# Stabilizing Congestion in Decentralized Record-Keepers*

Assimakis Kattis† Fabian Trottner‡

May 2020

**Abstract**

We provide an economic analysis of a market for record-keeping instantiated using succinct proof systems. Our model allows for adaptive adjustments in predicate (block) size according to pre-specified update rules. We show that when block rewards are absent, there exists a predicate size update rule, based on observed changes in difficulty, that induces stable equilibrium levels of transaction fees and congestion at varying levels of user demand. By using a Non-Interactive Zero-Knowledge proof as the consensus puzzle and integrating adaptive block size updates in the model, we ensure that adjustments in predicate size and thus block capacity incentivize miners to incur higher/lower computational cost in exchange for higher/lower transaction fee revenue. We provide an analysis of the various technological requirements needed to instantiate such a system in a commercially viable setting and identify relevant research directions.

*We would like to thank Joseph Bonneau, Tarun Chitra, and Georgios Konstantopoulos for helpful feedback and discussions.

†Courant Institute for Mathematical Sciences, New York University, kattis@cs.nyu.edu

‡Department of Economics, Princeton University, trottner@princeton.edu

# 1 Introduction

Blockchain technology (Nakamoto (2008)) offers a decentralized alternative to traditional record-keeping systems. In this work, we study the economic properties of record-keepers instantiated using succinct proof systems. Miners are responsible for producing proofs that a set of records (or transactions) provided to them by end-users is valid, and to add these to an append-only ledger while maintaining consensus. Such an approach generalizes the Bitcoin model, in that end-users can request a record of any truth claim (such as smart contract verification). Since the underlying proof-of-work (PoW) consensus mechanism ensuring correctness poses substantive costs (growing linearly with system size) (Abadi and Brunnermeier (2018); Budish (2018); Thum (2018)), it is desirable for it to output useful computation.

We work within a specific consensus framework, in which the proof of block validity and the consensus algorithm are combined into one process, while retaining the Nakamoto consensus (Nakamoto (2008)) security properties. In this model, a (suitably designed) Non-Interactive Zero-Knowledge (NIZK) proof is used as the PoW puzzle (Kattis and Bonneau (2020)) (instead of an arbitrary hash function). If the NIZK is performing state verification, this connection provides a link between miners' computational work in generating a proof (and thus a block) and the number of transactions that can be verified in a given block (also referred to as *predicate size*). Our analysis shows how to exploit this link so as to design decentralized record-keepers that maintain cost-efficiency at varying levels of demand. Our contributions are most closely related to previous work that extracts usefulness out of the PoW procedure (Ball et al. (2017); King (2013)), and recent initiatives that permit adaptive block sizes in cryptocurrency implementations (such as Ethereum Buterin (2014)).

## 1.1 NIZKs & Blockchains

Recent work has looked into solving various problems in distributed payment systems using NIZK proof systems (Buterin (2018 (accessed June 12, 2020); Gaži et al. (2019); Kerber et al. (2020); Sasson et al. (2014)). These are cryptographic primitives that are able to provide constant-size proof of knowledge guarantees for any NP statement (Ben-Sasson et al. (2013); Lapidot and Shamir (1990); Rackoff and Simon (1991)), regardless of its complexity. Their applications to blockchain-based protocols range from privacy enhancements to scaling and interoperability improvements. Proofs of block validity in blockchain systems can be encoded as NIZK proofs and appended to every new block without substantially increasing its size. In such a system, a miner needs to only publish in the block the minimal amount of information required to recreate the

state transition, since the proof will demonstrate that this transition is valid. This has been the focus of many initiatives to date, in which the state transition function of the underlying blockchain system is encoded as a *predicate*, which takes as input the state transition and a set of witness variables, outputting a succinct proof of correctness. This provides scaling benefits in two main ways:

1. Increases the number of transactions that can be processed at every block[1]. This is bounded above by the size of the transaction data included in each block.

2. End users and light-clients can verify block updates with minimal resources, using only a commitment to state (Kattis and Bonneau (2020); Meckler and Shapiro (2018)).

However, there is an additional challenge that such systems need to overcome. The generation of proofs at the scale required for a commercial payment network comes with large hardware and energy costs, meaning that system actors need to be properly incentivized to produce them. The naive approach of requiring miners to generate and add such a proof to each block header means that miners will need to perform non-trivial computations to generate the block before they begin searching for a PoW solution. Since the proof will only need to be created once, larger miners will face amortization benefits from this and skew incentives towards centralization. Approaches to achieve such an incentive-compatible system with succinct provers in Proof of Stake (PoS) (King and Nadal (2012)) protocols is also an open problem. This highlights a fundamental question that needs to be addressed: in a distributed payments protocol, how should system incentives be designed to ensure the expensive computation of generating these proofs?

### 1.1.1 Incentivizing Useful Computation

Recent work (Kattis and Bonneau (2020)) has shown that it is possible to conjoin the PoW and proof generation process to resolve the above incentive problem. Proof of Necessary Work (PoNW) achieves this by using the nonce to uniquely rerandomize the generated proof, ensuring that it also acts as a PoW puzzle. Resolving the incentive issue in this way does not come for free. The main security tradeoff in PoNW is captured by the ratio $\tau$ between proof generation and block times. If a proof takes too long to compute relative to block time, its generation can interfere with the PoW process. This is because a large portion of the miners will have to abandon their solutions mid-attempt at every step, leading to increased chances of forking and more

[1] See Buterin (2018 (accessed June 12, 2020)).

wasted work. Thus, it is desirable to reduce the cost of generating a single proof if we want low block times (which in turn implies higher throughput) without $\tau$ getting too large. Note that we cannot do better than Nakamoto consensus in a fundamental way: all miners will need to compute sequential proofs for an amount of time proportional to block time.

## 1.2 Variable Block Size

### 1.2.1 Operating at Maximal Block Size $S_{max}$

A fundamental question concerning the viability of such a system is its operating throughput. If the maximal amount of transactions that we can add is $S_{max}$, it is natural to consider a protocol which produces a proof for $S_{max}$ transactions in each block. This system would always be operating at maximal throughput, generating a proof for $S_{max}$ transactions regardless of how full the block is. If $S_{max}$ is large enough to act as a suitable ceiling[2], then the PoNW computation required would always be the same (subject to the difficulty parameter) and the costs of generating blocks are going to be maximal for all miners. Such a system is inefficient from an economic perspective, as miners will not receive any cost reduction in proof generation times from including fewer transactions in the block. Indeed, the properties of PoW mean that this puzzle would take the same amount of time to solve regardless of its inputs (which in this case is the transaction set, which can be full or empty). This implies a lower bound for the number of transactions required for miners to continue operating at a profit.

Further, a system always operating at maximal throughput is also 'brittle' to large deviations in demand. Carlsten et al. (2016) have shown that setting block rewards to zero creates an inherently unstable system if demand is sufficiently low so that all transactions can be included in a block. By relying solely on transaction fee revenue, miners are susceptible to large losses in periods when there is little demand. Although a drop in difficulty could rectify miners' losses, this creates a security vulnerability for the system, making it susceptible to attack. Thus, adaptable throughput capacity can help ensure that prices compensate miners sufficiently during periods of low demand. Indeed, by shrinking the predicate size $S$ with falling transaction fees, the system would bolster mining revenues by restricting the supply of proofs. This puts upwards pressure on difficulty and mitigates the system's susceptibility to the instability result of Carlsten et al. (2016).

[2]Recent developments in NIZK-based applications such as zk-rollups Buterin (2018 (accessed June 12, 2020) give $S_{max}$ values that correspond to ~ 500 transactions per second, which is substantial enough for commercial scaling purposes.

### 1.2.2 Elastic Blocks

Given that the adaptability of predicate sizes (parametrized by the number of transactions processed) is desirable, a natural next question is whether it should be miners or the protocol that picks block sizes at every step. Consider allowing miners to pick the number of transactions $S$ that a proof verifies as the PoW puzzle individually: i.e. if a miner herself wants to post $S$ transactions to the chain, she picks an $S$-sized NIZK. While this in principle enables miners to pick proof sizes based on demand, the varying computational times for different proof sizes would incentivize miners to choose small values for $S$ in order to maximize their chances of winning the next block.[3] Even if difficulty requirements change based on predicate size chosen, the fact that the *first* miner who finds a solution wins will skew the distribution of winners towards those with more frequently generated proofs, again favoring miners with smaller blocks. In light of these concerns, we instead propose that the protocol sets the accepted block size endogenously over time.

