STEAK: A Proof of Stake Protocol as Randomness Oracle on the Cardano Blockchain

Namano Gyuuniku

April 16, 2024

Abstract

We present the STEAK protocol, a proof of stake (PoS) protocol designed to serve as a sustainable and efficient source of randomness for dApps on the Cardano blockchain. The protocol is based on the Ouroboros protocol (the same protocol powering Cardano itself) and is implemented on top of Cardano using smart contracts. It aims to address the limitations of existing randomness solutions based on proof of work (PoW), such as the Fortuna protocol. By utilizing PoS instead of PoW, STEAK inherits all the advantages associated with PoS, including greater energy efficiency, reduced environmental impact, and enhanced scalability. STEAK allows stakeholders to register their stake in the network by creating stake pools and rewards them for creating blocks with a small share of the stake token. DApps can rely on frequently produced blocks that contain non-manipulable and unpredictable hashes. The protocol ensures sustainability, randomness, and security through its design and governance mechanisms.

1 Introduction and Motivation

Randomness on blockchains is desired for various applications, such as games and lotteries. However, the Cardano Blockchain natively does not provide a source of randomness, on the contrary it has a strong focus on determinism and predictability.

Fortuna [6], launched in September 2023, is a useful source of randomness on the Cardano blockchain. However, Fortuna is a proof of work protocol, which has several downsides, among them unclear sustainability and high effort of block production. As of April 2024, block production on Fortuna is infrequent, with new blocks being generated only every few days [10], and the cost of mining exceeds its rewards by large. Moreover, the protocol is not designed to serve as a direct source of randomness, requiring additional infrastructure to be built on top of it [9]. Most crucially, the protocol lacks native upgradeability, requiring a hard-fork to change its parameters.

This is where STEAK comes in. We propose a proof of stake protocol based on Ouroboros [5] that is designed to serve as a source of randomness and is sustainable, as it does not require expensive proof of work mining. It features native upgradeability. Through a sustainable and slow distribution of rewards, the protocol is designed to operate indefinitely and reward participants for their contribution to the network.

1.1 Background

In this section, we provide some background on the Cardano block chain, a short history of randomness in smart contracts and the concept of proof stake we rely on.

Cardano and the EUTxO model The Cardano blockchain is a third-generation blockchain that aims to provide a secure, scalable, and sustainable platform for decentralized applications [1]. It distinguishes itself from other blockchains by its rigorous scientific approach to development, peer-reviewed research, and commitment to sustainability and interoperability.

Transactions on the Cardano blockchain are built on top of the extended UTXO (EUTxO) model [3], which is based on the Bitcoin UTxO model and offers additional features and capabilities. This model is designed to be deterministic and highly parallelizable, enabling efficient transaction processing and scalability. However, the model is incompatible with the widely adopted account-based model implemented on Ethereum, making the Cardano Smart Contract environment incompatible with existing smart contracts tailored for the Ethereum Virtual Machine (EVM).

Randomness on Cardano By design, the determinism of the chain makes it difficult to implement randomness to be used by Smart Contracts on the Cardano blockchain [4]. Usual approaches have to rely on off-chain sources of randomness through oracles or commit schemes [2]. Oracles are not suitable for i.e. lotteries, as they can be manipulated by the operator. Neither are commit schemes ideal, where an operator commits to a random string s by publishing its hash $h(s)$, revealing s after a certain deadline. The

problem is that the committing party knows s before the reveal and can take benefits from this knowledge.

Alternatively, verifiable randomness can be used. It relies on the hash of a future block at a specified slot as a source of randomness. This hash is unpredictable but verifiable, as it is determined by the block at the specified slot. However, the caveat of this approach is that blocks are not visible to on-chain smart contracts. Therefore, users that rely on the randomness have to trust the operator of the contract that the randomness was generated correctly. They can at most sanction abuse after misuse, which is evident through the shared agreement on the actual block hash in the real world. The approach misses high vulnerability use cases though where a single case of abuse can be sufficiently catastrophic to make its application unacceptable.

The Fortuna protocol [6] was proposed as a solution to this problem. It is a proof of work protocol that generates a random number (the block hash) as part of the block generation process. However, the protocol is not designed to serve as a persisting source of randomness. In other words, it is impossible to restore an older state of the chain based on its current status. This limits the ability to prove on-chain that a specific block is the first minted after a given time for example. Therefore it requires an additional derivative chain as data source for dApps building on top of Fortuna [9].

