Minotaur: Multi-Resource Blockchain Consensus

Matthias Fitzi*
IOG
matthias.fitzi@iohk.io

Xuechao Wang*
University of Illinois
Urbana-Champaign
xuechao2@illinois.edu

Sreeram Kannan
University of Washington, Seattle
ksreeram@uw.edu

Aggelos Kiayias
University of Edinburgh & IOG
aggelos.kiayias@ed.ac.uk

Nikos Leonardos
National and Kapodistrian University
of Athens
nikos.leonardos@gmail.com

Pramod Viswanath
Princeton University
pramodv@princeton.edu

Gerui Wang
Beijing Academy of Blockchain and
Edge Computing
wanggerui@baec.org.cn

ABSTRACT

Resource-based consensus is the backbone of permissionless distributed ledger systems. The security of such protocols relies fundamentally on the level of resources actively engaged in the system. The variety of different resources (and related proof protocols, some times referred to as PoX in the literature) raises the fundamental question whether it is possible to utilize many of them in tandem and build *multi-resource* consensus protocols. The challenge in combining different resources is to achieve *fungibility* between them, in the sense that security would hold as long as the *cumulative* adversarial power across all resources is bounded.

In this work, we put forth Minotaur, a multi-resource blockchain consensus protocol that combines proof of work (PoW) and proof-of-stake (PoS), and we prove it *optimally* fungible. At the core of our design, Minotaur operates in epochs while continuously sampling the active computational power to provide a fair exchange between the two resources, work and stake. Further, we demonstrate the ability of Minotaur to handle a higher degree of work fluctuation as compared to the Bitcoin blockchain; we also generalize Minotaur to any number of resources.

We demonstrate the simplicity of Minotaur via implementing a full stack client in Rust (available open source [24]). We use the client to test the robustness of Minotaur to variable mining power and combined work/stake attacks and demonstrate concrete empirical evidence towards the suitability of Minotaur to serve as the consensus layer of a real-world blockchain.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**.

*The first two authors contributed equally to this work. For correspondence on the paper, please contact Matthias Fitzi at matthias.fitzi@iohk.io.

This work is licensed under a Creative Commons Attribution International 4.0 License.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3559356

KEYWORDS

proof-of-work, proof-of-stake, resource-based blockchain, security analysis

ACM Reference Format:
Matthias Fitzi, Xuechao Wang, Sreeram Kannan, Aggelos Kiayias, Nikos Leonardos, Pramod Viswanath, and Gerui Wang. 2022. Minotaur: Multi-Resource Blockchain Consensus . In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3559356

1 INTRODUCTION

Resource-based Consensus. The fundamental feature of the decentralized computing paradigm of permissionless blockchains is that participation in the consensus protocol is enabled by proving possession of a resource. Bitcoin pioneered this idea based on proof-of-work (PoW), and its implied energy wastage inspired new designs based on alternative resources such as proof-of-stake (PoS), proof-of-space (PoSp) and proof-of-elapsed-time (PoD). These different resources cover the variegated and multidimensional forms of "capital" that the participants bring. Each proof-of-X mechanism is interesting on their merit in appropriate settings (computation in PoW, memory and storage in PoSp and time/delay in PoD, tokens/capital in PoS). Further, the mechanisms trigger different incentives even within the same resource format; focusing on PoW for example, we see that the hashcash algorithm [2] instantiated by the SHA256 function [28] implemented in Bitcoin has been replaced by others, including scrypt [32] and ethash [12]. Each of these different proof-of-X mechanisms have led to different, and isolated, blockchain ecosystems.

Combining different resources. Given the diversity of incentivization embodied by different resources, it is a natural question whether it is feasible to combine their features into a single blockchain design (e.g., hybrid PoW-PoS permissionless blockchains). The key challenge to combining multiple resources is in determining the *exchange rate*, i.e., to what extent the different resources are translated into power of authority in the system and to extract security from these resources in an optimal manner. Adapting the

exchange rate dynamically to the participation levels in each resource type, resulting in a truly *fungible* notion of resource, is a basic and fundamental goal. By truly fungible, we mean that the security of the multi-resource consensus protocol is guaranteed as long as the honest players control a majority of the combined resources that consist of each resource type in the system — concretely, $\sum_{i=1}^{M} \beta_i < M/2$ where M is the number of resources and β_i the adversarial power in the i-th resource (this property is appropriately generalized to any linear combination in Definition 2). We note that achieving this type of fungibility is beneficial for the security of the underlying blockchain system as the adversary will be unable to launch a successful attack by commanding merely 51% in one of the underlying resources. Next we give some examples of *non-fungible* hybrid PoW/PoS protocol designs.

First order hybrid PoW/PoS protocols. Given the basic importance of incorporating multiple resources in a single blockchain design, there are several designs of hybrid PoW/PoS protocols in the literature [4, 10, 11, 19, 22]. These protocols, however, are either heuristic (lacking a formal security analysis) [4, 10, 22], or make assumptions that break fungibility (e.g. honest majority in stake [10, 11, 19] and/or static mining power [11]). Indeed, with sufficient assumptions the problem of a hybrid protocol is trivially resolved. For instance, if we assume an honest majority in stake (at all time), we can use a committee chosen randomly from the pool of stakeholders to assist a PoW ledger by regularly issuing *checkpoints* [19]. However, the security of this scheme is solely guaranteed by the stakeholders, and the trust is entirely supported by PoS (not from PoW).

Natural solutions. In fact, if we assume a static setting (where both the total mining power and total "active" stake are fixed and known to the protocol designer), a simple solution is the following: PoW and PoS mining occur in parallel; and whichever PoW or PoS succeeds first, goes ahead and extends the longest chain. However, there does not appear to be any straightforward (or otherwise) approach to extend this simple idea to the non-static setting, since we can no longer normalize the stake and work in the system when the total mining power is changing and unknown (§4). Even a fungible combination of work (i.e., total amount of honest work is more than total Byzantine work) emanating from two different hash functions (thus allowing two PoW blockchains using the same longest-chain consensus protocol to co-exist) has been unsolved.

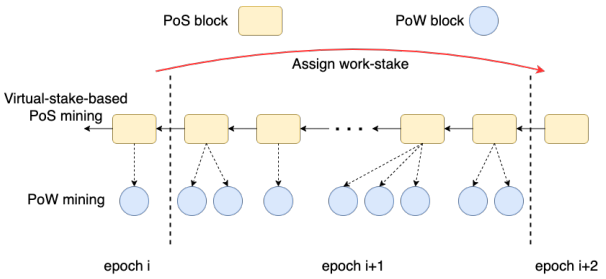
Minotaur Protocol. In this paper we present Minotaur, a blockchain protocol with proof of fungible work and stake. At its core, Minotaur is a longest chain protocol that bases its block-proposer schedule on our concept of *virtual stake* that fuses active *actual stake* and *work stake*. The work-stake distribution is determined per epoch by measuring the participants' contributions in hashing power during a prior epoch. Work contribution is achieved by concurrent PoW mining of *PoW blocks* (similar to "endorser inputs" [21] or "fruits" [31]) to be eventually referenced by main-chain blocks. The work-stake distribution is then assigned proportionally to a participant's share of endorser blocks referenced from the main chain during that epoch; and applying the fairness mechanisms in [17, 21, 31], this assignment indeed truly represents the PoW-resource distribution among the participants. Note that this process can be seen as a fine-grained adaptation of PoW-based committee election such as in [30]. Figure 1a illustrates the protocol (a detailed description is given in §5).

Minotaur security is optimal. We show that Minotaur is truly fungible between work and stake by showing it is secure when the sum of ω times the proportion of adversarial hash power (β_w) and $1-\omega$ times the proportion of the adversarial stake (β_s) is smaller than $1/2$, for any $\omega \in [0,1]$, a weighing parameter of PoW/PoS (see Figure 1b for $\omega = 0.5$). Figure 1b also compares the security guarantee of Minotaur and a couple of non fungible PoW/PoS protocols in the literature (the checkpointed ledger [19], the 2-hop blockchain [11], and a few finality gadgets [6, 27, 33, 35])—for more details, see the full version of the paper [14]. We give a formal security analysis in §6. The new challenge that we have to tackle in our security analysis is to show a *fairness* guarantee in the work-stake conversion (e.g., a miner with 10% mining power should gain 10% work stake) in the *non-static* setting and in the presence of an *adversarial majority* of mining power. This requires a new understanding of how to bound the evolution of mining difficulty in the system compared to techniques used in previous works and the Bitcoin blockchain, and presents a significant barrier to surmount in our analysis. One immediate consequence is that Minotaur can tolerate a 51% mining adversary, as long as there is a supermajority in honest stake (and vice verse, a 51% stake adversary). We prove our security guarantee to be optimal, see §3, in the sense that otherwise the adversary will control the majority of the combined "resource" in the system and an analogy of Nakamoto's private-chain attack [25] could break the security (formal and detailed proof in §3). We also show that Minotaur is capable of tolerating fluctuations of work much more effectively than Bitcoin, cf. §6.4.

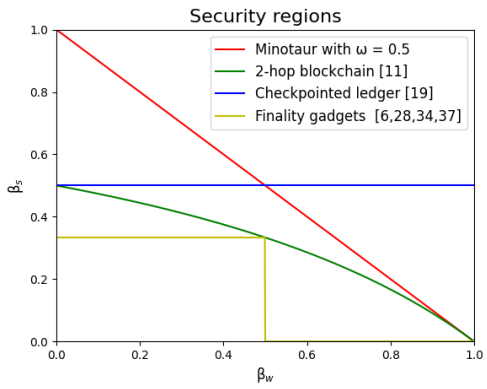
Implementation. We have implemented a prototype of a Minotaur client in about 6000 lines of Rust code [24] and provide experimental results to evaluate the performance of Minotaur under different scenarios. We also implemented Bitcoin/FruitChain/Ouroboros clients as benchmarks. We use that client to validate also experimentally that Minotaur can survive more drastic variations of network hash power, while Bitcoin/FruitChain is no longer live under the same scenario. We also evaluate practical security concerns, like private-chain and selfish-mining attacks, on our Minotaur client. Our implementation shows that Minotaur performs well even under these attacks, and makes a stronger case for the practical viability and robustness of the system. Details about the implementation and the experiments are given in §7.

Motivation. Combining multiple resources helps to protect against scenarios where the adversary may gain control over the majority of one single resource but not over the combination of all involved resources. For instance, to violate security in the PoW/PoS case with $\omega = 0.5$, an adversary controlling half of the hash power would now additionally need to control half of the stake.

