Optimal RANDAO Manipulation in Ethereum

Kaya Alpturer
Princeton University, NJ, USA

S. Matthew Weinberg
Princeton University, NJ, USA

Abstract

It is well-known that RANDAO manipulation is possible in Ethereum if an adversary controls the proposers assigned to the last slots in an epoch. We provide a methodology to compute, for any fraction α of stake owned by an adversary, the maximum fraction $f(\alpha)$ of rounds that a strategic adversary can propose. We further implement our methodology and compute $f(\cdot)$ for all α. For example, we conclude that an optimal strategic participant with 5% of the stake can propose a 5.048% fraction of rounds, 10% of the stake can propose a 10.19% fraction of rounds, and 20% of the stake can propose a 20.68% fraction of rounds.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Information systems → Digital cash; Security and privacy → Distributed systems security

Keywords and phrases Proof of Stake, Consensus, Blockchain, Ethereum, Randomness manipulation

Digital Object Identifier 10.4230/LIPIcs.AFT.2024.10

Supplementary Material *Software*: `https://github.com/kalpturer/randao-manipulation` archived at `swh:1:dir:4632247a5ed3767ade8c71d514997c9f99d52387`

Funding Supported by NSF CAREER CCF-1942497 and a grant from the Ripple University Blockchain Research Initiative.

Acknowledgements We are grateful to Noah Citron for introducing us to the problem, and helpful discussions in early phases of this work. We are also grateful to Yunus Aydın, István Seres, Aadityan Ganesh, and anonymous reviewers for feedback on earlier drafts of this work.

1 Introduction

Randomness is an essential component of blockchain protocols. With the invention of Proof of Work blockchains [20], a major innovation in Bitcoin was to use the randomness of the SHA256 function to select the next block proposer. In particular, a participant in the Bitcoin ecosystem is able to propose a block of their choice with probability proportional to their computational power. While this system satisfies many desirable properties, it is in many ways not desirable due to inefficiency. With the move to Proof of Stake blockchain protocols, the dependence on computation is replaced with stake in the digital currency itself. However, a new source of randomness is needed to select the next block proposer with probability *proportional to one's stake*. A major security requirement for this randomness is for it to be verifiable (i.e. everyone can verify that the block proposer lottery was not rigged) and unpredictable (i.e. before the lottery happens, no one can know the winner).

Several approaches exist to provide this source of randomness to Proof of Stake blockchain protocols. One approach is to use an external randomness beacon [13] which achieves similar guarantees as in Bitcoin. However, implementing such a beacon comes with trust centralization concerns. A more practical approach is to use protocols that rely on pseudorandom cryptographic primitives to select block proposers, which are adopted by Proof of Stake blockchains such as Ethereum.

While the system safety and liveness are not compromised by the randomness mechanism in current Proof of Stake blockchains, it is well-known that they are susceptible to manipulation (see [23, 7]). In particular, incentive-incompatibilities that result in block-withholding behavior exist in many Proof of Stake blockchain protocols – analogous to selfish mining for Bitcoin [10, 26, 22].

There is a recent line of work focusing on Proof of Stake incentive incompatibilities [4, 13, 15, 3, 14], some of which derive concrete bounds on optimally manipulating randomness for the Algorand protocol. However, while some analyses such as [23, 9] and simulation-based approaches such as [2] conclude that randomness manipulation is likely to be negligible for Ethereum, it is currently unknown how much more an adversary that *optimally* manipulates Ethereum can make. In this paper, we focus on answering this question and compute optimal strategies for randomness manipulation in Ethereum. Our approach relies on modeling the randomness manipulation game as a Markov decision process.

1.1 Brief overview of Proof of Stake Ethereum

We now briefly cover the relevant details of the Proof of Stake Ethereum protocol. In the Ethereum protocol, time is divided into epochs, each epoch is divided into 32 slots, and each slot is 12 seconds. Each epoch is assigned 32 block proposers (one for each slot) who can construct and broadcast a block to be added to the blockchain at that slot. If the block proposer fails to do so, the slot is *missed* (no block is added) and the blockchain moves on to the next slot.

Proof of Stake Ethereum provides randomness by a scheme that maintains a random value called the RANDAO (also known as `randao_reveal`) in each block [9]. As each block is proposed, the previous RANDAO value is mixed using the private key of the proposer. Since the private key is used to sign the epoch number and is mixed into the previous RANDAO value by the xor operation, the mixing is *verifiable*. Moreover, as the signature is assumed to be uniformly random and the private key is unknown to the public, it is *unpredictable*. These properties ensure that the only actions available to an adversary in influencing the RANDAO value is to choose between *broadcasting* or *withholding* a block.

At the end of each epoch, the RANDAO value is used to select a set of 32 new proposers for the next epoch[1]. For example, if an adversary controls multiple validators and happens to get assigned to propose in slot 30 and 31 (the last two slots of an epoch), after slot 29 passes, the adversary can use the RANDAO value at slot 29 to compute 4 different RANDAO outcomes for the next epoch. If the adversary withholds both 30 and 31, the RANDAO value remains the same. If the adversary withholds 30 and broadcasts 31, the RANDAO value is only mixed with the signature of the proposer of 31, and so on. By precomputing 4 possible outcomes, the adversary is able to select one of the 4 RANDAO values (at the cost of missing the relevant block rewards). Similarly, in general, an adversary that controls the last k proposers in an epoch is able to choose from 2^k RANDAO values that determine the proposers for the next epoch. We call the longest contiguous adversarial slots at the end of an epoch the tail. With this scheme, it is conceivable that an adversary may strategically withhold their block at specific slots to win the right to produce *more* blocks in expectation.

[1] Ethereum actually skips an epoch in this process so the RANDAO value at the end of epoch i determines the 32 proposers for epoch $i + 2$ – we discuss this in more detail in Section 3

1.2 Main contributions

Our main technical contributions in this paper are:

- We model and formalize the game that an adversary with $\alpha < 1$ proportion of the total staked Ethereum plays in manipulating the RANDAO value.
- We show that the RANDAO manipulation game can be formulated as a Markov Decision Process, and we show how to significantly reduce the state space so that policy iteration on a laptop quickly converges.[2]
- We present precise answers to the fraction of slots a strategic player can propose after optimally manipulating Ethereum's RANDAO.

1.3 Related Work

Manipulating Ethereum's RANDAO. The name RANDAO comes from (and the scheme is inspired by) an earlier project [1], and its manipulability has been acknowledged and discussed in the Ethereum community [6, 5]. Work of [2] focuses on modeling the RANDAO mechanism[3] in probabilistic rewrite logic and evaluating greedy strategies (analogous to Tail-max of Section 5.1). In evaluating the model, they follow a simulation based approach with epoch length 10 and report some biasability. In [23], some probabilistic analysis of how many blocks an adversary controlling the last two proposers in the current epoch can get in the next epoch is presented. In addition, it is demonstrated that in specific instances, some staking pools had the opportunity to control more than half the next epoch. Lastly, [9] shows that the number of proposers an adversary controls at the tail is expected to shrink as long as the adversarial stake is less than roughly 1/2. It also provides an strategy analysis that considers tails of length 0 and 1, concluding marginal improvement over the honest. These results are consistent with ours, and as expected the reported improvement in rewards is less than the optimal strategy we compute. For example, a 25% adversary with their single look-ahead strategy makes 2.99% more than the honest while we compute that the optimal strategy makes 4.09% more than the honest. In comparison to these works, our work nails down the optimal RANDAO manipulation, and via a principled framework that can accommodate slight modifications (such as epoch length) as well.

Computing Optimal Manipulations. The most related methodological papers are [22, 14], who also compute optimal strategic manipulations. [22] computes optimal manipulations in Bitcoin's longest-chain protocol, and [14] computes optimal manipulations in Algorand's cryptographic self-selection. Our work is methodologically similar, as we also formulate an MDP and use some technical creativity to solve it. On the methodological front, our state-space reduction in Section 4 is perhaps most distinct from prior work.

Manipulating Consensus Protocols, generally. There is a significant and rapidly-growing body of work on manipulating consensus protocols broadly [10, 22, 19, 8, 17, 16, 13, 11, 25, 24, 3, 14]. Aside from the aforementioned works, most of these do not compute *optimal* manipulations, but instead understand when profitable manipulations exist. For Ethereum's RANDAO, it is already well-understood that profitable manipulations exist for arbitrarily small stakers, and so the key open problem is how profitable they are (which our work resolves).

