On the (in)security of Proofs-of-Space based Longest-Chain Blockchains

Mirza Ahad Baig[1] and Krzysztof Pietrzak[1]

[1]ISTA, Austria

Abstract. The Nakamoto consensus protocol underlying the Bitcoin blockchain uses proof of work as a voting mechanism. Honest miners who contribute hashing power towards securing the chain try to extend the longest chain they are aware of. Despite its simplicity, Nakamoto consensus achieves meaningful security guarantees assuming that at any point in time, a majority of the hashing power is controlled by honest parties. This also holds under "resource variability", i.e., if the total hashing power varies greatly over time.

Proofs of space (PoSpace) have been suggested as a more sustainable replacement for proofs of work. Unfortunately, no construction of a "longest-chain" blockchain based on PoSpace, that is secure under dynamic availability, is known. In this work, we prove that without additional assumptions no such protocol exists. We exactly quantify this impossibility result by proving a bound on the length of the fork required for double spending as a function of the adversarial capabilities. This bound holds for any chain selection rule, and we also show a chain selection rule (albeit a very strange one) that almost matches this bound.

Concretely, we consider a security game in which the honest parties at any point control $\phi > 1$ times more space than the adversary. The adversary can change the honest space by a factor $1 \pm \varepsilon$ with every block (dynamic availability), and "replotting" the space (which allows answering two challenges using the same space) takes as much time as ρ blocks. We prove that no matter what chain selection rule is used, in this game the adversary can create a fork of length $\phi^2 \cdot \rho/\varepsilon$ that will be picked as the winner by the chain selection rule.

We also provide an upper bound that matches the lower bound up to a factor ϕ. There exists a chain selection rule (albeit a very strange one) which in the above game requires forks of length at least $\phi \cdot \rho/\varepsilon$.

Our results show the necessity of additional assumptions to create a secure PoSpace based longest-chain blockchain. The Chia network in addition to PoSpace uses a verifiable delay function. Our bounds show that an additional primitive like that is necessary.

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/F85

1 Introduction

Bitcoin was the first successful digital currency. What set it apart from previous attempts like Digicash [6] was the fact that it is permissionless. This means it is decentralized – so no single entity can shut it down or censor transactions – and moreover, everyone can participate in maintaining and securing the currency.

The key innovation in Bitcoin is the blockchain which realizes a "decentralized ledger". In the case of a digital currency, this ledger simply records all the transactions, but it can also hold richer data like smart contracts [12].

A blockchain is a hash-chain $b_0 \hookleftarrow b_1 \hookleftarrow b_2 \ldots \hookleftarrow b_j$ where each b_i is a data block that contains a "payload" (transactions, a time stamp, etc.), a hash $h_i = H(b_{i-1})$ of the previous block, and a "proof of work" (PoW) π_i.

The collision-resistance of the hash function H ensures that a block b_i commits all the previous blocks. The main novelty is the way proofs of work are used to make it computationally costly to add a block: To create a valid block b_i one must find a value π_i such that the hash of the previous block and π_i is below some threshold

$$0.H(b_{i-1}, \pi_i) < 1/D$$

here we think of the hash as a binary string: if the difficulty D is 2^k, then the hash $H(b_{i-1}, \pi_i)$ must start with at least k 0's. Parties called miners compete to find PoWs to extend the latest block. They are incentivized by rewards (block rewards and transaction fees) to contribute computing power towards this task. Bitcoin is permissionless in the sense that everyone can be a miner and the protocol does not need to know who currently participates [10]. Bitcoin can be shown to be secure (in particular, it does not allow for double spending), assuming that a majority of the hashing power is controlled by honest parties who follow the protocol rules. The most important rule just states that a miner should always work towards extending the heaviest valid chain (typically, the heaviest chain is also the longest one, hence the name "longest chain") they are aware of. Blockchains following this general rule are called "longest-chain blockchains", the protocol itself is referred to as "Nakamoto consensus".

Alternative Proof Systems. Nakamoto consensus uses computation as a resource so that a miner who holds an α fraction of the total resource will contribute an α fraction of all blocks in expectation, and thus get roughly an α fraction of the rewards.

Using computation as a resource has several negative implications. The main one is the ecological impact: currently Bitcoin mining is burning roughly as much energy as the Netherlands. It is thus natural to look for a more "sustainable" resource that could replace hashing power in a longest-chain blockchain.

The most investigated alternative are *proofs of stake* (PoStake), where the coins as recorded on the blockchain serve as a resource. More precisely, miners can stake their coins, which takes them temporarily out of circulation. They can then participate in the mining process, getting a fraction of the rewards which is proportional to the fraction of their stakes coins.

PoStake is extremely appealing as it is basically wasteless as it is not a "physical" resource, but it raises many technical questions and conceptual issues. One argument that is often raised is that PoStake is not really permissionless as the only way to participate in mining lies in acquiring coins from a limited supply in the first place.

In [8] *proofs of space* (PoSpace) were introduced. A PoSpace is a proof system where a prover convinces a verifier that it "wastes" disk space. The motivation for this notion was a replacement for proofs of work which is still a "physical resource ", and thus does not share many of the shortcomings of PoStake, but is also much more sustainable than proofs of work.

Proofs of Space. A proof of space [8] is an interactive protocol between a prover P and a verifier V. The main protocol parameter is a value N determining the disk space of an (honest) prover (a typical value would be $N = 2^{43}$ bits, which corresponds to one TB). In an initialization phase, which is executed once, the honest prover initializes his disk space space of size N with a file S, called a "plot". This phase should be very efficient for the verifier (or not involve the verifier at all [2]), while the prover should run in time $\tilde{O}(N)$. This is basically optimal as they must run in time N to just "touch" the entire disk space.

After the initialization phase, the prover can create valid proofs for random challenges very efficiently, in particular, only accessing a tiny portion of its local file S. The security property of a PoSpace states that any prover who instead of S stores some data S' of some size that is "sufficiently" smaller than S, will fail to "efficiently" create a proof for a random challenge with "significant" probability.

We will not discuss what exactly "sufficiently" and "significant" means here. Let us mention that for the application to longest-chain blockchains it is sufficient that for any $0 < \alpha < 1$, a prover storing $\alpha \cdot N$ bits will fail on a $1 - \alpha$ fraction of the challenges.

Concerning the "efficiently" in the statement above, note that a malicious prover can always create a valid proof even when storing almost nothing by simply running the initialization procedure after getting the challenge to create the plot S, and then computing the proof using the honest algorithm. Thus the best we can hope for is that a malicious prover needs $\tilde{\Omega}(N)$ computational work (i.e., the cost of computing the plot) when only storing a sufficiently compressed plot S'.

The observation above also implies that a prover with N space can "pretend" to have $k \cdot N$ space by creating k different plots sequentially using $k \cdot \tilde{O}(N)$ work. Note that when attacking a blockchain, one would need to do the replotting afresh for every block, as the challenge for a block is only known once the previous block is computed. Thus, creating a proof using such a *replotting attack* is extremely expensive compared to creating proofs honestly, and this attack is presumably not an issue when blocks arrive sufficiently frequently. *The results of this paper show that this intuition is wrong.*

Longest-Chain Blockchains from Proofs of Space. To construct a longest-chain blockchain from PoSpace we can use Nakamoto consensus, but replace the PoW with PoSpace. There are various challenges one must address which we outline below.

Interactive Resource Initialization: In Bitcoin, a miner with some mining hardware can start participating in mining at any time. For PoSpace this is in general not the case as there's an initialization phase. The earliest PoSpace longest-chain proposal (which remained purely academic) is Spacemint [11], which uses the pebbling-based PoSpace from [8]. This PoSpace has an interactive initialization phase after which the verifier holds a type of commitment to the plot created by the prover. In Spacemint the chain plays the role of the verifier, and the commitment must be uploaded by a miner to the chain as a special transaction before they can start mining. The function-inversion-based PoSpace from [2] has a non-interactive initialization, i.e., the verifier is not involved at all, and thus it can be used like a PoW in Bitcoin. This PoSpace is used in the Chia network [7] blockchain.

Bock-Arrival Times: In Bitcoin, the arrival time of blocks can be controlled by setting the difficulty. Unlike PoW, PoSpace (also PoStake) are efficient proof systems, where once the resource (a plot or staked coins) is available, creating a proof is cheap and fast, so we need another mechanism. The easiest approach is to simply assume all parties have clocks and specify that blocks are supposed to arrive in specific time intervals, say once every minute. This is the approach taken by Spacemint [11] or Ouroboros [9], while in the Chia network [7] verifiable-delay functions (VDFs) are used to enforce some clock-time between the creation of blocks.

