A Provably Secure Proof-of-Stake Blockchain Protocol

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

Ioannis Konstantinou†
University of Athens
i.konstantinoy@di.uoa.gr

Alexander Russell
University of Connecticut
acr@cse.uconn.edu

Bernardo David
Aarhus University
IOHK
bernardo.david@iohk.io

Roman Oliynykov
IOHK
roman.oliynykov@iohk.io

September 12, 2016

Abstract

We present a provably-secure blockchain protocol based on "proof of stake." As far as we are aware, this is the first proof of stake blockchain protocol which provides rigorous security guarantees. The security properties of the system are analyzed in the model of [9] and are comparable to the ones possessed by the bitcoin blockchain protocol which utilizes proof of work. Furthermore, an incentive mechanism for the protocol is also proposed.

1 Introduction

A primary consideration regarding the operation of blockchain protocols that are based on proof of work (PoW)—such as bitcoin [14]—is the energy that is required for executing the protocol. At the time of this writing, generating a single block in the bitcoin blockchain requires a number of hashing operations exceeding 2^{60}, which means that significant energy needs to be expended in order for the protocol to run. Early calculations placed the energy requirements of the protocol in the order of magnitude of a country, see e.g., [15].

This state of affairs has motivated the investigation of alternative blockchain protocols that would obviate the need for proof of work by substituting it with another, more energy efficient, mechanism that can provide similar guarantees. It is important to point out that the proof of work mechanism of bitcoin facilitates a type of randomized "leader election" process that elects one of the miners to issue the next block. Furthermore, this selection is performed in a randomized fashion proportionally to the computational power of each miner, provided that all miners follow the protocol (we note that deviations from the protocol may hurt this proportionality as exemplified by "selfish mining" strategies [7, 18]).

*Work partly supported by H2020 Project #653497 PANORAMIX.
†Work partly supported by ERC project #259152 CODAMODA.

A natural alternative mechanism is based on "proof of stake" (PoS). Rather than miners investing computational resources in order to participate in the leader election process, they instead run a process that randomly selects one of them proportionally to the stake that each possesses.

In effect, this generates a self-referential blockchain discipline: maintaining the blockchain relies on the stakeholders themselves and assigns work to them (as well as rewards) based on the amount of stake that is possessed by each one of them as reported in the ledger (without incurring any additional expenditures). In some sense, this sounds ideal; however, realizing such a proof-of-stake protocol appears to involve a number of definitional, technical, and analytic challenges.

Previous work. The concept of PoS has been discussed extensively in the bitcoin forum.[1] Proof-of-stake based blockchain design has been more formally studied by Bentov et al., both in cojunction with PoW [4] as well as the sole mechanism for a blockchain protocol [3]. Although Bentov et al. showed that their protocols are secure against some classes of attacks, they do not provide a formal model for analysing PoS based protocols or any security definitions/proofs. Heuristic proof-of-stake based blockchain protocols have been proposed (and implemented) in a number of cryptocurrencies.[2] Being based on heuristic security arguments, these cryptocurrencies have been frequently found to be deficient from a point of view of security. See [3] for a discussion of various attacks.

It is also interesting to contrast a PoS-based blockchain protocol with a more classical consensus blockchain that relies on a *fixed* set of authorities (such protocols have been proposed, see, e.g., [5]). What distinguishes PoS-based blockchains compared to such protocols is the fact that stake changes over time and hence the trust assumption evolves with the system.

Another alternative to PoW and PoS is the concept of *proof of space* [2, 6], which has been specifically investigated in the context of blockchain protocols [16]. In a proof of space setting, a "prover" wishes to demonstrate the utilization of space (storage / memory); as in the case of a PoW, this utilizes a physical resource but can be less energy demanding over time. A related concept is *proof of space-time* (PoST) [13]. In all these cases,s, a physical resource (either storage or computational power) is necessary.

The PoS Design challenge. A fundamental problem for PoS-based blockchain protocols is to simulate the leader election process. In order to achieve a randomized election among stakeholders, entropy has to be introduced in the system, and mechanisms to introduce entropy may be manipulated by the adversary. For instance, independently of the solution, an adversary controlling a set of stakeholders may choose to simulate the protocol execution trying different sequences of stakeholder participants so that it finds a favorable chain continuation that biases the leader election. To prevent this manipulation, honest stakeholders need to be able to add sufficient entropy and counter any lookahead performed by the adversary.

Our Results. To the best of our knowledge, we present the first PoS-based blockchain protocol that has a rigorous security analysis. In more detail, our results are as follows.

First, we provide a model and formalize the problem of realizing PoS-based blockchain protocols. The model we introduce is in the spirit of [9], focusing on *persistence* and *liveness*, two formal properties of a

[1] See "Proof of stake instead of proof of work." Bitcoin forum thread. Posts by user "QuantumMechanic" and others. (`https://bitcointalk.org/index.php?topic=27787.0`.).

[2] A non-exhaustive list includes NXT, Neucoin, Blackcoin, Tendermint, Bitshares.

robust transaction ledger. Persistence states that once a node of the system proclaims a certain transaction as "stable," the remaining nodes, if queried and respond honestly, will also report it as stable. Here, stability is to be understood as a predicate that will be parameterized by some security parameter k that will affect the certainty with which property holds. E.g., "more than k blocks deep," etc. Second, liveness ensures that once an honestly generated transaction has been made available for a sufficient amount of time to the network nodes, say u time steps, it will become stable. The conjunction of liveness and persistence provides a robust transaction ledger in the sense that honestly generated transactions are adopted and become immutable. Our model is suitably amended to facilitate PoS-based solutions.

Second, we describe a novel protocol for a blockchain based on PoS. Our blockchain protocol assumes that parties can freely create accounts and receive and make payments, and that stake shifts over time. We utilize a secure multiparty implementation of a coin flipping protocol to produce the randomness for the leader election process. This distinguishes the approach from other previous solutions that either defined such values deterministically based on the current state of the blockchain or used collective coin flipping as a way to introduce entropy [3]. Also, unique to our design approach is the fact that the system ignores round-to-round stake modifications. Instead, the set of stakeholders is taken in a snapshots in regular intervals called epochs, and in each such interval a secure multiparty computation takes place utilizing the blockchain itself as the broadcast channel. In each epoch a set of randomly selected stakeholders are responsible for executing the coin flipping protocol. The outcome of the protocol determines the set of next stakeholders to execute the protocol in the next epoch as well as the outcomes of all leader elections for the epoch.

Third, we provide a set of formal arguments establishing that no adversary can break persistence and liveness. Our protocol is secure under a number of plausible assumptions: (1) the network is highly synchronous, (2) the majority of the selected stakeholders is available as needed to participate in each epoch, (3) the astakeholders do not remain offline for long periods of time. At the core of our security arguments is a combinatorial probabilistic argument, regarding "forkable strings" which we formulate, prove and also verify experimentally.

Fourth, given our model we explore the various attacks and how they can be addressed within our model. Specifically, we discuss double spending attacks, transaction denial attacks, 51% attacks, desynchronization attacks and others.

Finally, we comment on the incentive structure of the protocol and discuss how participation on ledger maintainance can be incentivized by fees.

2 Model

Synchrony. We consider a setting where time is divided in discrete units called slots. Each slot can be associated with a single block. Players are aware of the current slot for which a block is to be determined. In general, each slot sl_r is indexed by an integer $r \in \{1, 2, \ldots\}$. We assume that the real time window that corresponds to each slot has the following properties.

- The current slot is determined by a publicly-known and monotonically increasing function of current time.
- Each player has access to the current time. Any discrepancies between parties' local time are insignificant in comparison with the length of time represented by a slot.

- The length of the time window that corresponds to a slot is sufficient to guarantee that any message transmitted by an honest party at the beginning of the time window will be received by any other honest party by the end of that time window (even accounting for small inconsistencies in parties' local clocks). In particular, while network delays may occur, they never exceed the slot time window.

Transaction Ledger Properties. A protocol Π implements a robust transaction ledger provided that the ledger that Π maintains is divided in "blocks" that determine the order with which transactions are incorporated in the ledger. It should also satisfy the following two properties.

- **Persistence.** Once a node of the system proclaims a certain transaction as *stable*, the remaining nodes, in case they are queried and they respond honestly, will also report it as stable. Here the notion of stability is a predicate that is parameterized by a security parameter k; specifically, a transaction is declared *stable* if and only if it is in a block that is more than k blocks deep in the ledger.
- **Liveness.** If all honest nodes in the system attempt to include a certain transaction then, after the passing of time corresponding to u slots (called the transaction confirmation time), there is a node which, if queried and responding honestly, will report the transaction as stable.

In [12] it was shown that persistence and liveness can be derived from the following three elementary properties provided that protocol Π derives the ledger from a data structure in the form of a blockchain.

- **Common Prefix (CP); with parameters** $k, l \in \mathbb{N}$**.** The chains C_1, C_2 possessed by two external observers at the onset of the slots sl_1, sl_2 with sl_2 at most l slots ahead of sl_1, are such that $C_1^{\lceil k} \preceq C_2$.
- **Chain Quality (CQ); with parameters** $\mu \in (0,1] \to \mathbb{R}$ **and** $k \in \mathbb{N}$**.** For any subset S of (possibly malicious) stakeholders with relative stake α and any portion of length k in a chain possessed by an honest party at the onset of a certain slot, the ratio of blocks originating from members of S can be at most $\mu(\alpha)$.
- **Chain Growth (CG); with parameters** $\tau \in (0,1], s \in \mathbb{N}$**.** Consider the chains C_1, C_2 posssessed by two honest parties at the onset of two slots sl_1, sl_2 with sl_2 at most s slots ahead of sl_1. Then it holds that $\text{len}(C_2) - \text{len}(C_1) \geq \tau \cdot s$. We call τ the speed coefficient.

Some remarks are in place. Regarding common prefix observe the importance of parameter l. In case $l = 0$, it coincides with the common prefix as originally formulated in [9]. A stronger formulation of common prefix would set l to be the lifetime of the system itself, see [12]. Restricting on a bound l smaller than the system's lifetime suggests that forks deeper than k blocks might be feasible in the chains of honest parties (or even of the same party) if the parties are observed between two rounds that are more than l slots away. This relaxation is necessary to be able to prove the common prefix property in the PoS setting. With foresight, maintaining the implication from common prefix to persistence, we will need the additional assumption that no honest stakeholder gets offline for more than k rounds, where k is a parameter.