As a concrete example, consider the launch of a new blockchain. A PoS chain can be easily bootstrapped with existing techniques such proof-of-burn [20], initial coin offering [23] and airdrop [1]. In contrast, the bootstrapping of a PoW chain is challenging, as the new system would presumably start off with a relatively small total computation power and thus be vulnerable to "51% attacks". Especially in view of the many PoW ledgers currently competing for the dedication of hashing power, a large established Bitcoin miner suddenly diverting their resources to the new system may



(a)



(b)

Figure 1: (a) Architecture of Minotaur **consensus protocol. (b) Comparison of security regions achieved by different hybrid PoW/PoS protocols (more details in [14]).**

easily result in their obtaining full control of the system. To defend against such situations, Minotaur allows to launch a (projected) PoW blockchain in pure PoS mode (setting parameter $\omega = 0$) and later transit into a pure PoW blockchain by gradually changing the weighing parameter ω to a higher value (§8), thus allowing to safely ramp up the applied hashing power until a safe level of participation has been reached.

Related work. The idea of hybrid PoW/PoS block production was first mentioned in [22]. A first concrete construction was given in [4] under the label "proof of activity", but without giving a rigorous security analysis. Their block-production mechanism essentially extends standard Bitcoin mining by having the mined block implicitly elect a set of stakeholders that are required to sign the block in order to validate it.

A similar construction was proposed in [11] and proven secure under a majority of adversarial hashing power—however, their security proof still implies an honest majority of stake. In particular, and contrary to their initial claims, the protocol is not proven secure under the condition that *any* minority of the combined resources is controlled by the adversary.

The work of [11] was extended in [10], to adapt to dynamic participation of both, hashing power and stake, and in [36], to combine PoW with multiple resources (rather than just PoS)—both works lack a rigorous security analysis.

The application of PoW/PoS hybrid block production for the goal of protecting early-stage PoW systems against initial periods of adversarial dominance in hashing power was explored in [7]. They propose to start the system with hybrid block production (where each resource contributes to a certain fraction of the blocks) and then slowly fade out the stake contribution to eventually turn the system into pure PoW. Also this protocol is not proven secure.

There also exist another class of hybrid PoW/PoS protocols [6, 27, 33, 35], which uses a BFT protocol (PoS) to build a finality gadget/layer on the top of a Nakamoto-style longest chain protocol (PoW) to achieve important properties such as accountability and finality. However, these protocols require an honest majority (or even supermajority) on both the set of miners and the set of stakeholders hence they are not fungible.

2 PRELIMINARIES

2.1 Security model

Time, slots, and synchrony. We consider a setting where time is divided into discrete units called *slots*. Players are equipped with (roughly synchronized) clocks that indicate the current slot. Slots are indexed by integers, and further grouped into epochs with fixed size R, i.e., epoch e composes of slots $\{(e-1)R+1, (e-1)R+2, \cdots, eR\}$. And we assume that the real time window that corresponds to each slot has the following two properties: (1) The current slot is determined by a publicly-known and monotonically increasing function of current time; (2) Each player has access to the current time and any discrepancies between parties' local time are insignificant in comparison with the duration of a slot.

We describe our protocols in the now-standard Δ-synchronous network model considered in [3, 8, 18, 29], where there is an (unknown) upper bound Δ in the delay (measured in number of slots) that the adversary may inflict to the delivery of any message. Similar to [18, 29], the protocol execution proceeds in slots with inputs provided by an environment program denoted by $\mathcal{Z}(1^\kappa)$ to parties that execute the protocol Π, where κ is a security parameter. The network is modeled as a diffusion functionality similar to those in [18, 29]: it allows message ordering to be controlled by the adversary $\mathcal{A}$, i.e., $\mathcal{A}$ can inject messages for selective delivery but cannot change the contents of the honest parties' messages nor prevent them from being delivered within Δ slots of delay — a functionality parameter. Specially, for $\Delta = 1$, the network model is reduced to the so-called *lock-step synchronous model*, where messages are guaranteed to be delivered within one slot.

Protocol player. The protocol is executed by two types of players, PoW-miners (miners for short) and PoS-holders (stakeholders for short), who generate different types of blocks, PoW blocks and PoS blocks. Note that we allow for any relation among the sets of miners and stakeholders, including the possibility that all players play both roles, or the two types of players are disjoint.

Random oracle. For PoW mining, we abstract the hash function as a random-oracle functionality. It accepts queries of the form $(\texttt{compute}, x)$ and $(\texttt{verify}, x, y)$. For the first type of query, assuming x was never queried before, a value y is sampled from $\{0,1\}^\kappa$ and it is entered to a table T_H. If x was queried before, the pair (x, y) is recovered from T_H. In both cases, the value y is provided as an answer to the query. For the second type of query, a lookup operation is performed on the table. Honest miners are allowed to ask one query per slot of the type `compute` and unlimited queries

of the type `verify`. The adversary $\mathcal{A}$ is given a bounded number of `compute` queries per slot and also unlimited number of `verify` queries. The bound for the adversary is determined as follows. Whenever a corrupted miner is activated the bound is increased by 1; whenever a query is asked the bound is decreased by 1 (it does not matter which specific corrupted miner makes the query).

Adversarial control of resources. We assume a Byzantine adversary who can decide on the spot how many honest/corrupted miners are activated adaptively. Note that we allow instantaneously adaptive corruption on miners, which means that both the number of honest miners and the number of corrupted miners in each slot are chosen by the adversary. For stakers, the adversary is only "moderately" adaptive (participant corruption only takes effect after a certain delay), just like in Ouroboros classic [21]. In a slot r, the number of honest miners that are active in the protocol is denoted by h_r and the number of all active miners in slot r is denoted by n_r. Note that h_r can only be determined by examining the view of all honest miners and is not a quantity that is accessible to any of the honest miners individually. In order to obtain meaningful concentration bounds on the number of PoW blocks in a long enough window, we set a lower bound α_0 on the fraction of honest mining power, i.e., $h_r \geq \alpha_0 n_r$ for all r. Note that α_0 does not have to be $1/2$ as required by Bitcoin, it can be a small positive constant. Sudden decreases of total mining power could hurt the liveness of the protocol due to too few PoW blocks mined in one epoch. Therefore, we need to restrict the fluctuation of the number of honest/adversarial queries in a certain limited fashion. Suppose $\mathcal{Z}, \mathcal{A}$ with fixed coins produces a sequence of honest miners h_r, where r ranges over all slots of the entire execution, we define the following notation.

Definition 1 (from [16]). *For $\gamma \in [1, \infty)$, we call a sequence $(x_r), r \in [LR]$, (γ, s)-*respecting *if for any set $S \subseteq [0, LR]$ of at most s consecutive slots,*

$$\max_{r \in S} x_r \leq \gamma \min_{r \in S} x_r,$$

*We say that $\mathcal{Z}$ is (γ, s)-*respecting *if for all $\mathcal{A}$ and coins for $\mathcal{Z}$ and $\mathcal{A}$, both the sequences of (h_r) and (n_r) are (γ, s)-respecting.*

Finally, Minotaur achieves consensus via proof of fungible work and stake. The following is the formal definition of fungibility and our major assumptions on the adversarial power.

Definition 2 (Fungibility of resources). *For a time window W, let β_s^W be the maximum fraction of adversarial stake in W, where the maximum is taken over all views of all honest players across all slots in W; and let β_w^W be the maximum fraction of adversarial mining power over all slots in W (i.e., $\beta_w^W = 1 - \min_{r \in W} h_r/n_r$). For $\theta \in [0, 1]$, we say the adversary $\mathcal{A}$ is (θ, m, ω)-bounded if for any time window W with at most m slots, we have $\omega\beta_w^W + (1-\omega)\beta_s^W \leq \theta$. We say a blockchain protocol achieves fungibility of work and stake if it is secure against such an adversary.*

To keep the paper simple, the Minotaur protocol, as described, will rely on Assumption 1 below that initializes the protocol with honest majority of stake; however this assumption is not essential; we refer the reader to §8 for a discussion on how to remove it.

Assumption 1 (Initialization). *During the initialization phase of the protocol, we assume:*

- *1.1 The initial stake distribution has honest majority, i.e., $\beta_s^0 \leq 1/2 - \sigma$, where β_s^0 is the fraction of initial stake controlled by the adversary.*
- *1.2 We have a good estimate of the initial honest mining power h_1. In particular, let $\tilde{h}_1$ be the estimate of h_1, we have $h_1/(1+\delta)\gamma^2 \leq \tilde{h}_1 \leq \gamma^2 h_1/(1-\delta)\alpha_0$.*

Post initialization, we relax the above to our intended fungible resource assumption described below.

Assumption 2 (Execution). *During the execution phase of the protocol, we assume:*

- *2.1 Stake-work bound: The adversary $\mathcal{A}$ is $(1/2 - 2\sigma, 2R, \omega)$-bounded.*
- *2.2 Work fluctuation bound: The environment $\mathcal{Z}$ is $(\gamma, 2R)$-respecting.*
- *2.3 Work participation bound: For any slot $r \in [LR]$, $h_r \geq \alpha_0 n_r$.*

Participation model. Minotaur can be constructed based on different PoS longest chain protocols. Various versions of Minotaur will take different subsets of the following assumptions on the participation model:

- (P1) Honest stakeholders are always online[1];
- (P2) Honest miners who mined PoW blocks in epoch e will stay online in epoch $e + 2$;[2]
- (P3) In case an honest party joins after the beginning of the protocol, its initialization chain C provided by the environment should match an honest party's chain which was active in the previous slot.

2.2 Blockchain security properties

Notation 1. *We denote by $C^{\lceil \ell}$ the chain resulting from "pruning" the blocks with timestamps within the last ℓ slots. If C_1 is a prefix of C_2, we write $C_1 \prec C_2$. The latest block in the chain C is called the head of the chain and is denoted by head(C). We denote by $C_1 \cap C_2$ the common prefix of chains C_1 and C_2. Given a chain C and an interval S (or $[r_1, r_2]$), let $C(S)$ (or $C[r_1, r_2]$) be the segment of C containing blocks with timestamps in S (or $[r_1, r_2]$). We say that a chain C is held by or belongs to an honest node if it is one of the best chains in its view.*

Definition 3 (Common Prefix (CP)). *The common prefix property with parameter $\ell_{\text{cp}} \in \mathbb{N}$ states that for any two honest nodes holding chains C_1, C_2 at slots r_1, r_2, with $r_1 \leq r_2$, it holds that $C_1^{\lceil \ell_{\text{cp}}} \prec C_2$.*

Definition 4 (Existential Chain Quality (∃CQ)). *The existential chain quality property with parameter $\ell_{\text{cq}} \in \mathbb{N}$ states that for any chain C held by any honest party at slot r and any interval $S \subseteq [0, r]$ with at least ℓ_{cq} consecutive slots, there is at least one honestly generated block in $C(S)$.*

Our goal is to generate a robust transaction ledger that satisfies *persistence* and *liveness* as defined in [21].