[2] Our implementation is available here: `https://github.com/kalpturer/randao-manipulation`

[3] The RANDAO model in [2] is an earlier variation of the RANDAO mechanism that uses a commit-reveal scheme. The induced game, however, is quite similar.

1.4 Roadmap

In Section 2, we briefly cover the relevant background on Markov Decision Processes. In Section 3 we formalize the RANDAO manipulation process as a game. In Section 4, we formulate the RANDAO manipulation game as a Markov Decision Process and reduce the state space to a tractable regime. In Section 5 and 6 we describe how to evaluate and solve for optimal policies. Lastly, in Section 7 we conclude with a discussion on modeling assumptions, future work, and block miss rates.

2 Preliminaries: Markov Decision Processes

In this section, we quickly review the necessary background for Markov chains and Markov decision/reward processes. The material in this section is largely drawn from [18] and [21]. These texts can be consulted for a more detailed treatment.

A *Markov chain* $C = (S, P)$ consists of a set of states S and transition probabilities $P : S \times S \to \mathbb{R}$. Given a current state $s \in S$, we transition to the next state with the probability distribution induced by $P(s, \cdot)$. Rewards can be added to this framework with a reward function $R : S \times S \to \mathbb{R}$ such that if we transition from state s to s', we get $R(s, s')$ reward. A Markov chain with rewards is a *Markov reward process (MRP)*.

An agent navigating a Markovian system can be modelled using a *Markov decision process (MDP)*. An MDP is a tuple $M = (S, A, \{P_a\}_{a\in A}, \{R_a\}_{a\in A})$ where S is a set of states, A is a set of actions, $P_a : S \times S \to \mathbb{R}$ is a transition probability function representing the probability of individual transitions given an action $a \in A$, and $R_a : S \times S \to \mathbb{R}$ represents the reward of transitioning between individual states with action $a \in A$.

A *policy* $\pi : S \to A$ is a function specifying which actions to take given the current state. Once we fix a policy in an MDP, we get an MRP. We only consider deterministic policies since the standard results [21] show that in the models we consider, an optimal deterministic policy exists.

We will be modeling the RANDAO manipulation game as an MDP. We now introduce some definitions and properties that will be useful when we introduce our MDP.

2.1 Properties of Markovian systems

One important property concerns whether some states are visited infinitely often.

A state s in a Markov Chain is *recurrent* if, conditioned on currently being at state s, the probability of later returning to state s is 1. If a state is not recurrent, it is called *transient*. A *recurrent class* of states is a set of recurrent states $\hat{S}$ such that, for all $s \in \hat{S}$, conditioned on being at state s: for all $s' \in \hat{S}$, the probability of visiting s' at a later time is > 0.

A Markov chain is *ergodic* if it consists of a single recurrent class of states. Similarly an MDP is ergodic if for every deterministic policy,[4] the Markov chain induced by the policy is ergodic. All MDPs we will consider will be ergodic so for the rest of this section we assume ergodicity.

A *stationary distribution* $\boldsymbol{\sigma}$ of a transition probability matrix $\mathbf{P}$ in a Markov chain is defined to be a solution to the following:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}\mathbf{P} \qquad \text{and} \qquad \sum_i \sigma_i = 1$$

[4] We only work with stationary policies which are policies that do not change over time. For the processes we consider a stationary optimal policy is guaranteed to exist.

▶ **Proposition 1** ([21]). *If a Markov chain is ergodic, then there exists a unique stationary distribution.*

2.2 Reward criteria

Now we define the reward criteria for a Markov decision process M. For our application, *average reward* is more appropriate than *discounted reward* since we are concerned with the infinite behavior of the system.

Let $\rho_{\pi,s}(m)$ be a random variable that is equal to the reward at time m while transitioning the state at time m to the state at time $m+1$ in some MDP when running policy π starting at state s.

The *average reward* of a policy π is defined as:

$$\Gamma_\pi = \lim_{N\to\infty} \mathbb{E}\left[\frac{1}{N}\sum_{m=0}^{N} \rho_\pi(m)\right]$$

where we rely on the following result to ignore the initial state.

▶ **Proposition 2** ([21]). *The average reward for ergodic MDPs is initial state independent.*

Let $q(s)$ be the expected reward of transitioning from state s. More formally $q(s) = \sum_{s'\in S} R_{\pi(s)}(s,s')P_{\pi(s)}(s,s')$. Then, Γ_π can be computed using the stationary distribution $\boldsymbol{\sigma}^\pi$ of the Markov chain induced by fixing policy π.

$$\Gamma_\pi = \sum_{s\in S} q(s)\sigma_s^\pi$$

Value Functions. In any recurrent process, it is a useful concept to understand the "value" of being in one state over another, due to the potential future rewards. With non-discounted rewards, this requires some subtlety to properly define (because the expected future reward from any state is infinite). One standard method is to define the value of a state as its *average adjusted sum of rewards*:

$$v_\pi(s) = \lim_{N\to\infty} \mathbb{E}\left[\sum_{m=0}^{N} (\rho_{\pi,s}(m) - \Gamma_\pi).\right]$$

That is, the average adjusted sum of rewards captures the additive difference between an unbounded process starting from state s and iterating π and an unbounded process that earns Γ_π (the average per-round reward of π) per round.

▶ **Lemma 3** ([18]). *For an ergodic MDP M,*

$$v_\pi(s) + \Gamma_\pi = q(s) + \sum_{s'\in S} P_{\pi(s)}(s,s')v_\pi(s')$$

Since we can compute Γ_π first given a policy, this equation determines all v_π up to an additive constant which we can solve for after setting $v_\pi(s) = 0$ for some $s \in S$.[5]

[5] The average reward Γ_π is sometimes called *gain* and what we call the *value* v_π, which is the average adjusted sum of rewards, is sometimes called *bias* in the literature.

To find the optimal policy with respect to the average reward criterion, we can run the policy iteration algorithm of [18, 21]. Starting from an arbitrary policy π_0, evaluate the policy to compute Γ_{π_0} and v_{π_0}. Then, a policy improvement step is performed which defines $\pi_{i+1}(s) := \arg\max_{a \in A}\{q_a(s) + \sum_{s' \in S} P_a(s,s')v_{\pi_i}(s')\}$. The following Bellman optimality equation guarantees that once this process stabilizes, we have an average reward optimal policy.

▶ **Theorem 4** (Bellman optimality equation for average-reward MDPs, [18]). *If a policy π^* satisfies the following equations in an MDP, then π^* is average reward optimal.*

$$v_{\pi^*}(s) + \Gamma_{\pi^*} = \max_{a \in A}\left\{q_a(s) + \sum_{s' \in S} P_a(s,s')v_{\pi^*}(s')\right\} \qquad \forall s \in S$$

3 The RANDAO manipulation game

We now review RANDAO in more detail, and formulate the RANDAO manipulation game. RANDAO is a pseudorandom seed that updates every block, and is used to select Ethereum proposers. Below, we use the terminology $R(b)$ to denote the RANDAO value after the b^{th} slot has finished.

Updating RANDAO. The process for updating $R(b)$ is quite simple. If no one proposes a block during slot b of epoch x, then $R(b) = R(b-1)$. If a block is proposed during slot b, the proposer must also digitally sign the epoch number x and the hash of this digital signature is XORed with $R(b-1)$ to produce $R(b)$. Note, in particular, that there is a unique private key eligible to propose a block during slot b, and therefore the only two possibilities for $R(b)$ are either $R(b-1)$ (if no block is proposed) or $R(b-1)$ XOR hash(signature of x by proposer for slot b).

Using RANDAO to seed epochs. The Ethereum blockchain consists of epochs, where each epoch contains 32 blocks. Within each epoch t, a seed $S(t)$ determines which private keys are eligible to propose during each slot. That is, for each of the 32 slots in an epoch, the proposer of that slot is a deterministic function of $S(t)$ (but $S(t)$ is a pseudorandom number). Moreover, if $S(t)$ is a uniformly random number, then each slot proposer is independently and uniformly randomly drawn proportional to stake.