Regarding chain quality, it will hold that the function μ that depends on α satisfies $\mu(\alpha) \geq \alpha$ for protocols of interest. An ideal setting of μ would be in fact the identity function and in this case, this would suggest that the percentage of blocks in any chain segment is proportional to the cumulative stake of a set of stakeholders (who potentially act maliciously).

It is worth noting that for bitcoin it holds that $\mu(\alpha) = \frac{\alpha}{1-\alpha}$, and this bound is in fact tight, (see [9] who argue this statement for chain quality using a variant of selfish mining [7]). The same will hold true for our protocol construction.

Finally chain growth deals with how fast the chain of honest parties. In the case of bitcoin, but also in our protocol, the power of the rule *longest* (*hardest* in the case of bitcoin) chain wins in each round provides for an easy proof for chain growth.

Security Model. We adopt the model introduced by [9] for analysing security of blockchain protocols. We denote by $\mathsf{VIEW}^{P}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ the view of party P after the execution of protocol Π with adversary $\mathcal{A}$, environment $\mathcal{Z}$, security parameter κ and auxiliary information z. We will only consider executions without auxiliary input $z = \epsilon$.

Contrary to [9], our analysis is in the standard model, assuming a trusted setup without a random oracle. The execution of the protocol is with respect to an adversary that corrupts a subset of the initial stakeholders. Beyond the initial stakeholders, the environment is allowed to introduce additional parties and generate transactions that are given as input to existing stakeholders. Each party has a private state that includes a public/secret-key pair $(\mathsf{pk}, \mathsf{sk})$.

The adversary $\mathcal{A}$ may, at any point of the execution, issue a special command $(\mathsf{Corrupt}, U)$ which will result in the the stakeholder U relinquishing its entire state to $\mathcal{A}$; from this point on, the adversary will be activated in place of the stakeholder U. The adversary can only corrupt a stakeholder if it is given permission by the environment $\mathcal{Z}$ running the protocol execution. The permission is in the form of a message $(\mathsf{Corrupt}, U)$ which is provided to the adversary by the environment.

- At each slot sl_j, the environment $\mathcal{Z}$ is allowed to activate any subset of stakeholders it wishes. Each one of them will possibly produce messages that are to be transmitted to other stakeholders.
- The adversary is activated last in each sl_j, is allowed to read all messages sent by honest parties and may deliver them in the next slot to each stakeholder in any order it wishes, potentially including messages of its own. Adversarial messages may be delivered only to a selected set of honest stakeholders.
- If a stakeholder is not activated in a certain slot then all the messages written to its communication tape are lost.

It is easy to see that the model above confers such sweeping power on the adversary that one cannot establish any significant guarantees on protocols of interest. It is thus important to restrict the environment suitably (taking into account the details of the protocol) so that we may be able to argue security. With foresight, the restrictions we will impose to the environment are as follows.

Restrictions imposed to the environment. The environment, which is responsible for activating the honest parties in each round, will be subject to the following constraints.

- At each slot there will be an identified set of elected shareholders, and the adversary will be permitted to corrupt only a minority of those.

- At each slot there will be a uniquely identified party that will be called the slot leader. If a stakeholder is honest and is the slot leader at a certain slot, the environment will activate it in the slot before and in the slot that it is the slot leader.
- In each round there will be at least one honest stakeholder that is activated (independently of whether it is a slot leader or not).
- There will be a parameter $k \in \mathbb{Z}$ that will signify the maximum number slots that an honest shareholder can be offline.

3 Our Protocol: Static State

3.1 Basic Concepts and Protocol Description

In the static stake case, we assume that a fixed collection of n stakeholders $U_1, \ldots, U_n$ interact throughout the protocol. Stakeholder U_i possesses s_i stake before the protocol starts. For each stakeholder U_i a verification and signing key pair $(\mathsf{vk}_i, \mathsf{sk}_i)$ for a signature scheme is generated and we assume without loss of generality that each verification key vk_i is known by all stakeholders. Before describing the protocol, we establish basic definitions following the notation of [9].

Definition 3.1 (Genesis Block) *The genesis block B_0 contains the list of stakeholders identified by their public-keys and their respective stakes $\{(\mathsf{vk}_1, s_1), \ldots, (\mathsf{vk}_n, s_n)\}$ and auxiliary information ρ.*

Definition 3.2 (Block) *A block B_i generated at a slot $sl \in \{sl_1, \ldots, sl_R\}$ contains the current state $st \in \{0,1\}^\lambda$, data $d \in \{0,1\}^*$, the slot number sl and a signature $\sigma = \mathsf{Sign}_{\mathsf{sk}_i}(st, d, sl)$ computed under sk_i corresponding to the stakeholder U_i generating the block. If no block is generated at slot i then $B_i = \emptyset$.*

Definition 3.3 (State) *A state is a string $st \in \{0,1\}^\lambda$.*

Definition 3.4 (Blockchain) *A blockchain (or chain) is a sequence of blocks $B_1, \ldots, B_n$ for which it holds that for each block B_i the state st_i is equal to $H(B_{i-1})$, where H is a prescribed collision resistant hash funcion.The* length *of a chain* $\mathrm{len}(C)$ *is its number of blocks. The rightmost block is the* head *of the chain, denoted* $\mathrm{head}(C)$. *Note that the empty string ε is also a legal chain; by convention we set* $\mathrm{head}(\varepsilon) = \varepsilon$.

Let C be a chain of length n and k be any non-negative integer. We denote by $C^{\lceil k}$ the chain resulting from removal of the k rightmost blocks of C. Note that if $k \geq \mathrm{len}(C)$ then $C^{\lceil k} = \varepsilon$. We let $C_1 \preceq C_2$ indicate that the chain C_1 is a prefix of the chain C_2.

Definition 3.5 (Epoch) *An epoch is a set of R adjacent slots $S = \{sl_1, \ldots, sl_R\}$. (The value R is a parameter of the protocol we analyze in this section.)*

Definition 3.6 (Adversarial Stake Ratio) *Let U_A be the set of stakeholders controlled by the adversary, the adversarial stake ratio is defined as*

$$\alpha = \frac{\sum_{j \in U_A} s_j}{\sum_{i=1}^{n} s_i}$$

where n is the total number of stakeholders and s_i is stakeholder U_i's stake.

Slot Leader Selection In the protocol described in this section, for $0 < j \leq R$ and slot sl_j, the *slot leader* E_j has the right to generate a block. For $1 \leq i \leq n$, a stakeholder U_i is selected as the slot leader with probability p_i proportional to its stake registered in the genesis block B_0. In the static stake case, the genesis block as well as the procedure for selecting slot leaders are determined by the ideal functionality $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$, defined in Figure 1. This functionality is parameterized by the list $\{(\mathsf{vk}_1, s_1), \ldots, (\mathsf{vk}_n, s_n)\}$ assigning to each stakeholder its respective stake, a distribution D that provides auxiliary information ρ and a *leader selection function* F defined below.

Definition 3.7 (Leader Selection Process) *A leader selection process* (D, F) *is a pair consisting of a distribution and a deterministic function such that, when* $\rho \leftarrow D$ *it holds that for all* $sl_j \in \{sl_1, \ldots, sl_R\}$, $\mathsf{F}(\rho, sl_j)$ *outputs* $U_i \in \{U_1, \ldots, U_n\}$ *with probability*

$$p_i = \frac{s_i}{\sum_{k=1}^{n} s_k}$$

where s_i *is the stake held by stakeholder* U_i.

Functionality $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$

$\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ is parameterized by the public keys and respective stakes of the stakeholders $\{(\mathsf{vk}_1, s_1), \ldots, (\mathsf{vk}_n, s_n)\}$, a distribution D and a function F so that (D, F) is a leader selection process. $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ interacts with stakeholders $U_1, \ldots, U_n$ as follows:

- Upon receiving $(\mathsf{genblock\_req}, U_i)$ from stakeholder U_i, $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ proceeds as follows. If $B_0 = \emptyset$, $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ samples $\rho \leftarrow D$ and sets $B_0 = \{(\mathsf{vk}_1, s_1), \ldots, (\mathsf{vk}_n, s_n), \rho\}$. Finally, $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ sends $(\mathsf{genblock}, B_0, \mathsf{F})$ to U_i.

Figure 1: Functionality $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$.

A Protocol in the $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$-hybrid model. We start by describing a simple PoS based blockchain protocol considering static stake in the $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$-hybrid model, i.e., where the genesis block B_0 (and consequently the slot leaders) are determined by the ideal functionality $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$. The stakeholders $U_1, \ldots, U_n$ interact among themselves and with $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ through Protocol π_{SPOS} described in Figure 2.

The protocol relies on a $\mathsf{maxvalid}(C, \mathbb{C})$ function that chooses a chain given the current chain C and a set of valid $\mathbb{C}$ that are available in the network. This function is parameterized by $k \in \mathbb{N}$, (a security parameter), and is defined as follows.

> Function $\mathsf{maxvalid}(C, \mathbb{C})$. Returns the longest chain from $\mathbb{C} \cup \{C\}$ that does not fork from C more than k blocks. If multiple exist it returns C, if this is one of them, or it returns the one that is listed first in $\mathbb{C}$.

Protocol π_{SPOS}

π_{SPOS} is a protocol run by stakeholders $U_1, \ldots, U_n$ interacting among themselves and with $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ over a sequence of slots $S = \{sl_1, \ldots, sl_R\}$. π_{SPOS} proceeds as follows:

1. **Initialization** When π_{SPOS} starts, each stakeholder $U_i \in \{U_1, \ldots, U_n\}$ sends $(\mathsf{genblock\_req}, U_i)$ to $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$, receiving $(\mathsf{genblock}, B_0, \mathsf{F})$ as answer. U_i sets an internal blockchain $C = B_0$ and a initial internal state $st = H(B_0)$.
2. **Chain Extension** For every slot $sl_j \in S$, every online stakeholder U_i performs the following steps:
   (a) Collect all valid chains received via broadcast into a set $\mathbb{C}$, verifying that for every chain $C' \in \mathbb{C}$ and every block $B' = (st', d', sl', \sigma') \in C'$ it holds that $\mathsf{Vrf}_{\mathsf{vk}'}(\sigma', (st', d', sl')) = 1$, where vk' is the verification key of the stakeholder $U' \leftarrow \mathsf{F}(r, sl')$. U_i calls the function $\mathsf{maxvalid}(C, \mathbb{C})$ to select a new internal chain $C \in \mathbb{C}$ and sets state $st = H(\text{head}(C))$.
   (b) If U_i is the slot leader determined by $\mathsf{F}(r, sl_j)$, it generates a new block $B = (st, d, sl_j, \sigma)$ where st is its current state, $d \in \{0,1\}^*$ is the transaction data and $\sigma = \mathsf{Sign}_{\mathsf{sk}_i}(st, d, sl_j)$ is a signature on (st, d, sl_j). U_i extends C by appending B, obtains $C = C|B$ and broadcasts the new C.