[1] This assumption can be easily relaxed. Namely, it is sufficient for an honest stakeholder to come online at the beginning of each epoch, determine whether it belongs to the slot leader set for any slots within this epoch, and then come online and act on those slots while maintaining online presence at least every k slots. See Appendix H of [8] for more detail.

[2] This can be relaxed similarly as P1.

Definition 5. *A protocol* Π *maintains a robust public transaction ledger if it organizes the ledger as a blockchain of transactions and it satisfies the following two properties:*

- *(Persistence) Consider the confirmed ledger* L_1 *on any node* p_1 *at any slot* r_1*, and the confirmed ledger* L_2 *on any node* p_2 *at any slot* r_2 *(here* u_1 *(*r_1*) may or may not equal* u_2 *(*r_2*)). If* $r_1 + \Delta < r_2$*, then* L_1 *is a prefix of* L_2*.*
- *(Liveness) Parameterized by* $u \in \mathbb{R}$*, if a transaction* tx *is received by all honest nodes for more than* u *slots, then all honest nodes will contain* tx *in the same place in the confirmed ledger.*

3 IMPOSSIBILITY RESULT

Consider any protocol Π executed by two types of players, miners and stakeholders, and let D_Π be the region of (β_w, β_s) such that Π can generate a robust public transaction ledger under an adversary controlling a β_w fraction of mining power and a β_s fraction of stake (see Figure 1b).

Theorem 3.1. *Any two points,* $X_1 = (p_1, q_1)$ *and* $X_2 = (p_2, q_2)$ *such that* $p_1 + p_2 \geq 1$ *and* $q_1 + q_2 \geq 1$*, cannot co-exist in* D_Π*.*

Proof. We show that if there exists a protocol Π secure under both points X_1 and X_2, then we can use Π to implement a robust transaction ledger with two players, one of them being Byzantine, which is impossible.

Let Alice and Bob be two players, one of them possibly malicious. Let Alice control a p_1 fraction of mining power and a $1 - q_2$ fraction of stake. Let Bob control a $1 - p_1$ fraction of mining power and a q_2 fraction of stake. In the case where Alice is malicious, a p_1 fraction of mining power and a $1 - q_2 \leq q_1$ fraction of stake are malicious, which is dominated by point X_1. In the case where Bob is malicious, a $1 - p_1 \leq p_2$ fraction of mining power and a q_2 fraction of stake are malicious, which is dominated by point X_2. Assuming that Π implements a robust public transaction ledger implies that also the protocol among the two players does while one of them is malicious. However, by Theorem 1 in [15], tolerating $f = 1$ malicious player requires at least $2f + 1 = 3$ players. This is a contradiction. □

Figure 2 gives a few examples of possible *maximum* security regions that satisfy the constraint by Theorem 3.1. By maximum, we mean the region cannot be enlarged. All these regions are bounded by a *non-increasing* curve that is *symmetrical* about the point $(1/2, 1/2)$. Thus, all these security regions have area $1/2$, which is an analogy of the $1/2$ fault tolerance in the single-resource systems. In this work, we focus on achieving the security regions bounded by a linear curve (the yellow and red ones in Figure 2) and leave the achievability of other possible regions (e.g., the green and blue ones) to future work. Note that the security region achieved by the checkpointed ledger [19] is also optimal (albeit not fungible), and we will see that this region can also be achieved by Minotaur with $\omega = 0$.

Corollary 1. *No protocol* Π *can generate a robust public transaction ledger under a* $(1/2, m, \cdot)$*-bounded adversary for any* m*.*

Corollary 1 thus implies that Assumption 2.1 is not only sufficient but also necessary.

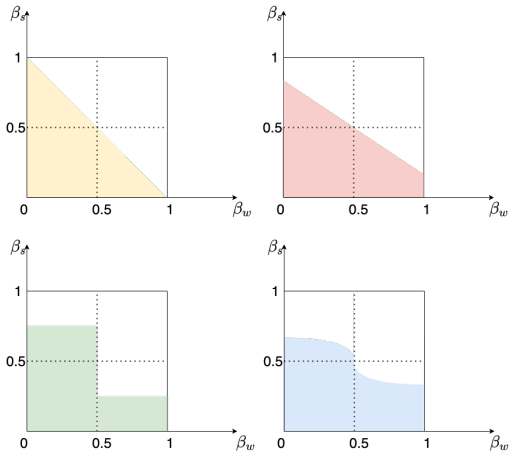


Figure 2: A few examples of possible security regions satisfying the constraint by Theorem 3.1.

4 BASELINE APPROACHES

In this section, we provide a couple of natural designs of hybrid PoW/PoS protocols, which fail to achieve the full goal of Minotaur.

Idea 1: Securing PoW chain via checkpointing. The checkpointed ledger [19] employs an external set of parties or a committee chosen randomly from the pool of stakeholders to assist a PoW ledger by finalizing blocks shortly after their creation. The finalized blocks are called checkpoints and the final ledger is formed by the chain of checkpoints. This mechanism can also secure a PoW ledger in the presence of adversarial mining majorities, but the security is solely guaranteed by the external set or the stakeholders.

Other checkpointing protocols [6, 27, 33, 35] achieve properties that are not possible with pure PoW protocols, such as accountability and finality. However, in contrast to our requirement to tolerate any adversarial minority of the *combined* resources, these protocols are based on the stronger assumption of honest majority (or even supermajority) of *both* the set of miners and the set of stakeholders.

Idea 2: Smooth interpolation among PoW and PoS blocks. In the static setting (where both the total mining power and total active stake are fixed and known to the protocol designer), there is a simple protocol that can also achieve the regions defined by the red line in Figure 1b. In this protocol, PoW and PoS mining occur in parallel, following the longest chain rule. In the initialization phase, we tune the mining targets such that the mining rate (i.e., number of blocks produced per unit time) of PoW blocks is ωf and the mining rate of PoS blocks is $(1 - \omega)f$. In the execution phase, whichever miners or stakeholders succeeds first, it goes ahead and extends the longest chain which accepts both types of blocks.

Compared with a pure PoS protocol (e.g., Ouroboros Praos [8]), the adversary has strictly smaller action space because it cannot equivocate with the PoW blocks. Thus, the security of this protocol follows directly from the security of Ouroboros Praos via either the forkable string argument [3] or the Nakamoto block method developed in [9]. We remark that the existence of this simple protocol makes the 2-hop blockchain [11] less interesting as it only works in the static setting and does not even allow weighing PoW and PoS.

However, it is very hard to extend this simple idea to the non-static setting, particularly with variable mining power. A natural approach to support variable mining power is to have fixed-length epochs and adjust the mining target of PoW blocks every epoch as in Bitcoin. From the simple protocol described above, we learn that in order to guarantee security in the non-static setting, we need to make sure that the ratio of total mining rates of PoW blocks and PoS blocks is a constant $\omega/(1-\omega)$ in every epoch. However, this is impossible to achieve in the variable mining power setting because the adversary can always decide to hide or release its PoW blocks so that there is no way to estimate the total mining power accurately. Inaccurate PoW mining target adjustment could lead to a different weighing parameter ω of the security region than the desired one. More importantly, the value of ω can be easily manipulated by the adversary (via selectively publishing its PoW blocks), and is unknown to the honest players.

5 THE FULL PROTOCOL

In this section, we provide a detailed description of our protocol Minotaur. The protocol is built on an epoch-based PoS longest chain protocol e.g., Ouroboros Classic [21], Praos [8], or Genesis [3]. Recall that in an epoch-based PoS protocol, time is divided into multiple epochs, each with a fixed number of slots. In each slot, one or multiple stakeholders are selected as block proposers by a PoS "lottery" process. In this process, the probability of being selected is proportional to the relative stake a stakeholder has in the system, as reported by the blockchain itself. See [3, 8, 14, 21] for more details.

In contrast to the original Ouroboros protocols, the block-proposer schedule accounts for both resources by means of *virtual stake*, which is a combination of the *actual stake* (representing stake as in the original protocol), and *work stake* representing the share in block-production rights attributed to the PoW resource.

Additionally, PoW miners participate by mining *endorser blocks* (along the lines of "endorser inputs" [21] or "fruits" [31]). These endorser blocks are not directly appended to the main chain, but are to be referenced by future main-chain blocks (i.e., PoS blocks scheduled by means of virtual stake). Each epoch is assigned a certain amount of work stake that is assigned to the PoW miners who succeed in mining PoW blocks; the work stake assigned to the respective miners is proportional to their contribution of PoW blocks referenced from the main chain during an epoch. PoS block production rights per epoch are then assigned by considering the sum of actual stake (contributed by tokens) and work stake (contributed by PoW blocks).

Note that the purpose of the PoW endorser blocks is to measure work fairly, not implying that they also need to carry the ledger transactions. Transaction inclusion is orthogonal to this aspect, and can still be implemented along the lines of Bitcoin (transaction inclusion in the main-chain blocks) or Input-Endorsers/FruitChain (transaction inclusion in the endorser blocks), or variants thereof.

We now treat the underlying PoS protocol as a black-box and present the protocol in detail. The protocol runs in fixed-time epochs with R slots each.