Costless Simulation/Grinding: The key difference between PoW and "efficient" proof systems like PoSpace or PoStake is the fact that producing proofs for $k > 1$ different challenges require k times as much of the resource in PoW, but it makes hardly a difference for PoSpace or Postake, as producing a proof is extremely cheap compared to acquiring the resource in the first place. This "costless simulation" property creates various issues in the blockchain setting. One such issue is grinding attacks. Consider a setting where an adversary can influence the challenge, a typical example is a blockchain like Bitcoin where the challenge for the next block depends on the current block, and the miner that creates the current block can e.g. choose which transactions to add. Such an attacker can "grind" through many different blocks until they find one that gives a challenge they like (say because with this challenge they can also win the next block).

A canonical countermeasure against grinding first proposed in Spacemint [11] is to "split" the chain in two. One chain only holds canonical values like proofs and is used for creating challenges, while another chain holds all the "grindable" values (transactions, time-stamps, etc.).

Costless Simulation/Double-Dipping: Even once grinding is no longer an issue, with costless simulation an adversary can still cheaply try to extend many of the past blocks, this way growing a tree rather than a chain. This strategy has been proven to virtually increase the adversarial resource by a factor of $e = 2.72$ [7]. An elegant countermeasure against this attack was proposed in [4], the basic idea is to only use the kth block for computing challenges, this "correlated randomness" technique decreases the advantage as k increases.

Costless Simulation/Bootstrapping: An adversary having some resource (space or stake) N can immediately create a long chain. Typically one would make up the time-stamps for this chain so it looks like a legit chain that has been created over a long period. In context of PoStake this is a well-known problem, while Spacemint [11] was the first instance this appeared in literature for PoSpace.

Replotting: In PoSpace a prover who has space of size N and gets a challenge c can pretend to have much more space by re-initializing the same space k times with different identifiers, this way creating k proofs pretending to have $k \cdot N$ bits of space. As replotting is fairly expensive, in a context like Blockchains, where challenges arrive frequently, it seems impossible (or at least not rational) to continuously replot. In this paper we show that this intuition is flawed; replotting attacks, in combination with bootstrapping, are used in our lower bound showing no PoSpace longest-chain blockchain is secure under resource variability. This was identified as a problem in [11].

1.1 Resource Variability

In this work, we prove that no PoSpace based longest-chain blockchain can be secure. Resource variability means that the amount of the resource dedicated to mining changes over time.

Bitcoin can be shown to be secure under resource variability as long as the honest parties hold more hashing power than a potential adversary at any point in time.

With PoStake the situation is more interesting. A PoStake based longest-chain protocol using the Bitcoin chain-selection rule where one picks the "heaviest" chain is not secure due to bootstrapping attacks.[1] On the positive side, the paper on Ouroboros Genesis [3] shows that a completely different chain selection rule does imply security even for PoStake based chains (with some additional assumptions, like assuming honest parties delete old keys). Their chain selection rule basically says that given two chains A and B one only looks at the weight of a short subchain starting at the point where A and B fork, and picks the chain whose subchain is heavier.

For PoSpace based chains the genesis chain selection rule is *not* secure, in fact, unlike the heaviest chain rule, the genesis rule is insecure even without

[1] More precisely, assume there's a point in time where a very high amount of coins is staked, say at the ith block the honest parties staked c_i^h coins, while the adversary $\mathcal{A}$ controls a $1/\phi < 1$ fraction of that, i.e., $c_i^a = c_i^h/\phi$. At this point, $\mathcal{A}$ bootstraps a private fork $b_i \hookleftarrow b_{i+1} \ldots \hookleftarrow b_{i+T}$ of some length T, each block having weight c_i^a. If for the next T blocks the amount staked by the honest parties is (on average) sufficiently smaller than c_i^a, the chain created by the honest parties will look lighter than the private fork of $\mathcal{A}$ at block $i+T$, and at this point $\mathcal{A}$ can release his private fork which will be adapted by all honest parties.

resource variability (i.e. when we assume the space of the honest and adversarial parties is static). There is a simple attack exploiting bootstrapping and *replotting*, which we'll sketch below. Informally, the reason this attack does not apply in the PoStake or PoW settings is that there's no analog of replotting in PoStake, while in the PoW setting, we do not have bootstrapping.

1.2 Modelling a Longest-Chain PoSpace Blockchain

To model a PoSpace based longest-chain blockchain we will make a few idealizing assumptions. As our main result is a lower bound, this only makes our result stronger, concretely

Resource: We assume the chain grows by exactly one fresh block per time unit, and each block exactly reflects the amount of space that was used. In reality, a blockchain like Chia or Bitcoin (in the PoW setting) only approximately reflects this amount. One can get a very good approximation of the space used by looking at a sufficiently long subsequence (this idea is used when recalibrating the difficulty). Alternatively one could consider a blockchain design where each block contains the best k proofs for some $k > 1$. The larger k, the lower the variance and thus the better the approximation. With "block" we do not necessarily model a single block, but rather the appearance of a fresh challenge, and this challenge can be used for multiple blocks (e.g. in Chia we have a fresh challenge every 10 minutes, but as Chia uses the correlated randomness technique to prevent double dipping, this challenge is used for up to 64 actual blocks).

Attacks: While we model bootstrapping and replotting, we assume there is no grinding or double dipping. This is justified as we have techniques to mostly prevent griding and double dipping, while there's no simple way to prevent replotting, and to prevent bootstrapping we need additional primitives like VDFs.

Resource Variability: The adversary can control the change in resource, but is restricted to change it only within some $1+\varepsilon$ factor with each block, where $\varepsilon > 0$. The quantitative lower bound and the matching upper bound depend crucially on this parameter.

1.3 Approach for Lower Bound

To prove our lower bound we let an adversary specify two possible forks, A and B of a chain by specifying how the space of the honest parties changes over time in both cases. Now assume we show that (for given ranges of parameters) by exploiting bootstrapping and replotting it is possible to create such forks where B can be "faked" using a fraction (say half) of the space the honest parties had in A, and vice versa, i.e., A can be faked using half the space of B.

This means that an adversary in a hypothetical world where A is the honest chain could fake chain B and vice versa, thus, no matter which chain selection rule is used, in one of the two worlds the adversary's fork will succeed (say the chain rule prefers A over B, then in a "world" where B is the correct profile, the adversary can create a fork A which will win over B).

For our upper bound, we show that a particular chain selection rule is secure almost up to the parameters for which our lower bound applies.

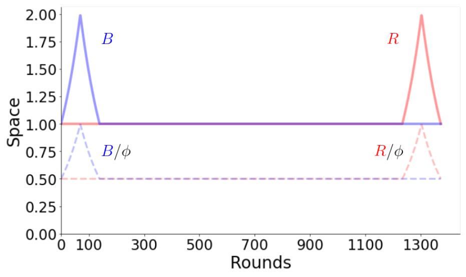

Fig. 1: Two profiles as used in our lower bound for $\varepsilon = 0.01, \phi = 2$ and $\rho = 4$.

1.4 Proof Sketch for Lower Bound

To prove our lower bound, we specify two profiles R and B reflecting the space honest users have, and then outline a strategy for how (by using bootstrapping and replotting) one can create profile R using a profile B/ϕ, where only a $1/\phi$ fraction of the space in B is available (i.e., the space the adversary has in world A) and vice versa. Moreover, in the profiles B and R the space is only allowed to change by a $1+\varepsilon$ per block. We will sketch the main idea using the profiles in Fig. 1. The profile R is plotted by the light red solid line and profile B is plotted by the light blue solid line. The solid lines show how much space honest parties control and the dashed lines of the respective colors show how much adversary controls for the respective solid space profile. The honest parties start at space 1, then we let the profile B increase to ϕ as fast as allowed (i.e., by a factor $1+\varepsilon$ per step), then we go back to 1 as fast as possible, and then there is a long flat part (the length will depend on our parameters). Profile R is the mirrored version of B.

Let us now sketch how the R (shown by light red solid line in Fig. 1) profile can be faked using a $1/\phi$ fraction of B (shown by solid light blue in Fig. 1). This is illustrated in the figures in Figs. 2(a) and 2(b). The adversary does nothing until block 70 when its space B/ϕ reaches its maximum.

