Blockchain Nash Dynamics and the Pursuit of Compliance

Dimitris Karakostas
University of Edinburgh
dkarakos@inf.ed.ac.uk

Aggelos Kiayias
University of Edinburgh and IOG
akiayias@inf.ed.ac.uk

Thomas Zacharias
University of Edinburgh
tzachari@inf.ed.ac.uk

ABSTRACT

We study "Nash dynamics" in the context of adversarial deviations in blockchain protocols. We introduce a formal model, within which one can assess whether the Nash dynamics can lead utility-maximizing participants to defect from the "honest" protocol operation, towards variations that exhibit one or more undesirable *infractions* that affect protocol security, like abstaining from participation and producing conflicting protocol histories. Blockchain protocols that lead to no such infraction states are deemed *compliant*. Armed with this model, we evaluate the compliance of various Proof-of-Work (PoW) and Proof-of-Stake (PoS) protocol families, under different utility functions and reward schemes, leading to the following results: i) PoW and PoS protocols exhibit different compliance behavior, depending on the lossiness of the network; ii) PoS ledgers can be compliant w.r.t. one realistic infraction (producing conflicting messages) but non-compliant (hence non-equilibria) w.r.t. others (abstaining or an attack we call selfish signing); iii) considering externalities, like exchange rate fluctuations, we quantify the benefit of economic penalties in the context of PoS protocols as mitigation for particular infractions that affect protocol security.

CCS CONCEPTS

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

KEYWORDS

blockchain, Nash dynamics, nothing-at-stake

1 INTRODUCTION

The advent of Bitcoin [52] brought the economic aspects of consensus protocols to the forefront. While classical literature in consensus primarily dealt with fail-stop or "Byzantine error models" [56], the pressing question post-Bitcoin is whether the participants' incentives align with what the consensus protocol asks them to do. This question is crucial since, if a large number of participants deviate from the protocol, all its known security properties become moot. Motivated by this, some works investigated if Bitcoin is an equilibrium under certain conditions [36, 43]. Others pinpointed deviations that are more profitable for some players, assuming others follow the protocol [19, 58]. The literature also includes tweaks towards improving the blockchain protocol in various settings [21, 42], game-theoretic studies of pooling behavior [1, 45], and equilibria that involve abstaining from the protocol [23] in high cost scenarios. Beyond consensus, economic mechanisms have also been considered in the context of multi-party computation to disincentivize "cheating" [13, 44]. Finally, various works optimized particular attacks e.g., : i) optimal selfish mining [58]; ii) quantitatively evaluating blockchain parameters and identifying optimal strategies for selfish mining and double-spending under network delays [28]; iii) strategies more profitable than selfish mining [53].

Though these works provide glimpses on these protocols' behavior from a game-theoretic perspective, they offer little guidance on how to design and parameterize new consensus protocols. This problem is of high importance, given the negative light shed on Bitcoin's energy consumption and carbon footprint [48] and the need for more efficient designs. Proof-of-Stake (PoS) is currently the most prominent alternative to Bitcoin's Proof-of-Work (PoW) mechanism. To create an acceptable block of processed transactions, PoW requires the expenditure of computational power. In contrast, PoS relies on each party's owned assets ("stake"), so blocks are created at (virtually) no cost beyond transaction processing. Interestingly, while it is proven that PoS protocols are Byzantine resilient [10, 29, 39] and are even equilibria under certain conditions [39], their security is heavily contested by PoW proponents via an economic argument termed *nothing-at-stake* [17, 46, 49]. This argument asserts that PoS ledgers' maintainers can maximize their expected rewards by producing conflicting blocks when possible.

What merit do these criticisms have? Participating in a blockchain protocol is a voluntary action that involves a participant downloading the software and committing resources to run it. Given the open source nature of these protocols, nothing prevents the participant from modifying the behaviour of the software in some way and engaging with the other parties via a modified strategy. There are a number of undesirable adjustments that a participant can do, e.g., i) run the protocol intermittently instead of continuously; ii) not extend the most recent ledger of transactions they are aware of; iii) extend simultaneously more than one ledgers of transactions. One can consider the above as fundamental *infractions* to the protocol's rules with potential serious security implications, in terms of both the consistency and the liveness of the underlying ledger.

To address these issues, many systems introduce additional mechanisms on top of standard incentives, frequently with only rudimentary formal analysis. These include: i) rewards for "uncle blocks" (Ethereum), i.e., blocks which are not part of the canonical chain [37]; ii) stake delegation (EOS, Polkadot, Cardano [34]), where users assign their participation rights to delegates or stake pools; iii) penalties for misbehavior, also referred to as *"slashing"* (Ethereum 2.0 [8, 9]). Unfortunately, the lack of thorough analysis of these mechanisms is, naturally, a serious impediment to wider adoption. For instance, consider penalties. Employing multiple replicas, for redundancy and crash-fault tolerance, may result in conflicting blocks, due to either a faulty configuration, if two replicas come alive simultaneously, or even software or hardware bugs. However,

AFT '22, September 19–21, 2022, Cambridge, MA, USA

ACM ISBN 978-1-4503-9861-9/22/09.
https://doi.org/10.1145/3558535.3559781

if a party employs no failover mechanism and experiences network connectivity issues, it may fail to participate. This highlights the flip side of such penalty mechanisms: participants may choose to not engage, (e.g., to avoid the risk of forfeiting funds or due to insufficient funds to make a deposit), or, if they do engage, they may steer clear of fault-tolerant good sysadmin practices, which could pose quality of service concerns and hurt the system in the long run.

The above considerations put forth the fundamental question that motivates our work: *How effective are blockchain protocol designs in disincentivizing particularly adverse protocol infractions?* In more detail, the question we ask is whether selfish behavior can lead to specific types of deviations, taking a blockchain protocol as the initial point of reference of honest — "compliant" — behavior.

Our Contributions and Roadmap. Our main question relates to the Nash dynamics of blockchain protocols. In the classical Nash dynamics problem [57], the question is whether allowing selfish players to perform step-wise payoff-improving moves leads the system to an equilibrium, and in how many steps this may happen; e.g., [20] considers the case of congestion games. In this perspective, the action space can be seen as a directed graph, where vertices represent vectors of player strategies and edges correspond to player moves. Notably, deciding whether the Nash dynamics converge to a (Nash or sink) equilibrium is particularly difficult, often being a NP-hard or PSPACE-complete problem [50].

This work adapts Nash dynamics to the setting of blockchain protocols, with a particular focus on studying specific undesirable protocol infractions. Importantly, instead of asking for convergence, we ask whether the "cone" in the directed graph positioned at the protocol contains strategies from a given infraction set $\mathcal{X}$ (Figure 1). If the cone is free of infractions, the protocol is deemed $\mathcal{X}$-compliant. In turn, we also consider ϵ-Nash-dynamics [12], i.e., considering only steps in the graph which represent best responses and improve the participant's payoff more than ϵ. Armed with this model, we investigate various protocols from a compliance perspective.

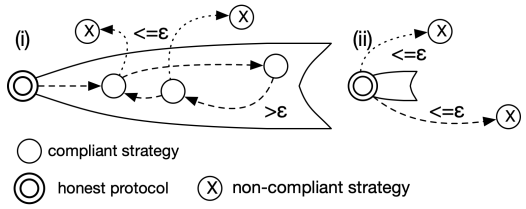

Figure 1: Illustration of a compliant protocol that does not exhibit an equilibrium (i), vs a protocol which is an approximate Nash equilibrium (ii).

A core motivation of our work is that $\mathcal{X}$-compliance enables to validate the incentive structure of a blockchain protocol w.r.t. specific disruptive behaviors (as captured by $\mathcal{X}$), while abstracting away any deviations that avoid the infractions. In this sense, $\mathcal{X}$-compliance of a protocol is a weaker notion compared to a Nash equilibrium, allowing a variety of possible protocol deviations as long as they do not fall into $\mathcal{X}$. This also enables a two-tiered analysis where compliance analysis rules out crucial deviations, while the set of all compliant behaviors can be analyzed in, say, worst-case fashion. Moreover, negative compliance results are immediately informative as they identify one or more specific infractions. This is helpful from the point of view of mechanism parameterisation for blockchain protocols, where various penalties (e.g., reward reduction or slashing of funds) are typically employed to mitigate specific deviant behaviors. So far, there exists no framework that enables a formal argument as to whether a specific penalty is sufficient to mitigate a certain behavior. Our work provides such framework and we illustrate its applicability in this setting, by analyzing an array of Nakamoto longest chain protocol families.

In detail, our paper is organized as follows.[1] Section 2 describes our model of *compliant* strategies and protocols. A strategy is compliant if a party that employs it never violates a predicate $\mathcal{X}$, which captures well-defined types of deviant behavior. Assuming a starting point where no party deviates and then parties choose their strategies in sequential steps, a protocol is compliant if no party eventually employs a non-compliant strategy. Section 3 describes compliance for blockchain protocols, proposing different infraction predicates that capture abstaining and producing conflicting blocks, and two types of utility, reward and profit (rewards minus cost). We then explore different reward schemes and protocol families. First, we analyze *block-proportional* rewards w.r.t. the chain adopted by an impartial observer of the system. Section 4.1 shows that PoW systems are compliant w.r.t. reward. Section 4.2.1 shows that single-leader PoS systems, i.e., which enforce that a single party participates at a time, are compliant under a synchronous network, but non-compliant under a lossy network (contrary to PoW). Section 4.2.2 shows that multi-leader PoS systems, i.e., which allow multiple parties to produce blocks for the same time slot, are not compliant. Notably, the latter result shows that a party can gain a *non-negligible* reward by being non-compliant under a certain network routing assumption, also highlighting the way the network interacts with protocol incentives. Section 5.1 shows that *resource-proportional* rewards, i.e., which depend only on a party's power, result in non-compliance w.r.t. profit. Section 5.2 highlights the distinction between compliance and Nash equilibria by showcasing a typical PoS protocol that is compliant w.r.t. an infraction predicate that captures realistic deviations (conflicting blocks) and non-compliant w.r.t. another realistic predicate (abstaining), hence not a Nash equilibrium. Next, in Section 6 we consider relative profits (as an extension to relative rewards [19, 36]) and introduce a deviation we call "selfish signing". Under block-proportional rewards, we show that the same PoS protocol is non-compliant w.r.t. to this deviation, but is compliant w.r.t. to conflicting blocks. Finally, Section 7 considers a varying exchange rate of the platform's underlying token, which models real-world prices, and external rewards, which come as a result of successful attacks. We show that penalties would be needed if a deviant behavior is synergistic to mounting the attacks, and we provide estimations for such penalties w.r.t. the ledger's parameters and the market's expected behavior.

[1] Due to space constraints, we refer to the extended version for an enhanced analysis, including proofs of all theorems [35].

2 COMPLIANCE MODEL

We assume a distributed protocol Π, which is executed by a set of parties $\mathbb{P}$ over a number of time slots. Every party $\mathcal{P} \in \mathbb{P}$ is activated on each time slot, following a schedule set by an environment $\mathcal{Z}$, which also provides the parties with inputs. Each party $\mathcal{P} \in \mathbb{P}$ is associated with a number $\mu_{\mathcal{P}} \in [0, 1]$. $\mu_{\mathcal{P}}$ identifies $\mathcal{P}$'s *percentage* of participation power in the protocol, e.g., its hashing or staking power, so $\sum_{\mathcal{P} \in \mathbb{P}} \mu_{\mathcal{P}} = 1$. We use the following notation: i) κ: Π's security parameter; ii) $\text{negl}(\cdot)$: a negligible function (asymptotically smaller than the inverse of any polynomial); iii) $[n]$: the set $\{1, \ldots, n\}$; iv) $E[X]$: the expectation of random variable X.

2.1 Preliminaries

We assume a peer-to-peer network, where parties communicate using the following variant of a *diffuse* functionality (cf. [27]).

Router. The *router* $\mathcal{A}$ is a special party that, on each time slot, retrieves all created messages and decides their order and time of delivery. In essence, $\mathcal{A}$ models the underlying communication network. Following, we consider three properties of interest for routers: i) *synchronous*: all messages are delivered at the end of the round during which they were created; ii) *lossy*: a message is omitted by $\mathcal{A}$, i.e., it is never delivered to any recipient, with probability d;[2] iii) *uniform*: the order of message delivery is uniformly randomized. The first two properties are mutually exclusive, but they might be combined with the third. Specifically, when considering plainly a synchronous router, we do not impose restrictions on the message ordering. Nonetheless, a synchronous uniform router is a special case of a synchronous router (equiv. for lossy uniform).

Diffuse Functionality. The functionality, parameterized by a router $\mathcal{A}$, initializes (to 1) a variable *slot* readable by all parties. It also maintains a string $\text{Receive}_{\mathcal{P}}()$ for each party $\mathcal{P}$. $\mathcal{P}$ is allowed to fetch the contents of $\text{Receive}_{\mathcal{P}}()$ at the beginning of each time slot. To diffuse a (possibly empty) message m, $\mathcal{P}$ sends to the functionality m, which records it. On each slot, every party completes its activity by sending a special Complete message to the functionality. When all parties submit Complete, the functionality delivers the messages, which are diffused during this slot, as follows. First, it sends all messages to $\mathcal{A}$. Following, $\mathcal{A}$ responds with a list of tuples $\langle \mathcal{P}, l_{\mathcal{P}} \rangle$, where $\mathcal{P} \in \mathbb{P}$ and $l_{\mathcal{P}}$ is an ordered list of messages. Subsequently, the functionality includes all messages in $l_{\mathcal{P}}$, following its specified order, in the $\text{Receive}_{\mathcal{P}}()$ string of $\mathcal{P}$. Hence, the received messages contain no information on each message's creator. Finally, the functionality increases the value of *slot* by 1.