### 1.2.3 From Adaptability to Stability

A crucial design choice in distributed payment systems is that of setting transaction fees so that they are stable over changes in demand ('demand shocks'). How Bitcoin sets fees with a limited amount of block space was analyzed in Huberman et al. (2019), in which closed form solutions for how transaction fees vary with system congestion are derived when congestion is positive (i.e. the number of pending transactions exceeds block space). If we parametrize proof size by the number of transactions $S$ that a given proof verifies (i.e. the number of transactions processed in the given block), then transaction fees, system congestion, and the computational costs to miners depend on the size of predicates. Intuitively, this dependence enables suitable update rules for predicate sizes to adapt throughput to changes in demand in a cost-efficient manner. We will provide a formal framework to verify this intuition, and to provide concrete suitable update rules.

## 1.3 Our Contributions

We provide an economic model of a market for record-keeping instantiated using succinct proof systems, whose predicate size adjusts according to pre-specified update

[3] Note that this is why the approach in PoNW is to randomize all inputs, making it infeasible to pick proof generation time adversarially while ensuring that the PoW process remains memoryless with the same success probability among all actors.

rules. Predicate size corresponds to block size (thus characterizing throughput), and miners generate the corresponding NIZK as a PoW puzzle. We show that the proposed protocol implies a stable equilibrium over time in the difficulty and predicate size parameters, where a shock to demand leads to proportional increases in throughput without any long-term changes in transaction fees and system congestion. This is due to two complementary features - PoNW consensus and an adaptive block size update rule - which together ensure that adjustments in predicate size and thus block capacity incentivize miners to incur higher/lower computational cost in exchange for higher/lower transaction fee revenue. This adaptability allows the system to achieve stability.

More specifically, we provide the following contributions:

1. An empirically relevant equilibrium framework permitting an analysis of the determinants of transaction fees and (entry) behavior of mining pools in a decentralized market for record-keeping featuring adjustments of system throughput according to arbitrary predicate size update rules.

2. A formal analysis of the dynamic equilibrium behavior induced by various update rules. Specifically, we derive an update rule that induces constant equilibrium transaction fees under arbitrary levels of demand (i.e., equilibrium elasticity of fees with respect to demand is zero due to offsetting throughput increases).

3. An incentive-compatible adaptive PoW system that also performs state verification in succinct blockchain architectures. This protocol admits stable levels of transaction fees that are robust to bounded demand shocks, where the bound can be made arbitrarily large. We provide an analysis of the design requirements for such a system and identify directions towards an efficient instantiation.

We model the behavior of miners and users to reason about the change in equilibrium transaction fees and computational cost posed by mining activity subject to demand shocks. We characterize the competitive equilibrium and use our model to formally derive the optimal adjustment of throughput to changes in difficulty under which changes in demand lead to efficient system cost changes. We prove that under the derived optimal rule, and when block rewards are zero, changes in demand lead to the balanced growth of throughput and mining revenues, while transaction fees and wait times remain stable.

Our analysis yields two conditions under which stable user fees and wait times across changing levels of demand can be sustained. First, block rewards need to equal zero. Only if block rewards are equal to zero, changes in the equilibrium entry, and

computational behavior of miners and thus difficulty can serve as a sufficient statistic for changes in demand. Although necessary at the early stages of network growth, positive block rewards act as a subsidy in this context and decrease market efficiency. Second, predicate size updates need to be gradual enough so that sudden large changes in demand do not destabilize the system. Up to some maximally tolerated change in demand, an update rule which takes into account changes in difficulty to alter predicate size proportionately to the shift in demand implies stability in its average transaction fees and wait times. Note that this admits a trade-off between the magnitude of the change in demand that can be tolerated and the time it takes for the system to converge to equilibrium.

We then analyze the implications of our theoretical analysis for the instantiation of a scalable, self-regulating record-keeping system. We first look at related work on distributed proof generation (Wu et al. (2018)) and confirm the functional equivalence of our model's assumptions to the observed data. This provides design specifications and proof system properties that can be used to construct payment systems with stability guarantees. We then detail the various areas of engineering relevance required in explicitly instantiating such a protocol. We outline the types of properties that the underlying system should possess, look at the relevant technical literature, and provide directions for future work.

# 2 Preliminaries

## 2.1 Proof of Work

Many blockchain protocols ensure security by requiring miners to publicly verify energy consumption (known as proof-of-work or PoW) in order to gain the right to write transactions on the blockchain. PoW is required in order to prevent denial-of-service attacks on the system: the potential for malicious entities to flood the system with cheaply produced 'valid' blocks can rapidly overwhelm honest participants and hamper the network's ability to process valid transactions. PoW was first introduced as a solution to the related problem of e-mail spam filtering (Dwork and Naor (1992)), and provides the security guarantees against such attacks in Bitcoin (Back et al. (2002)). By imposing a high energy cost for block creation, PoW makes it prohibitively expensive to produce many valid blocks and overwhelm the system, as doing so carries a non-trivial economic burden.

In the subsequent sections, we model this security constraint as the requirement that as demand $\lambda$ increases, some sufficient statistic (which in the case of Bitcoin is difficulty

$d$) also causes a proportional increase in miner costs.

## 2.2 Difficulty

Since the network is decentralized, the addition of a new block to the blockchain takes some time to propagate among all parties. This is due to the latency inherent in distributed networks, and imposes a trade-off between the size of the block propagated and the time taken for the rest of the network to receive the updated block. This limitation implies that the frequency with which the system accepts new blocks ('block frequency') needs to be sufficiently low so that each new block has enough time to reach the whole system. This is necessary to prevent forks (a 'split' in the chain), and in practice results in a trade-off between block size and frequency. Note that this also implies an upper bound on improving throughput through modifications to block size and/or frequency.

Since the total number of miners is an endogenous variable that varies over time, the protocol needs a way to ensure that a PoW solution is provided at a fixed frequency (on average) over *all* participants in the network. This is achieved by requiring the PoW puzzle to satisfy a 'difficulty' requirement, which would mean that the expected time taken to generate a solution can be calibrated according to a difficulty parameter that can be changed over time. Through the use of time-stamps, the Bitcoin PoW algorithm keeps track of the average block frequency in the last 2016 blocks and alters difficulty up or down accordingly. By performing this adjustment proportionately to the deviation of the block frequency from a hard-coded value (10 minutes/block in Bitcoin), the system 'self-corrects' the difficulty of the PoW puzzle to ensure constant block frequency. This mechanism is crucial to ensuring that blocks in Bitcoin are found every 10 minutes, so that the previous blocks had enough time to propagate in the network and the threat of forks is minimized. In the subsequent analysis, we model the block frequency as a random variable $\mu$ (the block arrival rate) with a constant expectation (the 'block time') mirroring the above technological constraints.

## 2.3 Difficulty Adjustment

The underlying system consists of a set of users and miners, who each are responsible for requesting and providing proofs from and to the system respectively. Everyone has access to the underlying state of the system, which consists of a sequence of valid transactions that have been processed into the system (the blockchain) along with their respective proofs of correctness: proofs that each block added to the chain is valid.