At this point $B/\phi \geq 1$ and the adversary can bootstrap the flat part of R for 1233 steps as shown in the top left graph of Fig. 2. At this point, the adversary only needs to fake the "tent" in the last 140 steps of the R profile. For this, it uses replotting.

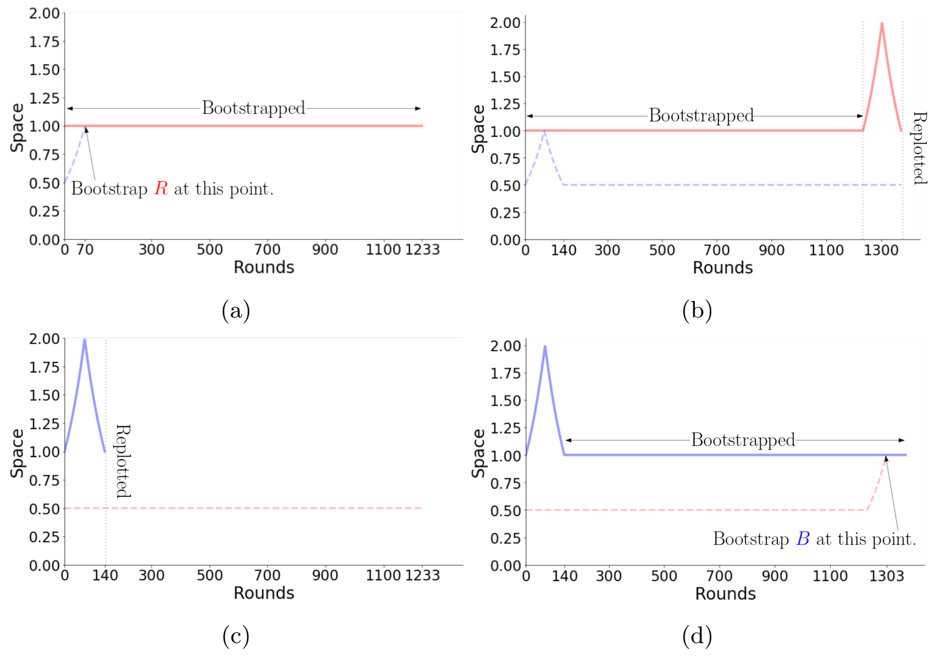

Fig. 2: The figs (a), (b) outline how the red profile from Fig 1 is faked using the blue profile for parameters $\varepsilon = 0.01, \phi = 2$ and $\rho = 4$. In the first step, we use bootstrapping to create the flat part of the solid red profile (once the solid blue profile reaches the "peak", a $1/\phi$ fraction of the solid blue profile is as high as the flat part of the solid red profile, and thus it can be bootstrapped). Then we use replotting to create the "tent" of the solid red profile (as $\rho = 4$, i.e., replotting takes four steps, it is sufficient that the remaining area below the blue profile is as large as the area under the red "tent". The two figures (b), and (c) illustrate how the blue profile is faked using the red one.

2 Model

2.1 Basic Notation for Chains

In the abstract model of the chain, we assume that time progresses in discrete steps $t_0, t_1, t_2, \ldots$. During the ith step $(t_{i-1}, t_i]$ the honest parties add a (super)block b_i. We'll always indicate the position of a block in the chain with a subscript like here.

The space available to the honest parties and the adversary at time i is denoted with $s_i^{\mathcal{H}}$ and $s_i^{\mathcal{A}}$, respectively. We assume that each block b_i perfectly reflects the amount of space that was used to create it, which is denoted by $s(b_i)$ (for the genesis block b_0 we set $s(b_0) = 1$).

We denote with $b_i \hookleftarrow b_{i+1}$ that b_{i+1} is extending (i.e., contains a hash of) block b_i. For a chain

$$\mathcal{C}_0^\ell = b_0 \hookleftarrow \ldots \hookleftarrow b_\ell,$$

we denote with $\mathcal{S}(\mathcal{C}_0^\ell)$ the *space-profile* of chain $\mathcal{C}_0^\ell$ which is defined as $(s(b_i))_{i=0}^\ell$, the sequence of space used to create each block b_i. We'll often use the notation $\mathcal{C}_i^j = b_i \hookleftarrow \ldots \hookleftarrow b_j$ to denote subchains.

Glossary:

$s_i^{\mathcal{H}} \in \mathbb{R}_{>0}$: The space available to the honest parties at step i.
$s_i^{\mathcal{A}} \in \mathbb{R}_{>0}$: The space available to the adversary $\mathcal{A}$ at step i.
$\phi > 1$: The honest/adversarial space ratio $\forall i : s_i^{\mathcal{H}}/s_i^{\mathcal{A}} = \phi$
$\varepsilon > 0$: The rate of change of the space $\forall i : s_i^{\mathcal{H}} \cdot \frac{1}{1+\varepsilon} \leq s_{i+1}^{\mathcal{H}} \leq s_i^{\mathcal{H}} \cdot (1+\varepsilon)$
$\rho \in \mathbb{N}_{\geq 2}^{+}$: Number of steps required for replotting

2.2 Chain Selection Rules

A chain selection rule takes as input two chains of the same length and outputs bit indicating the "winner"

$$\Lambda \colon \mathcal{C} \times \mathcal{C} \to \{0, 1\}$$

Consider two chains $\mathcal{C}_0^\ell, \tilde{\mathcal{C}}_0^\ell$

$$\mathcal{C}_0^\ell = b_0 \hookleftarrow \ldots \hookleftarrow b_\ell \qquad \tilde{\mathcal{C}}_0^\ell = \tilde{b}_0 \hookleftarrow \ldots \hookleftarrow \tilde{b}_\ell$$

The Bitcoin chain selection rule simply picks the chain of higher weight, adapted to our space setting this "highest weight rule" Λ_w is

$$\Lambda_w(\mathcal{C}_0^\ell, \tilde{\mathcal{C}}_0^{\ell'}) = 0 \iff \sum_{i=0}^{\ell} s(b_i) > \sum_{i=0}^{\ell'} s(\tilde{b}_i)$$

Note that the two chains may not always be of the same length but for our lower bound result we can assume without loss of generality that they are equal.

Most proposed and deployed longest-chain blockchains use a highest weight rule like this, in some cases augmented with checkpointing or finality gadgets that prevent miners from replacing their current chain with another chain that forks too far in the past even if it has a higher weight. An interesting exception is the rule suggested in Ouroboros Genesis [3] which, for some parameter $k \in \mathbb{N}$, chooses the winning chain based only on the weight of the k blocks following the forking point of the two chains. We'll denote the forking point (i.e., the index of the first blocks that differs) by λ as it looks like a forking chain, let

$$\lambda \overset{\mathsf{def}}{=} \min\{i \ : \ b_i \neq \tilde{b}_i\}$$

The genesis chain selection rule, adapted to our setting, can now be defined as

$$\Lambda^k_{\text{genesis}}(\mathcal{C}^j_0, \tilde{\mathcal{C}}^j_0) = 0 \iff \sum_{i=\lambda+1}^{\min(j,\lambda+k)} s(b_i) > \sum_{i=\lambda+1}^{\min(j,\lambda+k)} s(\tilde{b}_i)$$

The Ouroboros genesis rule was introduced as a proof-of-stake based longest-chain blockchain that is secure under resource variability. In Lemma 2 we observe that for a proof-of-space based chain, this rule is not secure, and our impossibility result from Theorem 1 shows that in fact, no secure rule exists.

2.3 Adversarial Options

We consider an experiment where honest farmers create a chain (the "honest chain") following the rules, while an adversary tries to create a private fork that can at some point be released and will be chosen as the winner over the honest chain by the chain selection rule.

Before specifying the game let us describe the option the adversary has in this game. Concerning the honest chain, if the adversary doesn't contribute at all, the honest parties at the end of the jth step (i.e., time t_j) will have created and agreed on a chain

$$b_0 \hookleftarrow b_1 \hookleftarrow \dots b_j \text{ where } \forall i \in [j] \ : \ s(b_i) = s^{\mathcal{H}}_i$$

The adversary could contribute to the honest chain, which would create a chain where

$$b_0 \hookleftarrow b_1 \hookleftarrow \dots b_j \text{ where } \forall i \in [j] \ : \ s^{\mathcal{H}}_i \le s(b_i) \le s^{\mathcal{H}}_i + s^{\mathcal{A}}_i$$

Intuitively, for the chain selection rule, there shouldn't be any advantage for an adversary to contribute to the honest mining as it should only make the honest chain better, and this will be the case in our attack proving the lower bound.