- PoW mining: In slot sl of epoch e, a miner mines a PoW block if
$$H(pk, sl, h, mr, nonce) < T_e,$$
where pk is the public key of the miner, h is the hash of the last confirmed PoS block (according to the confirmation rule of the underlying PoS protocol), mr is the Merkle root of the payload, T_e is the mining target in epoch e. Like in PoW blockchains, miners try different values of *nonce* to solve this hash puzzle. Following the notation in [18], we define the *difficulty* of this PoW block to be $1/T_e$. Besides possible transactions, also the public key of the miner is included in the payload. A PoW block is called recent in current slot sl_{now} if it refers to a confirmed PoS block mined no earlier than slot $sl_{\text{now}} - sl_{\text{re}}$, where sl_{re} is called the *recency parameter*.
- PoS mining: in slot sl of epoch e, one or multiple stakeholders are selected to propose PoS blocks extending the best chain (acc. to the chain-selection rule). The selection of PoS block proposers uses the same mechanism as in the underlying PoS protocol. But for each node, the probability of being selected is proportional to its relative "virtual" stake, instead of the relative actual stake. The virtual stake is the sum of actual stake and work stake. At the beginning of each epoch, we set the total work stake equal to $\frac{\omega}{1-\omega}$ times the total actual stake in the system. And the work-stake distribution used in epoch e is set to be the distribution of block difficulties from PoW blocks referred by PoS blocks in epoch $e-2$.
- Adjust the PoW mining target: For epoch e, T_e is adjusted according to D_{e-1}^{total} the total difficulty of PoW blocks referred by PoS blocks in slots $[(e-2)*R-k+1, (e-1)*R-k]$, i.e., $T_e = f^{(w)}R/D_{e-1}^{\text{total}}$, where $f^{(w)}$ is a protocol parameter representing the expected PoW mining rate in number of blocks per slot. (Specially, for $e=2, T_2 = f^{(w)}(R-k)/D_1^{\text{total}}$.) Note that around the boundary of two epochs $e-1$ and epoch e, the adversary has the option to use targets from both epochs, as PoW blocks don't have accurate timestamps.
- Chain selection rule: due to a long-range attack (see full version of the paper [14]), the longest chain rule would fail. We use the following chain selection rules, which have different security guarantees.
  - maxvalid-mc (from Ouroboros Praos [8], needs a trusted third party for bootstrapping, i.e., assumption (P3)): it prefers longer chains, unless the new chain forks more than k blocks relative to the currently held chain (in which case the new chain would be discarded).
  - maxvalid-bg (from Ouroboros Genesis [3], supports bootstrapping from genesis block): it prefers longer chains, if the new chain C' forks less than k blocks relative to the currently held chain C. If C' did fork more than k blocks relative to C, C' would still be preferred if it grows more quickly in the s slots following the slot associated with the last common block of C and C'.

6 SECURITY ANALYSIS

6.1 Main theorems

Under different security models and assumptions, we prove the security (in particular, *persistence* and *liveness* as defined in §2.2) of Minotaur constructed with three various versions of Ouroboros PoS protocols.

Theorem 6.1. *Under Assumption 1, Assumption 2 and assumptions (P1)-(P3), when executed in a lock-step synchronous model (i.e., Δ-synchronous model with $\Delta = 1$),* Minotaur *constructed with Ouroboros [21] generates a transaction ledger that satisfies persistence and liveness with overwhelming probability.*

Theorem 6.2. *Under Assumption 1, Assumption 2 and assumptions (P1)-(P3), when executed in a Δ-synchronous model,* Minotaur *constructed with Ouroboros Praos [8] generates a transaction ledger that satisfies persistence and liveness with overwhelming probability.*

Theorem 6.3. *Under Assumption 1, Assumption 2 and assumptions (P1)-(P2), when executed in a Δ-synchronous model,* Minotaur *constructed with Ouroboros Genesis [3] generates a transaction ledger that satisfies persistence and liveness with overwhelming probability.*

6.2 Security with Ouroboros

We first provide a proof sketch for Theorem 6.1. All the parameters used in our analysis are listed in Table 1.

h_r	number of honest PoW queries in slot r
n_r	number of total PoW queries in slot r
α_0	lower bound on the fraction of honest mining power
sl_{re}	recency parameter for PoW blocks
Δ	network delay in slots
κ	security parameter; length of the hash function output
$R \in \mathbb{N}$	duration of an epoch in number of slots
(γ, s)	restriction on the fluctuation of the number of honest queries across slots (Definition 1)
$f^{(s)}$	expected PoS mining rate in number of blocks per slot
$f^{(w)}$	expected PoW mining rate in number of blocks per slot
ϵ	quality of concentration of random variables
σ	PoW fairness parameter & advantage of honest parties (Assumption 1.1 and 2.1)
δ	"goodness" parameter of an epoch/slot (Definition 6)
λ	typicality parameter of the execution
ℓ	minimum number of slots for concentration bounds
L	total number of epochs in the execution
ω	the weighing parameter

Table 1: The parameters used in our analysis.

Proof sketch. The proof relies on two important arguments:

1) **Single-epoch argument.** Given honest majority in the virtual stake (normalized work-stake + normalized actual-stake) used in one epoch, we prove the security properties (CP and $\exists$CQ) for this single epoch. For this, we use the so-called *forkable strings* technique developed in the Ouroboros papers [3, 8, 21].
2) **FruitChain argument.** Given CP and $\exists$CQ of the PoS chain in an epoch, we prove *fairness* of PoW blocks in this epoch, i.e., a miner controlling a ϕ fraction of the computational resources will contribute a ϕ fraction of work.

Using these building blocks, we prove the security with maxvalid-mc inductively. Under the honest-majority assumption in the initial stake distribution (Assumption 1.1), we can prove security of the PoS chain in epochs 1&2. Applying the FruitChain argument, we get fairness of PoW blocks in epochs 1&2. Combining with Assumption 2.1, i.e., $\omega\beta_w + (1-\omega)\beta_s < 1$ (for any long enough window), we have honest majority in the virtual stake distribution used in epoch 3&4. And the proof goes on till the last epoch (Figure 3). □

6.2.1 Single-epoch security.

Theorem 6.4 (Theorem 3 from [8]). *Let $\kappa, R, \Delta \in \mathbb{N}$ and $\sigma \in (0,1)$. Let β_v be the fraction of adversarial virtual stake satisfying*

$$\beta_v \leq 1/2 - \sigma$$

for some positive constant σ. Then the probability that the adversary violates CP with parameter $\ell_{\text{cp}} = \kappa$ and $\exists$CQ with parameter $\ell_{\text{cq}} = \kappa$ throughout a period of R slots is no more than $Re^{-\Omega(\kappa)}$. The constant hidden by the $\Omega(\cdot)$-notation depends on σ.

In Minotaur, the main-chain blocks are scheduled as a function of the virtual-stake distribution only. The Ouroboros analysis thus directly translates by replacing its actual stake by virtual stake.

6.2.2 FruitChain argument.

Recall that h_r is the number of honest PoW queries in slot r and n_r is the number of total PoW queries in slot r. For a set of slots S, we define $h(S) = \sum_{r\in S} h_r$ and $n(S) = \sum_{r\in S} n_r$. In order to obtain meaningful concentration bounds on the number of PoW blocks in one epoch, there is a lower bound α_0 on the fraction of honest mining power, i.e., $h_r \geq \alpha_0 n_r$ for all r.

In the analysis of this subsection, we assume the main chain (PoS chain) satisfies properties CP with parameter $\ell_{\text{cp}} = \kappa$ and $\exists$CQ with parameter $\ell_{\text{cq}} = \kappa$. By the common prefix property, for a PoS chain C held by an honest node at slot r, the prefix $C^{\lceil\kappa}$ are stabilized, so to mine a PoW block at slot r an honest miner will refer the tip of $C^{\lceil\kappa}$ as the last confirmed PoS block. And we set the recency parameter $sl_{\text{re}} = 3\kappa + \Delta$, i.e., a PoW block B_w is recent w.r.t. a chain C at slot r if the confirmed PoS block referred by B_w is in C and has timestamp at least $r - 3\kappa - \Delta$. With this selection of the recency parameter, we can prove the following key property of the protocol: any PoW block mined by an honest miner will be incorporated into the stabilized chain (and thus never lost). We refer to this as the *Fruit Freshness Lemma*—PoW blocks stay "fresh" sufficiently long to be incorporated.

Lemma 6.5 (Fruit Freshness). *Suppose the PoS chain satisfies properties CP with parameter $\ell_{\text{cp}} = \kappa$ and $\exists$CQ with parameter $\ell_{\text{cq}} = \kappa$. Then, if $sl_{\text{re}} = 3\kappa + \Delta$, an honest PoW block mined at slot r will be included into the stabilized chain before slot $r + r_{\text{wait}}$, where $r_{\text{wait}} = 2\kappa + \Delta$.*

Proof. Suppose an honest PoW block B_w is mined at slot r_0 while the PoS chain C_0 is adopted, then B_w will reference the tip of $C_0^{\lceil\kappa}$ as the last confirmed PoS block. Further, by the $\exists$CQ property, the tip of $C_0^{\lceil\kappa}$ has timestamp $r_1 \geq r_0 - 2\kappa$. By slot $r_0 + \Delta$, all honest nodes will receive B_w. Let $r_2 = r_0 + 2\kappa + \Delta$ and C be any chain held by an honest node at slot r_2, then again by $\exists$CQ, there exists an honest block B_s on C whose timestamp r_3 is in the interval $[r_2 - 2\kappa, r_2 - \kappa)$. We check that B_w is still recent at slot r_3 as

$$r_3 - r_1 < (r_2 - \kappa) - (r_0 - 2\kappa) = 3\kappa + \Delta = sl_{\text{re}}.$$

As B_s is an honest block mined after $r_0 + \Delta$, B_s or an ancestor of B_s must include B_w. And since B_s is stabilized in C at slot r_2, we have that $r_{\text{wait}} = r_2 - r_0 = 2\kappa + \Delta$. □

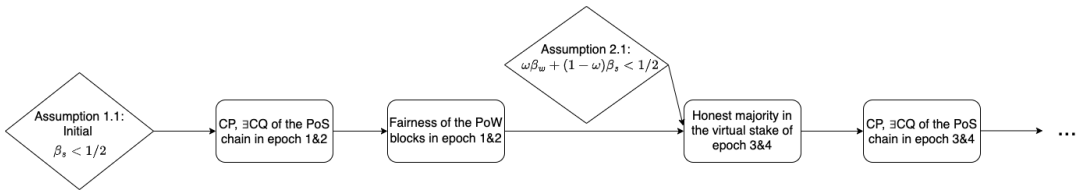


Figure 3: Proof sketch for §6.2.

In a (γ, s)-respecting environment, the following holds.

