2 blocks behind the canonical chain. It will **never** catch up no matter how long it mines.

6.2.2 Takeaway 2: Unverifiable timestamps lead to approximately linear attack duration. With unverifiable timestamps, the time taken to construct N blocks is approximately linear in N under our assumption that $\tilde{t}_{a_1} = t_{c_1}$. This implies that an adversary possessing mining power greater than 51pct can and will catch up any distance A provided it continues selfishly mining for long enough.

6.2.3 Future Work. We solved for the optimal attack an adversary can mount against naive honest miners assuming a very limited action set for the adversary. For instance, the adversary is not allowed to mine on the main chain. If that action was allowed, it is very easy to show that the adversary can improve on its desired outcome by mining on the main chain whenever its difficulty is low, and, leaving it to mine on its own chain whenever the difficulty increases. This is commonly known as chain hopping. Quantifying this optimal action depending on time verifiability is an immediate extension of this paper.

REFERENCES

[1] Dan A. Bard, Joseph J. Kearney, and Carlos A. Pérez-Delgado. Quantum advantage on proof of work. *Array*, 15:100225, 2021.

[2] Alex Boverman. Timejacking & bitcoin http://culubas.blogspot.com/. 2011.

[3] Michael Davidson and Tyler Diamond. On the profitability of selfish mining against multiple difficulty adjustment algorithms. *IACR Cryptol. ePrint Arch.*, page 94, 2020.

[4] Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, 2018.

[5] Amos Fiat, Anna Karlin, Elias Koutsoupias, and Christos H. Papadimitriou. Energy equilibria in proof-of-work mining. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 489–502. ACM, 2019.

[6] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.

[7] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 291–323. Springer, 2017.

[8] Guy Goren and Alexander Spiegelman. Mind the mining. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 475–487. ACM, 2019.

[9] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers*, volume 8438 of *Lecture Notes in Computer Science*, pages 72–86. Springer, 2014.

[10] Daniel Kraft. Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Netw. Appl.*, 9(2):397–413, 2016.

[11] Dmitry Meshkov, Alexander Chepurnoy, and Marc Jansen. Short paper: Revisiting difficulty control for blockchain systems. In Joaquín García-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, volume 10436 of *Lecture Notes in Computer Science*, pages 429–436. Springer, 2017.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[13] Kevin Alarcón Negy, Peter R. Rizun, and Emin Gün Sirer. Selfish mining re-examined. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 61–78. Springer, 2020.

[14] Shunya Noda, Kyohei Okumura, and Yoshinori Hashimoto. An economic analysis of difficulty adjustment algorithms in proof-of-work blockchain systems. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, page 611. ACM, 2020.

[15] Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle maker: (time)stamping out the competition in ethereum. *IACR Cryptol. ePrint Arch.*, page 1020, 2022.

[16] Aviv Yaish, Saar Tochner, and Aviv Zohar. Blockchain stretching & squeezing: Manipulating time for your best interest. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 65–88. ACM, 2022.