Figure 2: Protocol π_{SPOS}.

3.2 Forkable Strings

In our security arguments, we will treat strings over $\{0,1\}^*$ as an abstraction for (sub-)sequences of slots. If $w \in \{0,1\}^*$, we say that the the slot i is an adversarial slot if and only if $w_i = 1$. In this case, the string w will be the characteristic string of the sequence of adversarial slots. We start with some intuition on our approach to analyze the protocol.

Let $w \in \{0,1\}^n$ be a characteristic string of some sequence of slots S with $|S| = n$. Consider two observers that go offline immediately prior to the commencement of S. The two observers have the same view of the current chain which they believe it as correct; we denote it by C_0. The two observers come back online at the last slot of S and request an update of their chain. These two observers will have a diverging view over S if it is possible for the adversary to force the two observers to adopt two different chains C_1, C_2 whose common prefix is C_0.

We observe that not all characteristic strings permit this. For instance the string 0^n ensures that the two observers will adopt the same chain C which will have n new blocks on top of the joint state of the two observers C_0 prior to the commencement of the rounds in S. On the other hand, other strings do not guarantee this; in the case of 1^n, it is possible for the adversary to produce two completely different histories during the sequence of slots S and thus furnish to the two observers two distinct chains C_1, C_2 that only share the common prefix C_0.

Definition 3.8 (Characteristic String) *Fix an execution with genesis block B_0, adversary $\mathcal{A}$, environment $\mathcal{Z}$. Let $S = \{sl_i, \ldots, sl_j\}$ where $i < j$ is a sequence of slots of length $|S| = \ell$. The characteristic string $w \in \{0,1\}^\ell$ of S is such that $w_k = 1$ if and only if the adversary controls the slot leader of slot sl_k (i.e., upon receiving* $(\mathsf{leader\_req}, sl_k)$*, $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ answers with* $(\mathsf{leader\_ans}, sl_k, E_k)$ *such that E_k identifies a stakeholder controlled by the adversary).*

Forkable strings. Let $w = w_1, \ldots, w_n \in \{0,1\}^n$ be the characteristic string for a sequence of slots. A *fork* for the sequence w is a pair of increasing sequences,

$$f = (\alpha; \beta) = (\alpha_1, \ldots, \alpha_k; \beta_1, \ldots, \beta_\ell)$$

so that each α_i and β_i is an index of w (that is, an element of $\{1, \ldots, n\}$),

- $k \geq \ell$, and
- each honest index (for which $w_i = 0$) appears in *exactly one* of the two sequences.

We call the two sequences *tines*. Note that malicious indices may appear in neither, one, or both of the tines. We introduce some terminology for forks that captures the fact that honest indices "may only be added to the longer tine of a fork":

- We say that a fork for w is *incrementally legal* if either $w_n = 1$ or $w_n = 0$ and $\alpha_k = n$ (so that the longer tine of the fork contains this final honest index). That is, $w_n = 0 \Rightarrow \alpha_k = n$.
- If w is a prefix of the string $w' \in \{0,1\}^*$, $f = (\alpha, \beta)$ is a fork of w, and $f' = (\alpha', \beta')$ is a fork of w', we say that f is a prefix of f', written $f \sqsubseteq f'$, if each tine of f is a prefix of one of the tines of f'.
- In many cases, it is convenient to work with tines that do not "commit" anything beyond the last honest index. Specifically, we say that a fork is *closed* if $w_{\alpha_k} = w_{\beta_\ell} = 0$, so both tines end with honest indices.
- We say that a fork $f = (\alpha, \beta)$ for w is *legal* if there is a sequence

  $$f_0 = (\epsilon, \epsilon) \sqsubseteq f_1 \sqsubseteq \cdots \sqsubseteq f_n \sqsubseteq f$$

  of forks so that for each $i \in \{1, \ldots, n\}$ the fork f_i corresponds to the string $w_1 \ldots w_i$ and is incrementally legal and closed. (Note that both f_n and f are forks for the string w; we permit the possibility that they are actually different forks because we insist that f_n be closed—thus, in general f is obtained by adding further malicious indices to the end of the tines of f.) We call such a sequence a *transcript* (or a *transcript for* f, if we wish to emphasize the target fork in the sequence).

  (Note that the set of forks defined by these requirements would be unchanged if we removed the demand that the f_i are closed, but it is convenient for our purposes to adopt this extra assumption.)
- Finally, for a fork f we define the *gap* of f, denoted $\mathrm{gap}(f)$, to be $k - \ell$. We say that a fork is *flat* if $\mathrm{gap}(f) = 0$.

Definition 3.9 *Let $w \in \{0,1\}^n$. The string w is called* forkable *if there is a flat legal fork f for w.*

We define the *reserve* of a closed fork, denoted $\mathrm{reserve}(f)$, to be the number of malicious indices larger than β_ℓ, so

$$\mathrm{reserve}(f) = |\{i \mid w_i = 1 \text{ and } i \geq \beta_\ell\}|\,.$$

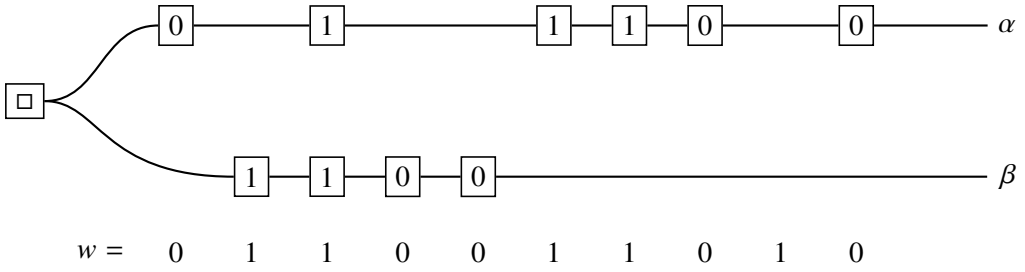


Figure 3: A fork for the string w. $\operatorname{gap}(f)$ indicates the difference in length between the two tines; in this case $\operatorname{gap}(f) = 2$. $\operatorname{reserve}(f) = |\{i \mid \beta_\ell < i \leq n \text{ and } w_i = M\}|$ indicates the number of "uncommitted" malicious indices, i.e., those appearing after β_ℓ; in this case $\operatorname{reserve}(f) = 3$.

We remark that this quantity, as with many of the other structural features discussed above, depends both on f and the specific string w associated with f.

We can treat the addition of a new symbol to the end of a string w as effecting a (non-deterministic) transition on the state space of all closed and incrementally legal forks. To approach this idea, we initially focus on the set of "0-terminating" strings (those which end with the symbol 0). Specifically, consider a closed fork f for a string w ending with a 0; extending w by the string 1^s0 results in another 0-terminalting string. We explore what possible (closed, incrementally legal) forks f' exist for $w' = w1^s0$ for which $f \sqsubseteq f'$. Indeed, there are two natural families of forks for w', depending on which tine is augmented with the new honest index:

Extend. It is always possible to add the new honest index to the longer tine (along with some of the s malicious indices of 1^s0); thus the longer tine is transformed from γ to $\gamma 1^t 0$ for some $t \leq s$. Note in this case that $t + 1$ is added to the gap and s is added to the reserve. (We remark that since f is closed, the last index corresponding to 0 appearing on the longer tine is indeed the last symbol of w.)

Crossover. If $s + \operatorname{reserve}(f) \geq \operatorname{gap}(f)$ (or, equivalently, $s + (\operatorname{reserve}(f) - \operatorname{gap}(f)) \geq 0$), the new honest index may be added to the shorter tine, creating a "crossover." In this case, the new gap is $1 + t$ for some $0 \leq t \leq s + \operatorname{reserve}(f) - \operatorname{gap}(f)$ and the reserve is s.

In general, a forkable string w may have many different (legal) forks. We can significantly simplify our reasoning about these dynamics by observing that for any forkable sequence w, there is an (essentially) unique canonical forking, which is produced by maximizing $\operatorname{margin}(f) = \operatorname{reserve}(f) - \operatorname{gap}(f)$ at each step.

Lemma 3.10 *Let $w = xy$ for two strings x and y, where x is 0-terminating, and consider a legal fork h for w. Let f be the fork associated with the prefix x by the transcript for h and let $\hat{f}$ be a closed, legal fork for x for which $\operatorname{margin}(\hat{f}) \geq \operatorname{margin}(f)$. Then there is a legal fork $\hat{h}$ for w with a transcript that associates $\hat{f}$ with x.*

Proof. The proof proceeds by induction on the number of 0's in the string y. Note that if y has no honest indices, then any closed fork f for x is also a closed fork for w and can be completed to a flat fork h (for w) if and only if the margin is non-negative. As $\operatorname{margin}(\hat{f}) \geq \operatorname{margin}(f)$, it follows that $\hat{f}$ can be completed to a flat fork as well.