Proposition 1 (Proposition 2 from [18]). *In a (γ, s)-respecting environment, let U be a set of at most s consecutive slots and $S \subseteq U$. Then, for any $h \in \{h_r : r \in U\}$ and any $n \in \{n_r : r \in U\}$ we have*

$$\frac{h}{\gamma} \le \frac{h(S)}{|S|} \le \gamma h,\ \frac{n}{\gamma} \le \frac{n(S)}{|S|} \le \gamma n\,,$$
$$h(U) \le \Big(1 + \frac{\gamma|U \setminus S|}{|S|}\Big)h(S),\ n(U) \le \Big(1 + \frac{\gamma|U \setminus S|}{|S|}\Big)n(S)\,.$$

Recall that T_e is the mining target in epoch e determined by the stabilized segment of the chain from epoch $e-1$. In order to obtain meaningful concentration bounds on the number of PoW blocks, we need T_e to be "reasonable" for each epoch. Similar to [16, 18], we define a notation of "good" epochs as follows. By abuse of notation, we write the expected PoW block rate $f^{(w)}$ simply as f in the analysis below.

Definition 6. *Epoch e is* good *if $(1-\delta)\alpha_0 f/\gamma \le p h^{(e)} T_e \le (1+\delta)\gamma f$, where $p = 1/2^\kappa$ and $h^{(e)} = h_{(e-1)R+1}$, i.e., the number of honest queries in the first slot of epoch e. A slot r is good if it is in a good epoch.*

Proposition 2. *Under a $(\gamma, 2R)$-respecting environment, if slot r is a good slot in epoch e, then $(1-\delta)\alpha_0 f/\gamma^2 \le p h_r T_e \le (1+\delta)\gamma^2 f$.*

Proof. By the definition of a good epoch, we have $(1-\delta)\alpha_0 f/\gamma \le p h^{(e)} T_e \le (1+\delta)\gamma f$. And under a $(\gamma, 2R)$-respecting environment and the fact that slot r is in epoch e, we know $1/\gamma \le h_r/h^{(e)} \le \gamma$. Therefore, $(1-\delta)\alpha_0 f/\gamma^2 \le p h_r T_e \le (1+\delta)\gamma^2 f$. □

Now, we prove *fairness* of PoW blocks in any large enough window, i.e., a miner controlling a ϕ fraction of the computational resources will contribute a ϕ fraction of work. We first define fairness.

Definition 7. *We say that $\mathcal{H}$ is a ϕ-*fraction honest subset *if miners in $\mathcal{H}$ (that may change over time) are honest and $n_r^{\mathcal{H}} > \phi n_r$ for any slot r, where $n_r^{\mathcal{H}}$ is number of queries in $\mathcal{H}$ at slot r.*

Definition 8 (Fairness). *We say that the protocol has* (approximate) (W_0, σ)−fairness *if, for any ϕ, $\alpha_0 \le \phi < 1$, any ϕ−fraction honest subset $\mathcal{H}$, and any honest miner holding chain $\mathcal{C}$ at slot r and any interval $S_0 \subseteq [0, r]$ with at least W_0 consecutive slots, it holds that the PoW blocks included in $\mathcal{C}(S_0)$ mined by $\mathcal{H}$ have total difficulty at least $(1-\sigma)\phi d$, where d is the total difficulty of all PoW blocks included in $\mathcal{C}(S_0)$.*

Define the random variable D_r equal to the sum of the difficulties of all PoW blocks computed by honest miners at slot r. And for fixed ϕ and ϕ−fraction honest subset $\mathcal{H}$, define the random variable $D_r^{\mathcal{H}}$ equal to the sum of the difficulties of all PoW blocks computed by miners in $\mathcal{H}$ at slot r. For a set of slots S, we define $D(S) = \sum_{r\in S} D_r$, $D^{\mathcal{H}}(S) = \sum_{r\in S} D_r^{\mathcal{H}}$, and $n^{\mathcal{H}}(S) = \sum_{r\in S} n_r^{\mathcal{H}}$. For a set of J adversarial queries, define the random variable $A(J)$, as the sum of difficulties of all the PoW blocks created during queries in J.

Next we define the notion of typical executions, which will be shown to occur with overwhelming probability.

Definition 9 (Typical execution). *An execution E is* typical *if the following conditions hold:*
(a) For any set S of at least ℓ consecutive good slots,

$$D(S) < (1+\epsilon) p h(S)\,.$$

(b) For any set S of at least ℓ consecutive good slots, let J be the set of adversarial queries in S. If we further know each query in J made at slot r with target T satisfies $(1-\delta)\alpha_0 f/\gamma^2 \le p n_r T \le (1+\delta)\gamma^2 f/\alpha_0$, then

$$D(S) + A(J) < (1+\epsilon) p(h(S) + |J|)\,.$$

(c) For any set S of at least ℓ/ϕ consecutive good slots,

$$D^{\mathcal{H}}(S) > (1-\epsilon) p n^{\mathcal{H}}(S)\,.$$

To obtain meaningful concentration, we should be considering a sufficiently long slot sequence of at least

$$\ell \triangleq \frac{2(1+\epsilon/3)}{\epsilon^2\gamma^3(1-\delta)\alpha_0 f}\lambda\,,$$

where λ is called the typicality parameter of the execution.

For our analysis to go through, the protocol parameters should satisfy certain conditions which we now discuss. First, we will require that the number ℓ defined above and the security parameter κ are appropriately small compared to R, the duration of an epoch.

$$R - 3\kappa - \Delta \ge \ell/\alpha_0 \ge \frac{\gamma}{\epsilon}(4\kappa + \Delta)\,. \tag{C1}$$

Note that (C1) implies $R \ge (4\kappa + \Delta)/\epsilon$. Second, the slack variables ϵ and δ should satisfy

$$4\epsilon \le \delta \le 1\,. \tag{C2}$$

Next we bound the probability of an atypical execution (Lem. 6.6) and show that all epochs are good in a typical execution by induction (Lem. 6.7). Therefore, all epochs are good with overwhelming probability.

Lemma 6.6. *For an execution $\mathcal{E}$ of LR slots, in a $(\gamma, 2R)$-respecting environment, the probability of the event "$\mathcal{E}$ is not typical" is bounded by $O(LR)e^{-\lambda}$.*

Lemma 6.7. *For a typical execution in a $(\gamma, 2R)$-respecting environment, all epochs are good.*

The proofs of Lemmas 6.6 and 6.7 can be found in the full version of the paper [14]. In order to address the dependency among PoW successes in the variable mining difficulty setting, the proof of Lemma 6.6 uses martingale arguments (from [18]) to provide useful bounds on the relevant random variables. The proof of Lemma 6.7

uses an induction argument to show that if all previous epochs are good, then the progress of the adversary in the next epoch will be bounded and as a result the next epoch is also good.

Finally, we prove the FruitChain/fairness argument.

Theorem 6.8 (Fairness). *For a typical execution in a $(\gamma, 2R)$-respecting environment, the protocol with recency parameter $sl_{\text{re}} = 3\kappa + \Delta$ satisfies (W_0, σ)−fairness, where $W_0 = \ell/\alpha_0 + 3\kappa + \Delta$ and $\sigma = 4\epsilon$.*

Proof. Fix ϕ, a ϕ−fraction honest subset $\mathcal{H}$, and an honest node holding chain C. Let $S_0 = \{u : r_1 \le u \le r_2\}$ be a window of at least W_0 consecutive slots. Let $C(S_0)$ be the segment of C containing PoS blocks with timestamps in S_0, let $\mathcal{B}$ be all PoW blocks included in $C(S_0)$, and d be the total difficulty of all PoW blocks in $\mathcal{B}$. The proof of Lemma 6.7 observes the following facts to be used below:

- Fact 1. For any PoW block $B \in \mathcal{B}$, B is mined after $r_1 - 4\kappa - \Delta$.
- Fact 2. For any PoW block $B \in \mathcal{B}$, B is mined before $r_2 + \kappa$.
- Fact 3. If a PoW block B is mined by $\mathcal{H}$ after r_1 and before $r_2 - 3\kappa - \Delta$, then $B \in \mathcal{B}$.

Now, let $S_1 = \{u : r_1 - (4\kappa + \Delta) \le u \le r_2 + \kappa\}$, $S_2 = \{u : r_1 \le u \le r_2 - (3\kappa + \Delta)\}$, and J be the set of adversary queries associated with $\mathcal{B}$ in S_1. Then by Fact 1 and Fact 2, we have that all PoW blocks in $\mathcal{B}$ are mined in S_1; by Fact 3, we have that all PoW blocks mined by $\mathcal{H}$ in S_2 are in $\mathcal{B}$. Similar to the arguments in Lemma 6.7, for each query in S_1 made by either an honest node or the adversary at slot r in epoch e, the target T must satisfy $(1-\delta)\alpha_0 f/\gamma^2 \le pn_r T \le (1+\delta)\gamma^2 f/\alpha_0$.

Further note that, to prove σ-fairness, it suffices to show that

$$D^{\mathcal{H}}(S_2) \ge (1-\sigma)\phi(D(S_1) + A(J)) .$$

Under a typical execution, we have

$$D^{\mathcal{H}}(S_2) > (1-\epsilon)pn^{\mathcal{H}}(S_2) \ge (1-\epsilon)\phi pn(S_2) ,$$

and

$$D(S_1) + A(J) < (1+\epsilon)p(h(S_1) + |J|) = (1+\epsilon)pn(S_1) .$$

By our choice of W_0, we have $|S_2| \ge \ell/\alpha_0 \ge \ell/\phi$. Furthermore, we may assume $|S_2| \le 2R$. This is because we may partition S_2 in parts such that each part has size between ℓ/α_0 and $2R$, sum over all parts to obtain the desired bound. Then by Proposition 1, we have

$$n(S_1) \le (1 + \frac{\gamma|S_1 \setminus S_2|}{|S_2|})n(S_2) \le (1 + \frac{\gamma(8\kappa + 2\Delta)}{\ell/\alpha_0})n(S_2) \overset{(C1)}{\le} (1+2\epsilon)n(S_2) .$$

Finally, by setting $\sigma = 4\epsilon$, we conclude the proof. □

6.2.3 Lifting argument from single-epoch to multiple-epoch. Theorem 6.4 gives bounds for CP and ∃CQ for a single-epoch run of the protocol with static stake distribution and perfect randomness. We now conclude our proof of Theorem 6.1 by showing that these blockchain properties hold throughout the whole lifetime of the system consisting of many epochs.