$S(t)$ is set based on RANDAO. Specifically, $S(t)$ is equal to the value of RANDAO at the end of epoch $t-2$. To be extra clear, there have been $32(t-2)$ slots completed by the end of epoch $t-2$, so $S(t) := R(32t-64)$.

Rewards. In practice, proposer rewards involve transaction fees, Maximal Extractable Value (MEV), and any payments made in the Proposer-Builder-Separation (PBS) ecosystem. To streamline analysis, and to be consistent with an overwhelming majority of prior work, we focus on the *fraction of slots* where an adversary proposes.[6] That is, we consider an adversary who aims to maximize the fraction of slots where they propose.

[6] Of course, it is an appropriate direction for future work to instead explicitly model transaction fees, MEV, etc. Such modeling would only make an adversary stronger, as their strategy can now depend on the value of each slot [8].

Ideal Cryptography. It is widely-believed that digital signatures of a previously-unsigned message using an unknown private key (and hashes of previously-unhashed inputs, etc.) are indistinguishable from uniformly-random numbers by computationally-bounded adversaries. However, such cryptographic primitives do not generate truly uniformly-random numbers. For the sake of tractability, and in a manner that is consistent with all prior work studying strategic manipulations of consensus protocols, we consider a mathematical model based on idealized cryptographic primitives (i.e. that hashing a previously-unhashed input produces a uniformly random number, independent of all prior computed hashes).

RANDAO Manipulation Game v1. We now formally define the RANDAO Manipulation Game (v1). After defining the game, we note its connection to Ethereum's RANDAO, and then proceed to simplify the game. Consistent with an overwhelming majority of prior work, we consider a single strategic manipulator optimizing against honest participants.[7]

▶ **Definition 5** (RANDAO Manipulation Game v1). *The RANDAO Manipulation Game proceeds in epochs* $1, 2, \ldots, n, \ldots$. *Each epoch has* $\ell := 32$ *slots. The strategic player has an* α *fraction of stake.*

- *Initialize* $\textsc{Reward}:=0$.
- *At all times, there is a RANDAO-generated list* $R := \langle R_1, \ldots, R_{32} \rangle \in \{S, H\}^\ell$. R *denotes the list of* ℓ *proposers based on the current value of RANDAO, and* R_i *denotes whether the strategic player (*S*) or an honest player (*H*) would propose in an epoch using the current value of RANDAO.*
- *Initialize* R *so that each coordinate of* R *is drawn iid, and equal to* S *with probability* α.
- *For each epoch* $n := 1, \ldots$
  - *Store* $R^n := R$ *and set the proposers for epoch* n *equal to* R^n.
  - *At all times during this epoch, for any set* B *of slots such that* $R^n_i = S$ *for all* $i \in B$, *Strategic Player can compute* $R|_B$, *which represents how the list of* 32 *proposers would update if Strategic Player were to propose a block in exactly slots* B *and no other blocks are proposed, given that the current RANDAO induces* R.
  - *For each slot* $i := 1, \ldots, \ell$:
    - *If* $R^n_i = H$:
      - *Update* R *to redraw each coordinate of* R *iid, and equal to* S *with probability* α.
      - *For all* B, *update* $R|_B$ *to redraw each coordinate of* R *iid, and equal to* S *with probability* α.
    - *If* $R^n_i = S$:
      - *Strategic Player chooses whether to propose or not.*
      - *If they choose to propose: (a) Add* $+1$ *to* $\textsc{Reward}$, *and (b) update* $R := R|_i$, *and* $R|_B := R|_{B \setminus \{i\}}$ *for all* B.
    - *Store* $\textsc{Reward}(n) := \textsc{Reward}$.

  Strategic Player's reward is $\liminf_{n \to \infty} \{\textsc{Reward}(n)/(\ell n)\}$.

Let us now overview the game above, highlight why it captures RANDAO manipulation on Ethereum, and where we've made stylizing assumptions.

- First, observe that the epoch's slot proposers are a deterministic function of RANDAO. We have skipped explicitly representing the RANDAO value, and focused only on the resulting proposers in R.

[7] An honest participant proposes a block during every round they are eligible.

- Next, observe that every time RANDAO changes *due to an Honest digital signature*, we've randomly redrawn each proposer i.i.d. and equal to S with probability α. This makes two stylizing assumptions.
  - First, we've assumed that uniformly RANDAO seed generates proposers iid proportional to stake. This may not be literally true, as Ethereum employs a more complicated sampling[8]. However, this stylizing assumption has negligible effect for the vast majority of real-world conditions[9].
  - Second, we've assumed that the hash of an honest digital signature is distributed uniformly at random from the perspective of Strategic Player. This assumes an Ideal hash function and Ideal digital signature (as consistent with prior work [12, 11, 14]), although in practice it only holds that the distribution is indistinguishable from uniform to a computationally-bounded adversary.
- In our game, the RANDAO value relevant for epoch t is whatever RANDAO is at the end of epoch $t-1$. In Ethereum proper, the RANDAO value relevant for epoch t is at the end of epoch $t-2$. However, we claim our modeling choice is *almost* wlog. Specifically, observe that there are essentially two RANDAO Manipulation Games being played: one on odd epochs, and one on even epochs. That is, the RANDAO value at the end of epoch $2t-1$ determines the proposers for epoch $2t+1$ for all t, just as in our RANDAO Manipulation Game. The only distinction to our game is that the RANDAO value at the start of epoch $2t+1$ is *not* equal to the RANDAO value at the end of epoch $2t-1$ (whereas in our RANDAO Manipulation Game, it is) – the RANDAO value can change during round $2t$. However, *as long as there is at least one Honest proposer during round* $2t+1$, the RANDAO value at the start of epoch $2t+1$ doesn't matter anyway, because it will be reset to uniformly random (at least, from the Strategic Player's perspective).
- To elaborate on the previous bullet, *as long as an Honest player proposes in at least one slot in every epoch*, our RANDAO Manipulation Game v1 correctly models all odd Ethereum epochs, and separately correctly models all even Ethereum epochs.
- Finally, observe that our Strategic Player receives a reward of one exactly when they propose a block, and their reward is indeed equal to the time-averaged fraction of rounds in which they propose.
- To summarize, our stylized game captures RANDAO manipulation in Ethereum with three exceptions: (a) it assumes Ideal cryptography for simplicity of analysis, (b) it assumes proposers in each epoch are drawn i.i.d. proportional to fixed stake, (c) it assumes that every epoch contains at least one Honest proposer. (a) is a natural assumption consistent with prior works, and essentially abstracts strategic manipulation away from breaking cryptography. The impact of (b) is negligible, as the distinction with Ethereum's shuffling and iteration based approach is negligible for an essentially uniform[10] set of over one million validators. (c) is also negligible, as the probability that a Strategic Player could ever induce the next epoch to be the first with no Honest proposers is at most $(2\alpha)^{32}$. [11]

---

[8] Roughly speaking, for each slot Ethereum shuffles the set of active validators and starts iterating over the shuffled list. A validator is selected to be the proposer for this slot with probability equal to its effective balance over 32.

[9] As long as most validators have effective balance equal to the maximum, Ethereum essentially selects the proposer using a uniformly random sample. Ethereum's real world conditions match this assumptions since the vast majority of validators have maximum effective balance.

[10] Here, by uniform, we are referring to the *effective balance* of validators. For Ethereum's current validator set, almost all have maximum effective balance which is 32 ETH.

[11] To see this, observe that Strategic Player has at most 2^{32} options to seed the subsequent epoch, and for each option the probability that it has no Honest proposers is α^{32}. The calculation follows by a union bound. Observe that even for $\alpha = 30\%$, this is 2^{-32}, meaning we would need to wait 2^{32} epochs, or over 150 years.

Manipulating RANDAO. To build intuition, we first give an example of how and why one might manipulate RANDAO. First, imagine that the Strategic Player proposes in slots 25 and 30 during epoch 1. Observe that RANDAO will get reset to uniformly random during epoch 31, and the RANDAO value between rounds 25 and 30 has no impact on any future proposers. Therefore, the Strategic Player gains nothing by skipping these proposal slots.