Approximate Nash Equilibrium. An approximate Nash equilibrium is a common tool for expressing a solution to a non-cooperative game involving n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Each party $\mathcal{P}_i$ employs a strategy S_i. The strategy is a set of rules and actions the party makes, depending on what has happened up to any point in the game, i.e., it defines the part of the entire distributed protocol Π performed by $\mathcal{P}_i$. There exists an "honest" strategy, defined by Π, which parties may employ; for ease of notation, Π denotes both the distributed protocol and the honest strategy. A *strategy profile* is a vector of all players' strategies. Each party $\mathcal{P}_i$ has a game *utility* U_i, which is a real function that takes as input a strategy profile. A strategy profile is an ϵ-Nash equilibrium when no party can increase its utility more than ϵ by *unilaterally* changing its strategy (Definition 2.1).

Definition 2.1. Let: i) ϵ be a non-negative real number; ii) $\mathbb{S}$ be the set of strategies a party may employ; iii) $\sigma^* = (S_i^*, S_{-i}^*)$ be a strategy profile of $\mathbb{P}$, where S_i^* is the strategy followed by $\mathcal{P}_i$; iv) S_{-i}^* denote the $n-1$ strategies employed by all parties except $\mathcal{P}_i$. We say that σ^* is an *ϵ-Nash equilibrium* w.r.t. a utility vector $\bar{U} = \langle U_1, \ldots, U_n \rangle$ if: $\forall \mathcal{P}_i \in \mathbb{P}\ \forall S_i \in \mathbb{S} \setminus \{S_i^*\} : U_i(S_i^*, S_{-i}^*) \geq U_i(S_i, S_{-i}^*) - \epsilon$.

For simplicity, when all parties have the same utility U, we say that σ^* is an ϵ-Nash equilibrium w.r.t. U. We also say that Π is an ϵ-Nash equilibrium w.r.t. U when $\sigma_\Pi = \langle \Pi, \ldots, \Pi \rangle$, where all parties follow the honest strategy, is an ϵ-Nash equilibrium for U.

2.2 Basic Notions

A protocol's execution $\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma,r}$ until a given time slot r is probabilistic and parameterized by: i) the environment $\mathcal{Z}$; ii) a router $\mathcal{A}$; iii) the strategy profile σ of the participating parties. As discussed, $\mathcal{Z}$ provides the parties with inputs and schedules their activation. For notation simplicity, when r is omitted, $\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}$ refers to the end of the execution, which occurs after polynomially many time slots.

An *execution trace* $\mathfrak{I}_{\mathcal{Z},\mathcal{A},\sigma,r}$ until a time slot r is the value that the random variable $\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma,r}$ takes for a fixed: i) environment $\mathcal{Z}$; ii) router $\mathcal{A}$; iii) profile σ; iv) random coins of $\mathcal{Z}$, each party $\mathcal{P} \in \mathbb{P}$, and every protocol-specific oracle (see below). A party $\mathcal{P}$'s view of an execution trace $\mathfrak{I}^{\mathcal{P}}_{\mathcal{Z},\mathcal{A},\sigma,r}$ consists of the messages that $\mathcal{P}$ has sent and received until slot r. For notation simplicity, we omit the subscripts $\{\mathcal{Z}, \mathcal{A}, \sigma, r\}$ from $\mathcal{E}$ and $\mathfrak{I}$, unless required for clarity.

The protocol Π defines two components, which are related to our analysis: (1) the oracle $\mathcal{O}_\Pi$, and (2) the "infraction" predicate $\mathcal{X}$.

The Oracle $\mathcal{O}_\Pi$. The oracle $\mathcal{O}_\Pi$ provides the parties with the core functionality needed to participate in Π. For example, in a Proof-of-Work (PoW) system, $\mathcal{O}_\Pi$ is the random or hashing oracle, whereas in an authenticated Byzantine Agreement protocol, $\mathcal{O}_\Pi$ is a signing oracle. On each time slot, a party can perform at most a polynomial number of queries to $\mathcal{O}_\Pi$; in the simplest case, each party can submit a single query per slot. Finally, $\mathcal{O}_\Pi$ is *stateless*, i.e., its random coins are decided upon the beginning of the execution and its responses do not depend on the order of the queries.

The Infraction Predicate $\mathcal{X}$. The infraction predicate $\mathcal{X}$ abstracts the deviant behavior that the analysis aims to capture. Given the execution trace and a party $\mathcal{P}$, $\mathcal{X}$ responds with 1 only if $\mathcal{P}$ deviates from Π in some well-defined manner. Definition 2.2 provides the core generic property of $\mathcal{X}$, i.e., that honest parties never deviate. With hindsight, our analysis will focus on infraction predicates that capture either producing conflicting messages or abstaining.

Definition 2.2. The infraction predicate $\mathcal{X}$ has the property that, for every execution trace $\mathfrak{I}$ and for every party $\mathcal{P} \in \mathbb{P}$, if $\mathcal{P}$ employs the (honest) strategy Π then $\mathcal{X}(\mathfrak{I}, \mathcal{P}) = 0$.

We stress that Definition 2.2 implies that $\mathcal{X}$ being 0 is a necessary but not sufficient condition for honesty. Specifically, for all honest parties $\mathcal{X}$ is always 0, but $\mathcal{X}$ might also be 0 for a party that deviates

[2] This router models a network with stochastic delays, where users reject messages delivered with later than a (protocol-specific) limit. For example, protocols like Bitcoin resolve message conflicts via delivery order; thus, delaying a message for long enough, such that a competing message is delivered first, is equivalent to dropping the message.

from Π, in a way not captured by $\mathcal{X}$. In that case, we say that the party employs an $\mathcal{X}-$compliant strategy (Definition 2.3). A strategy profile is $\mathcal{X}-$compliant if all its strategies are $\mathcal{X}-$compliant, so the "all honest" profile σ_Π, where all parties employ Π, is $\mathcal{X}-$compliant.

Definition 2.3 (Compliant Strategy). Let $\mathcal{X}$ be an infraction predicate. A strategy S is *$\mathcal{X}$-compliant* if and only if $\mathcal{X}(\mathfrak{I}, \mathcal{P}) = 0$ for every party $\mathcal{P}$ and for every trace $\mathfrak{I}$ where $\mathcal{P}$ employs S.

The observer Ω. We assume a special party Ω, the *(passive) observer*. This party does not actively participate in the execution, but it runs Π and observes the protocol's execution. Notably, Ω is *always online*, i.e., it bootstraps at the beginning of the execution and is activated on every slot, in order to receive diffused messages. Therefore, the observer models a user of the system, who frequently uses the system but does not actively participate in its maintenance. Additionally, at the last round of the execution, the environment $\mathcal{Z}$ activates only Ω, in order to receive the diffused messages of the penultimate round and have a complete point of view.

2.3 Compliant Protocols

To define the notion of an $(\epsilon, \mathcal{X})$-compliant protocol Π, we require two parameters: (i) the associated infraction predicate $\mathcal{X}$; (ii) a non-negative real number ϵ. Following Definition 2.3, $\mathcal{X}$ determines the set of compliant strategies that the parties may follow in Π. Intuitively, ϵ specifies the gain threshold after which a party switches strategies. In particular, ϵ is used to define when a strategy profile σ' is *directly reachable* from a profile σ, where σ' results from the unilateral deviation of a party $\mathcal{P}_i$ from σ and, by this deviation, the utility of $\mathcal{P}_i$ increases by more than ϵ, while σ' sets a *best response* for $\mathcal{P}_i$. Generally, σ' is *reachable* from σ, if σ' results from a "path" of strategy profiles, starting from σ, which are sequentially related via direct reachability. Finally, we define the *cone* of a profile σ as the set of all strategies that are reachable from σ, including σ itself.

We say that Π is $(\epsilon, \mathcal{X})$-compliant if the cone of the "all honest" strategy profile σ_Π contains only profiles that consist of $\mathcal{X}-$compliant strategies. Thus, if a protocol is compliant, then the parties may (unilaterally) deviate from the honest strategy only in a compliant manner, as dictated by $\mathcal{X}$. Formally, first we define "reachability" between two strategy profiles, as well as the notion of a "cone" of a strategy profile w.r.t. the reachability relation. Then, we define a compliant protocol w.r.t. its associated infraction predicate.

Definition 2.4. Let: i) ϵ be a non-negative real number; ii) Π be a protocol run by parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$; iii) $\bar{U} = \langle U_1, \ldots, U_n \rangle$ be a utility vector, where U_i is the utility of $\mathcal{P}_i$; iv) $\mathbb{S}$ be the set of all strategies a party may employ. We provide the following definitions.

(1) Let $\sigma, \sigma' \in \mathbb{S}^n$ be two strategy profiles where $\sigma = \langle S_1, \ldots, S_n \rangle$ and $\sigma' = \langle S'_1, \ldots, S'_n \rangle$. We say that σ' is *directly ϵ-reachable from σ w.r.t.* $\bar{U}$, if there exists $i \in [n]$ s.t. (i) $\forall j \in [n] \setminus \{i\} : S'_j = S_j$, (ii) $U_i(\sigma') > U_i(\sigma) + \epsilon$, and (iii) for every strategy profile $\sigma'' = \langle S''_1, \ldots, S''_n \rangle$ s.t. $\forall j \in [n] \setminus \{i\} : S''_j = S_j$, it holds that $U_i(\sigma'') \leq U_i(\sigma')$ (σ' sets a best response for $\mathcal{P}_i$).
(2) Let $\sigma, \sigma' \in \mathbb{S}^n$ be two distinct strategy profiles. We say that σ' is *ϵ-reachable from σ w.r.t.* $\bar{U}$, if there exist profiles $\sigma_1, \ldots, \sigma_k$ s.t. (i) $\sigma_1 = \sigma$, (ii) $\sigma_k = \sigma'$, and (iii) $\forall j \in [2, k]$ it holds that σ_j is directly ϵ-reachable from σ_{j-1} w.r.t. $\bar{U}$.
(3) For every strategy profile $\sigma \in \mathbb{S}^n$ we define the *$(\epsilon, \bar{U})$-cone of σ* as the set: $\mathsf{Cone}_{\epsilon,\bar{U}}(\sigma) := \{\sigma' \in \mathbb{S}^n \mid (\sigma' = \sigma) \vee (\sigma' \text{ is } \epsilon\text{-reachable from } \sigma \text{ w.r.t. } \bar{U})\}$.

Definition 2.5. Let: i) ϵ be a non-negative real number; ii) Π be a protocol run by the parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$; iii) $\mathcal{X}$ be an infraction predicate; iv) $\bar{U} = \langle U_1, \ldots, U_n \rangle$ be a utility vector, where U_i is the utility of party $\mathcal{P}_i$; v) $\mathbb{S}$ be the set of all strategies a party may employ; vi) $\mathbb{S}_\mathcal{X}$ be the set of $\mathcal{X}-$compliant strategies.

A strategy profile $\sigma \in \mathbb{S}^n$ is $\mathcal{X}-$compliant if $\sigma \in (\mathbb{S}_\mathcal{X})^n$.

The *$(\epsilon, \bar{U})$-cone of* Π, denoted by $\mathsf{Cone}_{\epsilon,\bar{U}}(\Pi)$, is the set $\mathsf{Cone}_{\epsilon,\bar{U}}(\sigma_\Pi)$, i.e., the set of all strategies that are ϵ-reachable from the "all honest" strategy profile $\sigma_\Pi = \langle \Pi, \ldots, \Pi \rangle$ w.r.t. $\bar{U}$, including σ_Π.

Π is *$(\epsilon, \mathcal{X})$-compliant w.r.t. $\bar{U}$* if $\mathsf{Cone}_{\epsilon,\bar{U}}(\Pi) \subseteq (\mathbb{S}_\mathcal{X})^n$, i.e., all strategy profiles in the $(\epsilon, \bar{U})$-cone of Π are $\mathcal{X}-$compliant.

2.4 Compliance and Approximate Nash Equilibria

In this subsection, we show that a protocol is an ϵ-Nash equilibrium w.r.t. some utility if and only if it is $(\epsilon, \mathcal{X})$-compliant w.r.t. the same utility, for any associated infraction predicate $\mathcal{X}$. We begin by a useful lemma stating that a protocol is an approximate Nash equilibrium if and only if the cone of the protocol includes only the all-honest strategy profile.