Theorem 6.9 (Restatement of Theorem 6.1). *Fix parameters κ, λ, ℓ, γ, σ, ϵ, δ, R and L satisfying conditions (C1) and (C2). Under Assumption 1, Assumption 2 and assumptions (P1)-(P3), when executed in a lock-step synchronous model (i.e., Δ-synchronous model with $\Delta = 1$),* Minotaur *constructed with Ouroboros generates a transaction ledger that satisfies persistence and liveness throughout a period of L epochs (each with R slots) with probability $1 - RL(e^{-\Omega(\kappa)} + e^{-\lambda})$.*

This part of the analysis proceeds similarly as in §5 of [21], and is given in the full version of the paper [14]. Applying the improved analysis in [5] (instead of the one in [21]), we obtain the refined bound $1 - RL(e^{-\Omega(\delta^3\kappa)} + e^{-\lambda})$ for the statement of Theorem 6.9. In the full version of the paper [14] we analyze a private-chain attack and show a corresponding lower bound. It reveals that $\delta^2\kappa$ needs to be bounded below by a constant and, consequently, the dependency on δ^3 cannot be improved much further.

6.3 Security with Ouroboros Praos/Genesis

By maxvalid-bg, an honest node prefers longer chains, if the new chain C' forks less than k blocks relative to the currently held chain C. If C' did fork more than k blocks relative to C, C' would still be preferred if it grows more quickly in the s slots following the slot associated with the last common block of C and C'. In this section, we analyze the security of the protocol with maxvalid-bg, but drop assumption (P3) (i.e., allow bootstrapping from genesis).

Proof sketch. This part is identical to the analysis in Ouroboros Genesis (§4.3-4.4 of [3]), so we only sketch it here. The proof proceeds in two steps:

1) For an honest node h_1 that is always online, we show that when replacing maxvalid-mc with maxvalid-bg, the overall execution of the protocol in h_1's view remains the same except with negligible probability. That is to show that with overwhelming probability, whenever h_1 receives a new chain C' that forks more than k blocks from h_1's local chain C, h_1 will always favor C, i.e., C grows more quickly than C' in the first s slots right after the fork.
2) For a newly joined node h_2, we consider a "virtual" node h_2' that holds no stake, but was participating in the protocol since the beginning and was honest all the time. Then we show h_2 will always adopt the same chains as h_2' after joining the network. □

6.4 Comparison to Bitcoin

In this paragraph, we observe that Minotaur (when executed in the pure PoW case, $\omega = 1$) is more robust against fluctuations in PoW participation than Bitcoin. In particular, we demonstrate that Bitcoin may not enjoy liveness if the number of parties is allowed to halve every two weeks—while Minotaur does. The theoretical results of this section are accompanied by experiments; see §7.2 and Figure 5. We point out that although Bitcoin mining power has gone up exponentially since its inception, other PoW blockchains that use the same difficulty adjustment rule may experience such drops in mining power.[3]

Consider Bitcoin for the case that, initially, there are n honest parties and that the target is T. Using the model and notation of [16] with $q = 1$, m is the number of blocks in an epoch and $p = 1/2^\kappa$. Thus, each slot is successful with probability $f = 1 - (1-pT)^n \approx$

[3]For example, ETC dropped from 19 TH/s to 1.5 TH/s in 7 months period https://2miners.com/etc-network-hashrate, but more impressively ERGO dropped from 23.65 to 0.5 TH/s in just 2 weeks' time https://2miners.com/erg-network-hashrate.

pTn. We will show that for $s = \lfloor m/pTn \rfloor$, liveness may fail in a $(2, s)$-respecting environment. In such an environment, the adversary is allowed to halve the number of parties every $s \approx m/f$ slots (note that m/f is the expected duration of an epoch, which for Bitcoin is two weeks).

The attack proceeds in stages of s slots, during which the adversary does not mine nor delays messages. Thus, in the beginning of each slot all parties have a chain of the same length. The adversary halves the number of parties at the beginning of each stage. Let N_i denote the number of honest parties in stage i, with $N_0 = n$ and $N_i = n/2^i$ for $i > 0$. The expected number of successful slots in stage i is

$$s \cdot [1 - (1 - pT)^{N_i}] < spTN_i = \frac{spTn}{2^i} \leq \frac{m}{2^i} .$$

Consider the stage j for which

$$k - 1 < \frac{m}{2^j} \leq 2(k - 1) .$$

The expected number of blocks computed in the first j stages is

$$\frac{m}{2} + \frac{m}{2^2} + \cdots + \frac{m}{2^j} = m - \frac{m}{2^j} < m - k + 1$$

and at most $k - 1$ in stage $j + 1$. Recalling that the median of a binomial distribution with mean μ is at most $\lceil \mu \rceil$, we obtain that with probability at least $1/4$ at most m blocks have been computed by the end of stage $j + 1$ and at most $k - 1$ of them have been computed in the last stage. Assuming the liveness parameter $u < s$, any transaction provided to all honest parties for the first u slots of the final stage, will be in depth less than k. It follows that liveness does not hold in the final stage of the attack.

Proposition 6.10. *Bitcoin's ledger does not satisfy liveness in a $(2, s)$-respecting environment, for any $u < s$ and $s < m/pTn$.*

In contrast, Minotaur does not suffer from such fluctuation. Intuitively, this is because it inherits its security from the underlying proof-of-stake protocol. In particular, the epochs are of fixed duration, implying that target recalculation occurs regularly. This is evident from Condition C1, which allows much greater values for R and γ.

7 EXPERIMENTS

We implemented a prototype instantiation of a Minotaur client in Rust and the code can be found at [24]. We also implemented Bitcoin, FruitChain and Ouroboros clients as benchmarks. Particularly, for better comparison we implemented the variable-difficulty version of FruitChain proposed in [37], which uses the same difficulty adjustment as Bitcoin. In this section, we describe the architecture of our implementation and present experimental results to evaluate the concrete performance of Minotaur under different scenarios.

7.1 Architecture

We implemented the Input-Endorsers/FruitChain variant of Minotaur, where transactions are exclusively included in the PoW blocks. In this way, Minotaur can potentially achieve optimal throughput up to the capacity of the network as shown by the implementation of Prism [38], a pure PoW consensus protocol with similar blockchain structure. The system architecture is illustrated in Figure 4. Functionally it can be divided into the following three modules:

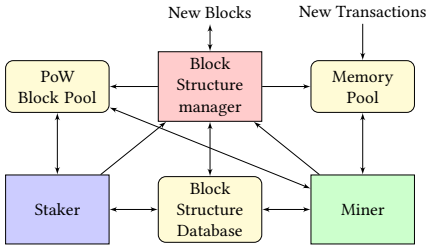


Figure 4: Architecture of our Minotaur client implementation.

(1) *Block Structure Manager*, which maintains the client's view of the blockchain, and communicates with peers to exchange new blocks.
(2) *Miner*, which assembles new PoW blocks.
(3) *Staker*, which assembles new PoS blocks.

The goal of a Minotaur client is to maintain up-to-date information of the blockchain. They are stored in the following three data structures:

(1) *Block Structure Database*, stores the graph structure of the blockchain (i.e., a directed acyclic graph (DAG) formed with PoW and PoS blocks).
(2) *Memory Pool*, stores the set of transactions that have not been mined into any PoW block.
(3) *PoW Block Pool*, stores the set of PoW blocks that have not been referenced by any PoS block.

At the core of the **Block Structure Manager** is an *event loop* which sends and receives network messages to/from peers, and a *worker thread pool* which handles those messages. When a new block arrives, the worker thread first checks its proof-of-work/proof-of-stake, and if valid, then proceeds to relay the block to peers that have not received it. Next, the worker thread checks whether all blocks referenced by the new block, e.g. its parent, are already present in the Block Structure Database. If not, it buffers the block in an in-memory queue and defers further processing until all the referenced blocks have been received. Finally, the worker validates the block (e.g., verifying transaction signatures), and inserts the block into the Block Structure Database. If the block is a PoW block, the Block Structure Manager checks the Memory Pool against the transactions in this new block and removes any duplicates or conflicts from the Memory Pool, and also puts the block into the PoW Block Pool.

The **Miner** module assembles new PoW blocks. It is implemented as a busy-spinning loop. At the start of each round, it polls the Block Structure Database and the Memory Pool to update the block it is mining. When a new PoW block is mined, it will be inserted into the Block Structure Database, then sent to peers by the Block Structure Manager. The Memory Pool and PoW Block Pool will also be updated accordingly.

The **Staker** module works similarly, but assembles new PoS blocks. At the start of each round, it polls the Block Structure Database and the PoW Block Pool to update the block it is assembling. When a new PoS block is generated, it will be inserted into the Block Structure Database, then sent to peers by the Block Structure Manager. The PoW Block Pool will also be updated accordingly.

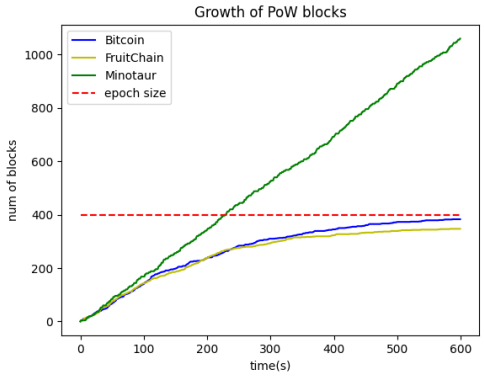


Figure 5: Behavior of Bitcoin, FruitChain **and** Minotaur **under decreasing mining power. When the total mining power is halved every two minutes, the main chains of** Bitcoin **and** FruitChain **stop growing as they can never reach the epoch size (400 blocks), while the number of PoW blocks referenced by the main chain grows linearly in** Minotaur**.**

7.2 Performance under variable mining power

In §6.4, we have seen that Minotaur can survive more drastic variations of network hash power. We design the following experiment on our Minotaur/Bitcoin/FruitChain codebase to verify this argument.

Recall that all three protocols have epoch-based target adjustment rules, varying the difficulty target of block mining based on the median inter-block time from the previous epoch. However, the epoch length in Minotaur is defined as a fixed number of slots instead of a fixed number of blocks (e.g., 2016 blocks in Bitcoin). This definition makes sense in Minotaur because the main-chain blocks, which are PoS blocks, always have accurate timestamps (even if they are proposed by an adversary). In this experiment, we will see that this small change allows Minotaur to enjoy better liveness than Bitcoin and FruitChain when the total mining power is decreasing.