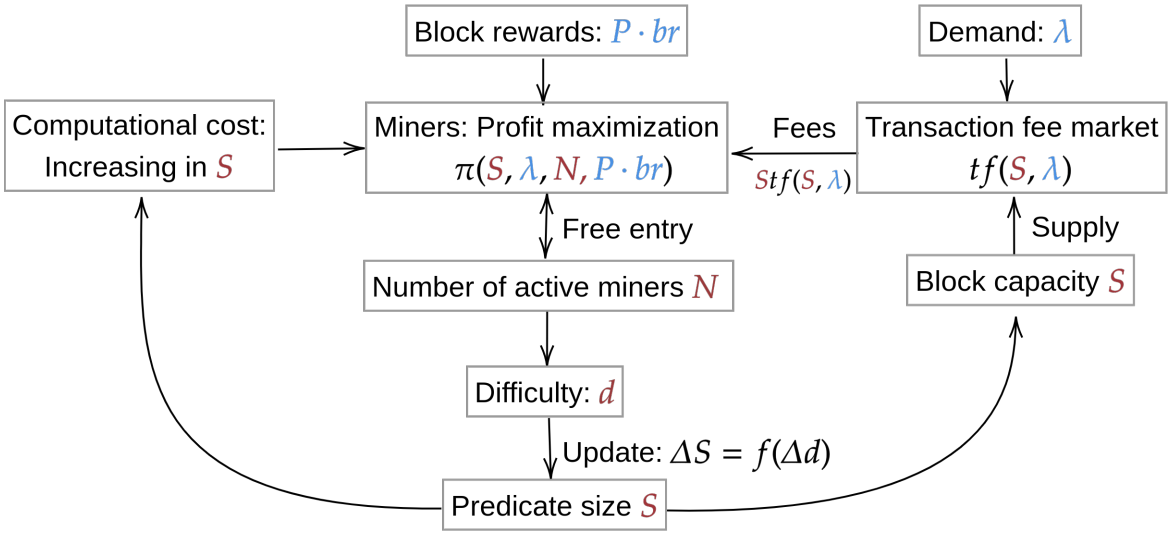
Users submit transactions (or 'records') to miners along with an associated transaction fee. The miners in turn are responsible for adding these transactions to some block which is then appended to the blockchain. This happens if the miner succeeds in generating a PoW solution that also acts as the proof of validity of all transactions in the block. We denote by $S$ the total number of transactions verified in the given block. In order for a miner to win the right to add the next block (and hence collect the corresponding transaction fees and block reward), they need to win the PoW game, which is a puzzle yielding a valid block in time proportional to the underlying difficulty parameter. This difficulty parameter functions equivalently to that in Bitcoin: through periodic updates, the difficulty parameter is adaptively changed to ensure constant block frequency. In this model, only miners are required to keep the whole blockchain (or some kind of 'state' data structure) in order to efficiently generate new blocks and proofs. End-users should be able to efficiently check provided proofs in order to find the correct chain with minimal bandwidth requirements.

## 2.4 State Verification as PoW

The process of generating succinct proofs can be quite resource intensive. One way to deal with this issue is by stipulating that the PoW procedure computes a NIZK as a by-product of each successful iteration. If the NIZK can be randomly resampled according to different nonces without affecting the underlying knowledge soundness claims, it can be used as an effective PoW puzzle. In such a model, the protocol can restrict miners to produce a NIZK attesting to the validity of $S$ transactions at any time step and accept it if a hash of the NIZK satisfies some difficulty condition. Using an update rule for $S$, such a system can be made to pick the number of transactions verified in each block based on endogenous system variables by stipulating how many transactions the NIZK needs to validate. We investigate how this extra degree of freedom can be leveraged to produce incentive-compatible equilibria in the number of transactions processed by the system per block.

Efficiency in this system is achieved by artificially constraining the block size to $S \leq S_{max}$, where $S_{max}$ a universal upper bound specified by a maximally accepted block size. This restriction to predicate size has the effect of decreasing proof generation times for miners. This is desirable in the context where $S_{max}$ is very large and blocks are (mostly) empty, as the majority of the computational work done to generate a NIZK of size $S_{max}$ will not be needed. Thus, enforcing a lower predicate size would remove the requirements for generating a very large NIZK even when the amount of transactions is substantially lower. Moreover, the restriction of block size to $S$ is a throughput bottleneck if the transaction demand rises sufficiently. We therefore need

Figure 1 A Decentralized Market for Throughput




Notes: This figure illustrates the key forces in our model. Exogenous variables are highlighted in blue, endogenous equilibrium outcomes are highlighted in red.

to investigate how to incentivize the underlying protocol to change the value of $S$ based on changes in transaction demand. This can only be achieved in a stable fashion by taking into account the incentives inherent to changing the size of the block. Since miner profits are based on transaction fees, they are directly proportional to predicate size. We use this observation as the 'driving force' behind throughput adaptability.

# 3 Economic Model

## 3.1 Overview

We model a decentralized market for throughput where transactions are verified in batches/blocks that arrive at a random rate $\mu$ in each unit of time. The chart displayed in figure 1 represents the key forces in our model.

Miners compete for the right to verify an incoming set of transactions in exchange for transaction fees and block rewards. They choose the profit-maximizing number of workers, among which to distribute proofs to simultaneously verify block correctness and generate a PoW solution. In equilibrium, all miners include the transactions offering the highest fees, up to block capacity $S$.[4] The probability that a miner gains the right to write the next block is proportional to the number of computations that she performs. The profitability of miners thus depends on the total number $N$ of

[4]In practice, miners may choose to include different transactions in each block. As we assume that miners observe the same pending transactions at any given point in time and miners have incentives to include transactions with higher fees, miners all aim to verify the same block of transactions.

active miners. Free entry requires that all miners make zero profits in expectation, characterizing the equilibrium number of miners $N$.

Predicate size $S$ corresponds to the number of transactions that can be verified in each new block. Users arrive at the pool of pending transactions at rate $\lambda$ and offer fees for transaction verification. The determination of transaction fees results from a VCG auction, where users optimally weigh the cost of higher fees against a reduction in their expected wait time. In equilibrium, average transaction fees $tf$ depend on both block capacity (and thus predicate size $S$), and on the level of demand $\lambda$.

Unlike standard economic markets, the level of supply - system throughput/block capacity $S$ - cannot be directly chosen by miners, but is instead set by the protocol. The inability of miners to choose both their computational cost *and* throughput poses an economic market failure that leads to inefficient adjustments of system cost to changes in demand. To address this limitation, we model a system that implements periodic updates to predicate size in accordance with an ex-ante specified update rule. The purpose of this updating rule is to grow throughput capacity in response to changes in demand. Our framework highlights that equilibrium system difficulty – the total proof computations performed by all miners – responds to changes in demand, and, therefore, serves as a natural candidate statistic to design such an updating rule.

Our framework allows us to analyze how key outcomes – transaction fees, the computational cost of the system, entry of miners – respond dynamically to changes in demand. A change in demand $\lambda$ induces changes in transaction fees offered by users, which changes difficulty by affecting the equilibrium number and size of active mining pools. The subsequent update in predicate size changes block capacity and, therefore, transaction fees, inducing, in turn, a change in mining profitability, the equilibrium number of miners, and, thus, difficulty. A steady-state is reached when difficulty and thus predicate size stabilize after the initial change in demand.

In this section, we focus on characterizing the problem of economic agents and defining a decentralized equilibrium. In the next section, we apply the framework to investigate the system's dynamic behavior under various updating rules for predicate size.

## 3.2 Miners

### 3.2.1 Hash Rate, Predicate Size, and Pools

Miners compete for the chance to mine the next arriving block and collect the associated mining rewards. To do so, miners distribute proof computation corresponding to predicate size $S$ to pools and provide PoW in terms of proofs per second $H$ to the

system. The relationship between proofs per second $H$ for a miner employing $n$ workers to solve proofs corresponding to a predicate size $S$ is given by:

$$\log H(n; S) = \log U + \sigma \log n - \log S \tag{3.1}$$

where $U$ is a technological constant. The parameter $\sigma$ governs the returns to pool size $n$ - the rate at which the hash rate increases as miners increase the size of their pools. This can be thought of as a parallelization coefficient, for which throughout the analysis satisfies $\sigma < 1$.[5] Note that a value of $\sigma = 1$ would imply perfect conservation of work in a parallel system of $n$ workers, and as such, no higher value of $\sigma$ is empirically achievable.

As evident in equation (3.1), an increase in predicate size $S$ reduces a pool's hash rate for any given pool size. An increase in $S$ thus poses a cost to the system as pools have to distribute proofs across more workers to maintain the same hash rate.

### 3.2.2 The Problem of Miners

Miners incur a cost $c_m \cdot n$ per time unit to distribute proofs across $n$ workers.[6] Further, (in each period) miners pay a fixed cost $f_m$ to open and maintain a mining pool. Miners compete for the chance to mine the next arriving block. In equilibrium, all miners include transactions offering the highest rewards up to block capacity $S$. Benefits of mining a block arise from total transaction fee revenue $Rev$ and block rewards $br$. Transaction fees are denominated in USD, as users propose the transaction fees. We specify $Rev$ further below after having solved the problem of users. As the system exogenously sets block rewards, their value to miners is impacted by the exchange rate of the currency to USD, which we denote by $P$. Total rewards from mining are given by $Rev + P \cdot br$.