While the honest parties will always create the ith block in the chain at time i, the adversary $\mathcal{A}$ who creates his private fork must not adhere to this, his only constraint is that his fork must have the same length as the honest chain when it is released.

In a PoSpace based chain, there are two ways in which $\mathcal{A}$ can exploit this, bootstrapping and replotting, outlined below.

Bootstrapping. If at time i $\mathcal{A}$ knows of a (prefix of a) chain $\mathcal{C}^j_0 = b_0 \hookleftarrow \dots \hookleftarrow b_j$, they can extend it immediately to a longer chain. The only constraint is that the space profile of the new blocks is at most the space available to $\mathcal{A}$, i.e., the new chain satisfies

$$\mathcal{C}^j_0 \hookleftarrow b_{j+1} \hookleftarrow \dots \hookleftarrow b_k \text{ where } \forall i > j \ : \ s(b_i) \le s^{\mathcal{A}}_i$$

Replotting. The replotting parameter $\rho \in \mathbb{N}^+$ specifies how many time steps it takes $\mathcal{A}$ to replot its space. By replotting the space k times – which requires $k \cdot \rho$ steps – they can create a superblock that looks as if they had $k+1$ times the space they actually hold. Formally, using replotting, at time i, they can start extending a chain $\mathcal{C}_0^j$ with an extra block $\mathcal{C}_0^j \hookleftarrow b_{j+1}$ where $s(b_{j+1}) \leq k \cdot s_i^{\mathcal{A}}$, and this will be done by time $i + k \cdot \rho$.[2]

2.4 The Forking Game

We now define the game in which an adversary $\mathcal{A}$ forks the honest chain with the goal of fooling the chain selection rule Λ to accept their fork as the winner.

Remark 1 (Probabilitsic vs. Deterministic). When analyzing actual blockchains there's always some probabilistic argument, e.g., in Bitcoin the probability that an adversary controlling $X\%$ (for $X < 50$) of the hashing power can double spend decreases exponentially with the confirmation time, but it is technically never 0. Our "game" on the other hand is completely deterministic (for given parameters and fork length $\mathcal{A}$ can win with probability 0 or 1) because we assume that each block perfectly reflects the amount of the resource (i.e., space) that was available to create it. While our analysis can be adapted to a probabilistic setting, we don't do so as it does not lead to any more interesting insight.

The game is parameterized by $\varepsilon > 1$, controlling how fast the amount of honest space changes; the changes are controlled by the adversary but allowed to change only by a factor $(1 + \varepsilon)$ per step. [3] The parameter $\phi > 1$ controls the amount of honest vs. adversarial space while $\rho \in \mathbb{N}^+$ is the number of steps required for replotting.

The $(\phi, \varepsilon, \rho, \Lambda)$-game is defined as follows:

[2] In an actual chain, the parameter $\rho = T_{replot}/T_{block}$ is defined by T_{replot}, the clock time required to replot, divided by T_{block}, the clock time in-between challenges (which is simply the block arrival time if the challenge for a block depends on the previous block like in Bitcoin). In practice, T_{block} should be large enough for a message to spread across the network, which is a few seconds. How large T_{replot} is, depends on many things, most importantly, on the type of PoSpace used. In the PoSpace based on function inversion [2] initialization is parallelizable, and thus T_{replot} can be in the order of seconds if the attacker has enough compute power (in particular, GPUs). As a consequence, ρ can only be assumed to be a small constant. In pebbling-based PoSpace [8], initialization is inherently sequential, and T_{replot} (and thus ρ) is much larger.

[3] Assuming that an adversary can precisely control the change of honest space is a strong assumption. But for our lower bound (i.e., an attack for any chain selection rule), arguably natural space profiles suffice. In particular, our attack works for any profile where the honest space stays below some bound s for a longer period of time, with the exception of a "peak" of height $\phi \cdot s$ in the middle (the more "narrow" this peak is, the shorter the overall period where the profile is below s can be). To break particular chain selection rules, even less demanding profiles are enough, e.g. for the rule used in Spacemint, it's sufficient that the profile at some point in time starts to decrease sufficiently much.

1. – The round counter is set to $i := 0$ and the "replotting lock" $\mathit{lock} := 0$.
   – The honest and adversarial chains are initialized with the genesis block $\mathcal{C}_0^0 = \tilde{\mathcal{C}}_0^0 = b_0$ (where w.l.o.g. $s(b_0) = 1$).
   – $\mathcal{A}$ chooses its initial space $s_0^{\mathcal{A}}$ and we set the honest space to $s_0^{\mathcal{H}} := s_0^{\mathcal{A}} \cdot \phi$.
2. In each round
   – Increase the round counter $i := i + 1$.
   – $\mathcal{A}$ chooses the space adjustment γ_i in the range $\frac{1}{(1+\varepsilon)} \leq \gamma_i \leq (1+\varepsilon)$ and the space is set to

$$s_i^{\mathcal{A}} := s_{i-1}^{\mathcal{A}} \cdot \gamma_i \quad , \quad s_i^{\mathcal{H}} := s_i^{\mathcal{A}} \cdot \phi$$

   – The honest chain is extended

$$\mathcal{C}_0^i := \mathcal{C}_0^{i-1} \hookleftarrow b_i$$

   with a block with space profile $s(b_i) = s_i^{\mathcal{H}}$
   – If $\mathit{lock} > 0$ (i.e., replotting is going on) set $\mathit{lock} := \mathit{lock} - 1$, otherwise $\mathcal{A}$ can extend its current chain $\tilde{\mathcal{C}}_0^j$ in two ways
   bootstrap: Extend $\tilde{\mathcal{C}}_0^j$ to

$$\tilde{\mathcal{C}}_0^{j'} = \mathcal{C}_0^j \hookleftarrow \tilde{b}_{j+1} \hookleftarrow \ldots \hookleftarrow \tilde{b}_{j'}$$

   where $\forall t, j+1 \leq t \leq j' \ : \ s(\tilde{b}_t) \leq s_i^{\mathcal{A}}$.
   replot: $\mathcal{A}$ can call a replot request by which the last block $\tilde{b}_j$ is replaced with a block $\tilde{b}'_j$ with space profile

$$s(\tilde{b}'_j) \leq s(\tilde{b}_j) + s_i^{\mathcal{A}}$$

   Set the replotting lock to $\mathit{lock} := \rho - 1$.
   – if $\mathit{lock} = 0$ (i.e., no replotting going on) and the length j of the adversarial chain $\tilde{B}_0^j$ is at least $j \geq i$, then $\mathcal{A}$ can stop the game.

If the chain selection rule prefers the current (i block prefix of the) adversarial chain to the honest one, i.e.,

$$\Lambda(\mathcal{C}_0^i, \tilde{\mathcal{C}}_0^i) = 1$$

then we say the game is ℓ-winning for the $\mathcal{A}$, where ℓ denotes the length of the fork (i.e., length of chain minus the length of the common prefix)

$$\ell = i - \max\{k \ : \ b_k = \tilde{b}_k\}.$$

2.5 Forking Existing Rules

In this section, we'll observe that the forking game can be won against the highest weight Λ_w and the genesis $\Lambda_{\text{genesis}}^k$ chain selection rules that we discussed in Section 2.2. To break Λ_w one only requires bootstrapping (but no replotting) and resource variability, i.e. a $\varepsilon > 0$. For $\Lambda_{\text{genesis}}^k$ the ϕ can be 1.

Lemma 1. *The $(\phi, \varepsilon, \rho, \Lambda_w)$-game can be ℓ-won for $\ell = \left\lceil \frac{\phi}{\varepsilon} \right\rceil$*

Proof. To win $(\phi, \varepsilon, \rho, \Lambda_w)$-game, $\mathcal{A}$ simply bootstraps a long chain and then reduces the amount of space dynamically in order to make the honest chain have weight less than the adversarial chain. $\mathcal{A}$ never contributes to the honest chain.