In our experiment, we set the epoch length in Bitcoin/FruitChain to be 400 blocks and the expected duration of an epoch is 2 minutes; while the epoch length of Minotaur is 2 minutes and the expected number of blocks in a epoch is 400 blocks. Then Bitcoin, FruitChain and Minotaur are experiencing the same mining power variation, i.e., starting with an insufficient mining rate (100 blocks per minute) and then halving the mining rate every epoch (2 minutes in the experiments). From Figure 5, we can see that the mining target never has a chance to be adjusted in Bitcoin/FruitChain as the length of the main chain can never reach 400 blocks, which leads to a liveness failure; while the mining targets are adjusted every epoch (2 minutes) in Minotaur to make sure that the number of PoW blocks referenced by the main chain grows linearly.

7.3 Performance under attacks

In the following experiments, we evaluate how Minotaur performs in the presence of private-chain and selfish-mining attacks. At a high level, a private-chain attack aims to violate the persistence of the protocol, while selfish mining aims to compromise (reward) fairness.

Private chain attack. In a private-chain attack, an attacker tries to privately generate an alternate chain faster than the public honest chain to displace a confirmed block [25] (e.g., a block buried k-deep in the longest chain). In a pure PoW/PoS protocol (e.g., Bitcoin/Ouroboros), an attacker with majority of mining/stake power can always succeed with the private-chain attack, no matter how large the confirmation depth is. In contrast, Minotaur can survive such an attack, as long as the honest players control a majority of the combined resources.

To verify this key security property, we implemented the private-chain attack on our code base, where an attacker can have various combinations of stake and mining power. The experiments of the private-chain attack are as follows: the attacker (with β_s fraction of stake and β_w fraction of mining power) tries to generate a private chain starting at a specific block (e.g., the first block in some epoch) to overwrite the honest public chain when the private chain becomes longer. We run the attack for one epoch and count the length of the longest private chain (pruning honest blocks in the prefix if there is any) whose length is no less than that of the honest chain generated at the same time. Note that the maximum length of the private chain in a successful attack reflects the minimum confirmation depth that is needed to guarantee security of the protocol. We chose $\omega = 0.5$ with a total PoS block rate being 1.58 block/s and a total PoW block rate being 2.70 block/s for Minotaur. Parameters β_s and β_w are both iterated in $[0, 1]$ with a step size of 0.1 for Minotaur experiments, and we present the results as a 2D heatmap in Figure 6. We also launched the same attack on Ouroboros Praos (Bitcoin resp.), where the attacker controls β_s fraction of stake (β_w fraction of mining power resp.) and the results are shown as 1D heatmaps in Figure 6. In these experiments, block generation is set to the same rate as PoS block rate in Minotaur experiments. For each data point, we repeated the experiments 10 times and took an average to minimize randomness effects. From Figure 6 we observe that as long as $\beta_s + \beta_w < 1$, the attacker can only succeed with very short private chain even when either β_s or β_w is close to 1, while Ouroboros Praos and Bitcoin would need $\beta_s < 0.5$ and $\beta_w < 0.5$ respectively. This provides empirical evidence that Minotaur is better than pure PoW or PoS protocols in term of security.

Selfish-mining attack. It has been known that Bitcoin is vulnerable to the selfish-mining attack [13, 26, 34], where a selfish miner withhold its mined blocks and release them later at an appropriate time to take the place of honest blocks in the longest chain. This attack hurts the fairness of the protocol, in the sense that the selfish miner with β_w fraction of mining power can have more than β_w faction of blocks in the main chain so that it will reap higher revenue. The FruitChain protocol [29] was proposed as a solution to selfish mining. We observe that the selfish-mining attack will not work on Minotaur Since it has similar blockchain structure as FruitChain. Moreover, Minotaur will achieve fairness even when $\beta_w \geq 0.5$ (as long as there is an honest super-majority in actual stake), while FruitChain would fail. To verify these observations, we implemented the following selfish-mining attacks on our Bitcoin/FruitChain/Minotaur code base:

- **On** Bitcoin**:** The selfish miner always mines on the block at the tip of the longest chain, whether the chain is private or public. Upon successful mining, the adversary maintains the block

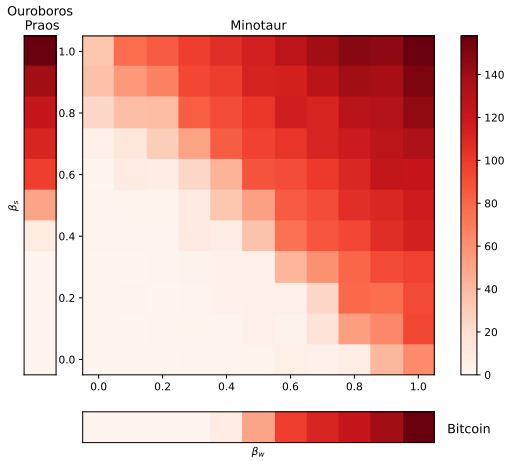


Figure 6: Longest private chain in private-chain attacks with various adversarial stake fraction β_s and adversarial mining power fraction β_w on Minotaur, Ouroboros Praos, and Bitcoin. When $\beta_s + \beta_w < 1$ and either β_s or β_w is close to 1, the private chain in Minotaur is way shorter, meaning that it remains secure whereas Ouroboros Praos and Bitcoin are insecure.

in private to release it at an appropriate time. In particular, when an honest miner publishes a block, the selfish miner will release a previously mined block at the same level (if it has one).

- **On FruitChain:** The selfish miner plays the same withholding-and-releasing strategy on its main-chain blocks as above, while its main-chain blocks contain fruits mined by itself exclusively.
- **On Minotaur:** The selfish miner plays the same withholding-and-releasing strategy on its PoS blocks as above, while its PoS blocks contain PoW blocks mined by itself exclusively.

We assume honest nodes will choose the attacker's block with probability p whenever there is a tie. In our experiments, we set $p > 0.5$ to simulate the case that the attacker has better network connection than honest nodes so its blocks are usually transmitted faster. Table 2 presents fractions of PoW blocks (or fruits) in the main chain of Bitcoin/FruitChain/Minotaur under various adversarial power (combinations of β_w and p). On Minotaur, the attacker controls 1/3 of the virtual stake, i.e., $\omega\beta_w + (1-\omega)\beta_s = 1/3$, where β_s is the fraction of the actual stake controlled by the attacker. In Table 2, we can see that Bitcoin is indeed vulnerable to selfish mining, meaning that the attacker has more fraction of PoW blocks in the main chain (and thus more block rewards in Bitcoin) than its fraction of mining power, particularly when $\beta_w \geq 0.5$; and FruitChain only resists the attack when $\beta_w < 0.5$. Meanwhile, in Minotaur, the attacker can only have β_w faction of PoW blocks in the main chain, no matter how much mining power it has (even when $\beta_w \geq 0.5$), and how the network favors it. This property is crucial to the security of Minotaur as we have seen in §6.2.2.

8 DISCUSSION

Variable weighing parameter. In our security analysis, we assume a fixed weighing parameter ω through the whole execution of the protocol. However, we point out that ω can change over time as long as we put proper assumptions on the adversarial stake/mining

Attacker's power		Fraction of PoW blocks				
β_w		0.75	0.67	0.50	0.33	0.25
$p = 1$	Bitcoin	0	0	0.007	0.513	0.669
	FruitChain	0	0.007	0.011	0.661	0.739
	Minotaur	0.248	0.332	0.498	0.665	0.746
$p = 0.7$	Bitcoin	0	0.001	0.098	0.618	0.72
	FruitChain	0	0.003	0.056	0.665	0.753
	Minotaur	0.248	0.333	0.499	0.666	0.750

Table 2: Fractions of honest PoW blocks (or fruits) in the main chain under selfish-mining attacks in Bitcoin/FruitChain/Minotaur. In these experiments, the attacker controls β_w fraction of mining power and the tie breaking rule favors the attacker's block with probability p.

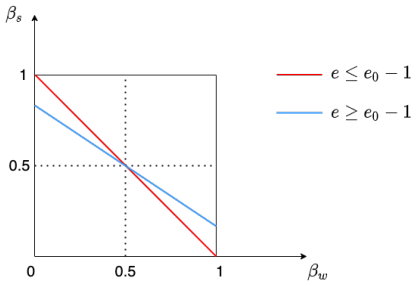


Figure 7: An example of variable weighing parameter ω in Minotaur. Suppose the value of ω changes from $1/2$ to $2/5$ at the onset of epoch $e_0 + 1$. Then the adversarial stake/mining power needs to be restricted below the red line in epochs $e \leq e_0 - 1$ and below the blue line in epochs $e \geq e_0 - 1$. In particular, the adversary is restricted by both lines in epoch $e_0 - 1$. When ω changes smoothly over time, it is reasonable to assume exactly this.

power. Suppose $\omega(e)$ is the weighing parameter of epoch e, i.e., the ratio of work stake and actual stake in epoch e is set to be $\omega(e)/(1-\omega(e))$. Note that the function $\omega(e)$ may be decided by the protocol designers and hard-coded in the genesis block, but the players can also reach an agreement (off-chain) to update it by doing a soft fork.

We give a brief security sketch, deferring the full analysis to future work. Recall that, in our protocol, the virtual stake of a player in epoch e composes of two parts: the actual stake drawn from the last PoS block in epoch $e-2$; and the work stake that is proportional to the amount of work it produced during epoch $e-2$. Thus, to guarantee the security in epoch e, we need the adversary to be $(1/2-2\sigma, 2R, \omega(e))$-bounded (see Def. 2) in epoch $e-2$ and epoch $e-1$. And similarly, the adversary needs to be $(1/2-2\sigma, 2R, \omega(e+1))$-bounded in epoch $e-1$ and epoch e. Therefore, the adversarial stake/mining power must be restricted by both above bounds for epoch $e-1$. As long as $\omega(e+1)$ does not differ too much from $\omega(e)$, this restriction on the adversary is reasonable to assume, and the weighing parameter can transit smoothly from $\omega(e)$ to $\omega(e+1)$. Figure 7 gives an example of how the assumptions shift when $\omega(e)$ is updated at the onset of epoch $e_0 + 1$.

We point out that such flexible weighing between work and stake is very useful in practice. For example, in general, PoS blockchains

are easy to launch with existing techniques, such as proof- of burn [20], initial coin offering [23] and airdrop [1]. Therefore, Minotaur can be launched as a pure PoS blockchain and later transit into a hybrid PoW/PoS one or a pure PoW blockchain. In addition, the security of Minotaur can be enhanced by assigning higher weight to the more decentralized resource.