Lemma 2.6. *Let: i) ϵ be a non-negative real number; ii) Π be a protocol run by the parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$; iii) $\bar{U} = \langle U_1, \ldots, U_n \rangle$ be a utility vector, with U_i the utility of $\mathcal{P}_i$. Then, Π is an ϵ-Nash equilibrium w.r.t. $\bar{U}$ (i.e., $\sigma_\Pi = \langle \Pi, \ldots, \Pi \rangle$ is an ϵ-Nash equilibrium w.r.t. $\bar{U}$) if and only if the $(\epsilon, \bar{U})$-cone of Π, $\mathsf{Cone}_{\epsilon,\bar{U}}(\Pi)$, is the singleton $\{\sigma_\Pi\}$.*

The statement in Lemma 2.6 resembles the well-known statement that a pure Nash equilibrium is a sink equilibrium that contains a single strategy profile [30]. Nonetheless, there are differences between the notions of a sink equilibrium and a cone. Recall that a sink equilibrium is a strongly connected component of the strategy profile graph that has no outgoing edges. On the other hand, according to Definition 2.4, the subgraph induced by the nodes of a cone of a strategy profile σ may not even be strongly connected (e.g., the cone could be a subtree rooted at σ). By applying Lemma 2.6, we offer this subsection's main result.

Theorem 2.7. *Let: i) ϵ be a non-negative real number; ii) Π be a protocol run by the parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$; iii) $\bar{U} = \langle U_1, \ldots, U_n \rangle$ be a utility vector, with U_i the utility of $\mathcal{P}_i$. Then, Π is an ϵ-Nash equilibrium w.r.t. $\bar{U}$ if and only if Π is $(\epsilon, \mathcal{X})$-compliant w.r.t. $\bar{U}$ for any associated infraction predicate $\mathcal{X}$.*

According to the equivalence proven in Theorem 2.7, the property that "a protocol is an approximate Nash equilibrium" can be interpreted as a composition of all possible statements that "it is not in any party's interest to be non-compliant", however compliance is specified by the associated infraction predicate.

Remark. It is easy to see that $\mathcal{X}$-compliance is a *strict relaxation* of the approximate Nash equilibrium notion. For example, consider a protocol Π^* that is not an ϵ^*-Nash equilibrium w.r.t. some utility $\bar{U}^*$ (cf. Theorem 5.2 for such a counterexample). Now set $\mathcal{X}^*$ to be the predicate that always returns 0. Clearly, by Definitions 2.3 and 2.5, the protocol Π^* is $(\epsilon^*, \mathcal{X}^*)$-compliant w.r.t. $\bar{U}^*$.

3 BLOCKCHAIN PROTOCOLS

In this work, we focus on blockchain-based distributed ledger protocols. In the general case, a ledger defines a global state, which is distributed across multiple parties and is maintained via a consensus protocol. The distributed ledger protocol defines the validity rules which allow a party to extract the final ledger from its view. A blockchain is a distributed database, where each message m is a block $\mathcal{B}$ of transactions and each transaction updates the system's global state. Therefore, at any point of the execution, a party $\mathcal{P}$ holds some view of the global state, which comprises of the blocks that $\mathcal{P}$ has adopted. We note that, if at least one valid block is diffused (w.r.t. the validity rules of the protocol), then every honest party can extract a final ledger from its execution view.

3.1 The Setting

Every blockchain protocol Π defines a *message validity* predicate $\mathcal{V}$. Party $\mathcal{P}$ accepts block $\mathcal{B}$, received during a time slot r, if $\mathcal{V}(\mathfrak{I}_r^{\mathcal{P}}, \mathcal{B}) = 1$. For example, in Proof-of-WorK (PoW) systems like Bitcoin, a block is valid if its hash is below a certain threshold; in Proof-of-Stake (PoS) protocols like Ouroboros [39], a block is valid if it was created by a specific party, given a known leader schedule. In all cases, $\mathcal{B}$ is valid if its creator submits at least one query for $\mathcal{B}$ to O_Π.

Each block $\mathcal{B}$ is associated with the following metadata: i) an index $index(\mathcal{B})$; ii) the party $creator(\mathcal{B})$ that created $\mathcal{B}$; iii) a set $ancestors(\mathcal{B}) \subseteq \mathfrak{I}^{creator(\mathcal{B})}$, i.e., blocks in the view of $creator(\mathcal{B})$ (at the time of $\mathcal{B}$'s creation) referenced by $\mathcal{B}$. Message references are implemented as hash pointers, given a hash function H employed by the protocol. Specifically, each block $\mathcal{B}$ contains the hash of all blocks in the referenced blocks $ancestors(\mathcal{B})$. Blockchain systems are typically bootstrapped via a global common reference string, i.e., a "genesis" block $\mathcal{B}_G$. Therefore, the blocks form a hash tree, stemming from $\mathcal{B}_G$ and $index(\mathcal{B})$ is the height of $\mathcal{B}$ in the hash tree. If $\mathcal{B}$ references multiple messages, i.e., belongs to multiple tree branches, $index(\mathcal{B})$ is the height of the longest one.

The protocol also defines the *message equivalency operator*, $\equiv$. Specifically, two messages are equivalent if their hashes match, i.e., $m_1 \equiv m_2 \Leftrightarrow \mathsf{H}(m_1) = \mathsf{H}(m_2)$. At a high level, two equivalent messages are interchangeable by the protocol.

Infraction Predicate. In our analysis of blockchain systems, we will consider two types of deviant behavior (Definition 3.1): i) creating conflicting valid messages of same origin, and ii) abstaining. We choose these predicates because they may lead to non-compliance in interesting use cases. The former refers to the widely discussed topic in blockchain systems of one participant extending two conflicting transaction histories. The latter deals with the issue of participants who intermittently engage in the system's maintenance, thus potentially hurting the safety of the deployed system; in particular, the more users participate in maintenance, the higher the level of resources that an adversary needs to reach to break a system's security. Other infraction predicates are of course also possible to define.

Definition 3.1 (Blockchain Infraction Predicate). Given party $\mathcal{P}$ and execution trace $\mathfrak{I}$, we define the following infraction predicates:

(1) *conflicting predicate*: $\mathcal{X}_{\text{conf}}(\mathfrak{I}, \mathcal{P}) = 1$ if there exist blocks $\mathcal{B}, \mathcal{B}' \in \mathfrak{I}$ s.t. $creator(\mathcal{B}) = creator(\mathcal{B}') = \mathcal{P} \wedge \mathcal{V}(\mathfrak{I}^{\mathcal{P}}, \mathcal{B}) = \mathcal{V}(\mathfrak{I}^{\mathcal{P}}, \mathcal{B}') = 1 \wedge index(\mathcal{B}) = index(\mathcal{B}') \wedge \mathcal{B} \not\equiv \mathcal{B}'$;
(2) *abstaining predicate*: $\mathcal{X}_{\text{abs}}(\mathfrak{I}, \mathcal{P}) = 1$ if there exists a time slot r such that $\mathcal{P}$ makes *no* queries to oracle O_Π during r;
(3) *blockchain predicate*: $\mathcal{X}_{\text{bc}}(\mathfrak{I}, \mathcal{P}) = 1$ if $(\mathcal{X}_{\text{conf}}(\mathfrak{I}, \mathcal{P}) = 1) \vee (\mathcal{X}_{\text{abs}}(\mathfrak{I}, \mathcal{P}) = 1)$.

We note that preventing conflicting messages is not the same Sybil attack resilience [15]. The latter restricts an attacker from creating multiple identities. Instead, our infraction predicate ensures that a user does not increase their utility by creating conflicting messages with one of its identities. Thus, a system may be compliant but not Sybil resilient, e.g., if a party participates via multiple identities without increasing its utility via conflicting messages.

Finally, at the end of the execution, the observer Ω outputs a chain $C_{\Omega,\mathfrak{I}}$. Typically, this is the longest valid chain, i.e., the longest branch of the tree that stems from genesis $\mathcal{B}_G$.[3] In case multiple longest chains exist, a choice is made either at random or following a chronological ordering of messages. The number of messages in $C_{\Omega,\mathfrak{I}}$ that are created by a party $\mathcal{P}$ is denoted by $M_{\mathcal{P},\mathfrak{I}}$.

3.2 Utility: Rewards and Costs

For each execution, the blockchain protocol defines a number of total rewards, which are distributed among the participating parties. For each party $\mathcal{P}$, these rewards are expressed via the *reward random variable* $R_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}$. For a specific trace $\mathfrak{I}_{\mathcal{Z},\mathcal{A},\sigma}$, the random variable takes a non-negative real value, denoted by $R_{\mathcal{P},\mathfrak{I}_{\mathcal{Z},\mathcal{A},\sigma}}$. Intuitively, $R_{\mathcal{P},\mathfrak{I}_{\mathcal{Z},\mathcal{A},\sigma}}$ describes the rewards that $\mathcal{P}$ receives from the protocol from the point of view of the observer Ω, i.e., w.r.t. the blocks output by Ω at the end of the execution.

Our analysis is restricted to systems where rewards are distributed to parties if and only if the genesis block is extended by at least one block during the execution, in which case at least one party receives a non-negative amount of rewards (Assumption 1).

Assumption 1. *Let $\mathfrak{I}$ be an execution trace. If no block is produced during $\mathfrak{I}$, then it holds that $\forall \mathcal{P} \in \mathbb{P} : R_{\mathcal{P},\mathfrak{I}} = 0$. If at least one block is produced during $\mathfrak{I}$, then it holds that $\exists \mathcal{P} \in \mathbb{P} : R_{\mathcal{P},\mathfrak{I}} \neq 0$.*

In addition to rewards, a party's utility is affected by cost. The *cost random variable* $C_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}$ expresses the operational cost of $\mathcal{P}$ during an execution $\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}$. For a fixed trace $\mathfrak{I}_{\mathcal{Z},\mathcal{A},\sigma}$, $C_{\mathcal{P},\mathfrak{I}_{\mathcal{Z},\mathcal{A},\sigma}}$ is a non-negative real value. Our analysis is restricted to cost schemes which are *linearly monotonically increasing* in the number of queries that a party makes to the oracle O_Π, with no queries incurring zero cost (Assumption 2). Intuitively, this assumption considers the electricity cost of participation, while the cost of equipment and other operations, such as parsing or publishing messages, is zero.

Assumption 2. *For every execution trace $\mathfrak{I}$, a party $\mathcal{P}$'s cost is $C_{\mathcal{P},\mathfrak{I}} = 0$ if and only if it performs no queries to O_Π in every time slot. Else, if during $\mathfrak{I}$ a party $\mathcal{P}$ performs t queries, then its cost is $C_{\mathcal{P},\mathfrak{I}} = t \cdot \lambda$, for some fixed parameter λ.*

[3] We assume that the longest chain (in blocks) contains the most hashing power, which is the metric used in PoW systems.

We define two types of utility. First is *Reward*, i.e., the expected rewards that a party receives when the cost is 0. Second is *Profit*, i.e., rewards minus participation cost.

Definition 3.2. Let σ be a strategy profile and $\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}$ be an execution during which parties follow σ. We define two types of blockchain utility $U_{\mathcal{P}}$ of a party $\mathcal{P}$ for σ:

(1) *Reward*: $U_{\mathcal{P}}(\sigma) = E[R_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]$
(2) *Profit*: $U_{\mathcal{P}}(\sigma) = E[R_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}] - E[C_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]$

For the computation of $U_{\mathcal{P}}$, the environment $\mathcal{Z}$ and the router $\mathcal{A}$ are fixed. Therefore, the expectation of the random variables $R_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}$ and $C_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}$ is computed over the random coins of $\mathcal{Z}$, $\mathcal{A}$, O_{Π}, and every party $\mathcal{P} \in \mathbb{P}$. Intuitively, utility depends on both the strategy choice and the underlying network (expressed via the router). As such, different routers may yield different optimal strategies for parties and possibly different equilibria ceteris paribus. Following, we evaluate the compliance of various Proof-of-Work (PoW) and Proof-of-Stake (PoS) blockchain protocols w.r.t. two types of rewards, *block-proportional* and *resource-proportional*.

4 BLOCK-PROPORTIONAL REWARDS

The arguably most common type of rewards in blockchain systems is *block-proportional* rewards. Each party is rewarded proportionally to the number of blocks it contributes to the final chain, at the end of the execution. Block-proportional rewards are a generalization of the *proportional allocation rule*, which, for example, is employed in Bitcoin. The proportional allocation rule states that a party $\mathcal{P}$'s *expected* rewards of a single block are $\mu_{\mathcal{P}}$. As shown by Chen *et al.* [11], this is the unique allocation rule that satisfies a list of desirable properties, namely: i) non-negativity, ii) budget-balance, iii) symmetry, iv) sybil-proofness, and v) collusion-proofness.

Our work expands the scope by considering proportional rewards w.r.t. blocks for the entirety of the execution. Specifically, Definition 4.1 describes block-proportional rewards, where a party $\mathcal{P}$'s rewards are *strictly monotonically increasing* on the number of blocks that $\mathcal{P}$ contributes to the chain output by the observer Ω. The definition considers a *proportional reward function* $\varrho(\cdot,\cdot)$ that takes as input the chain of Ω and $\mathcal{P}$ and outputs a value in $[0,1]$.