Concretely, in each step $i > 0$ the adversary decreases the space by the maximum allowed amount $s^{\mathcal{H}}_{i+1} = \frac{1}{(1+\varepsilon)} \cdot s^{\mathcal{H}}_i$. At $i = 1$ $\mathcal{A}$ bootstraps

$$\tilde{\mathcal{C}}^j_0 = \mathcal{C}^0_0 \hookleftarrow \tilde{b}_1 \hookleftarrow \tilde{b}_2 \hookleftarrow \cdots \hookleftarrow \tilde{b}_j$$

where $s(\tilde{b}_t) = s^{\mathcal{H}}_0/\phi = 1/\phi$ for all $t \in [1, j]$. After this adversary simply lets $\mathcal{C}_0$ catch up. The game ends on round j. This gives,

$$\textit{Weight of } \mathcal{C}^j_0 = \sum_{t=0}^{j} \left(\frac{1}{(1+\varepsilon)} \right)^t = \frac{1 - \frac{1}{(1+\varepsilon)^{j+1}}}{1 - \frac{1}{(1+\varepsilon)}} < \frac{1}{1 - \frac{1}{(1+\varepsilon)}}$$

while

$$\textit{Weight of } \tilde{\mathcal{C}}^j_0 = 1 + \frac{j}{\phi}$$

Thus when $j \geq \left\lceil \frac{1}{(1+\varepsilon)-1} \cdot \phi \right\rceil$ the weight of the adversarial chain is higher than the weight of the honest chain. Hence $\Lambda_w(\mathcal{C}^j_0, \tilde{\mathcal{C}}^j_0) = 1$. □

Lemma 2. *The $(\phi, \varepsilon, \rho, \Lambda^k_{\text{genesis}})$-game can be ℓ-won for $\ell = \lceil \phi \rceil \cdot k \cdot \rho$*

Proof. To win $(\phi, \varepsilon, \rho, \Lambda^k_{\text{genesis}})$-game, $\mathcal{A}$ simply replots many times to get an adversarial chain which in its k blocks after the fork has enough weight to be larger than the weight of the honest chain in the corresponding k blocks.

Concretely, the game runs till $l = \lceil \phi \rceil \cdot k \cdot \rho$. Throughout the game, $\mathcal{A}$ does not use the resource variability; $s^{\mathcal{H}}_i, s^{\mathcal{A}}_i$ remain constant at 1, $\frac{1}{\phi}$ respectively. Further, $\mathcal{A}$ doesn't contribute to the honest chain. This produces

$$\mathcal{C}^j_0 = b_0 \hookleftarrow b_1 \hookleftarrow \cdots \hookleftarrow b_l$$

such that $s(b_i) = 1 \forall i \in [0, l]$.

$\mathcal{A}$ does the following:

1. On round $i = 1$, it forks the chain to create
$$\widetilde{\mathcal{C}}^1_0 = b_0 \hookleftarrow \widetilde{b}_1$$
where $s(\widetilde{b}_1) = \frac{1}{(1+\varepsilon)}$. Set $j := 1$
2. Starting with round $i = 1$ and ending on round l, it does the following steps:
   (a) If $(i-1) \mod \lceil \phi \rceil \cdot \rho = 0$ and $i > 1$, put $j := j+1$ and add a new block $\widetilde{b}_j$, with $s(\widetilde{b}_j) = \frac{1}{\phi}$ to the adversarial chain
   $$\widetilde{\mathcal{C}}^j_0 := b_0 \hookleftarrow \widetilde{b}_1 \hookleftarrow \cdots \hookleftarrow \widetilde{b}_{j-1} \hookleftarrow \widetilde{b}_j.$$
   Then $\mathcal{A}$ puts *lock* $:= \rho - 1$ and starts replotting on $\widetilde{b}_j$.

(b) For next $\lceil \phi \rceil \cdot \rho - 1$ rounds $\mathcal{A}$ repeats replotting on $\widetilde{b}_j$ to increase its space to $\lceil \phi \rceil / \phi \geq 1$. Go back to step (a).
3. In the last round $\mathcal{A}$ bootstraps the chain to create additional $l - k$ blocks to get a chain of length $l + 1$.

The adversarial chain now is

$$\widetilde{\mathcal{C}}_0^l \ = b_0 \hookleftarrow \widetilde{b}_1 \ \hookleftarrow \cdots \hookleftarrow \widetilde{b}_{l-1} \ \hookleftarrow \widetilde{b}_l .$$

Here $\lambda = 0$ as the chains $\mathcal{C}_0^l, \widetilde{\mathcal{C}}_0^l$ forked at the first block. Thus we get $\sum_{i=1}^{\min(l,k)} s(b_i) = k$ while $\sum_{i=1}^{\min(l,k)} s(\widetilde{b}_i) \geq k$. Hence $\Lambda_{\text{genesis}}^k(\mathcal{C}_0^j, \widetilde{\mathcal{C}}_0^j) = 1$ and $\mathcal{A}$ wins. □

2.6 Our Contribution

Having introduced the forking game, we can now state our main result that under resource variability, no PoSpace longest-chain blockchain is secure.

Theorem 1 (Impossibility Result). *For every chain selection rule Λ, there exists an adversary $\mathcal{A}$ that wins the $(\phi, \varepsilon, \rho, \Lambda)$-forking game in*

$$\ell = \left\lceil \rho \cdot \phi^2 \cdot (1+\varepsilon) \cdot \left(\frac{(1+\varepsilon) - \frac{1}{\phi}}{\varepsilon} \right) \right\rceil + \left\lceil \rho \cdot \phi^2 \cdot \left(\frac{(1+\varepsilon) - \frac{1}{\phi}}{\varepsilon} \right) \right\rceil + 2 \cdot \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil \ \textit{steps.}$$

We sketched the general proof idea in Section 1.4. The full formal proof can be found in the Appendix A.1. While the expression in the theorem is somewhat complicated, typically we'd assume that ε is small, say < 0.1, while ϕ is bounded away from 1, say $\phi \geq 1.1$. In this case the bound becomes

$$\ell \leq O(\rho \cdot \phi^2 / \varepsilon) \ .$$

Next, we prove that the fork length in the attack from Theorem 1 is an optimal attack up to a factor ϕ. We show this by providing a simple chain selection rule Λ_{tent} such that $\mathcal{A}$ can not win a $(\phi, 1+\varepsilon, \rho, \Lambda_{\text{tent}})$ if the fork length is less than $\rho \left(\phi \cdot (1+\varepsilon) \cdot \frac{1-\frac{1}{\phi}}{\varepsilon} - \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil \right)$.

Before we give proof, we'll make a definition that will be useful. A $\Gamma = (\phi, x, y)$-tent, with $x \in \mathbb{N}$, $y \in \mathbb{R}_{>0}$, is the infinite sequence

$$\ldots, y_{x-1}, y_x, y_{x+1}, \ldots$$

where for $i > x$, $y_i = y_{i-1}/\phi$ and for $i < x$, $y_{2x-i} = y_{2x-i+1}/\phi$ (for an illustration see Fig. 1, where the first 36 steps of the dark blue curve are part of a ($\phi = 1.02, x = 18, y = 2$) tent. We say y is the size of the tent $\Gamma = (\phi, x, y)$ and that tent $\Gamma = (\phi, x, y)$ is larger than tent $\Gamma' = (\phi, x', y')$ if $y > y'$.

Theorem 2 (The attack from Theorem 1 is tight up to ϕ). *For any $(\phi, 1+\varepsilon, \rho)$ there's a chain selection rule Λ_{tent} for which no adversary can win the $(\phi, 1+\varepsilon, \rho, \Lambda_{\text{tent}})$ forking game in less than*

$$\rho \left(\phi \cdot (1+\varepsilon) \cdot \frac{1-\frac{1}{\phi}}{\varepsilon} - \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil \right) \textit{ steps}$$

The full proof can be found in Appendix A.2.

3 Discussion and Open Problem

In this paper, we showed that there's no chain selection rule that guarantees security (against double spending) for a permissionless longest-chain blockchain based on proofs of space assuming honest parties always hold more space than an adversary.

Overcoming our No-Go Result. Recall that our attacker can replot space and bootstrap the chain. Two existing PoSpace based chains, Chia and Filecoin, avoid our impossibility in different ways. Chia prevents bootstrapping by additionally using proofs of time, while Filecoin avoids replotting by using a BFT (rather than longest-chain) type protocol as we'll elaborate below.

Chia [7] combines proofs of space with "proofs of time", where the latter are instantiated with verifiable delay functions [5]. One can think of (a simplified version of) Chia as simply alternating PoSpace with VDFs, where the challenge for the next VDF (PoSpace) is computed from the previous PoSpace (VDF output). Bootstrapping such a chain is not possible as the main security property of a VDF requires that computing its output requires time.

In Filecoin parties must register their space before it can be used for mining. Moreover, parties must constantly compute and publish proofs for their registered space. Blocks are then created by the parties who registered space in a BFT-type protocol. Informally, this prevents replotting as only registered space can be used, and registering more space than one actually controls is not possible as one must constantly prove that space is available. Using the classification from [10], one can see our results as being in the fully permissionless setting (where the protocol has no knowledge about current participation), while Filecoin works in a quasi-permissionless setting (where parties must be known to the protocol and be always available).