Removing the initial constraint. As discussed in §2, we use an honest majority of stake without relying on PoW for the two initial epochs. We note that we can relax this initial constraint harmonizing even the initial two epochs with the rest of the execution by employing the smooth-interpolation technique (§4, Idea 2) where work blocks are mined based on 2-for-1 PoW [18] producing both, main-chain blocks (at a rate defined by ω) and endorser blocks (at a sufficient rate to guarantee fairness). This allows to start the protocol with any initial weighing by means of ω. We do not pursue this further as, (I) for most practical purposes we expect the present construction to be sufficient as is (namely it will be reasonable to launch Minotaur with PoW "turned off", at $\omega = 0$ and then begin to incrementally adjust ω to higher values at some time after two epochs), and, (II) as explained in §4, the analysis of the variation described above that allows $\omega > 0$ from launch will follow in a straightforward manner from the forkable string analysis and input-endorser technique of [3, 21] as well as the fairness argument of [31]. This is because the main difficulty in the analysis of PoW/PoS hybrid protocols is dealing with variable difficulty — and adopting the $\omega = 0$ convention for the initial two epochs allows us to focus on that.

Generalization to multiple resources. Following the idea in [36], Minotaur can be extended to more than two resources. In contrast to their construction, this is achievable without fundamentally changing the structure of the protocol.

Assume $M \geq 1$ different resources, and for each such resource, an independent lottery mechanism among the contributors to assign "successes" proportionally to a party's ratio of the total contributed resource. Defining a fixed amount of virtual stake V and weights $\omega_i > 0$, $\sum_{i=1}^{M} \omega_i = 1$, we have resource i control $\omega_i \cdot V$ of the virtual stake. During each epoch e, the different lotteries are run concurrently, wherein each success allows for the release of a respective block (to be eventually picked up by a block of the main chain), tied to the respective resource.

Main-chain block leadership for epoch $e + 2$ is assigned to resource i based on relative virtual stake ω_i, and further split among the contributors of that resource based on their production of respective "resource blocks" during epoch e. Naturally, this protocol tolerates

$$\sum_{i=1}^{M} \omega_i \beta_i < \frac{1}{2}, \tag{1}$$

where $\beta_i \in [0, 1]$ is the fraction of resource i held by the adversary. A pictorial example for $M = 3$ is given in Figure 8.

Although the underlying blockchain protocol is PoS-based, stake (in the classical, non-virtual sense) is not required to be one of the M resources, i.e., the protocol can be run solely on virtual stake. Also, full adversarial control of some of the resources can be tolerated as long as Eq. (1) is satisfied.

Finally, note that the resources can also be of the same type, e.g., Minotaur can combine work emanating from different hash functions such as SHA256, scrypt, and ethash—answering an open question raised in §1.

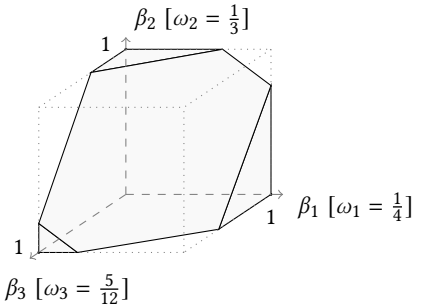


Figure 8: Tolerable resource corruption for $\omega = (\frac{1}{4}, \frac{1}{3}, \frac{5}{12})$. The volume enclosed by the gray surfaces is tolerated by the protocol: $\sum_{i=1}^{3} \omega_i \beta_i < \frac{1}{2}$.

Fairness with respect to the combined resources. Fairness with respect to PoW blocks (Definition 8) is a crucial property of Minotaur to guarantee fair assignment of work stake to the miners. However, this is not sufficient for aspects like fair reward sharing [31], since, for this purpose, fairness needs to be achieved for overall block production with respect to the *combined* resources. We note that fairness with respect to the combined resources can be achieved by scheduling endorser blocks for every involved resource, *including stake*. A given epoch reward can now be shared by assigning an ω_i fraction of the reward to the contributors of the i-th resource, and distributing each such fraction proportionally to the parties' contributions of endorser blocks for that resource.

Comparison to state-of-the-art protocols. In comparison to previous hybrid PoW-PoS protocols, [11] already generalized the conditions under which a permissionless blockchain can be securely operated by combining PoW and PoS. Our paper improves over [11] by demonstrating how to tolerate a strictly better bound, i.e., *any* adversarial minority of the combined resources, and furthermore proving this bound tight, thus settling the hybrid PoW-PoS question.

We also note that Minotaur's performance is comparable to previous state-of-the-art protocols:

- *State-of-the-art PoS.* As Minotaur resembles a PoS protocol (while operating on virtual stake derived from PoW), the only overhead it introduces over its underlying PoS protocol is the extra (insignificant) effort required for virtual-stake conversion—which impacts system throughput or transaction latency only marginally. The performance of our explicit Minotaur construction, based on Ouroboros, is thus directly inherited from the Ouroboros protocol.
- *State-of-the-art PoW.* Note that Nakamoto-style PoS mimics the workings of PoW Nakamoto consensus and thus yields similar stochastics as in the PoW case, and thus also comparable performance characteristics.

ACKNOWLEDGMENTS

We would like to thank Thomas Kerber and Alexander Russell for insightful discussions.

This research is supported in part by the US National Science Foundation under grants CCF-1705007 and CNS-1718270 and the US Army Research Office under grant W911NF1810332.

REFERENCES

[1] Airdrop. https://en.bitcoinwiki.org/wiki/Airdrop.

[2] Back, A. Hashcash—a denial of service counter-measure.

[3] Badertscher, C., Gaži, P., Kiayias, A., Russell, A., and Zikas, V. Ouroboros Genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018), pp. 913–930.

[4] Bentov, I., Lee, C., Mizrahi, A., and Rosenfeld, M. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review 42*, 3 (2014), 34–37.

[5] Blum, E., Kiayias, A., Moore, C., Quader, S., and Russell, A. The combinatorics of the longest-chain rule: Linear consistency for proof-of-stake blockchains. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020* (2020), S. Chawla, Ed., SIAM, pp. 1135–1154.

[6] Buterin, V., and Griffith, V. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437* (2017).

[7] Chen, L., Xu, L., Gao, Z., Lu, Y., and Shi, W. Protecting early stage proof-of-work based public blockchain. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)* (2018), IEEE, pp. 122–127.

[8] David, B., Gaži, P., Kiayias, A., and Russell, A. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2018), Springer, pp. 66–98.

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

[10] Duong, T., Chepurnoy, A., Fan, L., and Zhou, H.-S. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts* (2018), pp. 1–13.

[11] Duong, T., Fan, L., Katz, J., Thai, P., and Zhou, H.-S. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. In *European Symposium on Research in Computer Security* (2020), Springer, pp. 697–712.

[12] Ethereum. Ethash. https://web.archive.org/web/20220105100244/https://eth.wiki/en/concepts/ethash/ethash, 2022.

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

[14] Fitzi, M., Wang, X., Kannan, S., Kiayias, A., Leonardos, N., Viswanath, P., and Wang, G. Minotaur: Multi-resource blockchain consensus. *IACR ePrint Archive* (2022), 104.

[15] Garay, J., and Kiayias, A. Sok: A consensus taxonomy in the blockchain era. In *Cryptographers' Track at the RSA Conference* (2020), Springer, pp. 284–318.

[16] Garay, J., Kiayias, A., and Leonardos, N. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference* (2017), Springer, pp. 291–323.

[17] Garay, J. A., Kiayias, A., and Leonardos, N. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II* (2015), E. Oswald and M. Fischlin, Eds., vol. 9057 of *Lecture Notes in Computer Science*, Springer, pp. 281–310.

[18] Garay, J. A., Kiayias, A., and Leonardos, N. Full analysis of nakamoto consensus in bounded-delay networks. *IACR Cryptol. ePrint Arch. 2020* (2020), 277.

[19] Karakostas, D., and Kiayias, A. Securing proof-of-work ledgers via checkpointing. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2021), IEEE, pp. 1–5.

[20] Karantias, K., Kiayias, A., and Zindros, D. Proof-of-burn. In *International Conference on Financial Cryptography and Data Security* (2020), Springer, pp. 523–540.

[21] Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference* (2017), Springer, pp. 357–388.

[22] King, S., and Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August 19*, 1 (2012).

[23] Li, J., and Mann, W. Initial coin offering and platform building. *SSRN Electronic Journal* (2018), 1–56.

[24] Minotaur. Rust implementation of the minotaur consensus protocol. https://github.com/xuechao2/Minotaur.

[25] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review* (2008), 21260.

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

[27] Neu, J., Tas, E. N., and Tse, D. Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In *2021 IEEE Symposium on Security and Privacy (SP)* (2021), IEEE, pp. 446–465.

[28] NIST. Fips 180-2. http://csrc.nist.gov/encryption/tkhash.html, 2002.

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

[30] Pass, R., and Shi, E. Hybrid consensus: Efficient consensus in the permissionless model. *Cryptology ePrint Archive* (2016).

[31] Pass, R., and Shi, E. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing* (2017), pp. 315–324.

[32] Percival, C., and Josefsson, S. The scrypt password-based key derivation function. *IETF Draft URL: http://tools. ietf. org/html/josefsson-scrypt-kdf-00. txt (accessed: 30.11. 2012)* (2016).

[33] Sankagiri, S., Wang, X., Kannan, S., and Viswanath, P. Blockchain cap theorem allows user-dependent adaptivity and finality. *arXiv preprint arXiv:2010.13711* (2020).

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

[35] Stewart, A., and Kokoris-Kogia, E. Grandpa: a byzantine finality gadget. *arXiv preprint arXiv:2007.01560* (2020).

[36] Thai, P., Njilla, L., Duong, T., Fan, L., and Zhou, H.-S. A generic paradigm for blockchain design. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (2018), pp. 460–469.

[37] Wang, X., Muppirala, V. V., Yang, L., Kannan, S., and Viswanath, P. Securing parallel-chain protocols under variable mining power. *arXiv preprint arXiv:2105.02927* (2021).

[38] Yang, L., Bagaria, V., Wang, G., Alizadeh, M., Tse, D., Fanti, G., and Viswanath, P. Prism: Scaling bitcoin by 10,000 x. *arXiv preprint arXiv:1909.11261* (2019).