Definition 4.1 (Block-Proportional Rewards). For an execution trace $\mathfrak{I}$, let $C_{\Omega,\mathfrak{I}}$ be the chain output by Ω and $\mathcal{R}_{\Omega,\mathfrak{I}} \in \mathbb{R}_{\geq 0}$ be the total number of rewards which are distributed by the protocol, according to Ω. Let $M_{\mathcal{P},\mathfrak{I}}$ be the number of blocks in the chain output by Ω which are produced by $\mathcal{P}$. A *block-proportional reward* random variable $R_{\mathcal{P},\mathcal{E}}$ satisfies the following conditions:

(1) $\forall \mathfrak{I}\ \forall \mathcal{P} \in \mathbb{P} : R_{\mathcal{P},\mathfrak{I}} = \varrho(C_{\Omega,\mathfrak{I}}, \mathcal{P}) \cdot \mathcal{R}_{\Omega,\mathfrak{I}}$
(2) $\forall \mathfrak{I} : \sum_{\mathcal{P}\in\mathbb{P}} \varrho(C_{\Omega,\mathfrak{I}}, \mathcal{P}) = 1$
(3) $\forall \mathfrak{I}\ \forall \mathcal{P}, \mathcal{P}' \in \mathbb{P} : M_{\mathcal{P},\mathfrak{I}} > M_{\mathcal{P}',\mathfrak{I}} \Rightarrow \varrho(C_{\Omega,\mathfrak{I}}, \mathcal{P}) > \varrho(C_{\Omega,\mathfrak{I}}, \mathcal{P}')$

4.1 Bitcoin

First, we consider the Bitcoin [52] protocol. Bitcoin is a prime example of a family of protocols that links the amount of valid blocks, that each party can produce per execution, with the party's hardware capabilities, including: i) Proof-of-Work systems like Ethereum [60], Bitcoin NG [18], Zerocash [5]; ii) Proof-of-Space [16] and Proof-of-Space-Time [51] protocols.

Bitcoin is an Approximate Nash Equilibrium w.r.t. Reward. . Under our model Bitcoin is a $\Theta(\delta^2)$-Nash equilibrium w.r.t. utility *reward*, where δ is the success probability of each independent block production trial (query) and is a protocol-specific "difficulty" parameter. This result is in agreement with previous works [40, 43], while similar results exist w.r.t. *profit* [2]. Nonetheless, there are a few remarks to be made. A well-known implication from the selfish mining attack [19, 58] is that Bitcoin is not an equilibrium w.r.t. relative rewards. However, selfish mining relies on withholding a block's publication, which is a compliant behavior w.r.t. $\mathcal{X}_{bc}$, Definition 3.1 (a variation for PoS systems is treated in Section 6). Second, our analysis assumes fixed difficulty, while Bitcoin operates under variable difficulty, which is computed on regular intervals depending on the active mining power. Various interesting result exist for the variable difficulty setting: i) [33] showed that Bitcoin is not an equilibrium w.r.t. rewards, as selfish mining is more profitable; ii) [23] showed that Bitcoin is not an equilibrium w.r.t. profit, as miners stop performing *some* hashing queries to artificially reduce the difficulty; iii) [32] showed that Bitcoin is not an equilibrium w.r.t. profit in some cases (depending on the cost mechanism) and seemingly also non-compliant w.r.t. $\mathcal{X}_{\text{abs}}$, as miners take turns shutting down and resuming operations per difficulty adjustment epoch. So, Bitcoin's compliance under alternative utilities, variable difficulty, and alternative infraction predicates (e.g., with flexibility in the amount of hashing queries) is a promising line of future research.

4.2 Proof-of-Stake

Proof-of-Stake (PoS) systems differ from Bitcoin in a few points. Typically, the execution of PoS systems is organized in *epochs*, each consisting of a number l_e of time slots. On each slot, a specified set of parties is eligible to participate. Depending on the protocol, the leader schedule of each epoch may or may not be a priori public.

The core difference with PoW concerns the power $\mu_{\mathcal{P}}$. In PoS, $\mu_{\mathcal{P}}$ represents their *stake*, i.e., the number of coins that $\mathcal{P}$ owns. Stake is dynamic, therefore the system's coins may change hands and the leader schedule of each epoch depends on the stake distribution at the beginning of the epoch.[4] As in Bitcoin, each party participates proportionately to their power, so the expected ratio of slots for which $\mathcal{P}$ is leader over the total number of the epoch's slots is $\mu_{\mathcal{P}}$.

Also, in PoS protocols, the oracle O_{Π} does not perform hashing. Instead, it is parameterized by the leader schedule and typically performs signing. A signature output by O_{Π} is valid if and only if the input message is submitted by the slot leader in time. This introduces two important consequences: i) only the leader can produce valid messages for a given slot; ii) the leader can produce as many valid messages as the number of possible queries to O_{Π}.

Following we use this notation: i) C: a single query's cost; ii) R: the (fixed) reward per block; iii) e: the execution's number of epochs; iv) l_e: the epoch's number of slots; v) $\mu_{\mathcal{P},j}$: $\mathcal{P}$'s power on epoch j.

4.2.1 Single-Leader Proof-of-Stake. As before, we analyze a representative of a family of protocols; the family is single-leader PoS (SL-PoS) and the representative is Ouroboros [39]. The SL-PoS family includes systems like EOS[5] and Ouroboros BFT [38].

[4] In reality, the snapshot of the stake distribution is retrieved at an earlier point of the previous epoch, but we can employ this simplified version without loss of generality.
[5] https://developers.eos.io/welcome/latest/protocol/consensus_protocol

We again utilize the blockchain infraction predicates (cf. Definition 3.1). Ouroboros, as a consensus protocol, does not define rewards. Nonetheless, the Ouroboros implementation (in Cardano) employs block-proportional rewards, so we will also consider fixed rewards per block (cf. Definition 4.1).

On each slot, Ouroboros defines a single party, the "slot leader", as eligible to create a valid message. Specifically, the protocol restricts that a leader cannot extend the chain with multiple blocks for the same slot, therefore all honest parties extend their chain by at most 1 block per slot. The leader schedule is public and is computed at the beginning of each epoch via a secure, publicly verifiable Multi-Party Computation (MPC) sub-protocol, which cannot be biased by any single party. To prevent long-range attacks [7], Ouroboros employs a form of rolling checkpoints ("a bounded-depth longest-chain rule" [39]), i.e., a party ignores forks that stem from a block older than a (protocol-specific) limit from the adopted chain's head (it should be noted that subsequent versions of Ouroboros did not utilize the same logic, cf. Subsection 4.2.2).

Single-Leader Proof-of-Stake Protocol (SL-PoS)

The SL-PoS protocol Π has the following characteristics:

- the execution is organized in epochs;
- within each epoch, a single party (the *leader*) is eligible to produce a message per index;
- a party which is online considers the blocks of each past epoch finalized (i.e., does not remove them in favor of a competing, albeit possibly longer, chain);
- no party $\mathcal{P}$ with $\mu_{\mathcal{P}} < \frac{1}{2}$ can bias the epoch's leader schedule.

Figure 2: A generic SL-PoS protocol Π.

Synchronous network. First, we assume a diffuse functionality with a synchronous router (cf. Section 2.1). Theorem 4.2 shows that SL-PoS with block-proportional rewards is an ϵ-Nash equilibrium for negligible ϵ (hence, by Theorem 2.7, it is also $(\epsilon, \mathcal{X})$-compliant, $\mathcal{X}$ being any associated infraction predicate); this result is in line with the incentives' analysis of Ouroboros [39]. We remark that [6] explored the same setting (synchronous, longest-chain PoS) and identified a selfish mining-like attack against so-called "predictable" protocols, like Ouroboros. This result was later refined by [22], showing that such attacks are profitable for participants controlling more than 32.5% of total stake. Nonetheless, that line of research targeted relative rewards and relied on withholding a block's publication for some time; here, we consider absolute rewards and leave block withholding for consideration in Section 6.

Theorem 4.2. *Assume: i) a synchronous router $\mathcal{A}$; ii) $\forall \mathcal{P} \in \mathbb{P}$: $\mu_{\mathcal{P}} < \frac{1}{2}$. SL-PoS with block-proportional rewards (Definition 4.1 for fixed block reward R) is an ϵ-Nash equilibrium w.r.t. utility* Reward *and, if $R > C$, is an ϵ-Nash equilibrium w.r.t. utility* Profit, *both for negligible ϵ and under $\mathcal{A}$.*

Lossy network. Second, we assume a lossy, randomized router (cf. Section 2.1).[6] Theorem 4.3 shows that SL-PoS with block proportional rewards is *not* compliant w.r.t. the conflicting infraction predicate $\mathcal{X}_{\text{conf}}$; specifically, it shows that ϵ is upper-bounded by a typically non-negligible value.

Theorem 4.3. *Assume: i) a lossy, randomized router $\mathcal{A}$ with (non-negligible) parameter d (cf. Section 2.1); ii) $\forall \mathcal{P}' \in \mathbb{P}$: $\mu_{\mathcal{P}'} < \frac{1}{2}$; iii) $\mathcal{P}$ is the party with maximum power $\mu_{\mathcal{P}}$ across the execution and $s_{\mathcal{P}} = \sum_{j \in [1,e]} l_e \cdot \mu_{\mathcal{P},j}$ is the expected number of slots for which $\mathcal{P}$ is leader; iv) $(1-d) \cdot R \gg C$. SL-PoS with block-proportional rewards (cf. Definition 4.1) is* not *$(\epsilon, \mathcal{X}_{conf})$-compliant (cf. Definition 2.5) under $\mathcal{A}$ w.r.t. : i) utility* Reward *for (non-negligible) $\epsilon < (d - d^t) \cdot R \cdot s_{\mathcal{P}}$; ii) utility* Profit *for (non-negligible) $\epsilon < ((d-d^t) \cdot R - (t-1) \cdot C) \cdot s_{\mathcal{P}}$, where $t = \lfloor \frac{\ln(\frac{C}{R \cdot \ln(1/d)})}{\ln(d)} \rfloor$.*

The lossy network setting was observed on December 2019, when Cardano released its Incentivized Testnet (ITN)[7]. Stakeholders, i.e., users owning Cardano tokens, participated in PoS by forming stake pools. The ITN used proportional rewards and the SL-PoS execution model. Each pool was elected as a slot leader proportionally to its stake and received its share of an epoch's rewards based on its *performance*, i.e., the produced blocks compared to the *expected* blocks w.r.t. its stake. Thus, pool operators were highly motivated to always produce a block when needed. However, the network was unstable, so forks started to form. In turn, pools were incentivized[8] to "clone" their nodes, i.e., run multiple parallel instances, to increase network connectivity, reduce packet loss, and also extend all possible forks. To make matters worse, this solution both perpetuated forks and created new ones, as clones did not coordinate but produced different blocks, even when extending the same chain.

We note that, although a lossy network may render a PoS protocol non-compliant, the same does not hold for PoW. In the setting of Theorem 4.3, a party produces multiple blocks per slot to maximize the probability that one of them is output by Ω. Notably, since the PoS protocol restricts that at most one block extends the longest chain per slot, these blocks are conflicting. However, PoW ledgers do not enforce such restriction, so a party would instead create multiple consecutive (instead of conflicting) blocks, which yields maximal rewards even under a lossy network.

4.2.2 Multi-Leader Proof-of-Stake. We now turn to multi-leader PoS (ML-PoS) and Ouroboros Praos [14], a representative of a family alongside Ouroboros Genesis [3], Peercoin [41], and Tezos' baking system [59]. These protocols are similar SL-PoS, but with a core difference: multiple parties may be chosen as leaders for the same slot. As Theorem 4.4 shows, ML-PoS protocols are not compliant for block-proportional rewards. The core idea is the same as with SL-PoS under a lossy network: a party is incentivized to produce multiple blocks to decrease the probability that a competing leader's competing block is adopted over their own. We note that, although consensus doesn't enforce a tie-breaking policy for competing messages, parties typically opt for the message

[6] The randomized router is an example, which will be used to prove the negative result of Theorem 4.4. Other routers, which would model arguably more realistic networks, could also be considered to, possibly, achieve compliance results.
[7] https://staking.cardano.org/
[8] https://www.reddit.com/r/cardano/comments/ekncza

that arrives first. The dependency on randomized routing is also worth noting. Since alternative routers could yield positive results, an interesting research direction is to explore the class of routers under which compliance holds, possibly avoiding infractions via specially-crafted peer-to-peer message passing protocols.

Theorem 4.4. *Assume: i) a synchronous, randomized router $\mathcal{A}$ (cf. Section 2.1); ii) $\forall \mathcal{P}' \in \mathbb{P} : \mu_{\mathcal{P}'} < \frac{1}{2}$; iii) $\mathcal{P}$ is the party with maximum power $\mu_{\mathcal{P}}$ across the execution and $s_{\mathcal{P}} = \sum_{j\in[1,e]} l_e \cdot \mu_{\mathcal{P},j}$ is the expected number of slots s.t. $\mathcal{P}$ is leader; iv) $(1-d)\cdot R \gg C$. Let p_I be the (protocol-dependent) probability that multiple leaders are elected in a slot. ML-PoS with block-proportional rewards (Definition 4.1) is* not *$(\epsilon, \mathcal{X}_{conf})$-compliant (Definition 2.5) under $\mathcal{A}$ w.r.t. : i) utility* reward *for (non-negligible) $\epsilon < \frac{p_I}{2}\cdot R\cdot s_{\mathcal{P}}$; ii) utility* profit *for (non-negligible) $\epsilon < (\frac{t-1}{2\cdot(t+1)}\cdot p_I\cdot R-(t-1)\cdot C)\cdot s_{\mathcal{P}}$, where $t = \lfloor\sqrt{\frac{p_I\cdot R}{C}}\rfloor - 1$.*

5 RESOURCE-PROPORTIONAL REWARDS

As described in Section 2, a party $\mathcal{P}$ controls a percentage $\mu_{\mathcal{P}}$ of the system's power. Although this is set at the beginning of the execution, it is not always public. For instance, $\mathcal{P}$ could obscure its amount of hashing power by refraining from performing some queries. In some cases, each party's power is published on the ledger and, for all executions, can be extracted from the observer's chain. This is the case in non-anonymous PoS ledgers, where each party's power, denoted by its assets, is logged in real time on the ledger.

