Computing Optimal Manipulations in Cryptographic Self-Selection Proof-of-Stake Protocols

MATHEUS V. X. FERREIRA, University of Virginia, USA
AADITYAN GANESH, Princeton University, USA
JACK HOURIGAN, University of Pennsylvania, USA
HANNAH HUH, Princeton University, USA
S. MATTHEW WEINBERG, Princeton University, USA
CATHERINE YU, Princeton University, USA

Cryptographic Self-Selection is a paradigm employed by modern Proof-of-Stake consensus protocols to select a block-proposing "leader." Algorand [Chen and Micali, 2019] proposes a canonical protocol, and Ferreira et al. [2022] establish bounds $f(\alpha, \beta)$ on the maximum fraction of rounds a strategic player can lead as a function of their stake α and a network connectivity parameter β. While both their lower and upper bounds are non-trivial, there is a substantial gap between them (for example, they establish $f(10\%, 1) \in [10.08\%, 21.12\%]$), leaving open the question of how significant of a concern these manipulations are. We develop computational methods to provably nail $f(\alpha, \beta)$ for any desired (α, β) up to arbitrary precision, and implement our method on a wide range of parameters (for example, we confirm $f(10\%, 1) \in [10.08\%, 10.15\%]$).

Methodologically, estimating $f(\alpha, \beta)$ can be phrased as estimating to high precision the value of a Markov Decision Process whose states are countably-long lists of real numbers. Our methodological contributions involve (a) reformulating the question instead as computing to high precision the expected value of a distribution that is a fixed-point of a non-linear sampling operator, and (b) provably bounding the error induced by various truncations and sampling estimations of this distribution (which appears intractable to solve in closed form). One technical challenge, for example, is that natural sampling-based estimates of the mean of our target distribution are *not* unbiased estimators, and therefore our methods necessarily go beyond claiming sufficiently-many samples to be close to the mean.

CCS Concepts: • **Theory of computation** → **Algorithmic game theory and mechanism design**; • **Security and privacy** → *Cryptanalysis and other attacks.*

Additional Key Words and Phrases: blockchain, cryptocurrency, proof-of-stake, cryptography, strategic mining, provably correct estimations, fixed point estimations

ACM Reference Format:
Matheus V. X. Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Huh, S. Matthew Weinberg, and Catherine Yu. 2024. Computing Optimal Manipulations in Cryptographic Self-Selection Proof-of-Stake Protocols. In *The 25th ACM Conference on Economics and Computation (EC '24), July 8–11, 2024, New Haven, CT, USA.* ACM, New York, NY, USA, 27 pages. https://doi.org/10.1145/3670865.3673602

Authors' Contact Information: Matheus V. X. Ferreira, matheus@seas.harvard.edu, University of Virginia, Charlottesville, VA, USA; Aadityan Ganesh, aadityanganesh@princeton.edu, Princeton University, Princeton, NJ, USA; Jack Hourigan, hojack@seas.upenn.edu, University of Pennsylvania, Philadelphia, PA, USA; Hannah Huh, hannahchuh@gmail.com, Princeton University, Princeton, NJ, USA; S. Matthew Weinberg, smweinberg@princeton.edu, Princeton University, Princeton, NJ, USA; Catherine Yu, catyu6000@gmail.com, Princeton University, Princeton, NJ, USA.

EC '24, July 8–11, 2024, New Haven, CT, USA

ACM ISBN 979-8-4007-0704-9/24/07
https://doi.org/10.1145/3670865.3673602

1 Introduction

Blockchain protocols have attracted significant interest since Bitcoin's initial development in 2008 [Nakamoto, 2008], and several parallel research agendas and developments arose in that time. This paper lies at the intersection of two of these agendas: (a) strategic manipulability of consensus protocols, and (b) the return of Byzantine Fault Tolerant (BFT)-style consensus protocols via Proof-of-Stake (PoS). We briefly elaborate on both stories below, before discussing our contributions.

Manipulating Consensus Protocols. Initially following Nakamoto's whitepaper, Bitcoin and related blockchain protocols were studied through a classical security lens: some fraction of participants were honest, others were malicious, and the goal of study was to determine the extent to which a malicious actor can compromise security with a particular fraction of the computational power in the network. For example, Nakamoto's whitepaper already derives that with 51% of the computational power, a malicious actor could completely undermine Bitcoin's consensus protocol. However, the seminal work of Eyal and Sirer [2014], now referred to as "Selfish Mining", identified a fundamentally different cause for concern: an attacker with 34% of the computational power could manipulate the protocol in a way that does not violate consensus, but earns that attacker a $> 34\%$ fraction of the mining rewards.[1] This agenda has exploded over the past decade, and there is now a vast body of work considering strategic manipulation of consensus protocols (e.g. Bahrani and Weinberg, 2023, Brown-Cohen et al., 2019, Carlsten et al., 2016, Eyal and Sirer, 2014, Ferreira et al., 2022, Ferreira and Weinberg, 2021, Fiat et al., 2019, Goren and Spiegelman, 2019, Kiayias et al., 2016, Sapirshtein et al., 2016, Tsabary and Eyal, 2018, Yaish et al., 2023, 2022, Zur et al., 2020).