A question one can ask is whether simply committing to space without periodic checks would overcome our impossibility result. This would correspond to *dynamic availability* setting in [10]. The answer is no: an adversary can simply plot and commit to many different proofs of space in the honest chain and then later replot them when launching an attack. Thus our result precludes PoSpace based Nakamoto like longest chain blockchain in both fully permissionless and dynamic availability setting.

Open Problems. Our upper and lower bounds are separated by a gap ϕ, we believe this gap can be closed by coming up with a more sophisticated chain selection rule for Theorem 2.

References

1. The chia network blockchain. https://docs.chia.net/green-paper-abstract/ (2019)
2. Abusalah, H., Alwen, J., Cohen, B., Khilko, D., Pietrzak, K., Reyzin, L.: Beyond hellman's time-memory trade-offs with applications to proofs of space. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10625, pp. 357–379. Springer (2017). https://doi.org/10.1007/978-3-319-70697-9_13, https://doi.org/10.1007/978-3-319-70697-9_13
3. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 913–930. ACM (2018). https://doi.org/10.1145/3243734.3243848, https://doi.org/10.1145/3243734.3243848
4. Bagaria, V.K., Dembo, A., Kannan, S., Oh, S., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Proof-of-stake longest chain protocols: Security vs predictability. In: Soares, J.M., Song, D., Vukolic, M. (eds.) Proceedings of the 2022 ACM Workshop on Developments in Consensus, ConsensusDay 2022, Los Angeles, CA, USA, 7 November 2022. pp. 29–42. ACM (2022). https://doi.org/10.1145/3560829.3563559, https://doi.org/10.1145/3560829.3563559
5. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 757–788. Springer International Publishing, Cham (2018)
6. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982. pp. 199–203. Plenum Press, New York (1982). https://doi.org/10.1007/978-1-4757-0602-4_18, https://doi.org/10.1007/978-1-4757-0602-4_18
7. Cohen, B., Pietrzak, K.: The chia network blockchain. https://docs.chia.net/files/Precursor-ChiaGreenPaper.pdf (2019), this is an early proposal and differs significantly from the implemented version [1]
8. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology – CRYPTO 2015. pp. 585–605. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
9. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 357–388. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7_12, https://doi.org/10.1007/978-3-319-63688-7_12

10. Lewis-Pye, A., Roughgarden, T.: Permissionless consensus (2024), https://arxiv.org/abs/2304.14701
11. Park, S., Kwon, A., Fuchsbauer, G., Gazi, P., Alwen, J., Pietrzak, K.: Spacemint: A cryptocurrency based on proofs of space. In: Meiklejohn, S., Sako, K. (eds.) Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26 - March 2, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10957, pp. 480–499. Springer (2018). https://doi.org/10.1007/978-3-662-58387-6_26, https://doi.org/10.1007/978-3-662-58387-6_26
12. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger

A Proofs

A.1 Proof of Theorem 1

Theorem 1 (Impossibility Result). *For every chain selection rule Λ, there exists an adversary $\mathcal{A}$ that wins the $(\phi, \varepsilon, \rho, \Lambda)$-forking game in*

$$\ell = \left\lceil \rho \cdot \phi^2 \cdot (1+\varepsilon) \cdot \left(\frac{(1+\varepsilon) - \frac{1}{\phi}}{\varepsilon} \right) \right\rceil + \left\lceil \rho \cdot \phi^2 \cdot \left(\frac{(1+\varepsilon) - \frac{1}{\phi}}{\varepsilon} \right) \right\rceil + 2 \cdot \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil \text{ steps.}$$

Proof (Proof of Theorem 1). We already sketched the general idea in Section 1.4 and will make it more formal here.

To prove the theorem we must specify two space profiles $\mathcal{S} = (s_i^{\mathcal{H}})_{i=0}^{\ell}$ and $\tilde{\mathcal{S}} = (\tilde{s}_i^h)_{i=0}^{\ell}$ where $s_0^{\mathcal{H}} = \tilde{s}_0^h$ and $s_i^{\mathcal{H}}/(1+\varepsilon) \leq s_{i+1}^{\mathcal{H}} \leq s_i^{\mathcal{H}} \cdot (1+\varepsilon)$ and $\tilde{s}_i^h/(1+\varepsilon) \leq \tilde{s}_{i+1}^h \leq \tilde{s}_i^h \cdot (1+\varepsilon)$(same for $\tilde{s}^h$) such that

– Using $s_i^{\mathcal{A}} = s_i^{\mathcal{H}}/\phi$ the adversary can create a chain

$$\widetilde{\mathcal{C}}_0^l = b_0 \hookleftarrow \widetilde{b}_1 \hookleftarrow \cdots \hookleftarrow \widetilde{b}_{\ell-1} \hookleftarrow \widetilde{b}_\ell.$$

with space profile $(\tilde{s}_0^h, \tilde{s}_1^h, \ldots, \tilde{s}_\ell^h)$

– Using $\tilde{s}_i^a = \tilde{s}_i^h/\phi$ the adversary can create a chain

$$\mathcal{C}_0^l = b_0 \hookleftarrow b_1 \hookleftarrow \cdots \hookleftarrow b_{\ell-1} \hookleftarrow b_\ell.$$

with space profile $(s_0^{\mathcal{H}}, s_1^{\mathcal{H}}, \ldots, s_\ell^{\mathcal{H}})$

For this we first define $k := \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil$ and $l := \left\lceil \rho \cdot \phi^2 \cdot (1+\varepsilon) \cdot \left(\frac{1+\varepsilon-\frac{1}{\phi}}{\varepsilon} \right) \right\rceil + \left\lceil \rho \cdot \phi^2 \cdot \left(\frac{1+\varepsilon-\frac{1}{\phi}}{\varepsilon} \right) \right\rceil$. Notice, $\ell = 2k + l$.

In the first space profile in the first k rounds space increases by a factor $1+\varepsilon$ each round and then in the next k rounds it decreases by a factor $1+\varepsilon$ each round. In the remaining $l+1$ rounds the space stays constant at 1. In the second space profile the roles of k and $l+1$ rounds are reversed; in the first $l+1$ rounds the space stays constant at 1. Then in the next k rounds space increases by a factor $1+\varepsilon$ each round and finally in the last k rounds it decreases by a factor $1+\varepsilon$ each round. Formally,

$$s_i^{\mathcal{H}} = \begin{cases} (1+\varepsilon)^i & \text{for } 0 \le i \le k-1 \\ (1+\varepsilon)^{2k-i} & \text{for } k \le i \le 2k-1 \\ 1 & \text{for } 2k \le i \le l+2k \end{cases}$$

and

$$\tilde{s}_i^h = \begin{cases} 1 & \text{for } 0 \le i \le l \\ (1+\varepsilon)^{i-l-1} & \text{for } l+1 \le i \le l+k \\ (1+\varepsilon)^{l+2k-i} & \text{for } l+k+1 \le i \le l+2k \end{cases}$$

Lemma 3 ($\mathcal{A}$ creates $\widetilde{\mathcal{S}}$ from $\mathcal{S}$). *An adversary, $\mathcal{A}$, playing $(\phi, 1+\varepsilon, \rho, \Lambda)$-forking game can create a chain $\widetilde{\mathcal{C}}_0^\ell$ such that $\mathcal{S}(\widetilde{\mathcal{C}}_0^\ell) = (\tilde{s}_i^h)_{i=0}^\ell$ while honest chain is $\mathcal{C}_0^\ell$ with $\mathcal{S}(\mathcal{C}_0^\ell) = (s_i^{\mathcal{H}})_{i=0}^\ell$*

Proof (Proof of Lemma 3). The adversary, $\mathcal{A}$, does following to create $\widetilde{\mathcal{C}}_{i=0}^\ell$ while honest chain is $\mathcal{C}_{i=0}^\ell$:

1. At round $i=0$,
$$\mathcal{C}_0^0 = \widetilde{\mathcal{C}}_0^0 = b_0$$
where $s(b_0) = 1$.
2. For $1 \le i \le k-1$, set $1+\varepsilon_i = 1+\varepsilon$. Thus $s_i^{\mathcal{H}} = 1+\varepsilon \cdot s_{i-1}^{\mathcal{H}}$. Honest chain becomes
$$\mathcal{C}_0^i = \mathcal{C}_0^{i-1} \hookleftarrow b_i$$
where $s(b_i) = (1+\varepsilon)^i$. The adversarial chain
$$\widetilde{\mathcal{C}}_0^0 = b_0$$
remains unchanged.
3. For $i = k$, set $1+\varepsilon_i = 1+\varepsilon$. So, $s_i^{\mathcal{H}} = 1+\varepsilon \cdot s_{i-1}^{\mathcal{H}} = (1+\varepsilon)^k \ge \phi$. Thus $s_i^{\mathcal{A}} \ge 1$. Honest chain is
$$\mathcal{C}_0^i = \mathcal{C}_0^{i-1} \hookleftarrow b_i$$
where $s(b_i) = (1+\varepsilon)^i$. Now $\mathcal{A}$ bootstraps the adversarial chain to become
$$\widetilde{\mathcal{C}}_0^l = b_0 \hookleftarrow \widetilde{b}_1 \hookleftarrow \ldots \hookleftarrow \widetilde{b}_l$$
where $s(\widetilde{b}_j) = 1$ for all $j \in [1, l]$. This is demonstrated in Fig. 2(a).
4. For $k+1 \le i \le 2k$, set $1+\varepsilon_i = \frac{1}{(1+\varepsilon)}$. Thus, $s_i^{\mathcal{H}} = (1+\varepsilon)^{2k-i}$. Honest chain becomes
$$\mathcal{C}_0^i = \mathcal{C}_0^{i-1} \hookleftarrow b_i$$
where $s(b_i) = (1+\varepsilon)^{2k-i}$. The adversarial chain remains at $\widetilde{\mathcal{C}}_0^l$.

5. For $2k+1 \leq i \leq l+2k$, set $1+\varepsilon_i = 1$. Thus $s_i^{\mathcal{H}} = 1$. Honest chain continues as

$$\mathcal{C}_0^i = \mathcal{C}_0^{i-1} \hookleftarrow b_i$$

where $s(b_i) = 1$. For the adversarial chain, $\mathcal{A}$ uses replotting to create

$$\widetilde{\mathcal{C}}_{l+1}^{l+2kl} = \widetilde{b}_{l+1} \hookleftarrow \ldots \hookleftarrow \widetilde{b}_{l+2k}$$

such that

$$s(b_{l+i}) = \begin{cases} (1+\varepsilon)^i & \text{for } 1 \leq i \leq k \\ (1+\varepsilon)^{2k-i} & \text{for } k+1 \leq i \leq 2k. \end{cases}$$

This is demonstrated in Fig. 2(b). Thus it achieves an adversarial chain

$$\widetilde{\mathcal{C}}_0^{\ell} = \widetilde{\mathcal{C}}_0^{l} \hookleftarrow \widetilde{\mathcal{C}}_{l+1}^{l+2k}$$

with the space profile $\widetilde{\mathcal{S}}$.

To see why replotting can achieve the requisite space profile, note that $s_i^{\mathcal{A}} = 1/\phi \, \forall i \in [l+1, l+2k]$. In order to create a block b such that $s(b) = \alpha$, $\mathcal{A}$ needs to replot $\left\lceil \frac{\alpha - \frac{1}{\phi}}{\frac{1}{\phi}} \right\rceil = \lceil \alpha \cdot \phi - 1 \rceil$ times and this would take $\rho \cdot \lceil \alpha \cdot \phi - 1 \rceil$ rounds. Thus the total number of rounds required is

$$\begin{aligned}
&\sum_{i=1}^{k} \rho \cdot \left\lceil (1+\varepsilon)^i \cdot \phi - 1 \right\rceil + \sum_{i=k+1}^{2k} \rho \cdot \left\lceil (1+\varepsilon)^{2k-i} \cdot \phi - 1 \right\rceil \\
&\leq \rho \cdot \sum_{i=1}^{k} \left\lceil (1+\varepsilon)^i \cdot \phi - 1 \right\rceil + \rho \cdot \sum_{i=0}^{k-1} \left\lceil (1+\varepsilon)^i \cdot \phi - 1 \right\rceil \\
&\leq \rho \cdot \sum_{i=1}^{k} (1+\varepsilon)^i \cdot \phi + \rho \cdot \sum_{i=0}^{k-1} (1+\varepsilon)^i \cdot \phi \\
&= \rho \cdot \phi \cdot (1+\varepsilon) \cdot \frac{(1+\varepsilon)^k - 1}{\varepsilon} + \rho \cdot \phi \cdot \frac{(1+\varepsilon)^k - 1}{\varepsilon} \\
&\leq \rho \cdot \phi \cdot (1+\varepsilon) \cdot \frac{\phi \cdot (1+\varepsilon) - 1}{\varepsilon} + \rho \cdot \phi \cdot \frac{\phi \cdot (1+\varepsilon) - 1}{\varepsilon} \\
&\big(\text{as } \phi \leq \lceil \phi \rceil \leq (1+\varepsilon)^k \leq \phi \cdot (1+\varepsilon)\big) \\
&= \rho \cdot \phi^2 \cdot (1+\varepsilon) \cdot \frac{1+\varepsilon - \frac{1}{\phi}}{\varepsilon} + \rho \cdot \phi^2 \cdot \frac{1+\varepsilon - \frac{1}{\phi}}{\varepsilon} \\
&\leq \left\lceil \rho \cdot \phi^2 \cdot (1+\varepsilon) \cdot \left(\frac{1+\varepsilon - \frac{1}{\phi}}{\varepsilon} \right) \right\rceil + \left\lceil \rho \cdot \phi^2 \cdot \left(\frac{1+\varepsilon - \frac{1}{\phi}}{\varepsilon} \right) \right\rceil = l
\end{aligned}$$

Since total number of rounds is $\ell = l + 2k$, the adversary $\mathcal{A}$ after round $2k$ has l rounds to replot, which is sufficient.

At round $i = l + 2k = \ell$ we have the honest chain as $\mathcal{C}_0^\ell$ with space-profile $\mathcal{S}$ and adversarial chain as $\widetilde{\mathcal{C}}_0^\ell$ with space-profile $\widetilde{\mathcal{S}}$. This completes the proof of Lemma 3. □

Lemma 4 ($\mathcal{A}$ creates $\mathcal{S}$ from $\widetilde{\mathcal{S}}$). *An adversary, $\mathcal{A}$, playing $(\phi, 1+\varepsilon, \rho, \Lambda)$-forking game can create a chain $\mathcal{C}_0^\ell$ such that $\mathcal{S}(\mathcal{C}_0^\ell) = (s_i^{\mathcal{H}})_{i=0}^\ell$ while honest chain is $\widetilde{\mathcal{C}}_0^\ell$ with $\mathcal{S}(\widetilde{\mathcal{C}}_0^\ell) = (\tilde{s}_i^h)_{i=0}^\ell$*

Proof (Proof of Lemma 4). To create $\mathcal{C}_{i=0}^\ell$ while honest chain is $\widetilde{\mathcal{C}}_{i=0}^\ell$, the adversary, $\mathcal{A}$, does the reverse of Lemma 3, *i.e.* it first replots and then bootstraps. Formally,