These systems, where power distribution is public, can employ a special type of rewards, *resource-proportional rewards*. Specifically, the system defines a fixed, total number of rewards $\mathcal{R} > 0$. At the end of an execution, if at least one block is created, each party $\mathcal{P}$ receives a percentage $\xi(\mu_{\mathcal{P}})$ of $\mathcal{R}$, where $\xi(\cdot) : [0,1] \to [0,1]$; in the real world, ξ is usually the identity function. If no blocks are created during the execution, then every party gets 0 rewards.

Intuitively, resource-proportional rewards (Definition 5.1) compensate users for investing in the system. Unless no block is created (which typically happens with negligible probability when the parties follow the protocol), the reward level depends *solely* on a party's power, instead of the messages diffused in the execution.

Definition 5.1 (Resource-proportional Rewards). For a total number of rewards $\mathcal{R} \in \mathbb{R}_{>0}$ and $\xi : [0,1] \to [0,1]$ s.t. $\sum_{\mathcal{P}\in\mathbb{P}} \xi(\mu_{\mathcal{P}}) = 1$, a *resource-proportional reward* random variable $R_{\mathcal{P},\mathcal{E}}$ is:

$$\forall \mathfrak{I}\ \forall \mathcal{P} \in \mathbb{P} : R_{\mathcal{P},\mathfrak{I}} = \begin{cases} \xi(\mu_{\mathcal{P}})\cdot\mathcal{R}, & \text{if at least one valid block in } \mathfrak{I} \\ 0, & \text{otherwise} \end{cases}$$

5.1 Cost-induced Non-compliance

Given Definition 5.1, we observe that blockchains with resource-proportional rewards are ϵ-Nash equilibria w.r.t. utility *Reward* (Definition 3.2) for small ϵ. By Theorem 2.7, the latter implies that such protocols are also $(\epsilon, \mathcal{X})$-compliant w.r.t. the same utility for an arbitrary associated predicate $\mathcal{X}$. Intuitively, a party is rewarded the same amount regardless of their protocol-related actions, so nobody can increase their rewards by deviating from the honest strategy.

Nonetheless, when introducing operational costs to analyze profit, a problem arises: a user can simply abstain and be rewarded nonetheless. Such behavior results in a "free-rider problem" [4], where a user reaps some benefits while not under-paying them or not paying at all. Lemma 5.2 formalizes this argument and shows that a blockchain protocol, associated with the abstaining infraction predicate $\mathcal{X}_{\text{abs}}$ (cf. Definition 3.1), under resource-proportional rewards is *not* $(\epsilon, \mathcal{X}_{\text{abs}})$-compliant w.r.t. utility *Profit*, for reasonable values of ϵ.

Lemma 5.2. *Let: i) Π be a blockchain protocol run by the parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$; ii) $\mathcal{A}$ be a synchronous router (cf. Section 2.1); iii) $\bar{U} = \langle U_1, \ldots, U_n\rangle$ be a utility vector, where U_i is the utility* Profit *of party $\mathcal{P}_i$; iv) $\mathcal{R}$ be the total rewards distributed by the protocol; v) $\xi : [0,1] \to [0,1]$ be a resource-proportional reward function; vi) α be the probability that no blocks are produced when all parties follow the honest strategy.*
For $i \in [n]$, also let the following: i) q be the maximum number of queries that a party can make to the oracle $\mathcal{O}_\Pi$ in each time slot. ii) C be the cost of a single query to $\mathcal{O}_\Pi$; iii) C_i be the expected cost of $\mathcal{P}_i$ when $\mathcal{P}_i$ employs Π; iv) β_i be the probability that no blocks are produced when $\mathcal{P}_i$ abstains throughout the entire execution and all the other parties follow Π.
Assume that for every $i \in [n]$, it holds that $C > \beta_i \cdot \xi(\mu_{\mathcal{P}_i})\cdot\mathcal{R}\cdot q$. Then, for every $\epsilon \geq 0$ s.t. $\epsilon < \max_{i\in[n]}\{C_i - (\beta_i-\alpha)\cdot\xi(\mu_{\mathcal{P}_i})\cdot\mathcal{R}\}$, the protocol Π is not *$(\epsilon, \mathcal{X}_{\text{abs}})$-compliant w.r.t. $\bar{U}$.*

Subsequently, we examine the variables of the bound $\max_{i\in[n]}\{C_i - (\beta_i-\alpha)\cdot\xi(\mu_{\mathcal{P}_i})\cdot\mathcal{R}\}$. We note that, in the context of blockchain systems, a "party" is equivalent to a unit of power; therefore, a party $\mathcal{P}$ that controls $\mu_{\mathcal{P}}$ of the total power, in effect controls $\mu_{\mathcal{P}}$ of all "parties" that participate in the blockchain protocol.

To discuss α and β_i, we first consider the liveness property [26] of blockchain protocols. Briefly, if a protocol guarantees liveness with parameter u, then a transaction which is diffused on slot r is part of the (finalized) ledger of every honest party on round $r+u$. Therefore, assuming that the environment gives at least one transaction to the parties, if a protocol Π guarantees liveness unless with negligible probability $\text{negl}(\kappa)$,[9] then at least one block is created during the execution with overwhelming probability (in κ).

Now, we consider α and β_i. The former is negligible, since consensus protocols typically guarantee liveness against a number of crash (or Byzantine) faults, let alone if all parties are honest. The latter, however, depends on $\mathcal{P}_i$'s percentage of power $\mu_{\mathcal{P}_i}$. For instance, consider Ouroboros, which is secure if a deviating party $\mathcal{P}_i$ controls less than $\frac{1}{2}$ of the staking power and all others employ Π. Thus, if $\mu_{\mathcal{P}_i} = \frac{2}{3}$ and $\mathcal{P}_i$ abstains, the protocol cannot guarantee liveness, i.e., it is probable that no blocks are created. However, if $\mu_{\mathcal{P}_i} = \frac{1}{4}$, then liveness is guaranteed with overwhelming probability; hence, even if $\mathcal{P}_i$ abstains, at least one block is typically created. Corollary 5.3 generalizes this argument, by showing that, if enough parties participate, then at least one of them is small enough, such that its abstaining does not result in a system halt, hence it is incentivized to be non-compliant.

Corollary 5.3. *Let Π be a blockchain protocol, with security parameter κ, which is run by n parties, under the same considerations of Theorem 5.2. Additionally, assume that Π has liveness with security threshold $\frac{1}{x}$ in the following sense: for every strategy profile σ, if*

[9] Recall that κ is Π's security parameter, while $\text{negl}(\cdot)$ is a negligible function.

$\sum_{\mathcal{P}\in\mathbb{P}_{-\sigma}} \mu_{\mathcal{P}} < \frac{1}{x}$, *where* $\mathbb{P}_{-\sigma}$ *is the set of parties that deviate from* Π *when* σ *is followed, then* Π *guarantees liveness with overwhelming (i.e.,* $1-\mathsf{negl}(\kappa)$*) probability. If* $x < n$*, then for (non-negligible) values* $\epsilon < \max_{i\in[n]}\{C_i\} - \mathsf{negl}(\kappa)$, Π *is* not $(\epsilon, \mathcal{X}_{\text{abs}})$*-compliant w.r.t.* $\bar{U}$.

The minimal cost $C^{\perp}_{\mathcal{P}_i}$ of (honest) participation for party $\mathcal{P}_i$ depends on the underlying ledger. In PoW systems, where participation consists of repeated computations, cost increases with the percentage of mining power; for instance, controlling 51% of Bitcoin's mining power for 1 hour costs $\$1,700,000$.[10] In PoS systems, cost is typically irrespective of staking power, since participation consists only of monitoring the network and regularly signing messages; for example, a production-grade Cardano node costs $180 per month[11]. Therefore, considering Corollary 5.3, the upper bound of ϵ is typically rather large for PoS systems.

The free-rider hazard is manifested in Algorand[12], a cryptocurrency system that follows the Algorand consensus protocol [10, 29] and employs resource-proportional rewards, as defined above. Its users own "Algo" tokens and transact over a ledger maintained by "participation nodes", which run the Algorand protocol and extend the ledger via blocks. Each user receives a fixed reward[13] per Algo token they own [25], awarded with every new block. Users may also run a participation node, but are not rewarded [24] for doing so, and participation is proportional to the amount of Algos that the user owns. Therefore, a party that owns a few Algos will expectedly abstain from participation in the consensus protocol.

Remark. In summary, resource-proportional rewards in PoS protocols may incentivize users to abstain. This can impact performance, e.g., delaying block production and transaction finalization; in the extreme case, it could result in a "tragedy of the commons" situation [47], where all users abstain and the system grinds to a halt. Interestingly, this section illustrates a difference between PoW and PoS. In PoS systems, each party's power is registered on the ledger, without requiring any action from them. In PoW, power becomes evident only after the party puts their hardware to work. Therefore, the idea behind Theorem 5.2's proof, which relies on abstaining, does not necessarily hold in PoW systems, like Fruitchains [55], that define rewards (approximately) proportional to each party's mining power, as identified by their hashing operations.

5.2 Compliant Non-equilibrium PoS

So far, our positive results w.r.t. compliance relied on showing that the protocol is an equilibrium. We now demonstrate the distinction between the two notions via a protocol that is compliant w.r.t. a non-trivial infraction predicate, but not Nash equilibrium. The said protocol is a simple, yet typical, SL-PoS blockchain and its characteristics are presented in Figure 3.

We consider Π (Figure 3) under resource-proportional rewards (Definition 5.1) and profit and investigate compliance w.r.t. the two types of attacks captured by $\mathcal{X}_{\text{conf}}$ and $\mathcal{X}_{\text{abs}}$ (Definition 3.1). The goal of this study is to show that, under a well-defined interval of approximation factor values, the protocol, although non $\mathcal{X}_{\text{abs}}$-compliant (hence, also non approximate Nash equilibrium), operates in a $\mathcal{X}_{\text{conf}}$-compliant manner. We note that non $\mathcal{X}_{\text{abs}}$-compliance is consistent with Lemma 5.2; In particular, Theorem 5.4 applies Lemma 5.2 and assigns concrete values to the lemma's generic parameters.

[10] https://www.crypto51.app [February 2022]
[11] https://forum.cardano.org/t/realistic-cost-to-operate-stake-pool/40056 [Jan 2022]
[12] https://algorand.foundation
[13] The weekly reward per Algo is 0.00012 Algos. [AlgoExplorer, Feb 2022]

Single-Leader Proof-of-Stake Protocol (cont)

The extended SL-PoS protocol Π (Figure 2) has the following characteristics:

- The slot leaders are randomly elected, directly proportional to their staking power.
- A party $\mathcal{P}$'s staking power $\mu_{\mathcal{P}}$ is fixed across the execution (this always holds under resource-proportional rewards).
- If elected as slot r's leader, $\mathcal{P}$ makes a signing query to $\mathcal{O}_{\Pi}$ and casts the received block at r.
- The single query cost is C, a (typically small) polynomial on the security parameter κ.

Figure 3: A simple, yet typical, extension to the SL-PoS protocol Π of Figure 2.

Theorem 5.4. *Let: i)* Π *be the SL-PoS blockchain protocol specified in Section 5.2; ii)* $\mathcal{A}$ *be a synchronous router; iii)* $\mathcal{R}$ *be the total rewards distributed by the protocol; iv)* $\xi : [0,1] \to [0,1]$ *be the identity resource-proportional reward function, i.e.,* $\xi(\mu_{\mathcal{P}}) = \mu_{\mathcal{P}}$*; v)* $N \geq \kappa^c$ *be the number of time slots of the execution, where* κ *is the security parameter and* c *is a sufficiently large constant; vi)* $\mathcal{P}_{\max}$ *is the party with the maximum staking power* $\mu_{\mathcal{P}_{\max}}$*. If* $\mu_{\mathcal{P}_{\max}} < \frac{1}{2}$*, then the following hold under* $\mathcal{A}$*: i) for every* $\epsilon \geq 0$, Π *is* $(\epsilon, \mathcal{X}_{\text{conf}})$*-compliant w.r.t. utility* Profit*; ii) for* $\epsilon_{\max} := \mu_{\mathcal{P}_{\max}} \cdot N \cdot C - \mu^{N+1}_{\mathcal{P}_{\max}} \cdot \mathcal{R}$ *and every* $\epsilon < \epsilon_{\max}$, Π *is* not $(\epsilon, \mathcal{X}_{\text{abs}})$*-compliant w.r.t. utility* Profit.