These works study *several* different classes of protocols, and *several* avenues for manipulation: some study Proof-of-Work protocols while others study Proof-of-Stake, some study block withholding deviations while others manipulate timestamps, some focus on profitability denoted in the underlying cryptocurrency while others consider the impact of manipulation on that cryptocurrency's value. There are many important angles to this agenda, many of which are cited by practitioners as key motivating factors in design choices.[2] The primary goal of this agenda is to understand *under what conditions is it in every participant's interest to follow the prescribed consensus protocol?* That is, these works generally do not focus on understanding complex equilibria with multiple strategic players (and instead immediately consider it a failure of the protocol when it is not being followed), and instead seek to understand whether the strategy profile where all agents follow the protocol constitutes a Nash Equilibrium. That is, we seek to understand whether being honest is the best response when everyone else in the network is honest.

BFT-based Proof-of-Stake Protocols. As Bitcoin's popularity surged, the energy demands required to secure it comparably soared, and estimates place its global energy consumption at comparable levels to countries the size of Australia. This motivated discussions over alternate technologies that could still be permissionless and Sybil-resistant, and Proof-of-Stake emerged as a viable alternative. While Proof-of-Work protocols select participants to produce blocks proportional to their computational power, Proof-of-Stake protocols do so proportional to the fraction of underlying cryptocurrency they own. Initial Proof-of-Stake protocols predominantly followed the longest-chain consensus paradigm of Bitcoin [Daian et al., 2016, Kiayias et al., 2017], but modern proposals now look more like classical consensus algorithms from distributed computing [Chen and Micali, 2019, Gilad et al., 2017]. Specifically, BFT-based protocols run a consensus algorithm,

[1] Earlier work of [Babaioff et al., 2012] introduces the strategic manipulation aspects of the Bitcoin protocol, although the style of manipulation in [Eyal and Sirer, 2014] became more mainstream for subsequent work.

[2] For example, EIP-1559.

one block at a time, in order to reach consensus on a single block. Once consensus is reached, the block is finalized and consideration of the next block begins.

While Bitcoin still uses a longest-chain Proof-of-Work consensus protocol, and some large Proof-of-Stake cryptocurrencies such as Cardano still use longest-chain protocols [Kiayias et al., 2017], BFT-based protocols are now quite mainstream and are implemented, for example, in Algorand [Chen and Micali, 2019, Gilad et al., 2017] and Ethereum.[3]

Manipulating BFT-based Proof-of-Stake Protocols. In practice, there is no 'dominant' BFT-based protocol, and different cryptocurrencies each seem to have their own protocol. However, there are some unifying themes. Most BFT-based protocols have the concept of a *leader* in each round, and the consensus goal of each round is for everyone to agree on the leader's proposed block. Leader selection is challenging, though: it should be done proportional to stake, but in a way that neither relies on a trusted external source of randomness nor is manipulable by participants. This has proved to be quite challenging, and to-date there are no nonmanipulable proposals without heavyweight cryptography (such as Multi-Party Computation or Verifiable Delay Functions). While the underlying consensus protocols are often both complex and completely nonmanipulable (without sufficient stake to simply subvert consensus in the first place), the leader selection protocols are more vulnerable, and can be studied independently of the supported consensus protocol.

Algorand's initial proposal serves as a canonical process of study due to its elegance. The initial seed Q_1 is a uniformly drawn random number. Then, in each round t with seed Q_t, every wallet digitally signs the statement (Q_t, c) for every coin c they own, hashes it,[4] and broadcasts the hash as their credential $\textsc{Cred}_t^c$. The holder of the coin with the lowest credential is the leader, and their winning credential becomes the seed Q_{t+1} for round $t+1$. This elegant protocol has several desirable properties (for example, it is not vulnerable to any form of 'stake grinding' to influence next round's seed – you can either broadcast your credential or not),[5] but Chen and Micali [2019] acknowledge that it may still be manipulable by cleverly choosing not to broadcast credentials, and Ferreira et al. [2022] indeed establish that any size staker has such a profitable manipulation.