Otherwise, we write $y = 1^s 0 y'$ and consider the closed fork g assigned to $x' = x1^s0$ by the transcript for h. If g is obtained from f by extending, it holds that $\operatorname{margin}(g) \leq \operatorname{margin}(f) + s - 1$ by the discussion above. Consider the fork $\hat{g}$ (for x') obtained by extending $\hat{f}$ and setting $t = 0$ in the extension process. Then, $\operatorname{margin}(\hat{g}) = \operatorname{margin}(\hat{f}) + s - 1 \geq \operatorname{margin}(g)$, and the lemma follows by induction (as y' has one less 0). Otherwise, g is obtained from f by a crossover, in which case $\operatorname{margin}(g) \leq s - 1$ by the discussion above. Note that a crossover can only occur if $\operatorname{margin}(f) + s \geq 0$ and, as $\operatorname{margin}(\hat{f}) \geq \operatorname{margin}(f)$, we have $\operatorname{margin}(\hat{f}) + s \geq \operatorname{margin}(f) + s \geq 0$ and a crossover is also possible from the fork $\hat{f}$. Let $\hat{g}$ be the fork for x' obtained (from $\hat{f}$) by the crossover rule with $t = 0$. Then $\operatorname{margin}(\hat{g}) = s - 1 \geq \operatorname{margin} g$, and the lemma follows by induction. ∎

Given the above observe that if a string w is forkable, there is an (essentially) canonical strategy for the adversary: maximize margin at each step. (Where the notion of "step" here really refers to the addition of a suffix of the form 1^s0 to the end of the current string.) Specifically, the transformation rules above can be simplified so that they are deterministic. As above, consider a closed fork f for the 0-terminating string w and an extension of w by the string 1^s0. We explore the margin-maximizing (closed, incrementally legal) forks f' that exist for $w' = w1^s0$ for which $f \sqsubseteq f'$:

- If $\operatorname{margin}(f) + s < 0$, the only option is extension and margin is maximized by choosing $t = 0$. The resulting margin is $\operatorname{margin}(f) + s - 1$.
- If $\operatorname{margin}(f) > 0$, either of the transformations above are possible but margin is maximized by extending with $t = 0$. The resulting margin is $\operatorname{margin}(f) + s - 1$.
- If $-s \leq \operatorname{margin}(f) \leq 0$, either of the transformations above are possible, but margin is maximized by crossing over with $t = 0$. The resulting margin is exactly $s - 1$ (regardless of previous margin).

(When $\operatorname{margin}(f) = 0$, margin can be maximized in multiple ways, and we assume a crossover for concreteness.)

These deterministic transformation rules determine maximum margin as a function of the 0-terminating string w. Specifically, write $w = 1^{s_1}01^{s_2}0 \cdots 1^{s_k}0$ for some $k, s_1, \ldots, s_k \in \mathbb{N}$. We define $\operatorname{margin}(w)$ by the recursive rule

$$\begin{aligned} \operatorname{margin}(\epsilon) &= 0, \\ \operatorname{margin}(w'1^s0) &= \begin{cases} s - 1 & \text{if } -s \leq \operatorname{margin}(w') \leq 0, \\ \operatorname{margin}(w') + s - 1 & \text{otherwise.} \end{cases} \end{aligned}$$

Note, additionally, that any 0-terminating string w is forkable if and only if $\operatorname{margin}(w) \geq 0$.

For convenience, we extend this notion of margin to the set of all strings $\{0,1\}^*$. For a string $w \in \{0,1\}^*$, let $\operatorname{tail}(w)$ denote the number of trailing 1's in the string (that is, $\max\{k \mid w = w'1^k \text{ for some } w'\}$); then define $\operatorname{margin}(w)$ by the rule

$$\begin{aligned} \operatorname{margin}(\epsilon) &= 0, \\ \operatorname{margin}(w0) &= \operatorname{margin}(w) - 1, \\ \operatorname{margin}(w1) &= \begin{cases} \operatorname{margin}(w) + 1 & \text{if } \operatorname{margin}(w) + 1 \neq 0, \\ \operatorname{tail}(w1) & \text{if } \operatorname{margin}(w) + 1 = 0. \end{cases} \end{aligned}$$

Note that these two definitions of margin agree on 0-terminating strings.

We place a probability distribution on $\{0,1\}^n$ by independently selecting each $w_i \in \{0,1\}$ so that

$$\Pr[w_i = 0] = \frac{1+\epsilon}{2} = 1 - \Pr[w_i = 1]$$

and consider the random variables $X_t = \text{margin}(w_1 \dots w_t)$. Note that if it were not for the "exotic" behavior "around zero" (that is, the case that $\text{margin}(w) + 1 = 0$), these random variables would simply describe a biased random walk. In particular, they would arise from the familiar Markov chain of Figure 4 where

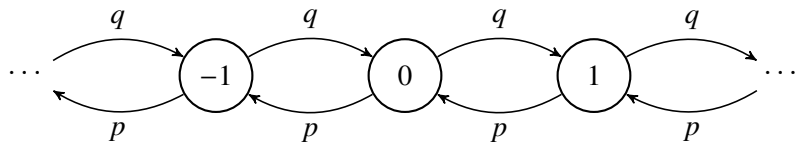


Figure 4: The simple biased walk.

$p = (1+\epsilon)/2$ and $q = 1 - p$.

With the exotic transition $\text{margin}(w1) = \text{tail}(w1)$ (when $\text{margin}(w) = -1$), we note that this process is no longer strictly "Markovian," as this transition depends on number of "recent" 1 symbols in the sequence. We can reflect this with a richer Markov chain over the state space $\mathbb{Z}\times\mathbb{Z}$, which we think of as a Markov chain on the state space $\mathbb{Z}$ (reflecting the current margin) which additionally remembers a "counter" corresponding to the position of the last 0 that was visited. This permits the chain to correctly handle the exotic rule associated with $\text{margin}() = -1$; the chain is described in Figure 5.

The basic event we wish to analyze is the event that after n steps on this Markov chain, the resulting margin is negative; in that case the corresponding string in $\{0,1\}^*$ is not forkable.

Theorem 3.11 *Let $\epsilon \in (0,1)$ and let w be a string drawn from $\{0,1\}^n$ independently assigning each $w_i = 0$ with probability $(1+\epsilon)/2$. Then $\Pr[w \text{ is forkable}] = 2^{-\theta(\sqrt{n})}$.*

Proof. Write $w = w^{(1)} \cdots w^{(\sqrt{n})}$ where $\lfloor\sqrt{n}\rfloor \le |w^{(i)}| \le \lceil\sqrt{n}\rceil$ for each i. Fix $\delta < \epsilon$ to be a small constant. Let $L^{(i)}$ denote the event that there is a contiguous sequence of "1" symbols of length exceeding $\delta\sqrt{n}$ in the string $w^{(i)}$. Then $\Pr[L^{(i)}] \le \sqrt{n}2^{-\delta\sqrt{n}} = 2^{-\theta(\sqrt{n})}$. We remark that these events are independent for distinct values of i, as they involve non-overlapping sets of symbols of w.

Let $m_t = \text{margin}(w^{(1)} \cdots w^{(t-1)})$. We define three events based on this margin:

Hot We let Hot_t denote the event that $m_t \ge 2\delta\sqrt{n}$ or $L^{(t-1)}$ occurred.

Volatile We let Vol_t denote the event that $-2\delta\sqrt{n} \le m_t < 2\delta\sqrt{n}$ and $L^{(t-1)}$ did not occur.

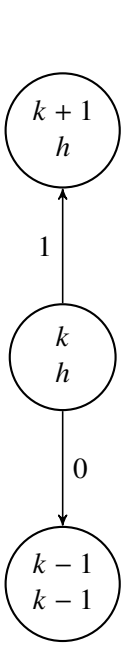
Cold We let Cold_t denote the event that $m_t < -2\delta\sqrt{n}$.

(We assume, by convention, that $L^{(-1)}$ does not occur.) Note that for each t, exactly one of these events occurs—they partition the probability space. Then we will establish that

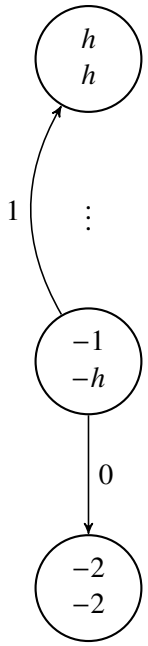
$$\Pr[\mathsf{Cold}_{t+1} \mid \mathsf{Cold}_t] \ge 1 - 2^{-\theta(\sqrt{n})}, \tag{1}$$

$$\Pr[\mathsf{Cold}_{t+1} \mid \mathsf{Vol}_t] \ge \theta(\epsilon), \tag{2}$$

$$\Pr[\mathsf{Hot}_{t+1} \mid \mathsf{Vol}_t] \le 2^{-\theta(\sqrt{n})}. \tag{3}$$



(a) The dynamics at nodes with $k \neq -1$.



(b) The dynamics at $k = -1$.

Figure 5: Diagram of the lifted Markov chain. The first coordinate, k, maintains the current margin; the second coordinate, h, maintains the position of the last appearance of the symbol 0.

Note that the event Vol_1 occurs by definition. We wish to show that the system is very likely to eventually become cold, and stay that way. Note that the probability that the system ever transitions from volatile to hot is no more than $2^{-\theta(\sqrt{n})}$ (as transition from Vol to Hot is bounded above by $2^{-\theta(\sqrt{n})}$, and there are no more than $\sqrt{n}$ possible transition opportunities). Note, also, that while the system is volatile, it transitions to cold with constant probability during each period. In particular, the probability that the system is volatile for the entire process is no more that $2^{-\theta(\sqrt{n})}$. Finally, note that the probability that the system ever transitions out of the cold state is no more than $2^{-\theta(\sqrt{n})}$ (again, there are at most $\sqrt{n}$ possible times when this could happen, and any individual transition occurs with probability $2^{-\theta(\sqrt{n})}$). It follows that the system is cold at the end of the process with probability $1 - 2^{-\theta(\sqrt{n})}$.

In preparation for establishing the three inequalities (1), (2), and (3), we note two facts about the simple biased walk (of Figure 4 above) with $p = (1 + \epsilon)/2$ and $\epsilon > 0$.

Constant escape probability. As $\epsilon > 0$, the probability that an infinite walk beginning at state 0 ever visits the state 0 again is a constant less than 1 (depending only on ϵ). (See, e.g., [10, Chapter 12].)

Concentration. Consider s steps of the Markov chain beginning at state 0; then the resulting value is tightly concentrated around $-\epsilon s$. Specifically, let $Z_1, \ldots, Z_S$ be i.i.d. $\{\pm 1\}$-valued random variables with $\Pr[Z_s = 1] = (1 - \epsilon)/2$, in which case the expected value $\mathbb{E}[\sum_{s=1}^{S} Z_s] = -\epsilon S$. Then

$$\Pr\left[\sum_s Z_s > -\frac{\epsilon S}{2}\right] = 2^{-\theta(S)}\,. \tag{4}$$

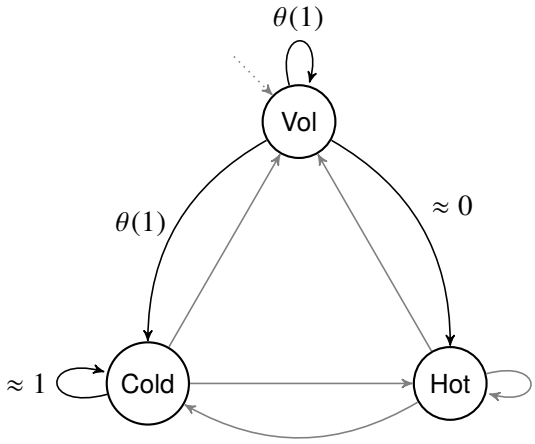


Figure 6: An illustration of the transitions between Cold, Vol, and Hot.