6 POS UNDER RELATIVE UTILITIES

We now continue our study of the SL-PoS protocol Π specified in Figure 3. In particular, we also assume that the protocol is *predictable*, i.e., the slot leader schedule for the entire execution is globally known to the parties in advance [6].[14] We consider block-proportional rewards (Definition 4.1) and a different utility that we call *Relative Profit*. This utility is defined as the fraction of the party's expected profit over the aggregate expected rewards of all parties, when the denominator is not 0 (and 0 otherwise).[15] Formally, for a party $\mathcal{P}$ and strategy profile σ and with $\mathcal{E}_{\sigma} = \mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}$:

[14] We can get similar results by studying (sufficiently large) fragments of the execution when the protocol is predictable.
[15] A seemingly plausible alternative approach would be to consider the fraction of the party's expected profit over the aggregate expected profit of all parties, i.e., $U_{\mathcal{P}}(\sigma) = \frac{E[R_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}} - C_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]}{E\left[\sum_{\hat{\mathcal{P}}\in\mathbb{P}} R_{\hat{\mathcal{P}},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}} - \sum_{\hat{\mathcal{P}}\in\mathbb{P}} C_{\hat{\mathcal{P}},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}\right]}$. However, in this approach, the denominator might become negative, so the utility would not provide intuition on the parties' payoffs. By considering only the (always non-negative) aggregate expected rewards in the fraction, we avoid such problematic cases while maintaining relativity.

$$U_{\mathcal{P}}(\sigma) = \begin{cases} \dfrac{E[R_{\mathcal{P},\mathcal{E}_\sigma} - C_{\mathcal{P},\mathcal{E}_\sigma}]}{E\left[\sum_{\hat{\mathcal{P}}\in\mathbb{P}} R_{\hat{\mathcal{P}},\mathcal{E}_\sigma}\right]}, & \text{if } E\left[\sum_{\hat{\mathcal{P}}\in\mathbb{P}} R_{\hat{\mathcal{P}},\mathcal{E}_\sigma}\right] > 0 \\ 0, & \text{if } E\left[\sum_{\hat{\mathcal{P}}\in\mathbb{P}} R_{\hat{\mathcal{P}},\mathcal{E}_\sigma}\right] = 0 \end{cases} =$$

$$= \begin{cases} \dfrac{E[R_{\mathcal{P},\mathcal{E}_\sigma}] - E[C_{\mathcal{P},\mathcal{E}_\sigma}]}{\sum_{\hat{\mathcal{P}}\in\mathbb{P}} E\left[R_{\hat{\mathcal{P}},\mathcal{E}_\sigma}\right]}, & \text{if } \sum_{\hat{\mathcal{P}}\in\mathbb{P}} E\left[R_{\hat{\mathcal{P}},\mathcal{E}_\sigma}\right] > 0 \\ 0, & \text{if } \sum_{\hat{\mathcal{P}}\in\mathbb{P}} E\left[R_{\hat{\mathcal{P}},\mathcal{E}_\sigma}\right] = 0 \end{cases} \tag{1}$$

Relative Profit is an extension of relative rewards [6, 19, 36, 40], where the utility is the fraction of the party's rewards over the total rewards, by now taking non-zero costs into account. As we will shortly prove, Π is $\mathcal{X}_{\text{conf}}$-compliant w.r.t. to Relative Profit, but non-compliant under a type of deviant behavior that we call *selfish singing*, described in Algorithm 1. In selfish signing, a party $\mathcal{P}$ that knows she is going to be elected for $d+1$ consecutive time slots, where d is the *depth* of the specific selfish signing event, can create a fork of $d+1$ consecutive blocks pointing to a block created $d+1$ steps earlier. This results in a new longest chain with the last d blocks of the old chain getting discarded. Since the discarded blocks belonged to other parties, selfish signing strictly improves the relative profit of $\mathcal{P}$.

ALGORITHM 1: Selfish signing of depth $d \geq 1$ during time slots $r, \ldots, r+d$.

Input: A sequence of $d+1$ blocks $\mathcal{B}_{r-d-1} \leftarrow \cdots \leftarrow \mathcal{B}_{r-1}$.
Output: A new sequence of $d+2$ blocks $\mathcal{B}_{r-d-1} \leftarrow \mathcal{B}'_r \leftarrow \cdots \leftarrow \mathcal{B}'_{r+d}$.

for $j \leftarrow 0$ **to** d **do**
  if $j == 0$ **then**
    As leader of slot r, create a new block $\mathcal{B}'_r$ pointing to $\mathcal{B}_{r-d-1}$;
  else
    As leader of slot $r+j$, create a new block $\mathcal{B}'_{r+j}$ pointing to $\mathcal{B}'_{r+j-1}$;

Remark. Selfish signing is akin to the "globally predictable selfish mining" attack presented in [6], with a major difference. In selfish signing, the deviant creates a fork that discards a sequence of past blocks, already created by other parties. In [6], the attacker withholds their blocks in order to discard future blocks, created *after* the attacker's actions. Consequently, selfish signing requires a shorter predictability "window" as, at any point in time, the attacker only needs to know the immediately following slots for which they are leader. Also, an attacker could employ an adaptive selfish mining attack, so as to choose whether to discard some or all of the available blocks, e.g., depending on their content.

Next, we describe S_{self} (Algorithm 2), a strategy where $\mathcal{P}$ takes advantage of Π's predictability and executes selfish signing at maximum depth whenever possible, under the condition that she never abstains or allows her selfish singing actions to discard her own existing blocks. For an execution of N time slots, the input is a string $\text{schedule}_{\mathcal{P}} \in \{0,1\}^N$ defined as follows: for $r \in [N]$, $\text{schedule}_{\mathcal{P}}[r]$ is 1, if $\mathcal{P}$ is the leader of slot r, and 0 otherwise.

Note that the output $\big((r_1, d_1), \ldots, (r_w, d_w)\big)$ of Algorithm 2 fully determines the behavior of $\mathcal{P}$ throughout the execution. Namely, $\mathcal{P}$ acts honestly until time slot r_1 when it performs selfish signing that lasts until $r_1 + d_1$, then it acts honestly at slots $r_1 + (d_1 + 1), \ldots, r_2 - 1$ and at slot r_2 it performs selfish signing that lasts until $r_2 + d_2$, etc.

ALGORITHM 2: The strategy S_{self} for party $\mathcal{P}$.

Input: A string of bits $\text{schedule}_{\mathcal{P}} \in \{0,1\}^N$, where N is the execution's time length and which contains 1 for the slots where $\mathcal{P}$ is leader.
Output: A sequence of pairs $\big((r_1, d_1), \ldots, (r_w, d_w)\big)$ indicating the time slots that selfish signing will take place at the respective depth.

Initialize a list $\text{strategy}_{\mathcal{P}} \leftarrow ()$;
Set $r \leftarrow 1$;
while $r \leq N$ **do**
  if $\text{schedule}_{\mathcal{P}}[r] == 0$ **then**
    Set $r \leftarrow r+1$; /* As no leader of r, $\mathcal{P}$ takes no action */
  else
    Set $k^* \leftarrow \max\{k | (\wedge_{j=0}^{k} \text{schedule}_{\mathcal{P}}[r+j]) == 1\}$;
    Set $\ell^* \leftarrow \max\{\ell | (\vee_{j=1}^{\ell} \text{schedule}_{\mathcal{P}}[r-j]) == 0\}$;
    if $(k^* == 0) \vee (\ell^* == 0)$ **then**
      Set $r \leftarrow r+1$; /* If no selfish signing is possible, $\mathcal{P}$ acts as an honest party */
    else
      Set $d \leftarrow \min\{k^*, \ell^*\}$;
      Add (r, d) to $\text{strategy}_{\mathcal{P}}$; /* $\mathcal{P}$ will execute Algorithm 1 in slot r at depth d */
      Set $r \leftarrow r + (d+1)$;
return $\text{strategy}_{\mathcal{P}}$;

Next, we define the infraction predicate $\mathcal{X}_{\text{self}}$:[16]

$$\mathcal{X}_{\text{self}}(\mathfrak{I}, \mathcal{P}) := \begin{cases} 0, & \text{if } \mathcal{P} \text{ never performs selfish signing in } \mathfrak{I} \\ 1, & \text{otherwise} \end{cases}.$$

Having introduced S_{self} and $\mathcal{X}_{\text{self}}$, we prove Theorem 6.1. We deploy the function $\delta(\mu) = 5 \cdot (1-\mu) \cdot \mu^2 + 6 \cdot (1-\mu)^2 \cdot \mu^3 + 3 \cdot (1-\mu)^2 \cdot \mu^4 + 3 \cdot (1-\mu)^3 \cdot \mu^4$, for $\mu \in (0,1)$. The function $\delta(\mu)$ sets a lower bound on the expected number of blocks that get discarded every 7 consecutive time slots, when a party with staking power μ unilaterally deviates from σ_Π by following S_{self}. It has a maximum at $\mu \approx 0.64469$ and $\delta(0.64469) \approx 1.03001$.

THEOREM 6.1. *Let: i) Π be the SL-PoS blockchain protocol specified in Section 5.2 with block-proportional rewards (Definition 4.1 for fixed block reward R), and assume that Π is also predictable; ii) a synchronous router $\mathcal{A}$; iii) $\mathcal{P}_{\max}$ is the party with the maximum staking power $\mu_{\mathcal{P}_{\max}}$.*

If $R > C$ and $\mu_{\mathcal{P}_{\max}} < \frac{1}{2}$, then the following hold under $\mathcal{A}$: i) for every $\epsilon \geq 0$, Π is $(\epsilon, \mathcal{X}_{\text{conf}})$-compliant w.r.t. utility Relative Profit; ii) for $\epsilon_{\max} := \frac{\mu_{\mathcal{P}_{\max}}}{\frac{7}{\delta(\mu_{\mathcal{P}_{\max}})} - 1} \cdot \frac{R-C}{R}$, where $\delta(\mu) = 5 \cdot (1-\mu) \cdot \mu^2 + 6 \cdot (1-\mu)^2 \cdot \mu^3 + 3 \cdot (1-\mu)^2 \cdot \mu^4 + 3 \cdot (1-\mu)^3 \cdot \mu^4$, and every $\epsilon \leq \epsilon_{\max}$, Π is not $(\epsilon, \mathcal{X}_{\text{self}})$-compliant w.r.t. utility Relative Profit.

7 EXTERNALITIES

In practice, blockchains coexist with other systems, which may affect the participants' behavior. This section enhances our analysis with parameters external to the ledger. We introduce an exchange rate, to account rewards in the same unit as costs, and analyze how it should behave to ensure compliance, assuming infractions yield an external utility, and finally take penalties into account.

7.1 Utility

In distributed ledger systems, rewards are denominated in the ledger's native token, but cost is typically denominated in fiat.

[16] Observe that $\mathcal{X}_{\text{self}}$ is a special case of the generic family of long-range attacks, when a party creates a fork by extending a block other than the longest chain's head.

So far, we assumed that convertibility between the two is fixed. Now, we introduce an *exchange rate* X, between the ledger's token and USD, which can be variable and help estimate a party's utility more precisely. $X_{\mathcal{E}}$ is a random variable, parameterized by a strategy profile σ. For a trace $\mathfrak{I}_\sigma$ under σ, the exchange rate takes a non-negative real value. The exchange rate is applied once, at the end of the execution. Intuitively, this implies that a party eventually sells their rewards at the end of the execution. Therefore, its utility depends on the accumulated rewards, during the execution, and the exchange rate at the end.

So far, we considered protocols in a standalone fashion, analyzing whether they incentivize parties to avoid infractions. In reality, a ledger exists alongside other systems, and a party's utility may depend on parameters external to the distributed ledger. For instance, double spending against Bitcoin is a common hazard, which does not increase an attacker's *Bitcoin rewards*, but awards them external rewards, e.g., goods that are purchased with the double-spent coins.

The external – to the ledger – reward is modeled as a random variable $B_{\mathcal{P},\mathcal{E}_\sigma}$, which takes non-negative integer values. Similarly to the rewards' random variable, it is parameterized by a party $\mathcal{P}$ and a strategy profile σ. The infraction utility is applied once and has the property that, for every trace $\mathfrak{I}$ during which a party $\mathcal{P}$ performs no infraction, $B_{\mathcal{P},\mathfrak{I}} = 0$, so a party receives these external rewards only by performing an infraction.

We define a new utility U, which also takes two forms, *Reward* and *Profit* (Definition 7.1). For rewards, U applies the exchange rate on the protocol rewards and adds the external reward. For profit, it also subtracts the cost. For ease of notation, we set the following: i) $\rho_{\mathcal{P},\sigma} = E[R_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]$; ii) $x_\sigma = E[X_{\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]$; iii) $b_{\mathcal{P},\sigma} = E[B_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]$; iv) $c_{\mathcal{P},\sigma} = E[C_{\mathcal{P},\mathcal{E}_{\mathcal{Z},\mathcal{A},\sigma}}]$. As in Section 3.2, when computing the utility, the environment and the router are fixed.

Definition 7.1. Let: i) σ be a strategy profile; ii) $\mathcal{E}_\sigma$ be an execution under σ; iii) x_σ be the (expected) exchange rate of $\mathcal{E}_\sigma$; iv) $b_{\mathcal{P},\sigma}$ be the (expected) external rewards of $\mathcal{P}$ under σ. We define two types of utility $U_{\mathcal{P}}$ of a party $\mathcal{P}$ for σ under externalities: 1) *Reward*: $U_{\mathcal{P}}(\sigma) = \rho_{\mathcal{P},\sigma} \cdot x_\sigma + b_{\mathcal{P},\sigma}$; 2) *Profit*: $U_{\mathcal{P}}(\sigma) = \rho_{\mathcal{P},\sigma} \cdot x_\sigma + b_{\mathcal{P},\sigma} - c_{\mathcal{P},\sigma}$.