To get brief intuition for a profitable manipulation, imagine that an attacker controls 10% of the coins. Perhaps they are also well-enough connected in the network so that they can choose which credentials of their own to broadcast in round t as a function of other participants' credentials (this corresponds to $\beta = 1$ – in general, the adversary is β-well-connected if they see a β fraction of honest credentials before broadcasting their own). Such an attacker might be in a position where they own (say) the three lowest credentials. In this case, the adversary could look one round ahead and determine which of these round-t-winning credentials gives them the best chance of winning round $t+1$, and broadcast only that credential. This particularly simple manipulation, termed the One-Lookahead strategy in Ferreira et al. [2022], is strictly profitable for any sized staker. While strictly better than honest, this strategy does not reap enormous profits: even with 10% stake, a 1-well-connected staker can lead at most 10.08% rounds. On the other hand, the only previous upper bounds derived on the maximum gains come from a loose analysis of an omniscient adversary who not only sees the credentials of honest wallets in round t, but can predict their future digital signatures to know exactly which hypothetical future rounds they'd win. This results in an upper

[3]Ethereum does maintain some longest-chain aspect to its protocol, but the key role that validators play make the protocol closer to BFT-based consensus.

[4]We will elaborate on this rigorously in Section 2.2. The role of the digital signature is simply to get a signature unique to the owner of coin c that no other player can predict, and the role of the hash function is to turn this into a uniformly drawn random number from $[0, 1]$.

[5]The live Algorand protocol seems to have recently pivoted from their initial proposal to a leader-selection protocol that has the winning credential of every k^{th} round set the seeds for the next k rounds.

bound of 21.12% on the maximum possible rounds led by a 10% staker. Needless to say, the level of concern that would arise from a 10% staker who is able to slightly increase their staking rewards by less than 1% to 10.08% is vastly different than what would arise from a 10% staker who can more than double their staking rewards to 21.12%.

Our Contributions. Using both theoretical and computational tools, we precisely nail down the manipulability of Algorand's canonical leader selection protocol. That is, we design computational methods to compute, for any fraction of stake α and network connectivity parameter β that an attacker might have,[6] the maximum fraction of rounds the attacker can lead (assuming other players are honest). We rigorously bound the error in our methods – some bounds hold with probability one (due to discretization, truncation, etc.) while others hold with high probability (due to sampling). We consider this methodology to be our main contribution. We also make the following adjacent contributions:

- We implement our computational procedure in Rust, and run it across several personal laptops and university clusters. We plot several of our findings in Section 5 (and future researchers can run our code to even higher precision, if desired). For example, we close the gap on the maximum profit of a 1-well-connected 10% staker from $[10.08\%, 21.12\%]$ to $[10.08\%, 10.15\%]$. We produce several plots in Section 5 demonstrating our results in comparison to prior bounds.
- One conclusion drawn from our simulations is that the gains from manipulation are quite small. For example, we confirm that 1-well-connected 10% staker can lead at most 10.15% of all rounds. A 0-well-connected 10% staker can lead at most 10.09% of the rounds. Even a 0-well-connected 20% staker can lead at most 20.21% of the rounds. This suggests that, while supralinear rewards are always a cause for concern as a potential centralizing force among stakers, the situation is unlikely to be catastrophic.
- A second conclusion drawn from our simulations is β plays a significant role in the magnitude of profitability. For example, a 0-well-connected 20% staker can lead at most 20.21% of the rounds, while there exists a strategy for a 1-well-connected 20% staker that leads at least 20.68% of the rounds – a 320% amplification in the marginal gains.
- Beyond our provably accurate computational methodology, we also provide two analytical results of independent interest.
  - We improve [Ferreira et al., 2022]'s analysis of the omniscient adversary, and in particular describe a recursive formulation that achieves an arbitrarily good approximation to the precise profit of an optimal omniscient adversary. This appears in Section 3.3.
  - Finally, we prove one conjecture and disprove another of Ferreira et al. [2022] characterizing "Balanced Scoring Functions." Balanced Scoring Functions are a tool used in leader selection to replace computing a digital-signature-then-hash per coin with computing a digital-signature-then-hash per wallet (in order to appropriately weight the hash before taking the minimum amongst all credentials). We state this result in Section 2.1.

As a whole, our results significantly improve our understanding of manipulating the canonical leader selection protocol first introduced in Algorand [Chen and Micali, 2019]. First, while supralinear rewards are always a cause for concern, the maximum achievable profits are at quite a small order of magnitude. Second, our results highlight the pivotal role that β plays in the rate of supralinear rewards. This suggests that protocol designers may wish to invest in augmentations to

[6]We define the network connectivity formally in Section 3.1.

bring β closer to zero.[7] Methodologically, our approach provides a blueprint for how similar leader selection protocols (such as Ethereum's) might be analyzed.