There has been another proposal for a randomised oracle on Cardano that relies only on a verified random function [8]. However, the technical documentation is meager and open-source implementations are not released until the completion of the project, making all statements about security, usability and fairness pure speculation. The documentation refers to a verifiable random function relying on a secret key owned by the lottery operator. Our conclusion from this is that the function does not achieve the high standard of unpredictability, because the owner of the secret key can predict the result of the oracle. For sufficiently delicate operations where the operator can not be trusted, such as a lottery, such an approach is unsuitable. Especially in the domain of pseudonomous blockchains, this allows the operator to act maliciously as an anonymous entity and gain benefit from its insider knowledge.

Proof of Stake & Ouroboros Proof of stake is an approach to consensus that is more sustainable than proof of work. Our implementation is based on Ouroboros [5], a proof of stake protocol that is implemented in Cardano. In this protocol, stakeholders register their stake in the network. Time is divided by slot length and each slot a single block can be appended to the block chain. Stakeholders are selected pseudo-randomly to create blocks in each slot based on the fraction of their stake on the overall stake in the network. They are rewarded for creating blocks with a small share of the stake token. This setup creates a natural incentive for stakeholders to act in the best interest of the network. Moreover, it is more sustainable than proof of work, as it does not require expensive mining hardware to decide the next block producer and preserves protection against Sybil attacks and censorship. Finally, stakeholders in the network are incentivized to hold and stake the token in order to participate in the network by the distributed rewards.

2 STEAK Protocol

The STEAK protocol is a proof of stake protocol that is designed to serve as a source of randomness on the Cardano blockchain. It is based on the Ouroboros protocol [5], which is a proof of stake protocol that is designed to be secure, scalable, and sustainable. The main feature is a chain of consecutive blocks, where each block is created by a stakeholder that is selected pseudo-randomly based on their stake in the network.

Important properties of the protocol are:

- **Sustainability:** The protocol is designed to be sustainable and operate indefinitely without requiring expensive proof of work mining (cf. Section 2.1).
- **Security:** The protocol is designed to be secure and resistant to attacks and manipulation. In particular, participants cannot manipulate a block hash to their advantage to influence either the randomness or the slot leader selection (cf. Sections 2.2 and 2.3).
- **Governance:** The protocol allows upgrades through built-in governance, giving immediate utility to the protocol token next to the staking rewards. Further, the token can be used to vote in a DAO that controls the protocol (cf Section 2.4).
- **Randomness:** The protocol is designed to serve as a source of randomness that can be used by smart contracts on the Cardano blockchain. In particular, no participant or observer can reliably predict the block hash of a given slot number in the future. Moreover dApps can rely on a block at a predetermined slot to be minted in time and contain a random number as a hash (cf. Section 2.5).

To participate in the protocol, stakeholders have to lock up their stake in a smart contract and register their pool with the protocol.

2.1 Stake Pools

Stakeholders can register their stake in the network by creating a stake pool. A stake pool is a smart contract that is registered with the protocol and contains the stake of the pool owner. The stake cannot be withdrawn unless the pool is deregistered from the protocol. To allow pooling of funds in stake pools, the protocol allows smart contracts as pool owners to register with the protocol. This allows multiple stakeholders to pool their funds together and participate in the protocol as a single entity.

The protocol checks during block creation that the pool owners signed the transaction with their private key. In case of smart contract owners, a corresponding withdrawal has to be present in the transaction, serving as an indicator of the smart contract "observing" the transaction.

