wishes within a bounded time window. We call the resulting primitive a "leaky resettable beacon" and show that our bootstrapping argument still holds in this stronger adversarial setting.

In the final refinement of our protocol, we show how it is possible to implement the leaky resettable beacon via a simple algorithm that concatenates the VRF outputs that were contributed by the participants from the blockchain and subjects them to a hash function that is modeled as a random oracle. This implementation explains the reasons behind the beacon relaxation we introduced: leakiness stems from the fact that the adversary can complete the blockchain segment that determines the beacon value before revealing it to the honest participants, while resettability stems from the fact that the adversary can try a bounded number of different blockchain extensions that will stabilize the final beacon value to a different preferred value.

Putting all the above together, we show how our protocol provides a "robust transaction ledger" in the sense that an immutable record of transactions is built that also guarantees that new transactions will be always included. Our security definition is in the Δ-semisynchronous setting with full adaptive corruptions. As mentioned above, security degrades gracefully as Δ increases, and this parameter is unknown to the protocol participants.

Note that implementing the beacon via hashing VRF values will make feasible a type of "grinding attack" where the adversary can trade hashing power for a slight bias of the protocol execution to its advantage. We show how this bias can be controlled by suitably increasing the relevant parameters depending on the hashing power that is available to the adversary.

Comparison to related work. The idea of proof-of-stake protocols has been discussed extensively in the bitcoin forum.[2] The manner that a stakeholder determines eligibility to issue a block is always publicly verifiable and the proof of eligibility is either computed publicly (via a calculation that is verifiable by repeating it) or by using a cryptographic mechanism that involves a secret-key computation and a public-key verification. The first example of the former approach appeared in PPCoin [KN12], and was followed by others including Ouroboros and Snow White [BGM14,KRDO17,DPS16]; while the first example of the latter approach (that we also employ in our work) appeared in NXT (cf. Section 2.4.1 of [Com14]) and was then also used elsewhere, most notably in Algorand [Mic16]. The virtue of the latter approach is exactly in its potential to control adaptive corruptions: due to the fact that the adversary cannot predict the eligibility of a stakeholder to issue a block prior to corrupting it, she cannot gain an advantage by directing its corruption quota to specific stakeholders. Nevertheless, none of these previous works isolated explicitly the properties of the primitives that are required to provide a full proof of security in the setting of adaptive corruptions. Injecting high quality randomness in the PoS blockchain was proposed by Bentov et al. [BLMR14,BGM16], though their proposal does not have a full formal analysis. The Ouroboros proof-of-stake protocol [KRDO17] is provably secure in a corruption model that excludes fully adaptive attacks by imposing a corruption delay on the corruption requests of the adversary. The Snow White proof-of-stake [DPS16] is the first to prove security in the Δ-semi-synchronous model but—as in the case of Ouroboros—adopts a weak adaptive corruption model.

A recent work close to ours is Algorand [Mic16] that also provides a proof-of-stake ledger that is adaptively secure. It follows an entirely different construction approach that runs a Byzantine agreement protocol for every block and achieves adaptive-corruption security via a novel, appealing concept of player-replaceability. However, Algorand is only secure against a 1/3 adversary bound; and while the protocol itself is very efficient, it yields an inherently slower block production rate compared to an "eventual consensus" protocol (like Bitcoin, Snow White, and Ouroboros). In principle, proof-of-stake blockchain protocols can advance at the theoretical maximum speed (of one block per communication round), while protocols relying on Byzantine agreement, like Algorand, would require a larger number of rounds to settle each block.

Sleepy consensus [PS16] puts forth a technique for handling adaptive corruptions in a model that also encompasses fail-stop and recover corruptions; however, the protocol can be applied directly only in a static stake (i.e., permissioned) setting. We note that in fact our protocol can be also proven secure in such mixed corruption setting, where both fail-stop and recover as well as Byzantine corruptions are allowed (with the former occurring at an arbitrarily high rate); nevertheless this is out of scope for the present exposition and we omit further details.

[2] Refer e.g., to the posts by QuantumMechanic and others from 2011 `https://bitcointalk.org/index.php?topic=27787.0` (Last Accessed 19/09/2017).

Note that the possibility of adversarial grinding in Ouroboros Praos is also present in previous work that derives randomness by hashing [Mic16,DPS16], as opposed to a dedicated coin-tossing protocol as in [KRDO17]. Following the examples of [Mic16,DPS16], we show that security can be guaranteed despite any adversarial bias resulting from grinding. In fact, we show how to use the q-bounded model of [GKL15] to derive a bound that shows how to increase the relevant security parameters given the hashing power that is available to the adversary.

Finally, in the present exposition we also put aside incentives; nevertheless, it is straightforward to adapt the mechanism of input endorsers from the protocol of [KRDO17] to our setting and its approximate Nash equilibrium analysis can be ported directly.

2 Preliminaries

We say a function $negl(x)$ is negligible if for every $c > 0$, there exists an $n > 0$ such that $negl(x) < 1/x^c$ for all $x \geq n$. The length of a string w is denoted by $|w|$; ε denotes the empty string. We let $v \,\|\, w$ denote concatenation of strings.

2.1 Transaction Ledger Properties

We adopt the same definitions for transaction ledger properties as [KRDO17]. A protocol Π implements a robust transaction ledger provided that the ledger that Π maintains is divided into "blocks" (assigned to time slots) 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 tx as *stable*, the remaining nodes, if queried, will either report tx in the same position in the ledger or will not report as stable any transaction in conflict to tx. 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), all nodes, if queried and responding honestly, will report the transaction as stable.

In [KP15,PSS17] 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 $k \in \mathbb{N}$. The chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest parties at the onset of the slots $sl_1 < sl_2$ are such that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$, where $\mathcal{C}_1^{\lceil k}$ denotes the chain obtained by removing the last k blocks from $\mathcal{C}_1$, and $\preceq$ denotes the prefix relation.

Chain Quality (CQ); with parameters $\mu \in (0, 1]$ **and** $k \in \mathbb{N}$. Consider any portion of length at least k of the chain possessed by an honest party at the onset of a round; the ratio of blocks originating from the adversary is at most $1 - \mu$. We call μ the chain quality coefficient.

Chain Growth (CG); with parameters $\tau \in (0, 1], s \in \mathbb{N}$. Consider the chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest parties at the onset of two slots sl_1, sl_2 with sl_2 at least s slots ahead of sl_1. Then it holds that $\text{len}(\mathcal{C}_2) - \text{len}(\mathcal{C}_1) \geq \tau \cdot s$. We call τ the speed coefficient.

2.2 The Semi-Synchronous Model

On a high level, we consider the security model of [KRDO17] with simple modifications to account for adversarially-controlled message delays and immediate adaptive corruption. Namely, we allow the adversary $\mathcal{A}$ to selectively delay any messages sent by honest parties for up to $\Delta \in \mathbb{N}$ slots; and corrupt parties without delay.