1.1 Very Brief Technical Highlight

We defer full details to our technical sections, but give a brief overview of the key technical challenges here. The optimal strategy can be phrased as a Markov Decision Process (MDP), and in some sense the obvious approach is to "write down the MDP and solve it." Unfortunately, states in our MDP are countably long lists of real numbers. That is, a state corresponds to (a) the list of credentials the attacker has in this round, but also (b) for each of those credentials i, and each other wallet j controlled by the adversary, the credential wallet j would provide the next round if wallet i wins this round (which can be computed as the seed for the next round is simply a digital signature plus hash of its credential i), and moreover (c) for each of those pairs of credentials (i, j), and each other wallet k controlled by the adversary, the credential wallet k would provide two rounds from now if credential i wins this round and credential j wins the next round, and (d) so on.

A first step is to truncate this countably long list of real numbers to (a) look only $T < \infty$ rounds in the future, (b) store only $k < \infty$ credentials per round, and (c) discretize each credential to a multiple of $\varepsilon > 0$. These steps can all be done with provable upper bounds on the error they induce. However, even with $k = 8$ and $T = 15$, states still correspond to a list of 8^{15} multiples of ε, and is clearly intractable.

Instead, our key idea is to reformulate the question as finding the distribution of future rewards that an optimal strategist receives. That is, consider the process of sampling a state for the attacker (a list of credentials for this round, hypothetical future credentials, etc.), and ask what future reward the attacker would get when playing optimally. If we can compute this distribution of rewards D, then its expected value is exactly the number we seek. We define an operator $\Theta(\cdot)$ that takes as input samples from some distribution F and produces samples from the distribution $\Theta(F)$, and establish that D is a fixed point of $\Theta(\cdot)$.

Again, the process now appears straight-forward: start from any distribution, and iterate $\Theta(\cdot)$ until it stabilizes. This is indeed our approach, and the remaining challenge is to account for sampling error. Essentially, we are looking for $\mathbb{E}[\Theta^{15}(F)]$ for some simple initial distribution F, and instead of computing $\Theta(\cdot)$ at each stage we'll take an empirical estimate $\hat{\Theta}(\cdot)$ instead. This appears ripe for a Chernoff plus union bound to bound the error due to sampling, except that $\mathbb{E}[\hat{\Theta}(\cdot)]$ is not an unbiased estimator for $\mathbb{E}[\Theta(\cdot)]$. So even establishing that we take sufficiently many samples to be close to the process's expected value does not guarantee we are close to $\mathbb{E}[\Theta^{15}(F)]$. Instead, at each round we both inflate (resp. deflate) our empirical $\hat{\Theta}^i(F)$ so that we know it stochastically dominates (resp. is stochastically dominated by) $\Theta^i(F)$ using a variant of the DKW inequality [Dvoretzky et al., 1956].

We share this to give the reader a sense of the technical developments necessary to analyze this particular Markov Decision Process, and ways in which it differs from more common MDPs.

1.2 Related Work

Manipulating Leader Selection Protocols. Chen and Micali [2019] propose the Algorand leader selection protocol, and acknowledge that it may be manipulable. They also prove an upper bound on the fraction of rounds an adversary can win after being honest in the previous round. Ferreira et al. [2022] provide a strategy that is strictly profitable for all β-well-connected α-sized stakers, and upper bound the attainable profit of an omniscient adversary who can predict future digital

[7] One such possibility is to broadcast credentials using commit-reveal: players make a large deposit along with a cryptographic commitment to their credential, and unlock their deposit only upon revealing it.

signatures of honest players. We provide provably accurate computational methodology to nail the optimal manipulability up to arbitrary precision (and implement our algorithms and draw conclusions from the results). In concurrent and independent work, [Cai et al., 2024] establish that any strictly profitable manipulation of our same leader selection protocol is statistically detectable (that is, an onlooker who sees only the seeds of each round can distinguish whether someone is profitably manipulating the protocol from when all players are honest but sometimes offline). This work is orthogonal to ours, but also provides an argument that solid defenses against manipulation exist (we argue that the manipulations are not particularly profitable, they argue that they are always detectable).