To disincentivize pool spamming, i.e., prevent stakeholders from creating multiple pools to block other stakeholders from participating in the network, pool creation is subject to a fee that is proportional to the amount of total stake in the network. This

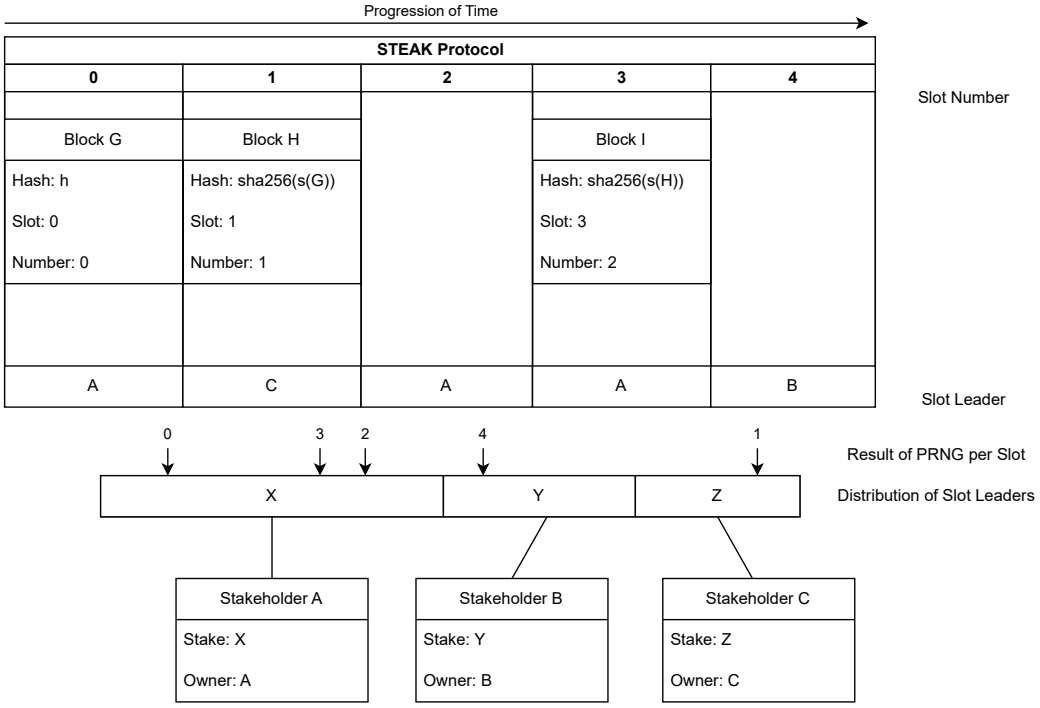

Figure 1: A sketch of the block creation process in the STEAK protocol. Each slot, a slot leader gets to create a block and submit it to the blockchain. The block hash is determined from the preceding block. The slot leader is determined based on the stake of each protocol participant. It may happen that slots remain unfilled such as slot 2 and 4 in this example.

fee is added back into the supply of stake tokens to be distributed as rewards for block creation.

2.2 Block Creation

Blocks are created by stakeholders that are selected pseudo-randomly based on their stake in the network. The selection process is based on the Ouroboros protocol, which is designed to be secure and unpredictable. Every slot, a slot leader is elected to create the next block. The slot leader is the only stakeholder that is allowed to create the block for that slot.

The protocol uses a verifiable random function (VRF) to select the slot leader. Specifically, it conducts a lottery among all registered stake pools, where the probability of winning the lottery is proportional to the stake of the pool. The random seed for the lottery for block b_t is the current hash of the preceding block b_{t-1} and the current slot number. The current hash of the preceding block b_{t-1} can only be influenced by the producer of block b_{t-2}. The slot number can not be influenced at all. This ensures that the selection process can not be influenced to increase ones own chances at getting

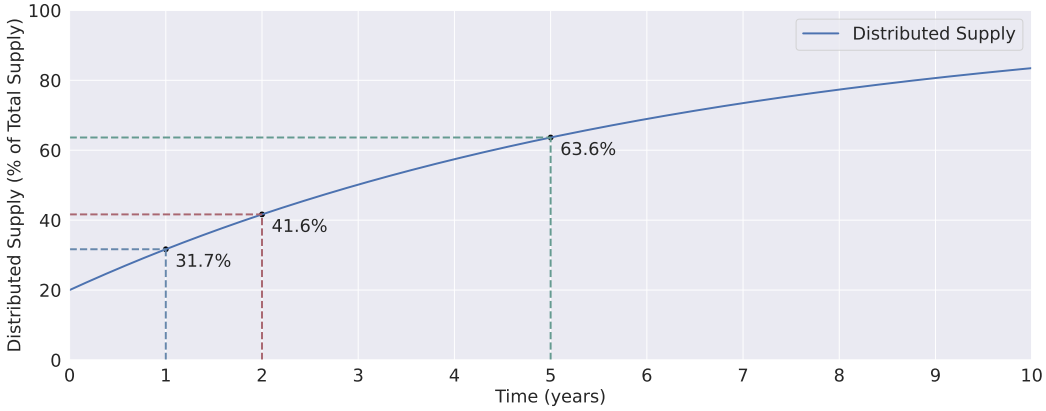

Figure 2: Token distribution over time starting at 20% initial token distribution.

elected for the next block, by moving dependencies further into the future. The process of slot leader election and block creation is sketched in Figure 1.