As in PoW, the probability that a given pool is selected to mine the next block depends on its hash rate relative to the aggregate hash rate of the system.[7] We denote individual pools by $i$ and the aggregate hash rate of the system by $\mathcal{H} = \sum_{i \in \{\text{Active Pools}\}} H_i$.

Taking predicate size $S$ and the aggregate hash rate $\mathcal{H}$ as given, a miner $i$ chooses pool

[5] State-of-the-art distributive systems of SNARK proofs achieve $\sigma \approx 0.88$ Wu et al. (2018).

[6] In practice, miners distribute and assemble proofs while workers solve small computational problems. The cost $c_m$ summarizes both margins.

[7] Chen et al. (2019) and Leshno and Strack (2019) show that proportional selection is the only allocation rule that satisfies a set of desirable properties, e.g. anonymity, collusion-resistance, and sybil-resistance.

size $n_i$ to maximize expected profits $\pi$ given by:

$$\pi\left(n_i; S, \mathcal{H}\right) = \frac{H\left(n_i, S\right)}{\mathcal{H}}\left(Rev + P \cdot br\right) - c_m \cdot n_i - f_m, \tag{3.2}$$

where $H\left(n, S\right)$ is defined in equation (3.1).

A Nash equilibrium requires that equilibrium pool sizes are pinned down by miners' best-responses:

$$n_i^* = \arg\max_n \pi\left(n; S, \mathcal{H}\right) \tag{3.3}$$

The solution to this problem is given by:[8]

$$n_i^* = \left(\frac{\sigma\left(Rev + P \cdot br\right)}{c_m(S/U)\mathcal{H}}\right)^{1/(1-\sigma)}. \tag{3.4}$$

### 3.2.3 Entry and the Equilibrium Number of Miners

Everybody is free to enter the system as a miner. In equilibrium, the number of miners $N$ adjusts to ensure that upon entry, mining pools make zero profits in expectation.

Since mining pools are symmetric, in equilibrium, all active pools employ the same number of workers, $n_i^* \equiv n^*$, and the aggregate hash rate, thus, is given by $\mathcal{H} = NH\left(n^*\right)$. Using equation (3.4), the optimal pool size can be shown to equal:

$$n^* = \frac{\sigma}{c_m} \cdot \frac{Rev + P \cdot br}{N} \tag{3.5}$$

The equilibrium probability that any miner is chosen to mine the next block equals $\frac{H_i}{\mathcal{H}} = \frac{H}{NH} = 1/N$. Substituting equation (3.5) into profits given by equation (3.2) and imposing zero expected profits upon entry, we can show that the equilibrium number of miners equals:

$$N = \frac{(1-\sigma)}{f_m} \cdot \left(Rev + P \cdot br\right) \tag{3.6}$$

The equilibrium number of miners is increasing in mining revenues and decreasing in the entry cost. By virtue of our design, miners that solve predicates of size $S$ and get chosen to write the next block can verify up to $S$ transactions. Since there is entry cost and miners cannot force other miners into a coalition, miners have no incentive to form coalitions with the intent to temper with transaction fees in order to increase

[8] When setting $\frac{\partial \pi}{\partial n_i} = 0$, we assume that miners are small in the sense that they don't internalize the effect their computations have on aggregate hash rate $\mathcal{H}$. Our qualitative results do not crucially depend on this assumption.

profits.[9] Therefore, miners include transactions that offer the highest transaction fees up to the capacity limit.

## 3.3 Demand for Transactions and Determination of Fees

We can now describe the problem of users, which yields the equilibrium level of transaction fees. The demand side of the model closely follows Huberman et al. (2019). Our model is distinct in one key dimension. The adaptive nature of our protocol implementation implies that block capacity $S$ and thus equilibrium system throughput is not a primitive, but rather an equilibrium outcome.

Users have single transactions and are heterogeneous in the cost that verification delay poses for them. New users enter the pool of pending transactions at a constant random rate per time unit and offer transaction fees for service. Knowing that a higher posted fee improves the chances of being included in a block sooner, users' willingness to pay arises out of the opportunity cost of delay.

### 3.3.1 Users

Users arrive at Poisson rate $\lambda$ at each point in time. $\lambda$ parametrizes demand intensity or the rate at which the pool of pending transactions grows over time. An arriving user is identified by a wait cost $c$, which is drawn from a continuous cumulative density function $F(c)$ with domain $C \equiv [0, \bar{c}] \subseteq \mathbb{R}_0^+$ and probability density function $f(c)$. The expected benefit of using the system to a user endowed with wait cost $c$ and offering transaction fee $tf$ is given by:

$$U(tf; W, c) = v - tf - cW(tf; G),$$

where $v > 0$ governs the value of a verified transaction to a user[10] and $W(tf, G)$ is the expected wait time for a user that posts fee $tf$. The wait-time $W$ depends on the (endogenous) distribution of transaction fees posted by all users $G(tf)$ in the system and is thus itself an equilibrium object.[11]

[9] For example, a large coalition may choose not to include transaction fees below a certain threshold.

[10] In practice, users have an outside option and only participate if their utility from doing so exceeds this outside option. Throughout the analysis, we assume that $v$ is sufficiently high so that users are willing to participate.

[11] The assumption is that users know the distribution of wait times $F$ and are assumed to anticipate others' optimal behavior correctly. However, users do not observe the current pool of pending transactions when submitting fee offers.

### 3.3.2 The Dependence of Fees on Wait Time

Each active user posts a fee $tf$ so as to maximize utility, weighing the cost of posting higher fees against the benefit of lowering the expected wait time. Standard arguments imply that equilibrium transaction fees are a continuous, monotonous function of user wait-cost $tf(c)$ satisfying $tf(0) = 0$ and $tf'(c) > 0$. Monotonicity implies that $G(tf(c)) = F(c)$. Equilibrium transaction fees $tf(c)$ solve the first order condition $W'(tf|G) = -\frac{1}{c}$ which is equivalent to the following differential equation:

$$\tilde{W}'(c|F)\, cf(c) = tf'(c),$$

where $\tilde{W}$ directly maps wait cost into wait times and satisfies $-W'(tf|G) = W'(c|F)$.[12] Integrating and using the property that users with no delay cost post zero fees, transaction fees can be fully expressed in terms of equilibrium wait times:

$$tf(c) = \int_0^c \tilde{c} f(\tilde{c}) \tilde{W}'(\tilde{c}|F)\, d\tilde{c}. \tag{3.7}$$

Inspecting equation (3.7) reveals that equilibrium fees have the externality correcting property inherent to the Vickrey-Groves-Clark (VCG) auction mechanism.[13] For our purposes, the fact that offered transaction fees are socially optimal implies that none of the potential cost-efficiencies exhibited by the decentralized equilibrium arise from suboptimal user behavior.

### 3.3.3 Equilibrium Wait Time

Blocks can process up to $S$ transactions per time period and arrive randomly at a Poisson rate $\mu$. The equilibrium wait time naturally depends on system congestion, which in our context is defined as $\rho \equiv \lambda/S\mu$ - the average ratio of demand $\lambda$ to the average number of transactions that can be processed per time unit. $\rho < 1$ implies that while users may experience delays, all transactions will eventually be processed by the system.

To characterize the equilibrium wait time, we build on results derived in Huberman et al. (2019). The authors show that for sufficiently high levels of block capacity, wait times are fully characterized by system congestion $\rho$.[14] We restate this result in the

[12] Note that $G(tf(c)) = F(c)$ implies $G'(b(c))b'(c) = f(c)$. Thus $W'(tf|G) = W'(tf|G)f(c)/b'(c)$.

[13] An externality arises as each user delays the verification of the transactions of other users. Despite the fact that users do not internalize the cost that they pose on others, the VCG auction mechanism incentivizes "truthful bidding," implying that the users with higher valuations submit higher bids.

[14] Huberman et al. (2019) show wait times are appropriately approximated for $S = 20$. Block capacity

following theorem.