(The constant hidden in the $\theta()$ notation depends only on ϵ. See, e.g., [1, Cor. A.1.14].)

Inequality (1) follows directly from the concentration statement above: Note that unless the Markov chain visits a state for which $k = -1$, it behaves like the simple unbiased walk. Let $D^{(i)}$ denote the event that there is a contiguous sequence of symbols x in $w^{(i)}$ for which $\#1(x) - \#0(x) \geq \delta\sqrt{n}$. By the same Chernoff bound of (4) above, $\Pr[D^{(i)}] \leq 2^{-\theta(\sqrt{n})}$. (Observe that such an event can only happen if the sequence of symbols has length at least $\delta\sqrt{n}$, in which case the Chernoff bound can be applied.) As the chain starts with $m_t < 2\delta\sqrt{n}$, unless D_t occurs, the Markov chain cannot possibly visit the state -1. It follows, again from (4), that the probability that $m_{t+1} \geq 2\delta\sqrt{n}$ is $2^{-\theta(\sqrt{n})}$. (In fact, the Chernoff bound shows that with high probability, the value of m_{t+1} has significantly decreased.) Finally, the probability of L_t is $2^{-\theta(\sqrt{n})}$. Thus the probability of $\overline{\mathsf{Cold}_{t+1}}$ is $2^{-\theta(\sqrt{n})}$, as desired.

As for inequality (2), note that if the system starts with $m_t \leq -1$ then with constant probability it will never visit -1 during (the rest of) $m^{(t)}$ and, conditioned on that, will end with a margin $< -2\delta\sqrt{n}$ with probability $1 - 2^{-\theta(\sqrt{n})}$ (by (4)). If, on the other hand, the system starts with $m_t > -1$ (but less than $2\delta\sqrt{n}$), again by a Chernoff bound it will visit the node $(-1, -1)$ with probability $1 - 2^{-\theta(\sqrt{n})}$ during the first half of the string $w^{(t)}$. As in the other case, the probability that it never returns to margin -1 and ends up below $-2\delta\sqrt{n}$ is a constant. The result follows.

Finally, consider inequality (3). Note that, first of all, by the union bound, $L^{(t)}$ occurs with probability no more than $\sqrt{n}2^{-\theta(\sqrt{n})} = 2^{-\theta(\sqrt{n})}$. The other way for the event Hot_{t+1} to occur is that the margin, initially smaller than $2\delta\sqrt{n}$, "escapes" to a value exceeding this. We separate this analysis into two cases: if the initial margin is positive (or zero), note with probability at least $1 - 2^{-\theta(\sqrt{n})}$, the margin will return to 0 during $w^{(t)}$ by (4). After this, note that assuming that neither $L^{(t)}$ or $D^{(t)}$ occur, the maximum possible margin that can appear in the remainder of $w^{(t)}$ is $2\delta\sqrt{n}$, as desired. (The factor of 2 arises due to the possibility that a transition through zero induces a tail() of size $\delta\sqrt{n}$.) On the other hand, if the initial margin is negative, as $L^{(t-1)}$ did not occur, the same argument concludes that the maximum final margin is no more than $2\delta\sqrt{n}$. ■

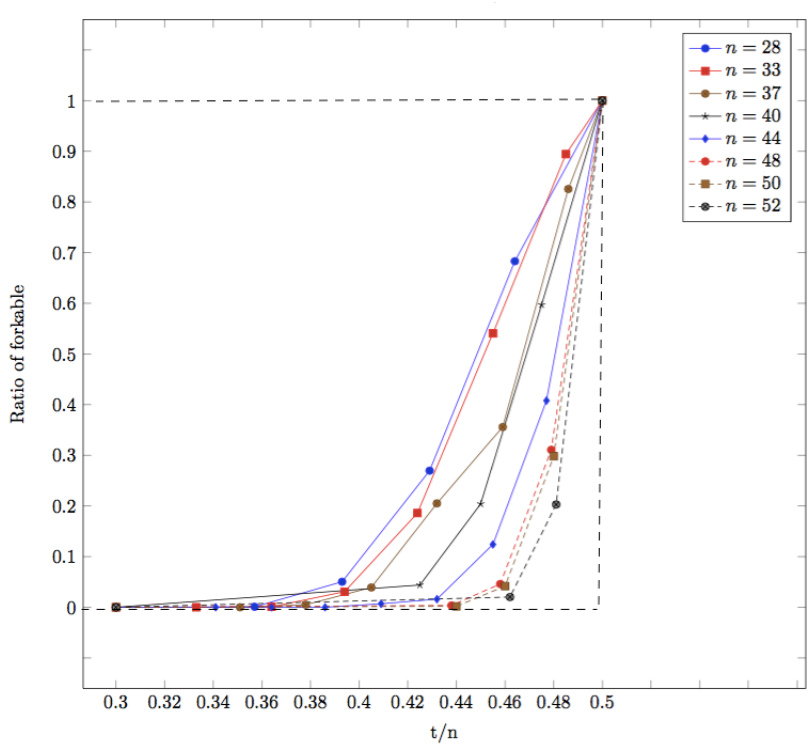


Figure 7: Ratio of forkable strings as the length n grows vs. the ratio of slots assigned to the adversary.

Experiments. In order to gain further insight regarding the density of forkable strings we performed exhaustive search experiments. For each string, an exhaustive search algorithm searches the space of possible forks. The experiments were run on a cluster of 4 servers equipped with Hexa-core Intel Xeon E5-2420 @ 1.90GHz, 16GB RAM, and one 1TB SATA disk, running CentOS 7 Linux.

Our results are presented in Figure 7. As one can observe, as n grows the ratio of forkable strings decays (for $\epsilon > 0$).

3.3 Common Prefix

The common prefix theorem is proven using the following approach. We observe that in order for a fork of length k to be created it should be the case that the characteristic string should have a substring of length k that is forkable. Given that the assignment of slots to stakeholders is a random procedure the number of slots assigned to the adversary follows the Binomial distribution with success probability equal to the adversarial stake. By assuming a bound on the number of forkable strings of a certain Hamming weight we obtain the following statement.

Theorem 3.12 *Let $k, R \in \mathbb{N}$ and $\epsilon \in (0, 1)$. The π_{SPOS} protocol satisfies the common prefix property with parameters k, R throughout a period of R slots with probability* $\exp(-c\sqrt{k} + \ln R)$ *against an adversary*

holding an $\frac{1-\epsilon}{2}$ portion of the total stake; the constant c depends only on ϵ.

Proof. Given the initialization of the protocol we call Bad the event that in any of the R rounds of execution, there exist the necessary conditions for the adversary to break the k-common-prefix property.

We observe the following:

$$\mathsf{Bad} \subseteq \bigcup_{a \in [1, R-k+1]} \mathsf{Bad}_a$$

where Bad_a is defined as the event that the characteristic string that corresponds to the slots $[a, a+k-1]$ is forkable.

To see why this is the case consider the event $\wedge_a \neg \mathsf{Bad}_a$. This means that no characteristic string starting at slot sl_a of length k is forkable.

For the sake of contradiction we will assume that the common prefix property is violated. Take two slots $sl_1 \leq sl_2$. Suppose that two honest players U_1, U_2 active at these two respective slots have chains for which it holds that $C_1^{\lceil k} \not\preceq C_2$. This means that C_2 forks from C_1 more than k blocks deep. Moreover, given that $sl_1 \leq sl_2$, it holds that C_2 is at least as long as C_1. Let $a-1$ be the slot that corresponds to the last common block between C_1, C_2 (equal to 0 if that is the genesis block). Observe that $sl_a + k - 1 < sl_1 \leq sl_2$ since both C_1 and C_2 have more than k blocks added after sl_{a-1}.

We will show that the characteristic string w of length k defined over the sequence of slots $\{sl_a, \ldots, sl_{a+k-1}\}$ is forkable. To achieve this we have to construct a flat legal fork. Consider the execution at slot sl_2 when the violation of the common prefix is observed. We construct a sequence of forks $f_1, \ldots, f_k$, $f_i = (\alpha_1^{(i)}, \alpha_2^{(i)})$, so that $f_i \sqsubseteq f_{i+1}$ for $i = 1, \ldots, k-1$. Let $f_0 = (\epsilon, \epsilon)$. Consider the rank $i \in \{1, \ldots, k-1\}$ of some slot in the sequence $\{sl_a, \ldots, sl_{a+k-1}\}$. We have the following cases.

- In case i is a malicious slot set $f_i = f_{i-1}$.
- In case i is an honest slot and the honest party leader of the slot extended some prefix of the chains C_1, C_2, say C_u, with $u \in \{1, 2\}$, define the tine $\alpha_u^{(i)}$ by appending to $\alpha_u^{(i-1)}$ all the malicious slots that correspond to the blocks in C_u, starting from the block that corresponds to the earliest slot not included in $\alpha_u^{(i-1)}$ and going up to $i-1$; finally append also i. The other tine is simply defined as $\alpha_{3-u}^{(i)} = \alpha_{3-u}^{(i-1)}$.
- In case i is an honest slot and the honest party leader of the slot extended some other chain C which is not a prefix of of C_1, C_2, perform the same steps as in the case above, picking $u \in \{1, 2\}$ to be the shortest tine (favoring $u = 1$ in case both tines are equal).

Observe that the fork f_i constructed in this way is incrementally legal and closed. Finally we pad the two tines of f_k with sufficient number of remaining adversarial slots from $\{sl_a, \ldots, sl_{a+k-1}\}$ to obtain a flat fork f such that $f_k \sqsubseteq f$. The fork f is flat and legal and thus the string w is forkable, which is a contradiction on $\neg \mathsf{Bad}_a$.

We proceed now to provide a bound on Bad. By the union bound we have that, $\mathbf{Pr}[\mathsf{Bad}] \leq R \cdot \varepsilon$ where ε is density of forkable strings of length k from which we obtain immediately the statement of the theorem using Theorem 3.11. ∎

3.4 Chain Growth and Chain Quality

We will start with the chain growth property.