Manipulating Consensus Protocols. We have already briefly cited a subset of the substantial body of work studying profitable manipulations of consensus protocols [Bahrani and Weinberg, 2023, Brown-Cohen et al., 2019, Carlsten et al., 2016, Eyal and Sirer, 2014, Ferreira et al., 2022, Ferreira and Weinberg, 2021, Fiat et al., 2019, Goren and Spiegelman, 2019, Kiayias et al., 2017, Sapirshtein et al., 2016, Tsabary and Eyal, 2018, Yaish et al., 2023, 2022]. Of these, [Brown-Cohen et al., 2019, Ferreira and Weinberg, 2021] also study Proof-of-Stake protocols, but longest-chain variants (and therefore have minimal technical overlap). Sapirshtein et al. [2016] bears some technical similarity, as they are the unique prior work that finds optimal manipulations (in Bitcoin's Proof-of-Work), and they also use computational tools with theoretical guarantees. We also use a lemma of theirs to reduce from maximizing the fraction of rounds won to maximizing reward in a linear MDP. Still, there is minimal technical overlap beyond these. For example, their problem can be phrased as a Markov Decision Process with countably-many states (i.e. a state in their setup is of the form "how many hidden blocks do you have?", which is an integer), and therefore the key steps in their provable guarantees are truncations. In comparison, we've noted that our problem is a Markov Decision Process with uncountably many states, and therefore the two MDPs have minimal overlap.[8]

2 Preliminaries

2.1 Primitives

In this section, we review various cryptographic primitives required to construct a cryptographic self-selection protocol. Since our model is identical, we use notations identical to Ferreira et al. [2022]. We begin by discussing a tool central to many Proof-of-Stake protocols– verifiable random functions. Verifiable random functions are useful in enabling a source of randomness endogenous to the blockchain for the leader election protocol.

Definition 1 (Ideal Verifiable Random Function (Ideal VRF))**.** *An ideal verifiable random function satisfies the following properties:*

(1) ***Setup:*** *There is an efficient randomized generator that can produce a pair (sk, pk) of a secret key and a public key that characterizes the instance $f_{sk}(\cdot)$.*
(2) ***Private computability:*** *For a string x, there exists an efficient algorithm to compute the encryption $f_{sk}(x)$ of x with the knowledge of sk.*
(3) ***Perfect randomness:*** *Without the knowledge of sk, the random variables $f_{sk}(x)$ and $f_{sk}(y)$ are distributed i.i.d. over $U[0,1]$. In particular, the random variable $f_{sk}(x) \sim U[0,1]$ even with the knowledge of $\left((y_i, f_{sk}(y_i))\right)_{1 \le i \le m}$ such that $y_i \neq x$ for all $1 \le i \le m$.*
(4) ***Verifiability:*** *Verifying the claim $y = f_{sk}(x)$ can be done efficiently conditioned on the knowledge of pk and a proof V_x, even if sk remains unknown. Generating a proof V_x such that a verifier confirms equality when $y \neq f_{sk}(x)$ is impossible.*

[8]This is also perhaps expected, as there is little technical similarity between creating forks in a longest-chain protocol and manipulating credentials in a leader-selection protocol.

An ideal VRF allows the holder of the secret key *sk* (through property 3) to provably generate a random number, i.e, show that the random number was generated through a prescribed process. However, it is impossible to construct an ideal VRF whose outputs are *statistically indistinguishable* from $U[0,1]$. On the other hand, it is possible to construct a VRF whose outputs are *computationally indistinguishable* from $U[0,1]$. For the sake of simplicity, we proceed with an Ideal VRF instead of computational – this results in only a negligible difference.

Example 1 (VRFs through digital signatures)**.** *Let σ be a digital signature scheme with a public key, secret key pair (pk, sk) and let h be a hash function. Then, $h(\sigma_{sk}(\cdot))$ is a verifiable random function. $y = h(\sigma_{sk}(x))$ can be computed efficiently with the knowledge of sk. With a proof $V_x = \sigma_{sk}(x)$, $y = h(\sigma_{sk}(x))$ can be verified as follows- verify that (i) the proof $V_x = \sigma_{sk}(x)$ with the public key pk and x and (ii) verify $y = h(V_x)$.*

Next, we proceed to discuss balanced scoring functions that enable electing a leader proportional to its stake.

Definition 2 (Balanced Scoring Functions)**.** *A scoring rule $S : [0,1] \times \mathbb{R} \to [0,1]$ is balanced if:*

(1) *For $X \sim U[0,1]$, the distribution of $S(X,\alpha)$ has no point masses for all $\alpha \in [0,1]$*
(2) *For all $n \in \mathbb{N}$ and $(\alpha_i)_{1\le i\le n} \in \mathbb{R}^n_{\ge 0}$,*

$$Pr_{X_1,\dots,X_n\sim U[0,1]}\Big(\arg\min_{1\le i\le n}\{S(X_i,\alpha_i)\} = j\Big) = \frac{\alpha_j}{\sum_{i=1}^n \alpha_i}$$

At a high level, a fair leader selection to elect a wallet with probability proportional to its stake can be conducted by choosing the wallet with the smallest score, while VRFs provide the source for the random variable X_i for a wallet i.