THEOREM 1 *1. For any $\rho \equiv \lambda/(\mu S) \in (0,1)$ the equilibrium wait time for a user with delay cost $c$, $W(c; S, \mu, \lambda)$ is given by:*

$$W(c;\rho) = \frac{\mu^{-1}}{1-(1+\alpha(\rho\bar{F}(c)))\,e^{-\alpha(\rho\bar{F}(c))}}, \tag{3.8}$$

*where $\bar{F}(c) \equiv 1 - F(c)$ and $\alpha(x)$ is the real root of the algebraic equation $e^{-\alpha} + x\alpha - 1 = 0$.*

*2. Equilibrium wait time is increasing and convex in congestion $\rho$ and decreasing in user wait cost $c$.*

*Proof.* See Lemma 14 in Huberman et al. (2019). □

The relationship between equilibrium wait times and optimal bidding behavior implies that users with higher wait cost offer higher fees, and wait less for transaction verification. Higher levels of system congestion increase wait times - and, therefore, transaction fees - for all users. Theorem 1 implies that wait times - and, therefore, transaction fees - are fully characterized by system congestion $\rho$,[15] implying that equilibrium stability in fees is equivalent to stability in system congestion $\rho$.

### 3.3.4 Equilibrium Transaction Fees

To summarize the optimal bidding behavior across all users, Theorem 1 and equation (3.7) allow to derive the average equilibrium fee revenue *Rev* per unit of time, given block capacity $S$ and system congestion $\rho$:

$$\begin{aligned} Rev(S,\rho) &= S \cdot \int_0^{\bar{c}} t f(c) dF(c) \\ &= S \cdot \rho \int_0^{\bar{c}} (\bar{F}(c) - c f(c))\, W(c;\rho)\, dc \equiv S \cdot \psi(\rho). \end{aligned} \tag{3.9}$$

in Bitcoin is at around 2000. As outlined earlier, we expect our proposed protocol implementation to be capable of handling a volume of transactions per block that outperforms Bitcoin by magnitudes. We, therefore, feel comfortable to use the characterization of wait times in equation (3.8) for the remainder of the analysis.

[15] In principle, wait-times depend on block capacity $S$ in addition to congestion. On the one hand for a given transaction, the inclusion in a block depends on how many pending transactions have accumulated at the time block is generated. This depends on system congestion $\rho$. On the other hand, the inclusion of the transaction might also depend on new transactions with higher priority that might arrive before the next block. The arrival of higher priority users creates significant randomness and thus affects wait times only if blocks have sufficiently small capacities such that new arriving transactions can significantly change the composition of included transactions. For sufficiently high $S$, however, wait times are nearly independent of this source of variation.

$\psi(\rho)$ in equation (3.9) is the average level of transaction fees per unit of time that users are willing to pay for service.[16] As *Rev* summarizes the optimal bidding behavior of users, demand side effects stemming from changes in either block capacity $S$ or demand $\lambda$ are fully captured by its properties.

We characterize the system's ability to induce increases in throughput so as to absorb an increase in demand $\lambda$ at constant average prices. The responsiveness of average fees $\psi(\rho)$ with respect to changes in demand $\lambda$ - or generally changes in congestion $\rho$ - is key for this purpose. We define the elasticity of $\psi(\rho)$ in equation (3.9) with respect to $\rho$ as $\varepsilon(\rho) \equiv \frac{\partial \log \psi(\rho)}{\partial \log \rho} \equiv \frac{\rho}{\psi(\rho)} \frac{\partial \psi(\rho)}{\partial \rho}$. $\varepsilon(\rho)$ captures the percentage increase in equilibrium fees in response to a percentage increase in congestion $\rho$. The following theorem characterizes key properties of $\varepsilon(\rho)$ and in particular shows that it admits a uniform upper bound, independently of the underlying distribution of wait cost $F(c)$. The proof is delegated to the appendix.

THEOREM 2 *The elasticity of equilibrium average transaction fees $\psi(\rho)$ with respect to system congestion $\rho$, $\varepsilon(\rho) \equiv \frac{\partial \log \psi(\rho)}{\partial \log \rho}$, is bounded below by 1, increasing and convex in $\rho$. Further, there exists $\overline{\varepsilon}(\rho)$ such that $\varepsilon(\rho) \leq \overline{\varepsilon}(\rho)$ for all $\rho \in (0,1)$ and any distribution of user wait cost $F(c)$.*

Figure 2 plots the uniform upper bound on the elasticity of average fees with respect to changes in congestion derived in Theorem 2.

## 3.4 Equilibrium for a Fixed Predicate Size $S$

### 3.4.1 Equilibrium Number of Miners

A partial equilibrium for a given predicate size $S$ and demand level $\lambda$ is defined by the condition that miners make zero profits in expectation upon entry.[17] Using equation (3.6) and the characterization of total mining fee revenues, the equilibrium number of miners $N$ at predicate size $S$, demand $\lambda$ and nominal block rewards $P \cdot br$ is given by:

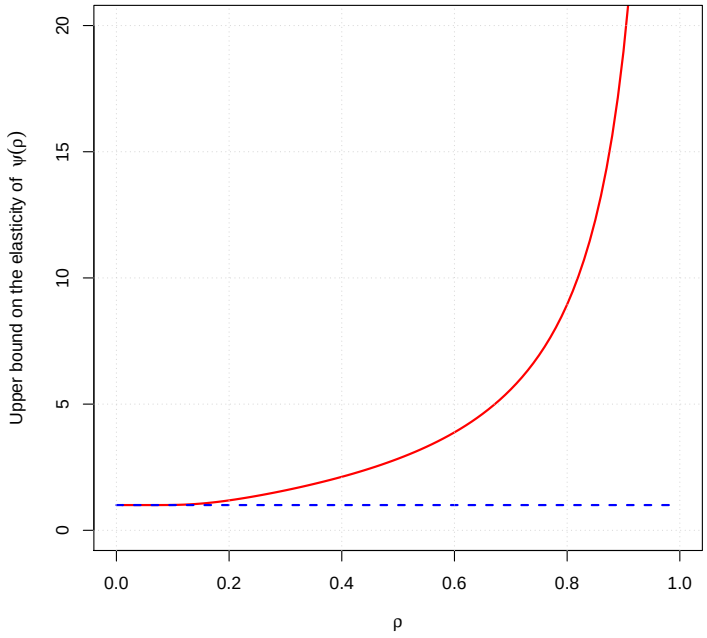
$$N(S, \lambda, P \cdot br) = \frac{1-\sigma}{f_m} \cdot (S\psi(\rho) + P \cdot br). \tag{3.10}$$

Predicate size $S$, demand $\lambda$ and nominal block rewards $P \cdot br$ summarize the profit incentives of miners. Adjustments in predicate size $S$, therefore, change the profit

[16] As we characterize steady states in a fee market where there is a constant flow of transactions in and out of a system that eventually verifies all transactions and where all users participate, the integral is taken over the entire domain of $c$.

[17] While this equilibrium is technically a steady-state - owing to the continuous entrance and exit of users - we effectively treat it as a static equilibrium condition.

Figure 2 Upper Bound on the Congestion Elasticity of Equilibrium Average Fees in the VCG Auction




Notes: This figure plots the uniform bound on the elasticity of equilibrium average fess given in equation (3.9) with respect to system congestion $\rho = \lambda/\mu S$ as a function of $\rho$ on the x-axis.

incentives and entry behavior of miners. Cost-efficiency in a decentralized equilibrium crucially depends on whether adjustments of predicate size $S$ can be optimally calibrated to induce incentive-compatible equilibrium behavior of miners that is *as if* miners internalized the externality that they pose on others through Nakamoto consensus. In order to build up an intuition for our later results, it is useful to briefly pause to discuss how adjustments in predicate size $S$ shape profit incentives of miners through the condition in equation (3.10).

An increase in demand $\lambda$ initially increases mining fee rewards through an increase in equilibrium user fees $\psi(\rho)$. This increases entry, the equilibrium number of miners, and the computational cost of the system without adding benefits for system capacity or users. To restore cost-efficiency, a subsequent increase in predicate size $S$ has to direct additional cost incurred by the system toward increases in throughput. Our results imply that competition by miners ensures that this is achieved by additional entry and lower success probabilities for each active pool.