On the other hand, imagine that the Strategic Player proposes in slots 31 and 32. The Strategic Player knows the RANDAO value going into slot 32, and therefore has two options to set the RANDAO for epoch 2. If they choose not to propose in slot 32, they miss out on a one-slot reward, but perhaps this leaves the RANDAO in a favorable place for epoch 2 as compared to the RANDAO value if they were to propose. In fact, the Strategic Player knows the RANDAO value going into slot 31, and has four choices between {propose twice, propose zero times, propose only in 31, propose only in 32}. Each of these will seed a different set of proposers for epoch 2, and forego a different number of rewards.

This example helps establish that the Strategic Player never benefits from foregoing a proposal before an Honest slot, but has 2^k options for the next epoch's RANDAO when they propose the last k slots. We therefore call the largest number of slots k such that the adversary controls the last k slots of an epoch the *tail* of the epoch. The adversary can influence the next epoch only through these slots and intuitively these slots represent how much predictive power the adversary holds for the next epoch.

Refining the RANDAO Manipulation Game. Since the length of the tail fully captures the manipulation power of the Strategic Player, we further analyze this and refine our RANDAO Manipulation Game. We first observe that the length of the tail for a single RANDAO draw is distributed according to a roughly geometric distribution. A tail of length t occurs if we have t slots at the end of an epoch controlled by the strategic player which happens with probability α^t, preceded by a single honest slot which happens with probability $(1-\alpha)$. We call the remaining non-tail slots that the adversary controls the *count*. The count follows a binomial distribution conditioned on the length of the tail. Specifically:

- $geom'(\alpha)$ is the distribution of the tail given an adversary with stake α. It is defined such that for $T \sim geom'(\alpha)$,

$$\Pr(T=t) = \begin{cases} (1-\alpha)\alpha^t & 0 \le t < \ell \\ \alpha^\ell & t = \ell \end{cases}$$

- $Binom'(\ell - t - 1, \alpha)$ is the distribution of the remaining count (how many slots the adversary gets from the non-tail part of the epoch) given that the tail is t. For $C \sim Binom'(\ell - t - 1, \alpha)$,

$$\Pr(C=c) = \begin{cases} \binom{\ell-t-1}{c}\alpha^c(1-\alpha)^{\ell-t-1-c} & 0 \le c < \ell - t \wedge 0 \le t < \ell - 1 \\ 1 & c = 0 \wedge (t = \ell - 1 \vee t = \ell) \\ 0 & c \neq 0 \wedge (t = \ell - 1 \vee t = \ell) \end{cases}$$

- $\mathcal{F}$ is the distribution of (C, T) where we first sample $T \sim geom'(\alpha)$ and then sample $C \sim Binom'(\ell - T - 1, \alpha)$.

Given our reasoning above, an optimal Strategic Player will always propose during any of the "count" rounds, and will only manipulate the "tail" rounds. In particular, this means that the Strategic Player need not know the full slate of proposers in an epoch, but *only the count and the tail*. With this in mind, we can now refine our RANDAO Manipulation Game v1 to an equivalent RANDAO Manipulation Game v2.

Using the definitions above, formally, the RANDAO manipulation game G is:

▶ **Definition 6** (RANDAO Manipulation Game v2)**.**

1. *Initialize* $\textsc{Reward} := 0$*, and* $\textsc{Rounds} := 0$*.*
2. *Initialize* (c, t) *drawn from* $\mathcal{F}$*.*
3. *For* $i = 1$ *to* t*,*
   - *For* $j = 1$ *to* $\binom{t}{i}$*,*
     - *Sample* $(c_{i,j}, t_{i,j})$ *from* $\mathcal{F}$*.*
4. *The Strategic Player chooses an* (i^*, j^*) *pair.*
5. *Update* $t := t_{i^*,j^*}$*, add* $c_{i^*,j^*} + t_{i^*,j^*} - i^*$ *to* $\textsc{Reward}$ *and* ℓ *to* $\textsc{Rounds}$*.*
6. *Repeat from step 3.*

The final payoff is $\liminf\{\textsc{Reward}/\textsc{Rounds}\}$*.*

RANDAO Manipulation Game v2 is equivalent to RANDAO Manipulation Game v1, after assuming that the Strategic Player optimally proposes during all non-tail slots. Our method of counting the rewards observes that in the next epoch we will always propose during the "count" rounds (and hence just add them directly to our reward), and miss i tail proposals in order to influence the RANDAO (and hence get only $t_{i,j} - i$ slot rewards from the tail).

Before we proceed with the analysis, we define two sets of interest. Let $\mathcal{O}$ be the set of all possible values of $(C_{i,j} - i, T_{i,j})$ when $(C_{i,j}, T_{i,j})$ gets sampled from $\mathcal{F}$ for all $0 \leq i \leq t$, $1 \leq j \leq \binom{t}{i}$ for some current tail $t \in \{0, \ldots, \ell\}$. Given the range of the tail, count and the epoch length, $\mathcal{O} = \{(\omega, t) : t \in [0..\ell] \wedge \omega \in [-\ell..\ell] \wedge ((t = \ell \wedge \omega \leq 0) \vee (t < \ell \wedge \omega \leq \ell - t - 1))\}$. Let Ω be the set of all possible multisets of observations. More formally, Ω is the set of all multisets $\{(C_{i,j} - i, T_{i,j}) : 0 \leq i \leq \ell, 1 \leq j \leq \binom{t}{i}\}$ for some tail length t. Note that this makes all observation multisets $Obs \in \Omega$ have size equal to some power of 2.

4 MDP formulation

We can now directly formulate the RANDAO manipulation game as an MDP given the RANDAO Manipulation Game v2.

▶ **Definition 7** (RANDAO MDP M_G)**.** *The Markov decision process representing the RANDAO manipulation game is* $M_G = (S, A, \{P_\pi\}_\pi, \{R_\pi\}_\pi)$ *where*

- $S = \{(t, Obs) : t \in \mathbb{N}, 0 \leq t \leq \ell, Obs \in \Omega)\}$*. Each state represents the length of the tail* t *and the observations available to the adversary corresponding to the RANDAO samples.*
- *The action space* $A = \mathcal{O}$*, each action is selecting a future state from the given observations.*
- $\pi \in \Pi$ *is the policy space where* Π *is the set of all functions* $\pi : \Omega \to \mathcal{O}$ *such that* $\pi(Obs) = (\omega, t) \in Obs$*.* π *chooses on of the sample in* Obs *to transition towards.*
- P_π *and* R_π *are determined by the process in the game where given a state* (t, Obs)*, we transition to* (t', Obs') *such that* $\pi(Obs) = (\omega', t')$ *and* Obs' *consists of* 2^t *pairs* $(c_{i,j} - i, t_{i,j})$ *each sampled using* $\mathcal{F}$ *as in step 2 of the game. The reward of this transition is* $\omega' + t'$*.*

▶ **Lemma 8.** *M_G is ergodic.*

Proof. It suffices to observe for any policy π, we can transition from any state to any other state in three steps with non-zero probability. Consider the Markov chain induced by fixing π in MDP M_G. Suppose we are at state (t, Obs) and we consider (t', Obs'). Let $t^* = \log(|Obs'|)$. We then observe that the following sequence of transitions have non-zero probability:

$$(t, Obs) \to (t_1, Obs_1) \to (t^*, Obs_2) \to (t', Obs')$$

where Obs_1 is the set of observations where all have tail equal to t^* and Obs_2 is the set of observations where all have tail equal to t'. ◀

It is fairly straight-forward to see that this MDP formulation accurately captures the RANDAO Manipulation Game v2 – the state captures the point in time after drawing $(c_{i,j}, t_{i,j})$ for all (i,j), and our only action at this point is to choose one such (i,j) and transition, getting reward Reward. Note also that every state transition corresponds to exactly an increase of ℓ in Rounds, so the time-averaged reward in this MDP is exactly the payoff in RANDAO Manipulation Game v2.

Unfortunately, the state space of this MDP is enormous, and we have absolutely no hope of even writing it down (let alone solving it). Luckily since our MDP is ergodic, we can exploit the structure of the optimal policy and drastically simplify the state space.