Ferreira et al. [2022] conjecture that $S(X, \sum_{i=1}^n \alpha_i)$ and $\min_{1\le i\le n}\{S(X,\alpha_i)\}$ are identically distributed for all balanced scoring functions S, $n \in \mathbb{N}$ and $\alpha_1,\dots,\alpha_n \in \mathbb{R}_{\ge 0}$[9]. Intuitively, their conjecture claims an adversary with a total stake $\sum_{i=1}^n \alpha_i$ cannot increase the probability of a smaller score and thus, the probability of getting elected by splitting their stake as $(\alpha_i)_{1\le i\le n}$ across n different wallets. We settle their conjecture.

Theorem 1. *Let $S(X,\alpha)$ be any balanced scoring function. Then, for all $n \in \mathbb{N}$ and $(\alpha_i)_{1\le i\le n}$, the random variables*

$$S(X, \sum_{i=1}^n \alpha_i) \text{ and } \min_{1\le i\le n}\{S(X_i,\alpha_i)\}$$

are identically distributed for $X, X_1, \dots, X_n \sim U[0,1]$.

The proof of Theorem 1 and further details on scoring functions are deferred to the full version of the paper.

2.2 Cryptographic Self-Selection

We are now ready to describe a cryptographic self-selection protocol.

Definition 3 (Cryptographic Self-Selection Protocol A (CSSPA); Ferreira et al., 2022)**.** *A Cryptographic Self-Selection Protocol A is the following:*

(1) *Every wallet i sets up an instance of an ideal VRF with public key, secret key pair (pk_i, sk_i) prior to round 1. Wallet i holds a stake α_i.*

[9]They also claim that $S(X,\alpha)$ is continuous in α. We provide a counterexample to their claim. However, we show that $Pr(S(X,\alpha) \ge s)$ is continuous in α. Refer the full version of the paper for a detailed discussion on scoring functions.

(2) *Q_t denotes the seed of round t. The seed Q_1 for the initial round is computed through an expensive multi-party computation and is distributed according to $U[0,1]$.*
(3) *In each round t, the user with wallet i computes its credential $\textsc{Cred}_t^i := f_{sk_i}(Q_t)$.*
(4) *Each user can choose either to broadcast its credential or remain silent. Any credential broadcast by a user is received by all other users*[10].
(5) *The wallet with the smallest score $S(\textsc{Cred}_t^i, \alpha_i)$ amongst all broadcasted credentials is elected the leader ℓ_t for round t.*
(6) *The seed for round $t+1$, $Q_{t+1} = \textsc{Cred}_t^{\ell_t}$, the credential of the winner of round t. All wallets learn the seed Q_{t+1}.*

Importantly, note that the blockchain cannot be forked in the CSSPA as in the case with many BFT-based consensus protocols including Algorand.

We consider strategic manipulations rather than network security attacks, and so the action space of users is restricted to distributing their stakes across multiple wallets and choosing between broadcasting and remaining silent for each of its wallet, as opposed to a network partition attack. An honest player keeps its stake in a single wallet and always broadcasts its credential. Conditioned on all players in the network being honest, observe that the probability of a wallet with stake α_j getting elected equals the probability that wallet i has the smallest score, which happens with a probability proportional to α_i.

We discuss choosing an explicit balanced scoring function for our model. Ferreira et al. [2022] argue that the game induced by the CSSPA is independent of the choice of the scoring function and show a bijection between the strategies of a strategic player in the games induced by two different scoring functions that preserve the player's rewards (Definition 7). We choose the logarithmic scoring function defined below.

Definition 4 (Logarithmic Scoring Function). *For $X \in [0,1]$ and $\alpha \in \mathbb{R}_{\geq 0}$,*

$$S_{\ln}(X,\alpha) = \begin{cases} \infty & \text{when } \alpha = 0 \\ 0 & \text{when } X = 0, \alpha \neq 0 \\ \frac{-\ln X}{\alpha} & \text{otherwise} \end{cases}$$

Definition 5 (Exponential Distribution). *The exponential distribution $\exp(\alpha)$ with rate α is the distribution with a cumulative density function (CDF) $F(x,\alpha) = 1 - e^{-\alpha x}$ and a probability density function (pdf) $f(x,\alpha) = \alpha e^{-\alpha x}$.*

Lemma 1 (Lemma 2.1 from Ferreira et al., 2022). *$S_{\ln}(X,\alpha)$ is distributed according to $\exp(\alpha)$ when $X \sim U[0,1]$.*

For notational convenience, we denote $S_{\ln}$ by S unless mentioned otherwise.