### 3.4.2 Difficulty

As opposed to for example Bitcoin, difficulty in our model depends only on the total number of proofs computed per time unit.[18] As pools are symmetric, solely distribute one proof computation among pool participants and operate at hash rate $H$, system difficulty is given by:

$$d = \log(\mu \times H \times N). \tag{3.11}$$

By using the equilibrium expression for the number of miners and the solution to the miner's problem, we can express difficulty as:[19]

$$d = \log\left(\alpha \times \frac{S\psi(\rho) + P \cdot br}{S}\right), \tag{3.12}$$

where $\alpha \equiv \mu U \left(\frac{\sigma}{c_m}\right)^{\sigma} \left(\frac{1-\sigma}{f_m}\right)^{1-\sigma}$ is a constant.

The inability of miners to autonomously adjust cost *and* throughput upwards as demand rises causes rising levels of system congestion, resulting in high wait times and fees for users, and costly increases in mining activity at no increase in throughput as demand rises. In light of this significant economic limitation, our protocol design's core innovation is to provide an "updating rule" for predicate size $S$ and, therefore, block capacity that correctly infers changes in demand. While demand $\lambda$ is not directly observable, equation (3.12) highlights that difficulty is a sufficient statistic for changes in demand when block rewards are zero. In this case, changes in equilibrium difficulty are proportional to changes in average transaction fees, which in turn depend solely on the congestion parameter $\rho$.

## 3.5 Dynamic Throughput Adjustment

We endow the system with the ability to periodically adapt throughput $S$ according to a pre-specified updating rule.

Definition 1 *Let* $\mathbf{X}$ *be a vector of statistics. A law of motion for throughput* $S$ *is a function* $f$ *such that:*

$$S_{t+1} = f(S_t, S_{t-1}, ...; \mathbf{X}_t, \mathbf{X}_{t-1}, ...). \tag{3.13}$$

[18] This is because the solution of a single proof commits to the nonce before distributing computation to the pool. This means that, unlike the distribution of double-SHA computations in Bitcoin (that effectively break up nonce generation among pool participants), here the nonce is set by the operator, and all computation forwarded to workers is nonce-specific.

[19] Note that the solution to the miners' problem implies that equilibrium hash rates are given by $U\left(\frac{\sigma f_m}{(1-\sigma)c_m}\right)^{\sigma} / S$.

Given an updating rule for predicate size $S$, we can define a dynamic equilibrium and the notion of a steady-state.

Definition 2 *Given a starting predicate size $S_0$, level of demand $\lambda$ and nominal block rewards $P \cdot br$, an equilibrium is a sequence of predicate sizes $\{S_t\}_{t=1,2,...}$ such that at each $t$, (i) transaction fees posted by users satisfy equation (3.9), (ii) the equilibrium number of miners $N_t$ satisfies equation (3.6) and (iii) the law of motion for $S$ is given by equation (3.13). A steady state is reached when changes in predicate size $S$ converge to 0.*

So far, we have provided a framework to analyze how the optimal behavior of miners and users determines fees and the energy usage of the system. In the following, we utilize this framework to analyze whether chosen laws of motion for throughput can replicate equilibrium outcomes of standard competitive markets where firms are able to autonomously adjust production in accordance with demand.[20]

# 4 Stabilizing Congestion

Unlike standard markets, the level of supply cannot be directly chosen by miners, but is set by the protocol through the predicate size $S$. Absent changes in $S$, increases in demand raise congestion in the system causing exorbitant cost at little benefit to users. In this section we show that suitable updates of throughput help overcome this limitation. Specifically, as equilibrium difficulty encodes changes in demand, we show that updates of predicate size $S$ in accordance with changes in difficulty allow to adjust throughput to changes in demand. This section outlines why stable levels of congestion are desirable, and shows by way of simple examples how our framework allows us to analyze the viability of various update rules.

## 4.1 Characterizing Efficient Outcomes

Here, we briefly outline why stable levels of congestion are desirable from a welfare perspective. In equilibrium, the net benefit that the system produces can be characterized in terms of value that transactions generate across all consumers, wait times as well as the cost of energy used by pools:

$$\nu\lambda - \text{Average Wait-time}\,(\lambda, S) - \text{Total Energy Cost}\,(S)\,.$$

[20] Definition 2 defines an equilibrium taking both block rewards $Pbr$ and demand $\lambda$ as given. The underlying assumption is therefore that adjustments in predicate size $S$ occur sufficiently frequently - or equivalently that the length of periods $dt$ is sufficiently small - for the system to reach steady states between changes in demand or block rewards.

Free entry implies that the total energy cost of the system must equal mining rewards. Abstracting from block rewards, total mining revenues increase at least proportionally to system congestion, as do average wait times. To ensure that the net benefits of the system do not decrease as demand $\lambda$ increases, changes in demand $\lambda$ should be offset by proportional changes in $S$. It is easy to see that this would imply that the benefit of the system would remain invariant to how many people use it.

In conclusion, an ideal adjustment of predicate size $S$ to changes in difficulty keeps wait times and congestion $\rho = \lambda/(\mu S)$ constant, allowing for proportional growth in demand and supply.

## 4.2 Floating Difficulty Level

Consider a policy rule that upon observing a change in difficulty between perdiods $t-1$ and $t$, updates predicate size $S$ in $t+1$ as follows:

$$\log\left(S_{t+1}/S_t\right) = \gamma\left(d_t - d_{t-1}\right). \tag{4.1}$$

We focus on analyzing transitions between equilibrium steady states in response to changes in demand when block rewards are zero. We analyze constraints on the updating parameter $\gamma$ stemming from two objectives. First, $\gamma$ should ensure convergence of the system to a new steady state. Second, economic efficiency requires that changes in demand yield minimal changes in congestion.

Assume the system is in steady state with predicate size $S_0$ and congestion $\rho^* = \lambda_0/\mu S_0$. We consider a change in demand $d\log\lambda \equiv \log(\lambda_1) - \log(\lambda_0)$ from $\lambda_0$ to $\lambda_1$. The object of interest is the elasticity of steady state system congestion with respect to this demand shock, $\frac{d\log\rho^*}{d\log\lambda}$. The following proposition summarizes the properties of this object.

Proposition 1 *Consider an initial steady state $\{S_0, \lambda_0\}$ and assume that block rewards are equal to zero. Under a floating difficulty regime, upon a change in demand $d\log\lambda \equiv \log(\lambda_1) - \log(\lambda_0)$,*

1. *the system converges to a new steady state if $\gamma < \frac{1}{\overline{\varepsilon}(\overline{\rho})}$, where $\overline{\varepsilon}(\rho)$ denotes the upper bound on $\varepsilon(\rho)$ derived in Theorem 2 and $\overline{\rho} \equiv \max\{\lambda_0/\mu S_0, \lambda_1/\mu S_0\}$,*
2. *if $\gamma \in \left(0, \frac{1}{\overline{\varepsilon}(\overline{\rho})}\right)$, then steady state changes in system congestion are strictly less than proportional to changes in demand: $\frac{d\log\rho^*}{d\log\lambda} \in \left(\frac{1}{2}, 1\right)$.*

*Proof.* Using the recursive structure of the system, the elasticity of steady state congestion with respect to a demand shock is:

$$\frac{d \log \rho^*}{d \log \lambda} = 1 + \sum_{t=1}^{\infty} (-\gamma)^t \left( \prod_{k=0}^{t} \varepsilon (\rho_t) \right).$$

Using Theorem 2, the proposition follows from standard properties of geometric sums. □

The first part of Proposition 1 shows that dynamic stability imposes a feasibility restriction on the rate at which predicate size (and thus throughput) can be increased in response to some demand shock. For $\gamma > 1/\overline{\varepsilon}(\overline{\rho})$, changes in predicate size would be large enough at every step to ensure that the system diverges from equilibrium. Thus, the value of $\gamma$ needs to be low enough so that this is prevented. Note that this imposes a new trade-off in efficiency: a lower $\gamma$ ensures that higher demand shock levels can be 'absorbed' without instability, but also means that the system will take longer to arrive at equilibrium (as predicate updates are less sensitive to changes in difficulty).