7.2 Compliance

To evaluate compliance under externalities, we will find when a reduction in asset prices counters external rewards, so parties are incentivized to remain compliant. Formally, Theorem 7.2 adapts the compliance bound for SL-PoS under a synchronous network; this analysis can be applied in a similar manner for the positive results of Subsections 4.1 and 5.1.

Theorem 7.2. *Assume i) a synchronous router $\mathcal{A}$ (cf. Section 2.1), ii) the conflicting predicate $\mathcal{X}_{\text{conf}}$, and iii) that $\forall \mathcal{P} \in \mathbb{P} : \mu_{\mathcal{P}} < \frac{1}{2}$. Also let: i) $\mathbb{S}_{-\mathcal{X}_{\text{conf}}}$: the set of all non $\mathcal{X}_{\text{conf}}$-compliant strategies; ii) x_{σ_Π}: the (expected) exchange rate under $\mathcal{E}_{\sigma_\Pi}$; iii) $x_{\sigma_{S_\mathcal{P}}}$: the (expected) exchange rate when only $\mathcal{P}$ employs some non $\mathcal{X}_{\text{conf}}$-compliant strategy $S_\mathcal{P}$; iv) $b_{\mathcal{P},\sigma_{S_\mathcal{P}}}$: the external utility that $S_\mathcal{P}$ yields for $\mathcal{P}$. SL-PoS with block-proportional rewards (cf. Definition 4.1, for fixed block reward R) under the aforementioned externalities and $\mathcal{A}$ is not $(\epsilon, \mathcal{X}_{\text{conf}})$-compliant (cf. Definition 2.5) w.r.t. utility* Reward *(cf. Definition 3.2) and, if $R > C$, it is also not $(\epsilon, \mathcal{X}_{\text{conf}})$-compliant w.r.t. utility* Profit*, if and only if* $\epsilon < \max\{\max_{\mathcal{P}\in\mathbb{P}}\{\max_{S_\mathcal{P}\in\mathbb{S}_{-\mathcal{X}_{\text{conf}}}} \{\rho_{\mathcal{P},\sigma_\Pi} \cdot (x_{\sigma_{S_\mathcal{P}}} - x_{\sigma_\Pi}) + b_{\mathcal{P},\sigma_{S_\mathcal{P}}}\}\}, 0\}$.

The previous sections offer non-compliance, negative results in PoS systems where (a) resource-proportional rewards are employed and (b) a party is incentivized to produce multiple conflicting messages, i.e., under a lossy network or multiple leaders per slot.

Regarding (a), Section 5 shows that resource-proportional rewards ensure compliance under utility *Reward*, but non-compliance regarding profit. Specifically, assuming a minimal participation cost $C^{\perp}_{\mathcal{P}}$, we showed that, if $\mathcal{P}$ abstains, they incur zero cost without any reward reduction. To explore compliance of resource-proportional rewards under externalities, we consider two strategy profiles $\sigma_\Pi, \sigma_{S_\mathcal{P}}$, as before. Notably, $S_\mathcal{P}$ is the abstaining strategy which, as shown in Section 5, maximizes utility in the standalone setting. For the two profiles, the profit for $\mathcal{P}$ becomes $\rho_{\mathcal{P},\sigma_\Pi} \cdot x_{\sigma_\Pi} - c_{\mathcal{P},\sigma_\Pi}$ and $\rho_{\mathcal{P},\sigma_{S_\mathcal{P}}} \cdot x_{\sigma_{S_\mathcal{P}}} + b_{\mathcal{P},\sigma_{S_\mathcal{P}}}$ respectively. Again, in both cases the party's rewards are equal. Therefore, since it holds that $C^{\perp}_{\mathcal{P}} \leq c_{\mathcal{P},\sigma_\Pi}$, $\mathcal{P}$ is incentivized to be $(\epsilon, \mathcal{X}_{\text{conf}})$-compliant (for some ϵ) if:

$$\rho_{\mathcal{P},\sigma_{S_\mathcal{P}}} \cdot x_{\sigma_{S_\mathcal{P}}} + b_{\mathcal{P},\sigma_{S_\mathcal{P}}} \leq \rho_{\mathcal{P},\sigma} \cdot x_{\sigma_\Pi} - c_{\mathcal{P},\sigma_\Pi} + \epsilon \Rightarrow$$
$$\Rightarrow C^{\perp}_{\mathcal{P}} + b_{\mathcal{P},\sigma_{S_\mathcal{P}}} \leq \rho_{\mathcal{P},\sigma_\Pi} \cdot (x_{\sigma_\Pi} - x_{\sigma_{S_\mathcal{P}}}) + \epsilon$$

If the abstaining strategy yields no external rewards, as is typically the case, $b_{\mathcal{P},\sigma_{S_\mathcal{P}}} = 0$, so the exchange rate needs to only counterbalance the minimal participation cost.

Regarding (b), we consider SL-PoS under a lossy network, since the analysis is similar for ML-PoS, and two strategy profiles $\sigma_\Pi, \sigma_{S_\mathcal{P}}$ as above. Now, under $\sigma_{S_\mathcal{P}}$, $\mathcal{P}$ produces k blocks during each slot for which it is leader, to increase the probability that at least one of them is output in the observer's final chain. Also, for simplicity, we set $b_{\mathcal{P},\sigma_{S_\mathcal{P}}} = 0$. These PoS systems become $(\epsilon', \mathcal{X}_{\text{conf}})$-compliant (for $\epsilon' = \frac{\epsilon}{(1-d^k\cdot(1-d)^2)\cdot R_{max}}$) if $x_{\sigma_{S_\mathcal{P}}} \leq \frac{1-d\cdot(1-d)^2}{1-d^k\cdot(1-d)^2} \cdot x_{\sigma_\Pi} + \epsilon'$ where $R_{max} = R \cdot \sum_{i\in[1,e]} l_e \cdot \mu_{\mathcal{P},i}$ and d, R, l_e are as in Subsection 4.2.2.

7.3 Penalties

Historically, attacks are profitable, so the market's response is typically insufficient to incentivize compliance. Interestingly, in many occasions the external utility was so high that, even if the exchange rate became 0, it would exceed the amount of lost rewards. Therefore, an additional form of utility reduction is necessary to prevent any specific infraction that is essential to mount long-range attacks. In many PoS systems, like Casper [8, 9] and Tezos [59], utility reduction is implemented via penalties. Thus, a party $\mathcal{P}$ deposits an amount of assets $g_\mathcal{P}$, which is forfeited if it violates a defined condition.

Consider profiles $\sigma_\Pi, \sigma_{S_\mathcal{P}}$ as before. Under penalties, with σ_Π, $\mathcal{P}$ receives $\rho_{\mathcal{P},\sigma_\Pi}$ and retains its deposit $g_\mathcal{P}$, both exchanged at rate x_{σ_Π}. With $\sigma_{S_\mathcal{P}}$, $\mathcal{P}$ forfeits its rewards and deposit, but receives external utility $b_{\mathcal{P},\sigma_{S_\mathcal{P}}}$. Thus, compliance is incentivized if the deposit and rewards are larger than the external utility. Note that the ϵ bound of Theorem 7.3 is tighter than Theorem 7.2's, so penalties can make infractions less appealing.

Theorem 7.3. *Assume i) a synchronous router $\mathcal{A}$ (cf. Section 2.1), ii) the conflicting predicate $\mathcal{X}_{\text{conf}}$, and iii) that $\forall \mathcal{P} \in \mathbb{P} : \mu_{\mathcal{P}} < \frac{1}{2}$. Also let: i) $\mathbb{S}_{-\mathcal{X}_{\text{conf}}}$: the set of all non-compliant strategies; ii) x_{σ_Π}: the (expected) exchange rate under σ_Π; iii) $x_{\sigma_{S_\mathcal{P}}}$: the (expected) exchange rate when only $\mathcal{P}$ employs some non-compliant conflicting strategy $S_\mathcal{P}$; iv) $b_{\mathcal{P},\sigma_{S_\mathcal{P}}}$: the external utility that $S_\mathcal{P}$ yields for $\mathcal{P}$; v) and $\rho_{\mathcal{P},\sigma} =$*

$E[R_{\mathcal{P},\mathcal{E}_\sigma}]$, *i.e., the expected rewards of* $\mathcal{P}$ *under profile* σ. *Finally, assume the block-proportional rewards (cf. Definition 4.1 for fixed block reward* R*) for which it also holds: for all* $\mathfrak{I}$ *and for all* $\mathcal{P} \in \mathbb{P}$, *if* $\mathcal{P}$ *produces no conflicting blocks during* $\mathfrak{I}$ *then* $R_{\mathcal{P},\mathfrak{I}} = \varrho(C_{\Omega,\mathfrak{I}}, \mathcal{P}) \cdot \mathcal{R}_{\Omega,\mathfrak{I}} + g_{\mathcal{P}}$, *otherwise* $R_{\mathcal{P},\mathfrak{I}} = 0$, *where* $g_{\mathcal{P}}$ *is a protocol-specific deposit value. SL-PoS with the above rewards and under the aforementioned externalities is not* $(\epsilon, \mathcal{X}_{\text{conf}})$*-compliant (cf. Definition 2.5) w.r.t. utility* Reward *(cf. Definition 3.2) and, if* $R > C$, *it is also not* $(\epsilon, \mathcal{X}_{\text{conf}})$*-compliant w.r.t. utility* Profit, *in both cases under* $\mathcal{A}$ *and if and only if* $\epsilon < \max\{\max_{\mathcal{P}\in\mathbb{P}}\{\max_{S_{\mathcal{P}}\in\mathbb{S}_{\neg\mathcal{X}_{\text{conf}}}} \{b_{\mathcal{P},\sigma_{S_{\mathcal{P}}}}\}\} - \rho_{\mathcal{P},\sigma_\Pi} \cdot x_{\sigma_\Pi}, 0\} - \text{negl}(\kappa)$.

The ϵ bounds in Theorems 7.2 and 7.3 depend on the external utility boost $b_{\mathcal{P},\sigma_{S_{\mathcal{P}}}}$. This highlights the inherent limitations of such systems' designers, since the bound depends on external (to the protocol) parameters. Intuitively, these bounds show that attacks which utilize the $\mathcal{X}_{\text{conf}}$ infraction can be prevented in two ways. First, larger deposits increase some attacks' profitability threshold. However, they also shut off small parties, with inadequate assets. Therefore, a tradeoff exists in preventing such attacks and enabling participation. Second, the longer an attack's duration, the more blocks an adversary needs to produce, hence the larger the rewards that it forfeits. Typically, the attack's duration depends on the required number of confirmations for a transaction to be finalized. So different confirmation limits, e.g., based on a transaction's value, could satisfy the tradeoff between fast settlement and security.

Considering the latter observation, we now briefly review users' behavior in SL-PoS (cf. Section 4.2.1) under deposits and penalties. In an SL-PoS execution, the percentage of parties that actively participate during each epoch is identifiable via the block density and the number of empty slots (when no block is diffused). Therefore, it is possible to estimate the level of double-signing that a party needs to perform to mount a double-spending attack, and then enforce a transaction finalization rule to dis-incentivize such attacks.

Let $\mathcal{P}$ be a user of an SL-PoS ledger. $\mathcal{P}$ requires k confirmations, i.e., finalizes a transaction after it is "buried" under k blocks. Let τ be a transaction, published on slot r, with value v_τ. After l slots, τ is buried under b blocks, with $b = x \cdot l$ for some $x \in (0, 1)$. In case we have full participation in the protocol and the adversary is bounded by $\frac{1}{2}$, it holds $x > \frac{1}{2}$; in the rest of the section, we will focus on this setting. Observe that $(1 - x) \cdot 100\%$ of slots will be – seemingly – empty. $\mathcal{P}$ will (on expectation) confirm τ after $\frac{1}{x} \cdot k$ slots, i.e., when k blocks are produced; of these, $\frac{1-x}{x} \cdot k$ are empty.

Let $\mathcal{M}$ be a party that wants to double-spend τ. $\mathcal{M}$ should produce a private chain with at least k blocks. Of these, at most $\frac{1-x}{x} \cdot k$ correspond to the respective empty slots, while $k - \frac{1-x}{x} \cdot k = \frac{2 \cdot x-1}{x} \cdot k$ conflict with existing blocks, i.e., are evidence of infraction. Let d be a deposit amount, which corresponds to a single slot. Thus, for a period of t slots, the total deposited assets $D = t \cdot d$ are distributed evenly across all slots. $\mathcal{M}$ can be penalized only for infraction blocks, i.e., for slots which showcase conflicting blocks. In a range of $\frac{1}{x} \cdot k$ slots, infraction slots are $\frac{2 \cdot x-1}{x} \cdot k$. Therefore, $\mathcal{M}$ forfeits at most $\frac{2 \cdot x-1}{x} \cdot k \cdot d$ in deposit and $\frac{2 \cdot x-1}{x} \cdot k \cdot R$ in rewards that correspond to infraction blocks. Thus, if $v_\tau > \frac{2 \cdot x-1}{x} \cdot k \cdot (d + R)$, $\mathcal{M}$ can profitably double-spend τ. Consequently, depending on the amount d of deposit per slot, the block reward R, and the rate $(1-x)$ of empty slots, for a transaction τ with value v_τ, $\mathcal{P}$ should set the confirmation window's size to: $k_\tau > \frac{v_\tau}{\frac{2 \cdot x-1}{x} \cdot (d+R)}$.