Consider a user distributing their stake α equally across n wallets for $n \to \infty$. The scores of each wallet is distributed according to $\exp(\frac{\alpha}{n})$. It is well-known that the minimum of n i.i.d random variables drawn from $\exp(\theta)$ is distributed according to $\exp(n\,\theta)$ (see Appendix A from Ferreira et al., 2022, for example). Therefore, the minimum score over all wallets of the user is distributed as per $\exp(\alpha)$. The following describes the distribution of the i^{th}-smallest score amongst the n wallets.

Lemma 2 (Lemma 4.3 from Ferreira et al., 2022). *Let $(X_i)_{i\in\mathbb{N}}$ be exponentially distributed i.i.d random variables such that $\min_{i\in\mathbb{N}}\{X_i\}$ is distributed according to $\exp(\alpha)$. Let Y_i be the random*

[10]We assume this to focus on the relevant aspects of the paper and is consistent in prior work that focuses on incentives [Bahrani and Weinberg, 2023, Carlsten et al., 2016, Eyal and Sirer, 2014, Ferreira and Weinberg, 2021, Ferreira et al., 2019, Kiayias et al., 2016, Sapirshtein et al., 2016]

variable denoting the i^{th}-smallest value in $(X_i)_{i\in\mathbb{N}}$. Then, $(Y_i)_{i\in\mathbb{N}}$ is distributed according to the following random process:

$$Y_1 \leftarrow \exp(\alpha) \text{ and } Y_{i+1} \leftarrow Y_i + \exp(\alpha)$$

We have the prerequisites to study the actions of a strategic player in the CSSPA in place.

3 Model

3.1 The Adversarial Game and Reward

In a network consisting of honest stakers, we study a single strategic adversary whose rewards are proportional to the fraction of rounds it is elected to propose a block. Conditioned on the stake the adversary holds in the system, we want to estimate the optimal marginal utility gained by the adversary from being strategic. We adopt the adversarial model described in Ferreira et al. [2022], which we review below.

We define the space of strategies available to the adversary. We abuse notation to denote the cryptographic self-selection protocol, the game played by the adversary and the space of strategies available to the adversary by CSSPA.

Definition 6 (CSSPA(α,β))**.** *In* CSSPA(α,β)*, the network consists of three players — the adversary with stake α, and two honest players B and C with stakes $\beta(1-\alpha)$ and $(1-\beta)(1-\alpha)$ respectively.Prior to round 1, the adversary learns the values of α, β and that B and C are honest. For $n \to \infty$, the adversary distributes its stake into a set A of n wallets, each containing a stake $\frac{\alpha}{n}$. The adversary makes the following decisions in round t:*

1. *The adversary learns the seed Q_t of round t.*
2. *The adversary computes the credentials $\textsc{Cred}_t^i$ for all wallets $i \in A$.*
3. *Further, the adversary learns the credentials $\textsc{Cred}_t^B$ of player B. The adversary knows that the credential of player C is drawn from $\exp((1-\beta)(1-\alpha))$ but does not learn $\textsc{Cred}_t^C$.*
4. *For any $r \geq 0$ and $(i_t, i_{t+1}, \ldots, i_{t+r}) \in (A \cup \{B\}) \times A^r$, the adversary precomputes the credentials $\textsc{Cred}_{t+r'}^{i_{t+r'}}$ for $1 \leq r' \leq r$ assuming $i_{t+\hat{r}}$ is elected to lead in round $t+\hat{r}$ for all $0 \leq \hat{r} < r'$.*
5. *The adversary either remains silent or broadcasts the credential of a wallet $i \in A$.*

The following discussion throws light on bullet 4 of Definition 6. Before broadcasting any credential in round t, the adversary observes the credentials of its own wallets and the credential of B. All credentials $\textsc{Cred}_t^i$ for $i \in A \cup \{B\}$ observed by the adversary are potential seeds for round $t+1$. Assuming one of these credentials as a hypothetical seed, the adversary can compute the credentials $\textsc{Cred}_{t+1}^i$ for all of its wallets $i \in A$. These hypothetical credentials are themselves potential seeds for round $t+2$. More generally, the adversary can precompute all possible future credentials of its wallets, assuming the precomputed credentials keep becoming the seed for successive rounds.