Reducing the state space

We now refine our formulation of the RANDAO manipulation MDP to make it tractable. We know that for each policy π, there exists a valuation $v_\pi : \{0, \dots, \ell\} \to \mathbb{R}$, the *average adjusted sum of rewards*. We also know from the Bellman optimality equation that the optimal policy will take the action maximizing the immediate reward plus the weighted sum of potential future states with their values. Hence, any plausibly optimal policy, given the set of samples $\{(c_{i,j}, t_{i,j}) : 0 \le i \le t, 1 \le j \le \binom{t}{i}\}$, will simply choose the one that maximizes $c_{i,j} - i + t_{i,j} + v_\pi(t_{i,j})$. Motivated by this observation we reformulate the RANDAO MDP M_G as the following reduced state space MDP M'_G.

Below, intuitively we no longer need to explicitly store all (count, tail) options, because any optimal policy can be fully specified by assigning a value to the tail. So our new state space is simply the tail, but it is now more complex to iterate a transition.

▶ **Definition 9** (RANDAO MDP M'_G)**.** *The reduced Markov decision process representing the RANDAO manipulation game is* $M'_G = (S, \Pi, \{P_\pi\}_\pi, \{R_\pi\}_\pi)$ *where*

- $S = \{t \in \mathbb{N} : 0 \le t \le \ell\}$*. Each state represents the length of the tail.*
- Π *is the policy space where* Π *is the set of all total orders on* $\mathcal{O}$*.*
- *We treat the action space as the same as the policy space. In other words, we only consider constant strategies that pick a total order on* $\mathcal{O}$*.*
- P_π *and* R_π *are determined as follows. Follow RANDAO Manipulation Game v2 in Steps 3-4 (drawing several* (c,t)*s and choosing one), where in Step 4 we choose the future state that is earliest in the total order according to* π*. We then transition according to the selected* t *(this defines* P_π*) and accumulate reward according to* $c + t - i$ *(this defines* R_π*).*[12]

Intuitively, the key difference between M_G and M'_G is at which point in the process we pause and determine a state. In M_G, we pause after seeing a large set of (count, tail) pairs and declare this a state. We then make a very simple decision (pick a pair), a very simple reward update (plus count, plus tail, minus number of missed slots), and a fairly simple state transition (draw the new collection of pairs from a known distribution based on the chosen tail).

In M'_G, we instead pause after selecting the tail, and declare this a state. We then make a complex decision (pick a total ordering over all plausible pairs), a complex and randomized reward update (sample the set of pairs according to the known distribution based on the state, pick the highest in the total order, and take the reward), and a complex state transition (sample the set of pairs according to the known distribution based on the state, pick the

[12] In Section 5, we explicitly describe how these transition probabilities and rewards are computed.

highest in the total order, and take the reward). That is, M_G has a very complicated state space but simple transitions, whereas M'_G has a very complicated action space but simple states. Moreover, we use the Bellman optimality principle to narrow down the plausibly optimal actions for consideration in M'_G. We now proceed to establish their equivalence formally.

▶ **Lemma 10.** *M'_G is ergodic.*

Proof. It suffices to observe that under any policy $\pi \in \Pi$ the transition probability between any pair of states is positive. Suppose that we are running policy π and are at state $t \in S$ and we consider destination tail $t' \in S$. Now, if all 2^t sampled future states have tail t', the next state is t'. Since this happens with small but non-zero probability, ergodicity holds. ◀

While we are no longer explicitly keeping track of individual observations in the state, the optimal policy for M'_G still achieves reward equal to the optimal reward in M_G (in expectation).

▶ **Proposition 11.** *If $\pi \in \Pi$ is an optimal policy for M_G and $\pi' \in \Pi'$ is an optimal policy for M'_G, then $\Gamma_\pi(M_G) = \Gamma_{\pi'}(M'_G)$.*

Proof. Suppose $\pi \in \Pi$ is an optimal policy for M_G and $\pi' \in \Pi'$ is an optimal policy for M'_G. We first show that given $\pi \in \Pi$ for M_G, there exists a corresponding policy $\pi^* \in \Pi'$ for M'_G such that $\Gamma_\pi(M_G) = \Gamma_{\pi^*}(M'_G)$. Since every optimal policy attains the same expected average reward, without loss of generality, assume that π satisfies the Bellman optimality equation. As a consequence, π selects the observation (ω, t) that maximizes $\omega + t + v_\pi(t)$. This precisely defines a total order on (ω, t) as v_π is fixed. Let π^* be the total order defined by maximizing $\omega + t + v_\pi(t)$. As both M_G and M'_G sample from the same distributions the same number of times, $\Gamma_\pi(M_G) = \Gamma_{\pi^*}(M'_G)$.

Hence, we know that given π_0^*, there exists a corresponding policy $\pi_0^{*'}$ playing M'_G such that $\Gamma_{\pi_0^*}(M_G) \leq \Gamma_{\pi_0^{*'}}(M'_G)$. By the optimality of π_1^*, we also know that $\Gamma_{\pi_0^{*'}}(M'_G) \leq \Gamma_{\pi_1^*}(M'_G)$. Therefore, $\Gamma_{\pi_0^*}(M_G) = \Gamma_{\pi_1^*}(M'_G)$ and the claim holds. ◀

5 Evaluating policies

In this section, we first analyze the policy that only cares about maximizing the tail length (Tail-max) as an instructive example. Intuitively, this policy can be implemented by (a) for each subset of slots that the Strategic Player can choose to propose, computing the resulting RANDAO value and hence the next epoch proposer assignments, and (b) the Strategic Player choosing to propose in the subset of slots that results in the longest tail in the next epoch. Subsequently, we describe how to evaluate arbitrary policies in our Markov decision process M'_G formulation of the RANDAO manipulation game.

5.1 Analyzing the Tail-max policy

The Tail-max policy can be defined as the policy π that given the current state t and 2^t samples $(C_{i,j}, T_{i,j}) \sim \mathcal{F}$, picks (i, j) that maximizes $T_{i,j}$. Note that in case of ties, we pick the transition with higher reward (i.e. breaking ties in favor of higher $C_{i,j} - i$).

To analyze Tail-max, we are interested in computing the following quantities.

- $P_{tail}(t, t')$: the probability of transitioning from state t to t' when running game G with Tail-max.
- $R_{tail}(t)$: the expected reward of transitioning from state t when running game G with Tail-max.

Let $maxTail(t)$ be a random variable representing the maximum sampled tail in the current round of the game. More formally, it is defined as the following:

$$maxTail(t) := \max_{\substack{(C_{i,j},T_{i,j})\sim\mathcal{F} \\ 0\le i\le t \\ 1\le j\le\binom{t}{i}}} \{T_{i,j}\}$$

which is identically distributed as $\max_{\substack{T_i\sim geom'(\alpha) \\ 0\le i\le 2^t-1}}\{T_i\}$. Then, we have the following probabilities.

$$\begin{aligned} P_{tail}(t,t') &= \Pr(maxTail(t)=t') \\ &= \begin{cases} \Pr(maxTail(t)\le t') - \Pr(maxTail(t)\le t'-1) & 0<t'\le\ell \\ \Pr(maxTail(t)\le t') & t'=0 \end{cases} \end{aligned}$$

where we use the following:

▶ **Lemma 12.** $\Pr(maxTail(t)\le T') = \begin{cases} \left(1-\alpha^{t'+1}\right)^{2^t} & 0\le t'<\ell \\ 1 & t'=\ell \end{cases}$

Proof. Using the fact that each $\{T_i\}$ are i.i.d.,

$$\begin{aligned} \Pr(maxTail(t)\le T') &= \Pr_{\substack{T_i\sim geom'(\alpha) \\ 0\le i<2^t}}\left(\bigwedge_{0\le i<2^t}(T_i\le t')\right) \\ &= \prod_{0\le i<2^t}\Pr_{T_i\sim geom'(\alpha)}(T_i\le t') \\ &= \left(\Pr_{T''\sim geom'(\alpha)}(T''\le t')\right)^{2^t} \\ &= \begin{cases} \left(1-\alpha^{t'+1}\right)^{2^t} & 0\le t'<\ell \\ 1 & t'=\ell \end{cases} \end{aligned}$$

◀

The transition reward can be computed in a similar way. Let $\preceq\in\Pi$ of M'_G be defined as $(t,v)\preceq(t',v')$ if and only if $t<t'$ or $(t=t'\wedge v\le v')$. Intuitively, these pairs represent choices that a policy can make where t is the tail and v is the amount of reward we get from the rest of the count. The Tail-max policy picks the maximum such pair according to $\preceq$. Equality and strict ordering are defined in the usual way. Also let $prev_{\preceq}(v,t)$ be defined as the previous pair in the ordering $\preceq$ if it exists and $\perp$ otherwise. Note that $\Pr(\cdot\preceq\perp)$ is interpreted as 0.