Theorem 3.13 *The π_{SPOS} protocol satisfies the chain growth property with parameters $\tau = 1 - \alpha, s \in \mathbb{N}$ throughout an epoch of R slots with probability $1 - \exp(-\epsilon^2 s + \ln R)$ against an adversary holding an $\alpha - \epsilon$ portion of the total stake.*

Proof. Define $\mathsf{Ham}_a(\alpha)$ to be the event that the Hamming weight ratio of the characteristic string that corresponds to the slots $[a, a + s - 1]$ is up to α. Given that the adversarial stake is $\alpha - \epsilon$, each of the k slots has probability $\alpha - \epsilon$ being assigned to the adversary and thus the probability that the Hamming weight is more than αs drops exponentially in s. Specifically, using the additive version of the Chernoff bound, we have that $\mathbf{Pr}[\neg\mathsf{Ham}_a(\alpha)] \leq \exp(-2\epsilon^2 s)$. It follows that,

$$\mathbf{Pr}[\mathsf{Ham}_\alpha] \geq 1 - \exp(-2\epsilon^2 s).$$

Given the above we know that when Ham_α happens there will be at least $(1 - \alpha)s$ honest slots in the period of s rounds. Given that each honest slot enables an honest party to produce a block, all honest parties will advance by at least that many blocks. Using a union bound, it follows that the speed coefficient can be set to $\tau = (1 - \alpha)$ and it is satisfied with probability $1 - \exp(-2\epsilon^2 s + \ln(R))$. ∎

Having established the chain growth property we now turn our attention to the chain quality property. Recall that the chain-quality property parameterized with k and it states that every k blocks in a chain observed at a certain slot the blocks corresponding to a set of stakeholders that hold cumulative stake ratio β are $\tau\beta$. In the next theorem we establish bounds for the parameter τ.

Theorem 3.14 *The π_{SPOS} protocol satisfies the chain quality property with parameters $\mu = \alpha/(1 - \alpha)$, $k \in \mathbb{N}$ throughout an epoch of R slots with probability*

$$1 - \exp(-\epsilon^2(1 - \alpha)^{-1}k + \ln R)$$

where $\alpha - \epsilon$ is the ratio of the cumulative stake of the set of malicious stakeholders.

Proof. First, using a similar argumentation as in the chain growth Theorem 3.13, we know that in a segment of s rounds the honest parties would advance by at least $(1 - \alpha)s$ blocks. Furthermore the adversary can produce at most αs blocks in the same period. It follows that in the chain of any honest party one would find at most $\alpha/(1 - \alpha)$ ratio of blocks originating from the adversary with probability $1 - \exp(-\epsilon^2 s + \ln R)$ among the blocks produced in the period that corresponds to that segment. It suffices to choose $s \geq (1 - \alpha)^{-1}k$. In this case we know that there will be at least k blocks produced in any segment of s rounds. This concludes the proof. ∎

4 Our Protocol: Dynamic Stake

4.1 Using a Trusted Beacon

In the previous protocol we assumed that stake was static during the whole execution (i.e., one epoch), meaning that stake changing hands inside a given epoch does not affect leader election. Now we consider a modification of Protocol π_{SPOS} that can be executed over multiple epochs in such a way that each epoch's leader election process is parameterized by the stake distribution at a certain designated point of the previous epoch, allowing for change in the stake distribution across epochs to affect the leader election process. Once again, we will construct the protocol in the $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$-hybrid model, assuming that the $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ ideal functionality provides randomness and auxiliary information for the leader election process at each epoch.

Before describing the protocol for the case of dynamic stake, we need to provide a modification of $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ that considers multiple epochs. We call such functionality $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ and allow stakeholders to query it for the leader selection process information specific to each epoch. $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ is parameterized by the initial stake of each stakeholder before the first epoch e_1 starts but in further epochs takes into consideration the stake distribution after the previous epoch's first $R - k$ slots, where k is the number of slots needed to achieve common prefix. Notice that it is necessary to consider the stake distribution of previous epochs only in the slots where it is guaranteed that common prefix is achieved, since an adversary who can force a fork would put the ideal functionality in an inconsistent (actually undefined) state.

We denote by $\{(\mathsf{vk}_1, s_1^j), \ldots, (\mathsf{vk}_n, s_n^j)\}$ the set of pairs of stakeholder verification keys vk_i and respective stakeholder's stake s_i^j after the first $R - k$ rounds of epoch e_j, where k is the number of slots necessary to ensure common prefix, for $1 \leq i \leq n$. We abuse notation in defining as $\{(\mathsf{vk}_1, s_1^0), \ldots, (\mathsf{vk}_n, s_n^0)\}$ the stake associated to each stakeholder (represented by its verification key) before protocol execution (and epoch e_1) starts. $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ is defined in Figure 8.

Functionality $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$

$\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ is parameterized by the public keys and respective stakes of the stakeholders $\{(\mathsf{vk}_1, s_1^0), \ldots, (\mathsf{vk}_n, s_n^0)\}$ before epoch e_1 starts, a distribution D and a leader selection function F. $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ interacts with stakeholders $U_1, \ldots, U_n$ as follows:

- Upon receiving $(\mathsf{genblock\_req}, U_i, e_j)$ from stakeholder U_i, if j is a valid epoch (within the current total number of epochs), $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ proceeds as follows. If $B_0^j = \emptyset$, $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ samples $\rho^j \leftarrow D$ and sets $B_0^j = \{(\mathsf{vk}_1, s_1^{j-1}), \ldots, (\mathsf{vk}_n, s_n^{j-1}), \rho^j\}$[a]. Finally, $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ sends $(\mathsf{genblock}, B_0^j, \mathsf{F})$ to U_i.

[a] Notice that here $\{(\mathsf{vk}_1, s_1^j), \ldots, (\mathsf{vk}_n, s_n^j)\}$ refers to the stake associated to each stakeholder in the epoch previous to the epoch for which the query was issued.

Figure 8: Functionality $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$.

For the case of dynamic stake we construct Protocol π_{DPOS}, which is a modified version of π_{SPOS} that updates its genesis block B_0 (and thus the leader selection process) for every new epoch. Protocol π_{DPOS} is described in Figure 9 and functions in the $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$-hybrid model.

Protocol π_{DPOS}

π_{DPOS} is a protocol run by stakeholders $U_1, \ldots, U_n$ interacting among themselves and with $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$ over a sequence of L slots $S = \{sl_1, \ldots, sl_L\}$. π_{DPOS} proceeds as follows:

1. **Initialization** When π_{SPOS} starts, each stakeholder $U_i \in \{U_1, \ldots, U_n\}$ sends $(\mathsf{genblock\_req}, U_i)$ to $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$, receiving $(\mathsf{genblock}, B_0, \mathsf{F})$ as answer. U_i sets an internal blockchain $C = B_0$ and a initial internal state $st = H(B_0)$.
2. **Chain Extension** For every slot $sl_j \in S$, every online stakeholder U_i performs the following steps:
   (a) If a new epoch e_k has started, U_i sends $(\mathsf{genblock\_req}, U_i, e_j)$ to $\mathcal{F}_{\mathcal{LS}}^{D,\mathsf{F}}$, receiving $(\mathsf{genblock}, B_0^j, \mathsf{F})$ as answer. U_i extends its internal blockchain with B_0^j and sets it as the new epoch's genesis block, storing ρ^{k-1} and parameterizing the leader selection function F with ρ^k contained in the new B_0^k. If more than one epoch has passed, U_i repeats this procedure for each new epoch.
   (b) Collect all valid chains received via broadcast into a set $\mathbb{C}$, verifying that for every chain $C' \in \mathbb{C}$ and every block $B' = (st', d', sl', \sigma') \in C'$ it holds that $\mathsf{Vrf}_{\mathsf{vk}'}(\sigma', (st', d', sl')) = 1$, where vk' is the verification key of the stakeholder $U' \leftarrow \mathsf{F}(\rho^k, sl')$ with F parameterized by ρ^k corresponding to the slot to which B' belongs (as determined by sl'). U_i calls the function $\mathsf{maxvalid}(C, \mathbb{C})$ to select a new internal chain $C \in \mathbb{C}$ and sets state $st = H(B_h)$, where $B_h = \text{head}(C)$.
   (c) If U_i is the slot leader determined by $\mathsf{F}(\rho^k, sl_j)$, it generates a new block $B = (st, d, sl_j, \sigma)$ where st is its current state, $d \in \{0,1\}^*$ is data and $\sigma = \mathsf{Sign}_{\mathsf{sk}_i}(st, d, sl_j)$ is a signature on (st, d, sl_j). U_i extends C by appending B, obtains $C = C|B$ and broadcasts the new C.

Figure 9: Protocol π_{DPOS}

4.2 Simulating a Trusted Beacon

While protocol π_{DPOS} handles multiple epochs and takes into consideration changes in the stake distribution, it still relies on $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ to perform the leader selection process. In this section, we show how to implement $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ through Protocol π_{DLS}, which allows the stakeholders to compute the randomness and auxiliary information necessary in the leader election.

Our starting point is the follow-the-satoshi algorithm, which takes as input uniform randomness and outputs a stakeholder U_i randomly selected with probability $p_i = \frac{s_i}{\sum_{k=1}^{n} s_k}$, where s_i is the stake held by stakeholder U_i. Intuitively, the leader selection process will be such that F is follow-the-satoshi and D is the uniform distribution. Protocol π_{DLS} will use a coin tossing protocol to generate unbiased randomness that can be used to run follow-the-satoshi given an honest majority of stakeholders. However, notice that the adversary could cause a simple coin tossing protocol to fail by aborting. Thus, we build a coin tossing scheme with guaranteed output delivery.