The number of blocks depends on the length of a slot, which is a parameter of the protocol. We propose a length of 60 seconds per slot, which results in a block roughly every minute. Note that forks can occur in the network but are handled by the underlying consensus protocol of Cardano. The protocol ensures that the preceding block is spent to create the new block, which allows only one possible chain to persist in each given fork of the Cardano protocol.

Such forks are possible, however, as a slot leader for $k > j > i$ can create a block for slot k as a successor to block at slot number i if it does not observe the block at slot number j. Again, this depends on the local view of the network and is resolved by the underlying consensus protocol of Cardano.

Note that, to increase randomness in the protocol, the protocol can elect several slot leaders for a single slot. By adding the id of the producing pool to the hash of the current block, it is harder to predict the next hash while it can also not be directly manipulated by the block producer. This strongly increases the uncertainty about block producers.

2.3 Reward Distribution

Stakeholders are rewarded for creating blocks with a small share of the stake token. The reward is distributed to the stake pool that created the block and is a fixed fraction of the remaining tokens to be distributed in the network. We propose a fixed fraction of $c = \frac{3}{10 \cdot 10^6} = 0.00003\%$ of the remaining tokens to be distributed with each block.

A plot of the expected amount of distributed tokens over time is shown in Figure 2, the mining rewards themselves can be found in Figure 3. Assuming an initial distribution of 20%, after two years, an additional 11.7% of the total supply is distributed, and after 5 years, roughly 63.6% of the total supply is in circulation. Note that this distribution schedule is very similar to the Bitcoin schedule, which halves the reward every 4 years. In an analytical interpolation, the Bitcoin reward schedule would have $c = \frac{3.3}{10 \cdot 10^6}$. In