Finally, the system should allow each participant to withdraw their deposit at some point. However, it should also enforce *some* limit, s.t. deposits can cover (possible) penalties. Intuitively, a party $\mathcal{P}$ should be able to withdraw a deposited amount that corresponds to a slot r, only if no transaction exists, s.t. r is part of the window of size k (computed as above). In other words, $\mathcal{P}$'s deposit should be able to cover all slots which $\mathcal{P}$ has led and which are in the confirmation window of at least one non-finalized transaction.

Acknowledgements. This work was supported by Input Output (iohk.io) through their funding of the Edinburgh Blockchain Technology Lab.

REFERENCES

[1] Nick Arnosti and S. Matthew Weinberg. 2019. Bitcoin: A Natural Oligopoly. In *ITCS 2019: 10th Innovations in Theoretical Computer Science Conference*, Avrim Blum (Ed.), Vol. 124. LIPIcs, San Diego, CA, USA, 5:1–5:1. https://doi.org/10.4230/LIPIcs.ITCS.2019.5

[2] Christian Badertscher, Juan A. Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. 2018. But Why Does It Work? A Rational Protocol Design Treatment of Bitcoin, See [54], 34–65. https://doi.org/10.1007/978-3-319-78375-8_2

[3] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. 2018. Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. In *ACM CCS 2018: 25th Conference on Computer and Communications Security*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, Toronto, ON, Canada, 913–930. https://doi.org/10.1145/3243734.3243848

[4] William J Baumol. 2004. Welfare Economics and the Theory of the State. In *The encyclopedia of public choice*. Springer, 937–940.

[5] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Berkeley, CA, USA, 459–474. https://doi.org/10.1109/SP.2014.36

[6] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S. Matthew Weinberg. 2019. Formal Barriers to Longest-Chain Proof-of-Stake Protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, Anna Karlin, Nicole Immorlica, and Ramesh Johari (Eds.). ACM, 459–473. https://doi.org/10.1145/3328526.3329567

[7] Vitalik Buterin. 2014. On stake. https://blog.ethereum.org/2014/07/05/stake/.

[8] Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437* (2017).

[9] Vitalik Buterin, Daniël Reijsbergen, Stefanos Leonardos, and Georgios Piliouras. 2019. Incentives in Ethereum's Hybrid Casper Protocol. *CoRR* abs/1903.04205. arXiv:1903.04205 http://arxiv.org/abs/1903.04205

[10] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. 2018. ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement. Cryptology ePrint Archive, Report 2018/377. https://eprint.iacr.org/2018/377.

[11] Xi Chen, Christos H. Papadimitriou, and Tim Roughgarden. 2019. An Axiomatic Approach to Block Rewards. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*. ACM, 124–131. https://doi.org/10.1145/3318041.3355470

[12] Steve Chien and Alistair Sinclair. 2011. Convergence to approximate Nash equilibria in congestion games. *Games Econ. Behav.* 71, 2 (2011), 315–327. https://doi.org/10.1016/j.geb.2009.05.004

[13] Bernardo David, Rafael Dowsley, and Mario Larangeira. 2019. ROYALE: A Framework for Universally Composable Card Games with Financial Rewards and Penalties Enforcement, See [31], 282–300. https://doi.org/10.1007/978-3-030-32101-7_18

[14] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain, See [54], 66–98. https://doi.org/10.1007/978-3-319-78375-8_3

[15] John R Douceur. 2002. The sybil attack. In *International workshop on peer-to-peer systems*. Springer, 251–260.

[16] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. 2015. Proofs of Space. In *Advances in Cryptology – CRYPTO 2015, Part II (Lecture Notes in Computer Science, Vol. 9216)*, Rosario Gennaro and Matthew J. B. Robshaw (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 585–605. https://doi.org/10.1007/978-3-662-48000-7_29

[17] Ethereum. 2018. Proof of Stake FAQs. https://eth.wiki/en/concepts/proof-of-stake-faqs.

[18] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-ng: A scalable blockchain protocol. In *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*. 45–59.

[19] Ittay Eyal and Emin Gün Sirer. 2014. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *FC 2014: 18th International Conference on Financial Cryptography and Data Security (Lecture Notes in Computer Science, Vol. 8437)*, Nicolas Christin and Reihaneh Safavi-Naini (Eds.). Springer, Heidelberg, Germany, Christ Church, Barbados, 436–454. https://doi.org/10.1007/978-3-662-45472-5_28

[20] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. 2004. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, László Babai (Ed.). ACM, 604–612. https://doi.org/10.1145/1007352.1007445

[21] Giulia C. Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, and Gerui Wang. 2019. Compounding of Wealth in Proof-of-Stake Cryptocurrencies, See [31], 42–61. https://doi.org/10.1007/978-3-030-32101-7_3

[22] Matheus V. X. Ferreira and S. Matthew Weinberg. 2021. Proof-of-Stake Mining Games with Perfect Randomness. In *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, Péter Biró, Shuchi Chawla, and Federico Echenique (Eds.). ACM, 433–453. https://doi.org/10.1145/3465456.3467636

[23] Amos Fiat, Anna Karlin, Elias Koutsoupias, and Christos H. Papadimitriou. 2019. Energy Equilibria in Proof-of-Work Mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, Anna Karlin, Nicole Immorlica, and Ramesh Johari (Eds.). ACM, 489–502. https://doi.org/10.1145/3328526.3329630

[24] Mehdi Fooladgar, Mohammad Hossein Manshaei, Murtuza Jadliwala, and Mohammad Ashiqur Rahman. 2020. On Incentive Compatible Role-Based Reward Distribution in Algorand. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020*. IEEE, 452–463. https://doi.org/10.1109/DSN48063.2020.00059

[25] Algorand Foundation. 2020. FAQs. https://algorand.foundation/faq

[26] Juan A. Garay and Aggelos Kiayias. 2020. SoK: A Consensus Taxonomy in the Blockchain Era. In *Topics in Cryptology – CT-RSA 2020 (Lecture Notes in Computer Science, Vol. 12006)*, Stanislaw Jarecki (Ed.). Springer, Heidelberg, Germany, San Francisco, CA, USA, 284–318. https://doi.org/10.1007/978-3-030-40186-3_13

[27] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *Advances in Cryptology – EUROCRYPT 2015, Part II (Lecture Notes in Computer Science, Vol. 9057)*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Heidelberg, Germany, Sofia, Bulgaria, 281–310. https://doi.org/10.1007/978-3-662-46803-6_10

[28] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the Security and Performance of Proof of Work Blockchains. In *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM Press, Vienna, Austria, 3–16. https://doi.org/10.1145/2976749.2978341

[29] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. Cryptology ePrint Archive, Report 2017/454. https://eprint.iacr.org/2017/454.

[30] M. Goemans, Vahab Mirrokni, and A. Vetta. 2005. Sink equilibria and convergence. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 142–151. https://doi.org/10.1109/SFCS.2005.68

[31] Ian Goldberg and Tyler Moore (Eds.). 2019. *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*. Lecture Notes in Computer Science, Vol. 11598. Springer, Heidelberg, Germany, Frigate Bay, St. Kitts and Nevis.

[32] Guy Goren and Alexander Spiegelman. 2019. Mind the Mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, Anna Karlin, Nicole Immorlica, and Ramesh Johari (Eds.). ACM, 475–487. https://doi.org/10.1145/3328526.3329566

[33] Cyril Grunspan and Ricardo Pérez-Marco. 2019. On profitability of selfish mining. arXiv:1805.08281 [cs.GT]

[34] Dimitris Karakostas, Aggelos Kiayias, and Mario Larangeira. 2020. Account Management in Proof of Stake Ledgers. In *SCN 20: 12th International Conference on Security in Communication Networks (Lecture Notes in Computer Science, Vol. 12238)*, Clemente Galdi and Vladimir Kolesnikov (Eds.). Springer, Heidelberg, Germany, Amalfi, Italy, 3–23. https://doi.org/10.1007/978-3-030-57990-6_1

[35] Dimitris Karakostas, Aggelos Kiayias, and Thomas Zacharias. 2022. Blockchain Nash Dynamics and the Pursuit of Compliance. arXiv:2201.00858 [cs.CR]

[36] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. 2016. Blockchain Mining Games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, Vincent Conitzer, Dirk Bergemann, and Yiling Chen (Eds.). ACM, 365–382. https://doi.org/10.1145/2940716.2940773

[37] Aggelos Kiayias and Giorgos Panagiotakos. 2017. On Trees, Chains and Fast Transactions in the Blockchain. In *Progress in Cryptology - LATINCRYPT 2017: 5th International Conference on Cryptology and Information Security in Latin America (Lecture Notes in Computer Science, Vol. 11368)*, Tanja Lange and Orr Dunkelman (Eds.). Springer, Heidelberg, Germany, Havana, Cuba, 327–351. https://doi.org/10.1007/978-3-030-25283-0_18

[38] Aggelos Kiayias and Alexander Russell. 2018. Ouroboros-BFT: A Simple Byzantine Fault Tolerant Consensus Protocol. Cryptology ePrint Archive, Report 2018/1049. https://eprint.iacr.org/2018/1049.

[39] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Advances in Cryptology – CRYPTO 2017, Part I (Lecture Notes in Computer Science, Vol. 10401)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 357–388. https://doi.org/10.1007/978-3-319-63688-7_12

[40] Aggelos Kiayias and Aikaterini-Panagiota Stouka. 2019. Coalition-Safe Equilibria with Virtual Payoffs. arXiv:2001.00047 [cs.GT]

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

[42] Elias Koutsoupias, Philip Lazos, Foluso Ogunlana, and Paolo Serafino. 2019. Blockchain Mining Games with Pay Forward. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 917–927. https://doi.org/10.1145/3308558.3313740

[43] Joshua A. Kroll, Ian C. Davey, and Edward W. Felten. 2013. The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries. In *The Twelfth Workshop on the Economics of Information Security (WEIS 2013)*.

[44] Ranjit Kumaresan, Tal Moran, and Iddo Bentov. 2015. How to Use Bitcoin to Play Decentralized Poker. In *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM Press, Denver, CO, USA, 195–206. https://doi.org/10.1145/2810103.2813712

[45] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. 2015. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*. Citeseer, 919–927.

[46] Wenting Li, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. 2017. Securing proof-of-stake blockchain protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 297–315.

[47] William Forster Lloyd. 1833. *Two lectures on the checks to population*. JH Parker.

[48] Katie Martin and Billy Nauman. 2021. Bitcoin's growing energy problem: 'It's a dirty currency'. https://www.ft.com/content/1aecb2db-8f61-427c-a413-3b929291c8ac.

[49] Julian Martinez. 2018. Understanding Proof of Stake: The Nothing at Stake Theory. https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027.

[50] Vahab S. Mirrokni and Alexander Skopalik. 2009. On the complexity of nash dynamics and sink equilibria. In *Proceedings 10th ACM Conference on Electronic Commerce (EC-2009), Stanford, California, USA, July 6–10, 2009*, John Chuang, Lance Fortnow, and Pearl Pu (Eds.). ACM, 1–10. https://doi.org/10.1145/1566374.1566376

[51] Tal Moran and Ilan Orlov. 2019. Simple Proofs of Space-Time and Rational Proofs of Storage. In *Advances in Cryptology – CRYPTO 2019, Part I (Lecture Notes in Computer Science, Vol. 11692)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 381–409. https://doi.org/10.1007/978-3-030-26948-7_14

[52] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

[53] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2015. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. Cryptology ePrint Archive, Report 2015/796. https://eprint.iacr.org/2015/796.

[54] Jesper Buus Nielsen and Vincent Rijmen (Eds.). 2018. *Advances in Cryptology – EUROCRYPT 2018, Part II*. Lecture Notes in Computer Science, Vol. 10821. Springer, Heidelberg, Germany, Tel Aviv, Israel.

[55] Rafael Pass and Elaine Shi. 2017. FruitChains: A Fair Blockchain. In *36th ACM Symposium Annual on Principles of Distributed Computing*, Elad Michael Schiller and Alexander A. Schwarzmann (Eds.). Association for Computing Machinery, Washington, DC, USA, 315–324. https://doi.org/10.1145/3087801.3087809

[56] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (1980), 228–234. https://doi.org/10.1145/322186.322188

[57] R. W. Rosenthal. 1973. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory* 2 (1973), 65–67.

[58] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal Selfish Mining Strategies in Bitcoin. In *FC 2016: 20th International Conference on Financial Cryptography and Data Security (Lecture Notes in Computer Science, Vol. 9603)*, Jens Grossklags and Bart Preneel (Eds.). Springer, Heidelberg, Germany, Christ Church, Barbados, 515–532.

[59] Tezos. 2020. Proof-of-stake in Tezos. https://tezos.gitlab.io/whitedoc/proof_of_stake.html

[60] Gavin Wood. 2014. Ethereum yellow paper.