Leader Selection Process. Follow-the-satoshi is parameterized by the total amount of satoshis (i.e., smallest fractions of coins) τ in the system and we will denote it by $F(\rho, sl_j)$; ρ is auxiliary information and $sl_j \in \{sl_1, \ldots, sl_R\}$. In our concrete case, $\rho = (\rho_1, \rho_2)$, where $\rho_1 \leftarrow \{0,1\}^{R \log \tau}$ and $\rho_2 = \{(\mathsf{vk}_1, s_1^j), \ldots, (\mathsf{vk}_n, s_n^j)\}$ (i.e., the distribution of stake in an epoch e_j). Let each satoshi be uniquely identified by a label $\varsigma \in \{0,1\}^{\log' \tau}$. The output of $F(\rho, sl_j)$ is computed by selecting the satoshi identified by label $\varsigma = \varsigma_{(1-j)\log \tau}, \ldots, \varsigma_{j \log \tau}$ of ρ and then outputting the stakeholder U_i who currently owns that satoshi

(i.e., the stakeholder whose address was that last one to receive that satoshi as output in a transaction according to the distribution of stake $\rho_2 = \{(\mathsf{vk}_1, s_1^j), \ldots, (\mathsf{vk}_n, s_n^j)\}$). Notice that, since ρ_1 is a uniformly random string, each satoshi is selected at random. Hence, given that a stakeholder U_i has a number s_i of satoshis, one of its satoshis is selected by follow-the-satoshi (causing it to be selected as the final output) with probability $p_i = \frac{s_i}{\sum_{k=1}^{n} s_k}$. The leader selection process is defined as (D, F), where D is the joint distribution of uniformly random binary strings of length $R \log \tau$ and stake distributions of the form $\{(\mathsf{vk}_1, s_1^j), \ldots, (\mathsf{vk}_n, s_n^j)\}$, and F is the follow-the-satoshi function. Notice that an easy way to reduce the amount of initial uniform randomness necessary to run F (i.e., reduce the size of ρ) is to start with a smaller uniformly random string $s \in \{0,1\}^\kappa$ and use it to seed a pseudorandom number generator $PRG(s)$ in order to obtain $r \in \{0,1\}^{R \log \tau}$, where κ is a security parameter of PRG.

Commitments and Coin Tossing. A coin tossing protocol allows two or more parties to obtain a uniformly random string. A classic approach to construct such a protocol is by using commitment schemes. In a commitment scheme, a *committer* carries out a *commitment phase*, which sends evidence of a given value to a *receiver* without revealing it; later on, in an *opening phase*, the committer can send that value to the receiver and convince it that the value is identical to the value committed to in the commitment phase. Such as scheme is called *binding* if it is hard for the committer to convince the receiver that he was committed to any value other than the one for which he sent evidence in the commitment phase, and it is called *hiding* if it is hard for the receiver to learn anything about the value before the opening phase. We denote the commitment phase with randomness r and message m by $\mathsf{Com}(r, m)$ and the opening as $\mathsf{Open}(r, m)$.

In a standard two-party coin tossing protocol, one party starts by sampling a uniformly random string u_1 and sending $\mathsf{Com}(r, u_1)$. Next, the other party sends another uniformly random string u_2 in the clear. Finally, the first party opens u_1 by sending $Open(r, u_1)$ and both parties compute output $u = u_1 \oplus u_2$.

Verifiable Secret Sharing (VSS). A secret sharing scheme allows a *dealer* P_D to split a secret σ into n *shares* distributed to parties $P_1, \ldots, P_n$, such that no adversary corrupting up to t parties can recover σ. In a Verifiable Secret Sharing scheme [8], there is the additional guarantee that the honest parties can recover σ even if the adversary corrupts the shares held by the parties that it controls and even if the dealer itself is malicious. We define a VSS scheme as a pair of efficient dealing and reconstruction algorithms $(\mathcal{D}, \mathcal{R})$. The dealing algorithm $\mathcal{D}(n, \sigma)$ takes as input the number of shares to be generated n along with the secret σ and outputs shares $\sigma_1, \ldots, \sigma_n$. The reconstruction algorithm $\mathcal{R}$ takes as input shares $\sigma_1, \ldots, \sigma_n$ and outputs the secret σ as long as no more than t shares are corrupted (unavailable shares are set to $\perp$ and considered corrupted). A simple VSS construction based on discrete logarithms that can be used is by Schoenmakers, [19].

Constructing Protocol π_{DLS}. The main problem to be solved when realizing $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$ with a protocol run by the stakeholders is that of generating uniform randomness for the leader selection process while tolerating adversaries that may try to interfere by aborting or feeding incorrect information to parties. In order to generate uniform randomness ρ_1 for follow-the-satoshi F we will employ a coin tossing scheme for which all honest parties are guaranteed to receive output as long as there's an honest majority. In the first round, for $1 \leq i \leq n$, stakeholder U_i samples a uniformly random string $u_i \in \{0,1\}^{R \log \tau}$ and randomness r_i for the

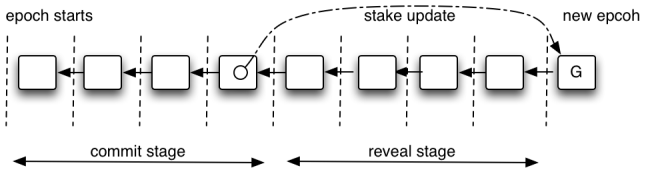


Figure 10: The two stages of the protocol π_{DPOS} that use the blockchain as a broadcast channel.

underlying commitment scheme, generates shares $\sigma_1^i, \ldots, \sigma_n^i$, posts $\mathsf{Com}(r_i, u_i)$ to the blockchain together with the encryptions of the all the shares under the public-key of each respective shareholder. After k slots, when common prefix is reached, if commitments from a majority of stakeholders are posted on the blockchain and if shares from a majority of stakeholders have been received, for $1 \leq i \leq n$, stakeholder U_i posts $\mathsf{Open}(r_i, u_i)$ to the blockchain. If a stakeholder U^a does not post an opening to its commitment, the honest parties can use shares $\sigma_1^a, \ldots, \sigma_n^a$ and $\mathcal{R}(\sigma_1^a, \ldots, \sigma_n^a)$ to reconstruct u^a. Given that all the values should be revealed, the shareholders can post the openings of the commitments independently of whether U_i posts the correct opening value or not. Next, each stakeholder uses the values u_i obtained in the second round to compute $\rho_1 = \sum_i u_i$. Finally, in the next epoch, each honest stakeholder sets $\rho = (\rho_1, \rho_2)$, where $\rho_2 = \{(\mathsf{vk}_1, s_1^{j-1}), \ldots, (\mathsf{vk}_n, s_n^{j-1})\}$. Protocol π_{DLS} is described in figure Figure 11.

The two stages of the protocol are presented in Figure 10.

Protocol π_{DLS}

π_{DLS} is a protocol run by stakeholders $U_1, \ldots, U_n$ interacting among themselves over a sequence of L slots $S = \{sl_1, \ldots, sl_L\}$ and proceeds as follows for every epoch e_j that lasts $R = 3k$ slots:

1. **Commitment Phase** ($2k$ slots) When epoch e_j starts, for $1 \leq i \leq n$, stakeholder U_i samples a uniformly random string $u_i \in \{0,1\}^{R \log \tau}$ and randomness r_i for the underlying commitment scheme, generates shares $\sigma_1^i, \ldots, \sigma_n^i \leftarrow \mathcal{D}(n, u_i)$, posts $\mathsf{Com}(r_i, u_i)$ to the blockchain and encrypts each share σ_k^i under stakeholder U_k public-key.
2. **Reveal Phase** ($2k$ slots) After slot $R - k$, (when it is guaranteed that common prefix was achieved for the blocks containing the commitments), for $1 \leq i \leq n$, stakeholder U_i opens its commitment by posting $\mathsf{Open}(r_i, u_i)$ to the blockchain.
   (Recovery) For any stakeholder U^a, for $1 \leq i \leq n$, U_i submits its share σ_i^a for insertion to the blockchain. When all shares $\sigma_1^a, \ldots, \sigma_n^a$ are available, each stakeholder U_i can compute $\mathcal{R}(\sigma_1^a, \ldots, \sigma_n^a)$ to reconstruct u_a (independently of whether U^a opens the commitment or not).
3. **Leader Selection Parameters Generation** Let $\mathbb{L} = \{l_1, \ldots, l_h\}$ be the set of indexes for which a value u_l was successfully obtained or recovered. For $1 \leq i \leq n$, U_i uses the values computes $\rho_1 = \sum_{l \in \mathbb{L}} u_l$. Finally, in the next epoch, each honest stakeholder sets $\rho = (\rho_1, \rho_2)$, where $\rho_2 = \{(\mathsf{vk}_1, s_1^{j-1}), \ldots, (\mathsf{vk}_n, s_n^{j-1})\}$ and run $\mathsf{F}(\rho, \cdot)$ to determine the leader of slots in epoch e_j.

Figure 11: Protocol π_{DLS}.

Security Proof Sketch. Notice that, if a majority of the stakeholders are honest, they either obtain enough values u_l to compute a uniformly random string ρ_1 by the end of the reveal phase or manage to recover such

values if the respective stakeholders do not open their commitments. Given that ρ_1 is a uniformly random string and ρ_2 is represents the stake distribution from last epoch, $\mathsf{F}(\rho, \cdot)$ behaves as the leader selection process described before, thus selecting a leader with the same probability of $\mathcal{F}_{\mathcal{DLS}}^{D,\mathsf{F}}$.

4.3 Robust Transaction Ledger

Recall that in the dynamic stake case, we would have to conceive a way to prevent deep forks. To see this formally consider a player who is offline and joins the system after a number of epochs have passed. Even if in the system execution the current set of stakeholders satisfies honest majority, it could be the case that honest majority is violated in one of the previous epochs by this time and hence the adversary may produce an alternative history consistent with the view of honest party. In order to capture the interaction between security and the modification of stake we introduce the following property.

Definition 4.1 *Consider two slots sl_1, sl_2, an honest player U and an execution $\mathcal{E}$. The stake shift w.r.t. U between sl_1, sl_2 is the statistical distance of the two follow-the-satoshi distributions that are defined using the stake reflected in the chain $\mathcal{C}$ of U in the most recent blocks before sl_1 and sl_2 respectively as reflected in the execution $\mathcal{E}$.*

Taking into account the definition above we can now express the following theorem about the common prefix property.

Theorem 4.2 *Fix parameters $k, R, L \in \mathbb{N}, \epsilon, \sigma \in (0, 1)$. Let R be the epoch length and L the total lifetime of the system. Assume the adversary is restricted to $\frac{1-\epsilon}{2} - \sigma$ relative stake and that the π_{SPOS} protocol satisfies the common prefix property with parameters R, k and probability of error ϵ_{CP}, the chain quality property with parameters $\mu \geq 1/k, k$ and probability of error ϵ_{CQ} and the chain growth property with parameters $\tau \geq 1/2, k$ and probability of error ϵ_{CG}.*

Then, the π_{DPOS} protocol satisfies the persistence with parameters k and liveness with parameters $u = 2k$ throughout a period of L slots with probability $1 - R(\epsilon_{\mathsf{CQ}} + \epsilon_{\mathsf{CG}} + \epsilon_{\mathsf{CG}})$, assuming that σ is the maximum stake shift over $2k$ slots and no honest player is offline for more than k slots.