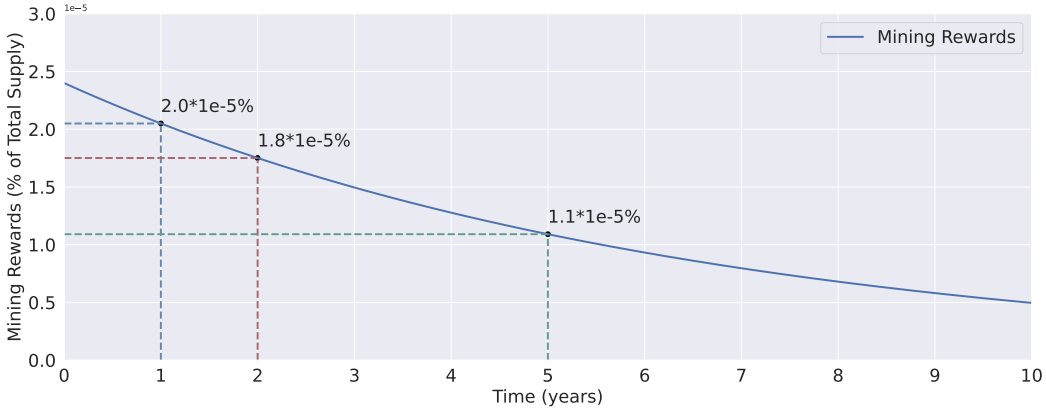

Figure 3: Mining rewards over time

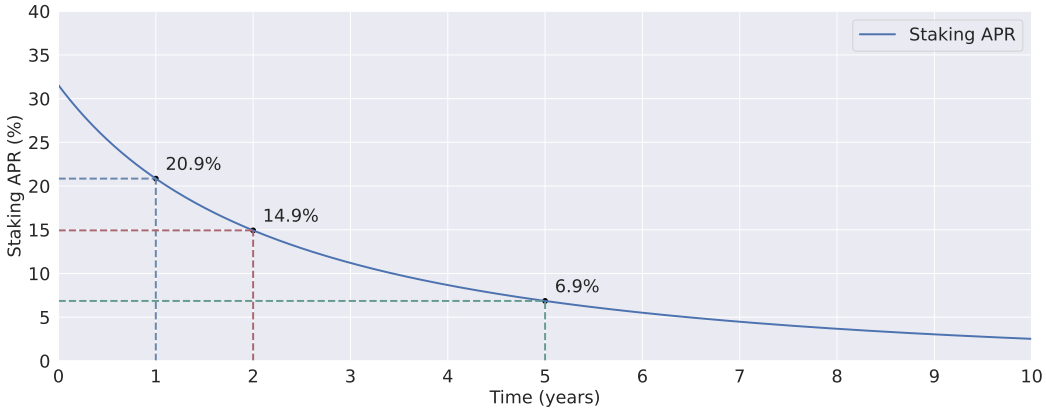

Figure 4: Staking APR (%) assuming that all yield is reinvested immediately.

the STEAK scheme, the rewards halve roughly every 4 years and 5 months. The reward scheme ensures a continuously high APR in the first years which slowly fades out to around 5% after 6 years, assuming that all rewards are reinvested, as shown in Figure 4.

2.4 Governance and Upgrades

The protocol is designed to be decentralized and governed by the stakeholders in the network. Stakeholders can vote on proposals to change the protocol parameters, such as changing the reward distribution fraction or the slot length. Moreover, it is possible to initiate a fork of the protocol by moving the stake to a new smart contract that implements a new iteration of the protocol.

In an initial version, the governance is based on a simplified agreement protocol. Each block B may contain auxiliary data d which is freely determined by the current block producer. We capture whether the datum d of B represents a request for an upgrade in the variable $1_{up}(B)$. We now allow an upgrade of the protocol if n consecutive blocks contain upgrade requests, i.e. $\{B_i\}_{i=k}^{l}, s.t. \forall i \in [k,l] : suc(B_i, B_{i+1}) \wedge 1_{up}(B_i) \wedge n = l - k$.

This approach has two important benefits:

1. **Liveness**: It can not happen that this protocol depends on the agreement of inactive staking pools (which would block the process), because only blocks produced by active participants are counted.

2. **Representation**: The protocol makes it sufficiently unlikely that a proposal will be accepted if less than a fraction of x of the active stake support the proposal when n is chosen adequately.

To compute n we will compute the probability of a proposal being accepted depending on x and n. We will assume that no pools register or deregister during the n blocks. The probability $P(1_{up}(B_i))$ of block number i containing an upgrade request is y, where y is the percentage of the total stake that is i) active and ii) approves the upgrade. Moreover, the probability of $P(\forall i \in [k,l] : 1_{up}(B_i)) = \prod 1_{up}(B_i)_k^l$ assuming independence between block production. Assume that $y < x$. Then $P(\forall i \in [k,l] : 1_{up}(B_i)) < x^n$. This allows to choose a suitable n so that proposals are only accepted with suitable agreement based on x. An illustration of how difficult it is for a proposal to be accepted with x approval for different n is presented in Table 1. As an additional safeguard, an admin key is designated that is additionally required to approve the upgrade. This grants the admin a veto right, but no option to unilaterally decide an upgrade.

Alternative Governance Models A generally more secure model would rely on a proof that more than x of the total stake is in favor of the upgrade. This could be shown by reconstructing the slot leader and its weight as part of the total supply. However, this implies either significantly growing the chain state or reconstructing the entire list of voting pools for each block during the proof, both of which are computationally expensive or even infeasible. Note also that this approach could rely on the agreement of inactive stake pools, which could block upgrades entirely.

x \ n		1	2	5	7	10	15	20
0.5	x^n	5.0e-01	2.5e-01	3.1e-02	7.8e-03	9.8e-04	3.1e-05	9.5e-07
	k slots	2	4	32	128	1024	32768	1048576
0.7	x^n	7.0e-01	4.9e-01	1.7e-01	8.2e-02	2.8e-02	4.7e-03	8.0e-04
	k slots	2	3	6	13	36	211	1254
0.9	x^n	9.0e-01	8.1e-01	5.9e-01	4.8e-01	3.5e-01	2.1e-01	1.2e-01
	k slots	2	2	2	3	3	5	9

Table 1: Values of x^n for $x \in \{0.5, 0.7, 0.9\}$ and $n \in [1, 20]$. The second row displays the expected number of slots k until an upgrade-approving sequence would occur.