Let $maxPair_{\preceq}(t)$ be a random variable representing the maximum tail, and the non-tail reward pair in the current round of the game according to the total order $\preceq$. More formally, it is defined as the following:

$$maxPair_{\preceq}(t) := \operatorname*{\overset{\preceq}{max}}_{\substack{(C_{i,j},T_{i,j})\sim\mathcal{F} \\ 0\le i\le t \\ 1\le j\le\binom{t}{i}}} \{(T_{i,j},C_{i,j}-i)\}$$

Note that $\preceq$ is a total order and we have the property that $maxPair_{\preceq}(t) \preceq (a,b)$ if and only if $(T_{i,j}, C_{i,j} - i) \preceq (a,b)$ for all $0 \le i \le t, 0 \le j \le \binom{t}{i}$. Therefore,

$$\begin{aligned} R_{tail}(t) &= \mathbb{E}\left[T' + V' \mid maxPair_{\preceq}(t) = (T', V')\right] \\ &= \sum_{\substack{0 \le t' \le \ell \\ -t \le v' \le \ell - t'}} \Pr(maxPair_{\preceq}(t) = (t', v'))(t' + v') \\ &= \sum_{\substack{0 \le t' \le \ell \\ -t \le v' \le \ell - t'}} (\Pr(maxPair_{\preceq}(t) \preceq (t', v)) - \Pr(maxPair_{\preceq}(t) \preceq prev_{\preceq}(t', v)))(t' + v) \end{aligned}$$

where we can use the following:

▶ **Lemma 13.** *Let $\preceq \in \Pi$ of M'_G be the* Tail-max *policy. For $f(x) := \Pr_{T \sim geom'(\alpha)}(T < x)$, $g(x) := \Pr_{T \sim geom'(\alpha)}(T = x)$, and $h(x,y) := \Pr_{C \sim Binom(\ell - x - 1, \alpha)}(C \le y)$,*

$$\Pr(maxPair_{\preceq}(t) \preceq (t', v)) = \prod_{0 \le i \le t} \left(\Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \right)^{\binom{t}{i}}$$

and

$$\Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) = \begin{cases} f(t') & t' = \ell \wedge 0 > v' + i \\ f(t') + g(\ell) & t' = \ell \wedge 0 \le v' + i \\ f(t') + g(t')h(t', v' + i) & \textit{otherwise} \end{cases}$$

Proof. Using standard properties, we observe that

$$\begin{aligned} \Pr(maxPair_{\preceq}(t) \preceq (t', v)) &= \Pr_{\substack{(C_{i,j}, T_{i,j}) \sim \mathcal{F} \\ 0 \le i \le t \\ 1 \le j \le \binom{t}{i}}} \left(\bigwedge_{\substack{0 \le i \le t \\ 1 \le j \le \binom{t}{i}}} ((T_{i,j}, C_{i,j} - i) \preceq (t', v)) \right) \\ &= \prod_{\substack{0 \le i \le t \\ 0 \le j \le \binom{t}{i}}} \Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \\ &= \prod_{0 \le i \le t} \left(\Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \right)^{\binom{t}{i}} \end{aligned}$$

and

$$\begin{aligned} &\Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \\ &\quad = \Pr_{(C,T) \sim \mathcal{F}}(T < t' \vee (T = t' \wedge C - i \le v')) \\ &\quad = \Pr_{T \sim geom'(\alpha)}(T < t') + \Pr_{(C,T) \sim \mathcal{F}}(T = t' \wedge C \le v' + i) \\ &\quad = \Pr_{T \sim geom'(\alpha)}(T < t') + \Pr_{T \sim geom'(\alpha)}(T = t') \Pr_{(C,T) \sim \mathcal{F}}(C \le v' + i \mid T = t') \\ &\quad = \Pr_{T \sim geom'(\alpha)}(T < t') \\ &\qquad + \begin{cases} 0 & t' = \ell \wedge 0 > v' + i \\ \Pr_{T \sim geom'(\alpha)}(T = \ell) & t' = \ell \wedge 0 \le v' + i \\ \Pr_{T \sim geom'(\alpha)}(T = t') \Pr_{C \sim \mathrm{Binom}(\ell - t' - 1, \alpha)}(C \le v' + i) & \text{otherwise} \end{cases} \end{aligned}$$

◀

We can now compute the stationary distribution in order to directly compute average reward of the Tail-max policy. This is a lower bound to the optimal reward ratio we can obtain from this game.[13]

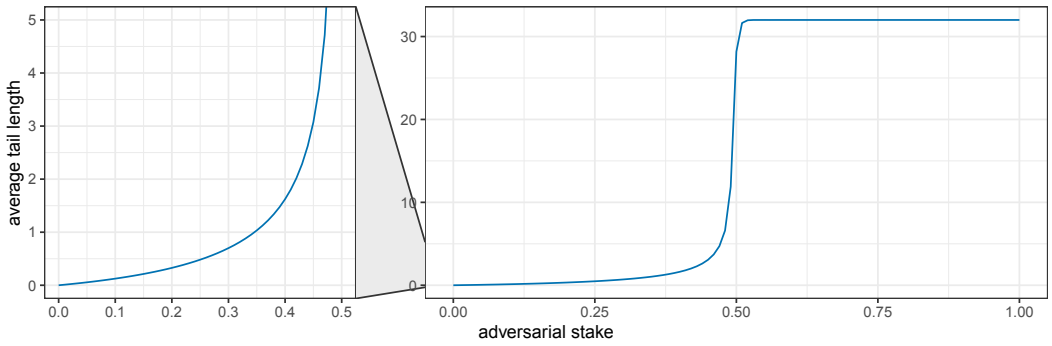

Figure 1 Average tail length attained for each α when running Tail-max. The adversary controls a larger tail value as α rises as expected. There is a quick jump when approaching $\alpha = 50\%$, indicating that the adversary can propose almost all blocks.

5.2 Policy evaluation in the general case

We proceed similar to the Tail-max analysis. Consider policy $\preceq \in \Pi$ so it is some total order on $\mathcal{O}$. Recall that $maxPair_{\preceq}(t)$ is a random variable defined (in Subsection 5.1) to be the maximum (tail, non-tail reward) pair given that the adversary currently controls a tail of length t. Similar to the analysis of Tail-max, we then have

$$P_{\preceq}(t,t') = \sum_{-\ell \le v \le \ell - t'} \Pr(maxPair_{\preceq}(t) = (t', v))$$

$$R_{\preceq}(t) = \sum_{\substack{0 \le t' \le \ell \\ -t \le v' \le \ell - t'}} \Pr(maxPair_{\preceq}(t) = (t', v))(t' + v)$$

Now, it suffices to describe how to compute the CDF of $maxPair_{\preceq}(t)$ since

$$\Pr(maxPair_{\preceq}(t) = (t', v)) = \Pr(maxPair_{\preceq}(t) \preceq (t', v)) - \Pr(maxPair_{\preceq}(t) \preceq prev_{\preceq}(t', v))$$

▶ **Lemma 14.** *Let $\preceq \in \Pi$ be an arbitrary policy in M'_G.*

$$\Pr(maxPair_{\preceq}(t) \preceq (v, t')) = \prod_{0 \le i \le t} \left(\sum_{\forall (t^*, v^*) \preceq (t', v)} \Pr_{(C,T) \sim \mathcal{F}}((T, C) = (t^*, v^* + i)) \right)^{\binom{t'}{i}}$$

[13] Note that Tail-max does not necessarily outperform Honest – it could be that in an attempt to increase the tail by one, Tail-max misses several proposal slots, and yet also does not take good advantage of the increased tail. However, our results show that Tail-max does outperform the honest strategy for all α. See Figure 2 and 3 for a comparison with Honest and the optimal policy. The average tail length of Tail-max, however, serves as an upper bound on the average tail length of any feasible strategy, including the optimum.