Proof. (sketch) Observe that with probability of error $\epsilon_{\mathsf{CQ}} + \epsilon_{\mathsf{CG}} + \epsilon_{\mathsf{CG}}$ the π_{SPOS} protocol executed in the first epoch, given the assumptions imposed to the environment, will enable the parties to use the blockchain as a broadcast channel to simulate the trusted beacon and produce the randomness required to seed the leader election in the next round (this combines Theorems 3.12, 3.14, 3.13). This can be seen as follows: given that chain growth holds with coefficient $1/2$, the chain of all honest parties will grow for at least k blocks during the commitment phase. Moreover, given chain quality there will at least one block that will be inserted by an honest party. This will contain the commitments and the VSS sharings of all honest parties. With a similar argument in the reveal phase there will be at least one block included in the chain by an honest party that will contain all the openings for the commitments that were made in the first stage. Observe that this would be the case independently of small forks occurring during the stages (as long as no deeper than k forks occur).

Note that as shown in Figure 10 a delay in calculating the stake for the next epoch will result in a bias in the proper calculation of the leader election for the next epoch. This is accounted for by the further restriction that is imposed on the adversarial stake in the statement of the theorem.

It follows that with probability $1 - (L/R)(\epsilon_{\mathsf{CQ}} + \epsilon_{\mathsf{CG}} + \epsilon_{\mathsf{CG}})$ all epochs in the lifetime of the system will be seeded correctly and the π_{SPOS} protocol can be bootstrapped and will continue to operate properly in each next epoch. Persistence and liveness with the stated parameters follow. ■

4.4 Input Endorsers

We next present an extension of our basic protocol that assigns two different roles to stakeholders. As before in each epoch there is a set of elected stakeholders that runs the secure multiparty coin flipping protocol and are the slot leaders of the epoch. Together with those there is a (not necessarily disjoint) set of stakeholders called the endorsers. Now each slot has two types of stakeholders associated with it; the slot leader who will issue the block as before and the slot *endorser* who will endorse the input to be included in the block. Moreover, contrary to slot leaders, we can elect multiple slot endorsers for each slot. While this seems like an insignificant modification it gives us a room for improvement because of the following reason: endorsers' contributions will be acceptable even if they are $2k$ slots late.

Note that in case no valid endorser input is available when the slot leader is about to issue the block, the leader will go ahead and issue an empty block, i.e., a block without any actual inputs (e.g., transactions in the case of a transaction ledger). Note that slot endorsers just like slot leaders are selected by follow-the-satoshi and thus they are a representative sample of the stakeholder population. In the case of a transaction ledger the same transaction might be included by many input endorsers simultaneously. In case that a transaction is multiply present in the blockchain its first occurrence only will be its "canonical" position in the legder.

The enhanced protocol, π_{DPOSwE}, can be easily seen to have the same persistence and liveness behaviour as π_{DPOS}: the modification with endorsers does not provide any possibility for the adversary to prevent the chain from growing, accepting inputs, or being consistent. However, if we measure chain quality in terms of number of *endorsed inputs* included this produces a much more favorable result: it is easy to see that the number of endorsed inputs originating from a set of stakeholders S in any k-long portion of the chain is proportional to the relative stake of S with high probability. This stems from the fact that it is sufficient that a single honest block is created for all the endorsed inputs of the last $2k$ slots to be included in it. Given that any set of stakeholders S will be an endorser in a subset of the $2k$ slots with probability proportional to its cumulative stake the result follows.

5 Attacks Discussion

We next discuss a number of practical attacks and indicate how they are reflected by our modeling.

Double spending attacks. In a double spending attack, the adversary wishes to revert a transaction that is confirmed by the network. The objective of the attack is to issue a transaction, e.g., a payment from an adversarial account holder to a victim recipient, have the transaction confirmed and then revert the transaction by, e.g., including in the ledger a second conflicting transaction. Such an attack is not feasible under the conditions of Theorem 4.2. Indeed, persistence ensures that once the transaction is confirmed by an honest player, all other honest players from that point on will also confirm it. Thus it will be impossible to bring the system to a state where the confirmed transaction is invalidated (assuming all preconditions of the theorem hold).

Transaction denial attacks. In a transaction denial attack, the adversary wishes to prevent a certain transaction from becoming confirmed. For instance, the adversary may want to target a specific account and prevent the account holder from issuing an outgoing transaction. Such an attack is not feasible under the conditions of Theorem 4.2. Indeed, liveness ensures that, provided the transaction is attempted to be inserted for a sufficient number of slots by the network, it will be eventually confirmed by an honest party.

Desynchronization attacks. In a desynchronization attack, a shareholder behaves honestly but is nevertheless incapable of synchronizing correctly with the rest of the network. This leads to ill-timed issuing of blocks and being offline during periods when the shareholder is supposed to participate. Such an attack can be mounted by preventing the party's access to a time server or any other mechanism that allows synchronization between parties. Moreover, a desynchronization may also occur due to exceedingly long delays in message delivery. Our model allows parties to become desynchronized by incorporating them into the adversary. No guarantees of liveness and persistence are provided for desynchronized parties.

Eclipse attacks. In an eclipse attack, message delivery to a shareholder is violated due to a subversion in the peer-to-peer message delivery mechanism. As in the case of desynchronization attacks, our model allows parties to be eclipse attacked by incorporating them into the adversary. No guarantees of liveness or persistence are provided for such parties.

51% attacks. A 51% attack occurs whenever the adversary controls more than the majority of the stake in the system. It is easy to see that any sequence of slots in such a case is with very high probability forkable and thus once the system finds itself in such setting the honest stakeholders may be placed in different forks for long periods of time. Both persistence and liveness can be violated.

Nothing at stake and past majority attacks. As stake moves our assumption is that only the *current* majority of stakeholders is honest. This means that past account keys (which potentially do not hold any stake at present) may be compromised. This leads to a serious vulnerability for any PoS system since a set of malicious shareholders from the past can build an alternative blockchain exploiting such old accounts and the fact that it is effortless to build such a blockchain. In light of Theorem 4.2 such attack can only occur against shareholders who are not frequently online to observe the evolution of the system or in case the stake shifts are higher than what is anticipated by the preconditions of the theorem. This is a special instance of the "nothing at stake" problem which refers in general to attacks against PoS blockchain systems that are facilitated by shareholders continuing simultaneously multiple blockchains exploiting the fact that little computational effort is needed to build a PoS blockchain. With respect to nothing at stake it is worth noting that, contrary to PoW-based blockchains, in our protocol it is infeasible to have a fork generated in earnest by two shareholders. This is because slots are uniquely assigned and thus at any given moment there is a single uniquely identified shareholder that is elected to advance the blockchain. Players following the longest chain rule will adopt the newly minted block (unless the adversary presents at that moment an alternative blockchain using older blocks).

6 Incentive Structure

While the analysis we perform is in the cryptographic setting of [9], we include in this section a discussion regarding the incentive structure of our system. Note that game theoretic analysis is also very important, see [11] for a recent analysis of bitcoin. We focus our analysis in the variant of our system with endorsers described in Section 4.4.

As in bitcoin, shareholders that issue blocks are incentivized to participate in the protocol by collecting transaction fees. Contrary to bitcoin of course, one does not need to incentivize shareholders to invest computational resources. Rather, *availability* is incentivized. Any shareholder, at minimum, must be online in the following circumstances.

- In the slot prior to a slot she is the elected shareholder so that she queries the network and obtains the currently longest blockchain.
- In the slot during which she is the elected shareholder so that she issues the block.
- In a slot during the commit stage of an epoch where she is supposed to issue the VSS commitment of her random string.
- In a slot during the reveal stage of an epoch where she is supposed to issue the required opening shares as well as the opening to her commitment.
- In general, in sufficient frequency, to check whether she is an elected shareholder for the next or current epoch.
- In a slot during which she is the elected input endorser so that she issues the endorsed input (e.g., the set of transactions).

In order to incentivize the above actions in the setting of a transaction ledger, fees can collected from those that issue transactions to be included in the ledger which can then be transfered to the block issuers. In bitcoin for instance fees can be collected by the miner that produces a block of transactions as a reward. In our setting, similarly, a reward can be given to the parties that are issuing blocks and endorsing inputs. The reward mechanism does not have to be immediate as advocated in [17]. For instance it is possible to collect all fees of transactions included in a sequence of k blocks in a pool and then distribute that pool to all shareholders that participated during these k slots. For instance all input endorsers that were active may receive proportionally to the number of inputs they endorsed during the period of k rounds (independently of the actual number of transactions they endorsed).

Other ways to distribute transaction fees are also feasible (including the one that is used by bitcoin itself - even though the bitcoin method is known to be vulnerable to attacks, e.g., the selfing-mining attack). It is beyond the scope of the current exposition to provide a formal analysis of the incentive structure discussed above. This analysis should be performed in a game theoretic setting that also takes into account costs of being online vs. expected rewards from participating in the protocol.

References

[1] Noga Alon and Joel Spencer. *The Probabilistic Method*. Wiley, 3rd edition, 2008.

[2] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 538–557. Springer, 2014.

[3] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. *CoRR*, abs/1406.5694, 2014.

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

[5] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In *23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[6] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 585–605. Springer, 2015.

[7] Ittay Eyal and Emin Gun Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2014.

[8] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 427–437. IEEE Computer Society, 1987.

[9] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 281–310, 2015.

[10] Charles M Grinstead and J Laurie Snell. *Introduction to Probability*. American Mathematical Society, 2nd edition, 1997.

[11] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In Vincent Conitzer, Dirk Bergemann, and Yiling Chen, editors, *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 365–382. ACM, 2016.

[12] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. `http://eprint.iacr.org/2015/1019`.

[13] Tal Moran and Ilan Orlov. Proofs of space-time and rational proofs of storage. Cryptology ePrint Archive, Report 2016/035, 2016. http://eprint.iacr.org/2016/035.

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

[15] Karl J. O'Dwyer and David Malone. Bitcoin mining and its energy footprint. *ISSC 2014 / CIICT 2014, Limerick, June 26–27*, 2014.

[16] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. Spacemint: A cryptocurrency based on proofs of space. *IACR Cryptology ePrint Archive*, 2015:528, 2015.

[17] Rafael Pass. Cryptography and game theory. Securty and Cryptography for Networks, 2016, invited talk., 2016.

[18] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. *CoRR*, abs/1507.06183, 2015.

[19] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 148–164. Springer, 1999.