Instead, the governance process will upgrade to using MuesliSwap Onchain Governance [7], or a variation thereof, where we count stake pools towards the weight of the vote. This allows the removal of the veto right entirely.

2.5 Usage as a source of Randomness

A derivative dApp that builds on top of the STEAK protocol can use the block hash of a future block as a source of randomness. To ensure that the block hash of the current block can not be predicted or manipulated, we force the miner to use a strongly deterministic value to create the block: the serialization of the entire preceding state and the current slot number. The addition of the slot number ensures that even based on the preceding block, it is impossible to determine the next hash, as it is unclear which next slot will be successfully minted. Further, if several producers are allowed for the same slot, uncertainty increases as the hash of the block takes into account which of these producers took the block. To further prevent that stake pools register or deregister in order to increase their own odds at becoming elected a slot leader, we recommend a fee for pool registration and deregistration that exceeds the staking rewards, thus making it financially uninteresting to manipulate the state. Thus, this hash is unpredictable but verifiable, as it is determined by the block at the specified slot. The dApp can use the hash as a seed for a random number generator to generate random numbers for various applications, such as games and lotteries.

An important feature of the STEAK protocol is that dApps can also trace back past block hashes. Imagine a dApp where users lock funds until a certain slot j, and the funds are distributed based on a random number generated from the first block hash generated after that time. Note that if we use slots rather than block numbers, we can exactly fixate a time until which funds are locked as opposed to relying on i.e. a block number which has unclear bounds on when it will be minted. For this dApp we need the ability to prove on-chain that a block is not only minted after j but also that it is the first block minted after j. However, at any time, only one block is referencable on-chain, the most recent block. It can happen that i) either no block is minted in the slot $j + 1$ because the elected slot leader did not submit a block or ii) the off-chain batcher has an

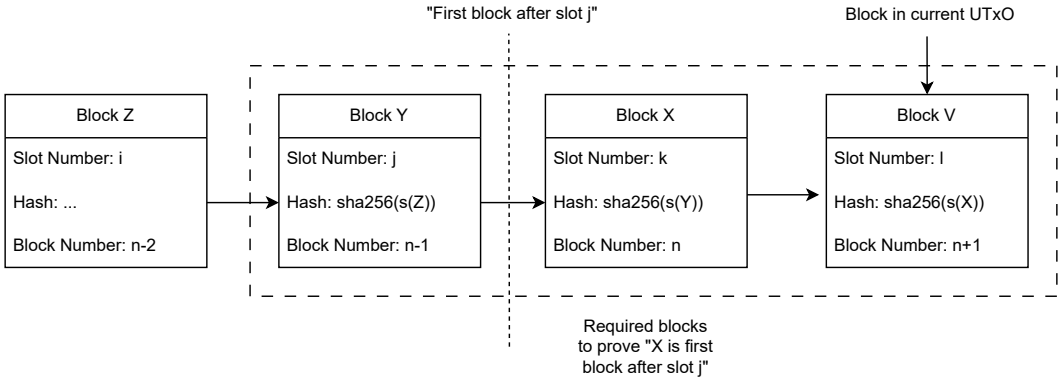

Figure 5: A graphical depiction of how to reconstruct the first block after cutoff slot j using the current state of the UTxO. This is necessary as dApp smart contracts can only see the current UTxO and none of the history.

incentive to wait until after $j + 1$ for a block hash that suits its own purposes best [9]. Therefore we need to reconstruct a part of the chain that is long enough to reach beyond $j + 1$, starting at j and ending at $i >= j + 1$, where i is the slot number of the latest block of the chain, i.e. the block of the chain that is currently in the UTxO. This block is the only block that can be passed to the dApp smart contract by reference because only blocks in the current UTxO can be used as reference inputs to smart contracts. By providing a reconstructed chain, the smart contract can verify that the provided block X is indeed the first minted after j.