Proof. Using standard properties of independent samples, we observe that:

$$\begin{aligned}
\Pr(maxPair_{\preceq}(t) \preceq (v, t')) &= \Pr_{\substack{(C_{i,j},T_{i,j})\sim\mathcal{F} \\ 0\le i\le t \\ 1\le j\le \binom{t}{i}}} \left(\bigwedge_{\substack{0\le i\le t \\ 1\le j\le \binom{t}{i}}} ((T_{i,j}, C_{i,j} - i) \preceq (t', v)) \right) \\
&= \prod_{\substack{0\le i\le t \\ 1\le j\le \binom{t}{i}}} \Pr_{(C,T)\sim\mathcal{F}} ((T, C - i) \preceq (t', v)) \\
&= \prod_{\substack{0\le i\le t \\ 1\le j\le \binom{t}{i}}} \sum_{\forall (t^*,v^*)\preceq(t',v)} \Pr_{(C,T)\sim\mathcal{F}} ((T, C - i) = (t^*, v^*)) \\
&= \prod_{0\le i\le t} \left(\sum_{\forall (t^*,v^*)\preceq(t',v)} \Pr_{(C,T)\sim\mathcal{F}} ((T, C) = (t^*, v^* + i)) \right)^{\binom{t'}{i}}
\end{aligned}$$

◀

Therefore we can also evaluate arbitrary policies by computing the quantities above.

6 Solving for optimal strategies

We can now run policy iteration [18] in our policy space.

1. Start with an arbitrary policy $\preceq \in \Pi$.
2. Compute $P_{\preceq}$ and $R_{\preceq}$.
3. Compute average reward $\Gamma_{\preceq}$ using $R_{\preceq}(t)$ and the stationary distribution of $P_{\preceq}$.
4. Determine v_t for each $t \in \{0, \dots, \ell\}$ by setting $v_0 = 0$ and solving the system of linear equations given by

$$\Gamma_{\preceq} + v_{\preceq}(t) = R_{\preceq}(t) + \sum_{t'=0}^{\ell} P_{\preceq}(t, t') v_{t'} \qquad \text{for } t \in \{0, \dots, \ell\}.$$

5. Let $\preceq'$ be a new total order constructed by sorting each (ω, t) pair by the quantities $\omega + t + v_t$ in ascending order.
6. If $\preceq$ is equal to $\preceq'$, we converged and $\preceq$ is optimal. Otherwise let $\preceq := \preceq'$ and repeat from step 2.

Note that this is equivalent to policy iteration as described in Section 2 since sorting by immediate reward plus future state value to get new policy $\preceq$ is equivalent to picking the maximum at each state.

For $\ell = 32$ and all α, policy iteration always converged in less than 10 steps. This enables us to plot the following results on optimal manipulations for RANDAO:

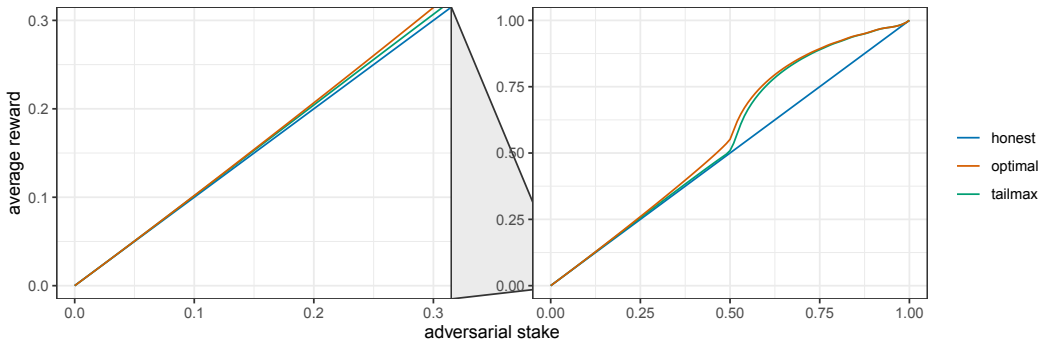

Figure 2 Average reward of the optimal policy and TAIL-MAX for $\ell = 32$. The figure on the left shows the $0 < \alpha \leq 0.3$ range, the figure on the right show the entire range of $0 < \alpha < 1$.

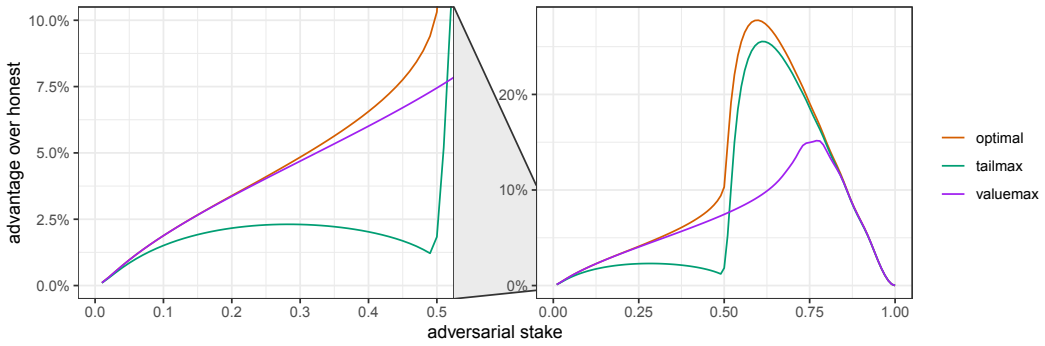

Figure 3 Percentage improvement of the optimal policy and TAIL-MAX over the honest policy for $\ell = 32$. Improvement is defined as (policy average reward)/(honest average reward) $- 1$. We also analyze the strategy VALUE-MAX here which we define as the strategy that maximizes the reward in the next epoch (chooses the pair that maximizes $c_{i,j} + t_{i,j} - i$).

Table 1 Average reward of the optimal policy. In expectation, the honest reward is equal to α. We see in the table that the optimal policy is strictly more profitable.

α	optimal reward
1%	1.00107%
5%	5.04834%
10%	10.18807%
15%	15.39960%
20%	20.67770%
25%	26.02472%
30%	31.45164%
35%	36.97348%
40%	42.62435%
45%	48.49184%

A key strength of our methodology is the fact that it is constructive, we explicitly construct the optimal policy for each α. In order to implement the optimal policy we compute, the values can be used as in step 5 of the policy iteration routine described above. For instance, for $\alpha = 0.2$, the optimal policy assigns the the following values to tails of length 0 to 32 (rounded to two decimal points):

(0.00, 0.90, 1.66, 2.30, 2.86, 3.35, 3.79, 4.19, 4.55, 4.89, 5.21,
5.50, 5.78, 6.05, 6.30, 6.55, 6.78, 7.00, 7.22, 7.43, 7.63, 7.82,
8.01, 8.19, 8.37, 8.54, 8.71, 8.87, 9.03, 9.19, 9.34, 9.49, 9.64)

Considering $\ell \neq 32$

One benefit of our approach is that it trivially extends to any ℓ. This allows one to easily answer, for example, whether RANDAO would be more or less manipulable with different epoch lengths and by how much.

Policy iteration runs smoothly for $\ell = 32$ and smaller. However, when we consider $\ell = 64$ or 128, numerical instability becomes a concern, and our experiments are no longer *provably* accurate. In particular, 64-bit floats we use in our machine introduce precision error that explode when evaluating the expression in Lemma 14 with $\ell > 32$.

In order to improve the numerical stability of the expression in Lemma 14, when $\ell > 32$ we evaluate the inner sum directly to 1 instead of taking the exponential when the sum reaches within 10^{-14} of 1 since the error ϵ introduced by the floating point representation cause $(1 \pm \epsilon)^N$ to become 0 or a large constant for $N \gg 1$. The following figure shows running our evaluation for different ℓ, using this modification.