The second part of the above result demonstrates that such a record keeping system with $\gamma > 0$ does strictly better than one where $S$ is fixed. This is due to the upper bound on the elasticity of steady state system congestion, which shows that congestion will increase at a fraction of the non-adaptive case. Although they will increase comparatively less, it is immediate that transaction fees remain unbounded: an arbitrary sequence of positive demand shocks will always increase difficulty arbitrarily high, pushing up equilibrium fees as well. Further, to ensure dynamic stability at possibly high levels of congestion requires sufficiently low $\gamma$, which impedes the speed of convergence. In fact, the only choice for $\gamma$ that ensures dynamic stability under a floating difficulty regime as $\lambda \to \infty$ is $\gamma = 0$.

In the traditional Bitcoin protocol, an unbounded difficulty parameter is required due to the security benefits that it provides. This is by making the cost of disrupting the network scale with $d$. Note, however, that an increase in difficulty here is not necessarily required for system security: the costs of participation are increasing with respect to both difficulty *and* predicate size. This means that 'difficulty' in the sense of Bitcoin is actually better understood to be an increasing function $g(S, d)$ here, as both of these parameters are monotonically increasing with respect to costs per unit time. This motivates the question of whether *constant* equilibrium transaction fees can be achieved without compromising system security.

## 4.3 Pegged Difficulty Level

Consider a policy rule that updates predicate size $S$ as follows:

$$\log \frac{S_{t+1}}{S_t} = \gamma \left( d_t - d^* \right), \tag{4.2}$$

where $d^*$ is a *pre-specified* level of difficulty.

Again, consider a demand shock $d \log \lambda$. The following proposition draws out conditions on the update parameter $\gamma$ under which the system maintains a stable steady state level of congestion $\rho$ under zero block rewards.

Proposition 2 *Assume that block rewards are equal to zero, and difficulty is pegged to $d^*$. Denote $\bar{\rho}^* \equiv \psi^{-1}(\exp(d^*/\alpha))$ with $\alpha$ given in equation (3.12). Under a pegged difficulty regime, upon a change in demand $d \log \lambda \equiv \log(\lambda_1) - \log(\lambda_0)$ yielding a change from $\rho^*$ to $\rho$, $\frac{d \log \rho^*}{d \log \lambda} = 0$ for any $\gamma \in \left(0, \frac{2}{\bar{\varepsilon}(\overline{\rho})}\right)$, where $\overline{\rho} \equiv \max\{\rho^*, \rho\}$.*

*Proof.* Under a pegged difficulty regime, equilibrium elasticity of congestion with respect to a demand shock is:

$$\frac{d \log \rho^*}{d \log \lambda} = \varepsilon(\rho_0) \prod_{1=1}^{\infty} \left(1 - \gamma \varepsilon(\rho_t)\right).$$

The result follows from Theorem 2 and the above expression. □

Rule 4.2 allows for the *a priori* selection of some acceptable congestion level $\rho^*$ by fixing (or 'hardcoding') the corresponding $d^*$ in the update rule. In this case, the system will increase predicate size $S$ accordingly until system difficulty *drops* back to $d^*$. Therefore, the long-term equilibrium effect on congestion is zero. Such a system has the desirable property that transaction fees at equilibrium will always revert to constant: the effects of any demand shock will be fully compensated by changes in predicate size, which will revert transaction fees to their previous equilibrium level.

Technically, the crucial change that permits such desirable behavior lies in the 'decoupling' between the sufficient statistics for system security and demand in the above update rule. Indeed, the costs of participation rise monotonically with $S$, while costs of transactions are only functions of $d$. Thus, by ensuring that the system will always update $S$ as a reaction to changes in demand so as to fully compensate for any changes in $d$, we are able to keep equilibrium transaction fees constant *regardless* of the demand shock as long as $\gamma$ is low enough.

### 4.4 Positive Block Rewards

We conclude our analysis by briefly discussing how positive block rewards, $br > 0$, affect our key results. In the presence of positive nominal block rewards, miners' profit incentives are not fully determined by mining fees and therefore entry and changes in difficulty only are an imperfect measure of demand. Consequently, changes in demand cannot be fully absorbed through proportional scaling of throughput. Further, changes in nominal block rewards - either due to an increase in $P$ or a (scheduled) decrease in block rewards $br$ - will cause changes in throughput that are (potentially) detached from fundamental user demand. While nominal block rewards serve as a subsidy to sustain mining incentives at low levels of initial demand, here they pose a cost in terms of economic efficiency.

## 5 Towards an Instantiation

The feasibility of the construction in the preceding sections depends heavily on the properties of the underlying proof system utilized, as well as on the fulfillment of technical requirements for the underlying PoW scheme. In this section, we (1) outline the engineering ingredients essential to the efficient realization of such a system, (2) illustrate that the economic assumptions made are consistent with empirical observations, and (3) identify potential design challenges.

### 5.1 Proof System Choice

Research into the design and implementation of proof systems relevant for use in the above context has seen rapid advancements in recent years. Current implementations of Succinct Non-interactive ARguments of Knowledge (SNARKs) (see Ben-Sasson et al. (2013); Groth (2016); Parno et al. (2013) for definitions) provide extremely fast proof verification times and small proof sizes, seeing active use in blockchain protocols (see Sasson et al. (2014) for an example). Of such systems, the main relevant properties are:

1. Succinctness: Proofs are small enough to add to the blockchain.
2. Trustlessness: Requires no additional trust assumptions.
3. Variability: Easy to produce/verify many different predicates.
4. Recursion: Proofs can efficiently verify previous proofs.

Such restrictions have been the focus of active research in proof system design. Proof systems with the property of succinctness, or proof sizes that don't vary with respect to predicate size, generate proofs that can be efficiently computed and are small enough (< 1 kB) in state-of-the-art implementations such as Groth (2016). However, the majority of such systems are either reliant on cryptographic assumptions that are known to be falsifiable in the quantum setting, or come with changes in the trust model such as the need for an initial setup by a trusted third-party, thus violating notions of trustlessness. Moreover, such implementations do not provide the flexibility needed by the above model to adaptively change the number of transactions being verified in each block - doing so here comes with the corresponding prohibitive cost of rerunning the trusted setup procedure each time we update the size of $S$. This is because in these constructions the creation/validation of proofs for a given predicate require the initial generation of a predicate-specific Structured Reference String (SRS) that is required in all subsequent computation. Thus, the notion of variability is also relevant for deployable prototypes. We should also note that many current pairing-based implementations also have a 'maximal' predicate size that they can effectively compute, based on the 2-adicity of the underlying elliptic curve. Looking at ways to raise this bound would also be important in attaining even higher predicate sizes.

Recent work has looked into resolving the above challenges while maintaining efficiency. SNARK constructions that are based on a *universal* SRS have been developed Groth et al. (2018); Maller et al. (2019) in which the SRS generation procedure needs to be done only once for all potential predicates. This means that one SRS suffices to generate proofs for any number of transactions (i.e. any predicate size). Other designs with different trade-offs have also been developed (for more information, see Ben-Sasson et al. (2018); Bowe et al. (2019); Chiesa et al. (2019a,b); Gabizon et al. (2019)).

Indeed, the 'ideal' proof system should be (1) trustless (rely on no trusted setup procedure), (2) plausibly quantum-resistant, (3) support proofs for multiple predicates through a universal SRS, and (4) support recursive proof composition - or the ability to verify previous proofs efficiently. The first three technological requirements together would be required for an effective instantiation of the above protocol in the quantum setting, while (4) would cut the bandwidth and computational costs needed by new end users to enter the system to negligible.

## 5.2 Integrating Proof of Work

The core of the proposed protocol relies on a PoW procedure that generates a proof of correctness for a set number of transactions that are verified (the 'predicate size')

in each block. For the proof generation procedure to act as a suitable PoW function, certain restrictive properties need to be fulfilled. This is achieved for a simple payments protocol in Kattis and Bonneau (2020), built on top of a recursive proof system. Since the ability to use proof generation as a suitable PoW puzzle is essential for the present proposal, techniques to generate PoW-suitable predicates and/or proof systems is an immediate area of interest for future work. Moreover, assessing whether the approach in Kattis and Bonneau (2020) can be easily extended to proof systems based on different underlying assumptions (such as proof systems that support universal SRS generation) is a natural next question, as is assessing whether this approach can also be used in more complex predicates that support a wider array of features (such as arbitrary code execution/smart contracts).