1. At round $i = 0$,
$$\mathcal{C}_0^0 = \widetilde{\mathcal{C}}_0^0 = b_0$$
where $s(b_0) = 1$. Note that here $\widetilde{\mathcal{C}}$ is the honest chain while $\mathcal{C}$ is the adversarial chain.
2. For round $1 \le i \le l$, set $1 + \varepsilon_i = 1$. Thus, $s_i^{\mathcal{H}} = 1$. The honest chain becomes
$$\widetilde{\mathcal{C}}_0^i = \widetilde{\mathcal{C}}_0^{i-1} \hookleftarrow \widetilde{b}_i$$
where $s(\widetilde{b}_i) = 1$. $\mathcal{A}$ replots a chain
$$\mathcal{C}_0^{2k} = b_0 \hookleftarrow b_1 \hookleftarrow \ldots \hookleftarrow b_{2k}$$
such that
$$s(b_i) = \begin{cases} (1+\varepsilon)^i & \text{for } 1 \le i \le k \\ (1+\varepsilon)^{2k-i} & \text{for } k+1 \le i \le 2k. \end{cases}$$
This is demonstrated in Fig. 2(c). As argued in Lemma 3 Item 5 l rounds are sufficient to replot $\mathcal{C}_0^l$.
3. For round $l + 1 \le i \le l + k - 1$, set $1 + \varepsilon_i = 1 + \varepsilon$. Thus $s_i^{\mathcal{H}} = (1+\varepsilon)^{i-l}$. The honest chain becomes
$$\widetilde{\mathcal{C}}_0^i = \widetilde{\mathcal{C}}_0^{i-1} \hookleftarrow \widetilde{b}_i$$
where $s(\widetilde{b}_i) = (1+\varepsilon)^{i-l}$. The adversarial chain remains unchanged.
4. For round $i = l + k$, set $1 + \varepsilon_i = 1 + \varepsilon$. Thus $s_i^{\mathcal{H}} = (1+\varepsilon)^k \ge \lceil \phi \rceil$. Therefore, $s_i^{\mathcal{A}} \ge \frac{\lceil \phi \rceil}{\phi} \ge 1$. The honest chain continues as
$$\widetilde{\mathcal{C}}_0^i = \mathcal{C}_0^{i-1} \hookleftarrow \widetilde{b}_i$$
where $s(\widetilde{b}_i) = (1+\varepsilon)^k$. $\mathcal{A}$ uses space $s_i^{\mathcal{A}} \ge 1$ to bootstrap the adversarial chain to form
$$\mathcal{C}_0^{2k+l} = \mathcal{C}_0^{2k} \hookleftarrow b_{2k+i} \hookleftarrow \ldots \hookleftarrow b_{2k+l}$$
where $s(b_{2k+i}) = 1 \; \forall i \in [1, l]$. This is demonstrated in Fig. 2(d).

5. For round $l+k+1 \leq i \leq l+2k$, set $1+\varepsilon_i = \frac{1}{(1+\varepsilon)}$. Thus $s_i^{\mathcal{H}} = (1+\varepsilon)^{l+2k-i}$. The honest chain continues as
$$\widetilde{\mathcal{C}}_0^i = \widetilde{\mathcal{C}}_0^{i-1} \hookleftarrow \widetilde{b}_i$$
where $s(\widetilde{b}_i) = (1+\varepsilon)^{l+2k-i}$. The adversarial chain remains unchanged.

At round $i = l+2k = \ell$ we have the honest chain as $\widetilde{\mathcal{C}}_0^\ell$ with space-profile $\widetilde{\mathcal{S}}$ and adversarial chain as $\mathcal{C}_0^\ell$ with space-profile $\mathcal{S}$. This completes the proof of Lemma 4. □

Consider any chain selection rule Λ. From Lemmas 3 and 4 we get two forking games: first, where the honest chain is $\mathcal{C}_0^\ell$ and $\mathcal{A}$ creates $\widetilde{\mathcal{C}}_0^\ell$ and second, where the honest chain is $\widetilde{\mathcal{C}}_0^\ell$ and $\mathcal{A}$ creates $\mathcal{C}_0^\ell$. If $\Lambda(\mathcal{C}_0^\ell, \widetilde{\mathcal{C}}_0^\ell) = 0$, then in the first forking game the chain selection rule would choose the adversarial chain. If $\Lambda(\mathcal{C}_0^\ell, \widetilde{\mathcal{C}}_0^\ell) = 1$, then in the second forking game the chain selection rule would choose the adversarial chain. Therefore in either case it is possible to fool the chain selection. Note that we made no restriction on Λ; we only used replotting and resource variability. Hence, we can conclude that there does not exist a secure chain selection rule under resource variability. □

A.2 Proof of Theorem 2

Theorem 2 (The attack from Theorem 1 is tight up to ϕ). *For any $(\phi, 1+\varepsilon, \rho)$ there's a chain selection rule Λ_{tent} for which no adversary can win the $(\phi, 1+\varepsilon, \rho, \Lambda_{\text{tent}})$ forking game in less than*
$$\rho\left(\phi \cdot (1+\varepsilon) \cdot \frac{1-\frac{1}{\phi}}{\varepsilon} - \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil\right) \textit{ steps}$$

Proof. Given two chains $\mathcal{C}_0^j = b_0 \hookleftarrow \cdots \hookleftarrow b_j$ and $\widetilde{\mathcal{C}}_0^j = \widetilde{b}_0 \hookleftarrow \cdots \hookleftarrow \widetilde{b}_j$ selection rule first determines the forking point f, i.e., the first block where chains diverge to have $b_f \neq \widetilde{b}_f$. Let $l := j - f$ denote the fork length.

Let $\mathcal{S} = (a_1, \ldots, a_l) \stackrel{\text{def}}{=} (s(b_f), \ldots, s(b_j))$ and $\widetilde{\mathcal{S}} = (\widetilde{a}_1, \ldots, \widetilde{a}_l) \stackrel{\text{def}}{=} (s(\widetilde{b}_f), \ldots, s(\widetilde{b}_j))$ be the space profiles of the fork. The Λ_{tent} rule now picks the chain whose fork covers the larger tent. More formally, let μ be maximal such that there exists a tent $\Gamma = (\phi, x, \mu)$ under $\mathcal{S}$, where $f \leq x \leq j$ and for all $i, 0 \leq i \leq j$ we have $a_i \geq y_i$. Similarly, we define tent $\widetilde{\Gamma} = (\phi, \widetilde{x}, \widetilde{\mu})$ for $\widetilde{\mathcal{S}}$. With these definitions, the chain selection rule is defined as $\Lambda_{\text{tent}}(\mathcal{C}_0^j, \widetilde{\mathcal{C}}_0^j) = 0$ if $\mu \geq \widetilde{\mu}$.

Let's assume $\mathcal{C}_0^j$ is the honest chain, while $\widetilde{\mathcal{C}}_0^j$ is the adversarial one created in the $(\phi, 1+\varepsilon, \rho, \Lambda_{\text{tent}})$ forking game. To win the game it must hold that $\widetilde{\mu} > \mu$. We know that during the fork (when the honest parties created $b_f, \ldots, b_j$) the honest parties never controlled more μ space (otherwise we could have embedded a tent larger than μ under A), and thus the adversary never controlled more than μ/ϕ space in that phase. But during that phase he must have created a chain with space profile $\widetilde{\mathcal{S}}$, which in particular contains a tent of size $\widetilde{\mu}$, and thus also a tent $\widetilde{\varGamma}' = (\phi, \widetilde{x}, \mu)$ of size μ (as $\mu < \widetilde{\mu}$). While the adversary might have used bootstrapping to create this it could only have bootstrapped space up to μ/ϕ.

We want a lower bound on the amount of space above μ/ϕ in the tent $\widetilde{\varGamma}'$ after the forking point. Let $\widetilde{z} \coloneqq \widetilde{x} - f$ and $k = \left\lceil \frac{\log \phi}{\log(1+\varepsilon)} \right\rceil$. We have the following scenarios

1. If $\widetilde{z} \geq k$ or $l - \widetilde{z} \geq k$, then the fork is long enough to contain a sequence of $\mu, \frac{\mu}{(1+\varepsilon)}, \cdots, \frac{\mu}{(1+\varepsilon)^{k-1}}$ above the $\frac{\mu}{\phi}$. Number of rounds required to replot this space, using $\frac{\mu}{\phi}$, is
$$\sum_{i=0}^{k-1} \rho \cdot \left\lceil \frac{\frac{\mu}{(1+\varepsilon)^i} - \frac{\mu}{\phi}}{\frac{\mu}{\phi}} \right\rceil = \sum_{i=0}^{k-1} \rho \cdot \left\lceil \frac{\phi}{(1+\varepsilon)^i} - 1 \right\rceil$$
$$\geq \rho \cdot \sum_{i=0}^{k-1} \left(\frac{\phi}{(1+\varepsilon)^i} - 1 \right) = \rho \cdot \left(\phi \cdot (1+\varepsilon) \cdot \frac{1 - \frac{1}{\phi}}{\varepsilon} - k \right).$$
2. If $\widetilde{z} < k$ and $l - \widetilde{z} < k$, then the fork is too short for replotting as every step of the tent in the forked chain is above μ/ϕ and for each of them we need at least ρ steps of replotting. Thus in total at least $\rho * l$ steps are required. Since $\rho \geq 2$, replotting is not possible. Therefore, $\mathcal{A}$ cannot win the game and we have a contradiction.

This finishes the proof that to win $(\phi, 1+\varepsilon, \rho, \varLambda_{\text{tent}})$ forking game the fork length must be at least $\rho \cdot \left(\phi \cdot (1+\varepsilon) \cdot \frac{1-\frac{1}{\phi}}{\varepsilon} - k \right)$. □