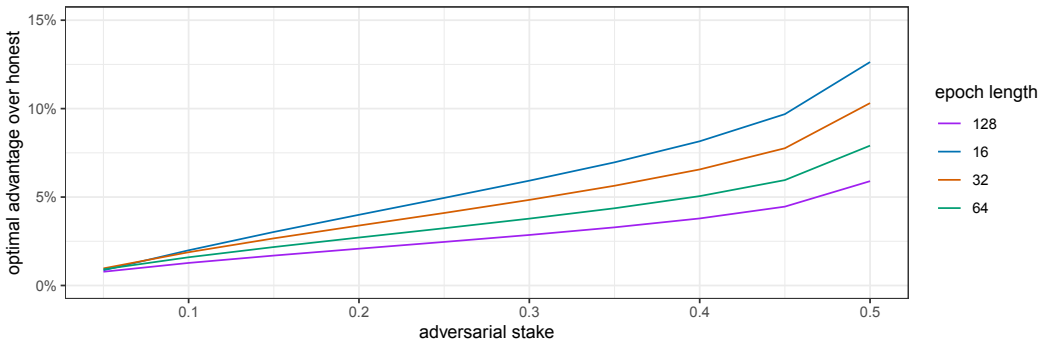

Figure 4 Percentage improvement over honest for $\ell \in \{16, 32, 64, 128\}$. Improvement is defined as (optimal average reward)/(honest average reward) $-$ 1.

We conjecture that this plot is representative of how the results scale with ℓ, although unlike our main results the experiments are not provably accurate due to the aforementioned numerical instability. If one desires provable numerical guarantees on these quantities, one would need an analysis of numerical error induced by floating point representations of the machines that run the evaluation.

7 Discussion

We model optimal RANDAO manipulation in Proof-of-Stake Ethereum and frame it as an MDP. Our main modeling contribution is getting from RANDAO manipulation in practice to RANDAO Manipulation Game v2, and our key technical insight is getting from there to the reduced RANDAO MDP M'_G. From here, simple policy iteration on a laptop suffices to analyze the optimal strategy. Our main results shed light on exactly how manipulable Ethereum's RANDAO is. One could compare our results, for example, to those of [14] for Proof-of-Stake protocols based on cryptographic self-selection. For example, [14] establishes that a well-connected Strategic Player with 10% of the stake can propose between 10.08% an 10.15% of the rounds in cryptographic self-selection protocols, and our work establishes that a Strategic Player with 10% of the stake can propose a 10.19% fraction of rounds in Ethereum Proof-of-Stake. While our work introduces methodology to compute these numbers, we leave *interpretation* of their significance to the Ethereum community since many different factors come into play when designing the consensus mechanism.

A clear direction for future work would be to consider the impact of slot-varying rewards as in [8]. This will clearly increase the manipulability (as now the Strategic Player can use the value of a slot when deciding whether to propose), but it is not obvious by how much. A second direction would be to consider the impact of idiosyncratic details such as Ethereum's sync committees (extra rewards every 256 blocks).

Lastly, we briefly discuss the empirical signature of randomness manipulation. The results immediately lead to the following question: are there any entities currently manipulating the RANDAO value? The signature of such an attack would affect the block miss rates especially around the tail. Some prior analyses suggest that while there has been ample opportunities that would result in short term gains for certain entities, none have been observed to manipulate RANDAO [23]. In the figure below, the block miss rates by epoch slot index is displayed from epoch 146876 to 272341.

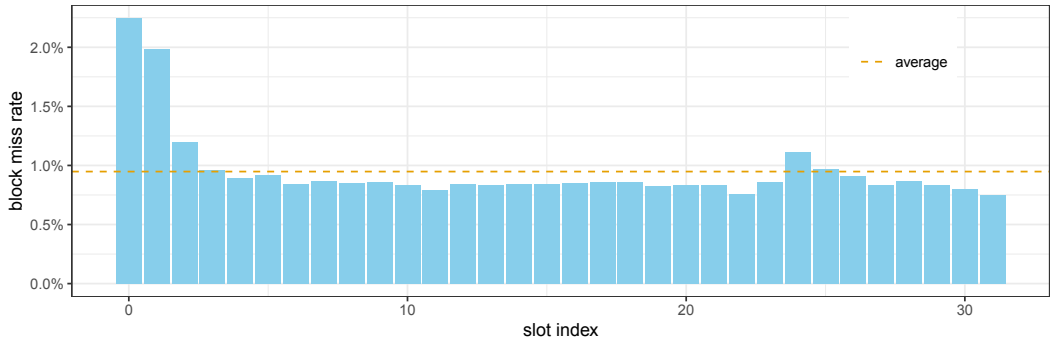

Figure 5 Block miss rate by slot index from epoch 146876 to 272341. The average block miss rate is 0.9482%.

Our interpretation of this data is that slot index 0, 1, and 2 is missed frequently since validators have less time to react to their proposer assignments and slot index 24 and 25 is missed with slightly higher frequency than the baseline due to votes crossing the 2/3 majority. We do not observe any significant elevation in block miss rates around the tail of the epoch. It is also an interesting direction for future work to examine whether *undetectable* profitable strategies exist for RANDAO manipulation (i.e. strategies that strictly outperform Honest, but produce the same miss rate for all slots).

References

1 Randao: A dao working as rng of ethereum, March 2019. URL: https://github.com/randao/randao/.

2 Musab A Alturki and Grigore Roşu. Statistical model checking of randao's resilience to pre-computed reveal strategies. In *Formal Methods. FM 2019 International Workshops*, pages 337–349, Porto, Portugal, 2020. Springer.

3 Maryam Bahrani and S. Matthew Weinberg. Undetectable selfish mining. In *EC '24: The 25th ACM Conference on Economics and Computation*. ACM, 2024.

4 Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S. Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, pages 459–473, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3328526.3329567.

5 Vitalik Buterin. Randao beacon exploitability analysis, round 2, May 2018. URL: https://ethresear.ch/t/randao-beacon-exploitability-analysis-round-2/1980.

6 Vitalik Buterin. Rng exploitability analysis assuming pure randao-based main chain, April 2018. URL: https://ethresear.ch/t/rng-exploitability-analysis-assuming-pure-randao-based-main-chain/1825/1.

7 Vitalik Buterin. Vitalik's annotated ethereum 2.0 spec, 2020. URL: https://notes.ethereum.org/@vbuterin/SkeyEI3xv#Time-parameters.

8 Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 154–167, 2016. doi:10.1145/2976749.2978408.

9 Ben Edgington. Upgrading ethereum, 2023. Capella edition. URL: https://eth2book.info/.

10 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

11 Matheus V. X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 89–114. ACM, 2022. doi:10.1145/3490486.3538337.

12 Matheus V. X. Ferreira and S. Matthew Weinberg. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, pages 683–712. ACM, 2020. doi:10.1145/3391403.3399495.

13 Matheus V. X. Ferreira and S. Matthew Weinberg. Proof-of-stake mining games with perfect randomness. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, EC '21, pages 433–453, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465456.3467636.

14 Matheus V.X. Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Hu, S. Matthew Weinberg, and Catherine Yu. Computing optimal manipulations in cryptographic self-selection proof-of-stake protocols. In *EC '24: The 25th ACM Conference on Economics and Computation*. ACM, 2024.

15 Matheus V.X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC '22, pages 89–114, New York, NY, USA, 2022. Association for Computing Machinery.

16 Amos Fiat, Anna Karlin, Elias Koutsoupias, and Christos H. Papadimitriou. Energy equilibria in proof-of-work mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 489–502, 2019. doi:10.1145/3328526.3329630.

17 Guy Goren and Alexander Spiegelman. Mind the mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 475–487, 2019. doi:10.1145/3328526.3329566.

18 R.A. Howard. *Dynamic programming and Markov processes*. Technology Press of Massachusetts Institute of Technology, 1960.

19 Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 365–382, 2016. doi:10.1145/2940716.2940773.

20 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: http://www.bitcoin.org/bitcoin.pdf.

21 M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 2014.

22 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, pages 515–532, 2016. doi:10.1007/978-3-662-54970-4_30.

23 Toni Wahrstätter. Selfish mixing and randao manipulation, July 2023. URL: https://ethresear.ch/t/selfish-mixing-and-randao-manipulation/16081.

24 Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle maker: (time)stamping out the competition in ethereum. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 135–149. ACM, 2023. doi:10.1145/3576915.3616674.

25 Aviv Yaish, Saar Tochner, and Aviv Zohar. Blockchain stretching & squeezing: Manipulating time for your best interest. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 65–88. ACM, 2022. doi:10.1145/3490486.3538250.

26 Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient mdp analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, pages 113–131, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3419614.3423264.