## 5.3 Distributing Proof Computation

Since the proof generation techniques that can be potentially used in the current context require substantial computational resources, in the absence of hardware-acceleration techniques this is only feasible if proof generation can be adequately distributed among workers. This approach is similar to the 'mining pool' model in current PoW-based cryptocurrencies, wherein a mining pool operator (the "miner") distributes part of the PoW computation to workers who, in turn, are guaranteed a percentage of returns in exchange for working for the specific pool. Such an arrangement can provide workers with the ability to receive less income albeit more frequently, and thus has the benefit of distributing large PoW computational burdens among many small participants.

In the context of proof generation, the work of Wu et al. (2018) provides the first framework for SNARK proof distribution among many nodes. Since the guarantees of the proposed protocol depend heavily on the proof generation cost function, empirical performance of SNARK generation in a distributed setting is crucial to assessing the accuracy of the present model. Indeed, the fundamental assumption on the functional form of the production function in this model is directly observed in the empirical results of Wu et al. (2018), indicating that this model is representative of distributed proof production dynamics, at least when using Groth (2016). Moreover, these results suggest a potential $\sigma$ value of $\sigma \approx 0.88$.

Furthermore, distributed proving techniques for proof systems with desirable properties need to also be developed. More specifically, techniques such as those employed in Wu et al. (2018) should be applied to other proof systems that support additional features, such as trustlessness or universal SRS generation. Finally, in order to fully emulate the threat model inherent to current mining pools, an effective solution needs

to be devised to ensure that mining pool participants cannot submit invalid computation to the pool operators. In the present model this would require the workers to also provide a proof-of-correctness for their individual computation to the operator that is both cheap to verify (comparatively to performing the computation itself) and efficient to perform. Although this is not necessarily required in the context of trusted pools, it would be an important feature for any pool open to a public set of (potentially adversarial) workers.

### 5.4 Data Availability Considerations

An important issue that is relevant in the present discussion is that of data availability. Since miners are providing proofs of correctness, in order for participants to update their state accordingly the transactions that resulted in the updated blocks need to also be released. If these are added to the propagated blocks, this becomes a bottleneck in the potential throughput of such a system since block sizes cannot exceed certain levels ($\sim$ 1 MB). This directly corresponds to the $S_{max}$ maximum predicate size. A promising line of work in resolving this problem uses Coded Merkle Trees Yu et al. (2019), which can be used to reconstruct the relevant transaction data even in the context of a single honest node. By substantially raising the value of $S_{max}$, the system can scale to much higher levels.

## 6 Conclusion

Drawing on recent developments in the literature aimed at optimizing the scaling of distributed payment systems, we have provided a protocol specification that addresses economic cost efficiencies from correctness verification, PoW security guarantees and throughput adaptability. We have shown how to link throughput to an observable system statistic that adequately reflects changes in user demand and have demonstrated by way of an economic analysis how to autonomously regulate incentives of system participants so as to ensure cost-efficient scaling to user demand. We believe that an implementation of this approach is feasible given the current state of knowledge, provided that engineering efforts yield the necessary tools that our theoretical analysis highlights.

# References


Joseph Abadi and Markus Brunnermeier. Blockchain Economics. NBER Working Papers 25407, National Bureau of Economic Research, Inc, December 2018. URL https://ideas.repec.org/p/nbr/nberwo/25407.html.

Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of useful work. *IACR Cryptology ePrint Archive*, 2017:203, 2017.

Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Annual cryptology conference*, pages 90–108. Springer, 2013.

Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021, 2019. https://eprint.iacr.org/2019/1021.

Eric Budish. The Economic Limits of Bitcoin and the Blockchain. Working paper, 2018. URL http://faculty.chicagobooth.edu/eric.budish/research/Economic-Limits-Bitcoin-Blockchain.pdf.

Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2014. URL https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 2016-08-22.

Vitalik Buterin. *On-chain scaling to potentially 500 tx/sec throughmass tx validation.*, 2018 (accessed June 12, 2020). URL https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477.

Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016.

Xi Chen, Christos Papadimitriou, and Tim Roughgarden. An axiomatic approach to block rewards. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT âĂŹ19, page 124âĂŞ131, New York, NY, USA, 2019. Association

for Computing Machinery. ISBN 9781450367325. doi: 10.1145/3318041.3355470. URL https://doi.org/10.1145/3318041.3355470.

Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Report 2019/1047, 2019a. https://eprint.iacr.org/2019/1047.

Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. Cryptology ePrint Archive, Report 2019/1076, 2019b. https://eprint.iacr.org/2019/1076.

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

Peter Gaži, Aggelos Kiayias, and Dionysis Zindros. Proof-of-stake sidechains. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 139–156. IEEE, 2019.

Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 305–326. Springer, 2016.

Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. Cryptology ePrint Archive, Report 2018/280, 2018. https://eprint.iacr.org/2018/280.

Gur Huberman, Jacob Leshno, and Ciamac C. Moallemi. An economic analysis of the Bitcoin Payment System. Columbia Business School Research Papers 17-92, Columbia Business School, 2019. URL https://ideas.repec.org/p/cpr/ceprdp/13506.html.

Assimakis Kattis and Joseph Bonneau. Proof of necessary work: Succinct state verification with fairness guarantees. Cryptology ePrint Archive, Report 2020/190, 2020. https://eprint.iacr.org/2020/190.

Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. Mining for privacy: How to bootstrap a snarky blockchain. Cryptology ePrint Archive, Report 2020/401, 2020. https://eprint.iacr.org/2020/401.

Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, 1(6), 2013.

Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Conference on the Theory and Application of Cryptography*, pages 353–365. Springer, 1990.

Jacob Leshno and Philipp Strack. Bitcoin: An Impossibility Theorem for Proof-of-Work based Protocols. Cowles Foundation Discussion Papers 2204R, Cowles Foundation for Research in Economics, Yale University, October 2019. URL https://ideas.repec.org/p/cwl/cwldpp/2204r.html.

Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Report 2019/099, 2019. https://eprint.iacr.org/2019/099.

Izaak Meckler and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. *O (1) Labs whitepaper. May*, 10:4, 2018.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. http://bitcoin.org/bitcoin.pdf.

Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. Cryptology ePrint Archive, Report 2013/279, 2013. https://eprint.iacr.org/2013/279.

Charles Rackoff and Daniel R Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Annual International Cryptology Conference*, pages 433–444. Springer, 1991.

Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

Marcel Thum. The Economic Cost of Bitcoin Mining. *CESifo Forum*, 19(1):43–45, March 2018. URL https://ideas.repec.org/a/ces/ifofor/v19y2018i1p43-45.html.

Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. {DIZK}: A distributed zero knowledge proof system. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 675–692, 2018.

Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains. Cryptology ePrint Archive, Report 2019/1139, 2019. https://eprint.iacr.org/2019/1139.

# A Proof of Theorem 2

*Proof.* Let $\psi(\rho) = \rho \int_0^{\bar{c}} (\bar{F}(c) - cf(c)) W(c,\rho)\, dc$. First, the derivative of the wait time is equal to:

$$\frac{\partial W(\rho)}{\partial \rho} = e^{-\alpha(\rho)} \alpha(\rho)^3 W(\rho)^3.$$

Therefore, the elasticity is given by:

$$\frac{\rho}{W(\rho)} \frac{\partial W(\rho)}{\partial \rho} = \rho e^{-\alpha(\rho)} \alpha(\rho)^3 W(\rho)^2. \tag{A.1}$$

Evidently, this elasticity is increasing in $\rho$. Thus, the elasticity of average transaction fees is given by:

$$\frac{d \log \psi(\rho)}{d \log \rho} = 1 + \rho \int_0^{\bar{c}} \omega_c \bar{F}(c) e^{-\alpha(\rho \bar{F}(c))} \alpha(\rho \bar{F}(c))^3 W(\rho \bar{F}(c))^2 dc,$$

where $\omega_c \equiv \frac{\bar{F}(c) - cf(c) W(c,\rho)}{\int_0^{\bar{c}} (\bar{F}(c) - cf(c)) W(c,\rho) dc}$. Given that $\alpha(\rho) \to \infty$ as $\rho \to 0$, a uniform bound on this elasticity is given by:

$$\frac{d \log \psi(\rho)}{d \log \rho} \leq 1 + \rho e^{-\alpha(\rho)} \alpha(\rho)^3 W(\rho)^2 \equiv \overline{\varepsilon}(\rho).$$

□