In order to achieve the chain reconstruction, an off-chain batcher can provide the necesseary data to the dApp to reconstruct the chain from block V at slot i. In general, the hash of follow-up block B' of B is exactly the hash of B (i.e. $B'.hash = sha256(s(B))$) in the STEAK protocol. Therefore, by providing B, the smart contract is able to verify that B is indeed the predecessor to B' by computing its hash. This can be repeated indefinitely (up to the limits of computation given in the Cardano EUTxO model) to reconstruct chains of arbitrary length grounded in the currently most recent and visible block V of the staking protocol. A graphical depiction of the current UTxO and the data passed in by an off-chain batcher is depicted in Figure 5. Note that the simple governance protocol of Section 2.4 makes use of this history reconstruction and thus already provides a Proof of Concept implementation.

Note that it can in general not be proven on-chain that the provided predecessors of the current block exist in the chain, as past chain states are not visible on-chain. However, it is sufficiently difficult to provide spoofed values for the predecessors of the current block that it can be assumed that the predecessor was also part of the actual chain when its hash matches.

2.6 Bootstrapping

Bootstrapping of a proof of stake protocol is difficult [11]. A part of the staking tokens needs to be distributed to participants in advance to the blockchain launching, because participants need to already hold the token in order to participate in the protocol. We are aiming to initially distribute 20% of the total supply to the community.

3 Conclusion

The STEAK protocol provides a sustainable and efficient source of randomness on the Cardano blockchain. By utilizing a proof of stake consensus mechanism, STEAK addresses the limitations of existing randomness solutions, such as the Fortuna protocol, which relies on proof of work. The protocol incentivizes stakeholders to participate in the network by rewarding them for creating blocks and allows for decentralized governance and upgrades.

The protocol's design ensures sustainability, randomness, and security, making it a valuable addition to the Cardano ecosystem. As the demand for reliable sources of randomness grows, STEAK has the potential to become a widely adopted solution for various applications, such as games and lotteries.

Future work may include further analysis of the protocol's security properties, optimization of the reward distribution schedule, and the development of additional features and use cases. The results of this work may be integrated into the existing protocol using the built-in governance and upgrade mechanisms. The STEAK protocol represents a significant step towards a more sustainable and efficient blockchain ecosystem, and its implementation on the Cardano blockchain demonstrates the platform's commitment to innovation and progress.

References

[1] Sven Barac et al. "Cardano - What Is It and How to Start Working with It". In: ed. by Dragan Cisic et al. IEEE, 2023.

[2] Lars Brünjes. *Randomness in a smart contract.* https://cardano.stackexchange.com/a/393. 2022.

[3] Manuel M. T. Chakravarty et al. "The Extended UTXO Model". In: ed. by Matthew Bernhard et al. Springer, 2020.

[4] Micheal Peaton Jones and Roman Kireev. *Plutus Technical Report.* https://ci.iog.io/job/input-output-hk-plutus/master/x86_64-linux.packages.plutus-report/latest/download/1. 2024.

[5] Aggelos Kiayias et al. "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol". In: ed. by Jonathan Katz and Hovav Shacham. Springer, 2017.

[6] Cardano Miners. *Fortuna - Bitcoin style proof of work in smart contract form.* https://github.com/cardano-miners/fortuna/tree/main. 2023.

[7] MuesliSwapTeam. *MuesliSwap On-Chain Governance Platform*. https://github.com/MuesliSwapTeam/muesliswap-onchain-governance/blob/main/report/main.pdf. 2024.

[8] NuCast.io and Lovelace Club. *Technical Documentation: On-Chain Random Number Generation Library for Cardano Blockchain*. https://faint-albacore-c32.notion.site/Technical-Documentation-On-Chain-Random-Number-Generation-Library-for-Cardano-Blockchain-8b002ecb0d9d4122a1c876c284e6988c. 2024.

[9] Piefayth. *Perchance - Discussion*. https://discord.com/channels/946071061567529010/1146201677209292891/1206664321564803192. 2024.

[10] QCPOL Stake Pool. *Fortuna Explorer - Blocks*. https://fortuna-explorer.stakepool.quebec/blocks, archived at: https://archive.is/7EJmt.

[11] Suryanarayana Sankagiri and Pramod Viswanath. *Bootstrapping Blockchains*. https://courses.grainger.illinois.edu/ece598pv/sp2021/lectureslides2021/ECE_598_PV_course_notes17_v2.pdf. 2021.