β denotes the network connectivity of the adversary. The stake of C, and therefore the probability of C having a small score and being selected, decreases with β. Thus, for large values of β, it is much more unlikely for a credential not precomputed by the adversary, namely $\textsc{Cred}_t^C$, to become the seed Q_{t+1} for the next round. Since both B and C have non-negative stakes, $\beta \in [0,1]$.

Remember that the honest strategy collects all stake into a single wallet and broadcasts the credential of the wallet each round. We assume both B and C play the honest strategy. We normalize the total stake to 1 and as a consequence, use the stake and the fraction of stake held in a wallet interchangeably.

Definition 7 (Reward of a Strategy)**.** *For a strategy π describing the actions taken by the adversary in each round of* CSSPA(α,β)*, let the Bernoulli random variable $X_t(\alpha,\beta;\pi)$ be 1 if the adversary is*

elected in round t and 0 otherwise. Then, the expected reward

$$\mathrm{Rew}(\alpha, \beta; \pi) = \mathbb{E}\Big[\liminf_{T\longrightarrow\infty} \frac{\sum_{t=1}^{T} X_t(\alpha, \beta; \pi)}{T}\Big]$$

equals the fraction of rounds led by the adversary in expectation over the outcomes of the VRFs in each round.

When clear from the context, we drop the parameters α and β and denote $\mathrm{Rew}(\alpha, \beta; \pi)$ and $X_t(\alpha, \beta; \pi)$ by $\mathrm{Rew}(\pi)$ and $X_t(\pi)$.

The above model of the CSSPA appears quite restrictive in more than one aspect – the adversary can broadcast at most one credential, the adversary cannot strategically distribute its stake into multiple accounts prior to round 1, and there are only two honest players in the network. In Appendix A, we recap a very general model of CSSPA(α, β) discussed in Ferreira et al. [2022] and their results showing that the above restricted version of the CSSPA has the same optimal adversarial reward as the more general version.

3.2 Biased Seeds and Stopping Times

We aim to estimate the reward $\mathrm{Rew}(\pi) = \mathbb{E}\Big[\liminf_{T\longrightarrow\infty} \frac{\sum_{t=1}^{T} X_t(\pi)}{T}\Big]$ the adversary wins by playing a strategy π. A tractable closed-form expression for $X_t(\pi)$ is hard to find and computing its expected values for all $1 \leq t \leq \infty$ is infeasible. Therefore, it becomes imperative to find a round τ such that the expected adversarial reward can be estimated without computing the expected value of $X_{\tau+r}(\pi)$ for any $r > 0$. We call such a round τ a stopping time. The expected adversarial reward has a much simpler expression in terms of stopping times.

Lemma 3 (Lemma 4.1 from Ferreira et al., 2022)**.** *Suppose the strategy π has an expected finite stopping time τ in* CSSPA(α, β)*. Then,*

$$\mathrm{Rew}(\pi) = \frac{\mathbb{E}[\sum_{t=1}^{\tau} X_t(\pi)]}{\mathbb{E}[\tau]}$$

where τ is a random variable denoting a stopping time.

Suppose we reach a round $\tau + 1$ such that the adversary is indifferent between the current seed $Q_{\tau+1}$ and a fresh draw from $U[0, 1]$. We say such a seed $Q_{\tau+1}$ is *unbiased*. The adversary's rewards from the rounds following $\tau+1$ is similar to restarting CSSPA(α, β) from round 1, whose initial seed Q_1 is drawn from $U[0, 1]$ (in practise, this is done through an expensive multi-party computation and is not susceptible to manipulation). The expected adversarial reward can be computed by estimating only the distributions of $X_1(\pi), X_2(\pi), \ldots, X_\tau(\pi)$ and thus, τ is a stopping time.

For an arbitrary round t, it is hard to determine whether the seed Q_{t+1} is unbiased and whether t is a stopping time. We define forced stopping times so that they are much easier to identify. In the next few paragraphs, we motivate forced stopping times through an example adversarial strategy.

For a stake α and a random seed Q_t in round t, the probability that the adversary gets elected is at most α. However, the adversary could have multiple wallets whose scores are smaller than the score of the honest wallets B and C. When $\beta = 1$, the adversary knows the smallest honest score and that broadcasting the credentials of any of its wallets with a smaller score would ensure an adversarial wallet getting elected in round t. It is convenient to explicitly christen these candidate adversarial wallets.

Definition 8 (Potential Winners and Adversarial Potential Winners)**.** *In round t with seed Q_t, let $\hat{W}(Q_t)$ be the set of all adversarial wallets with a score less than that of B. Then, $\hat{W}(Q_t)$ is the set of adversarial potential winners and $W(Q_t) = \hat{W}(Q_t) \cup \{B\}$ is the set of potential winners in round t.*

Out of these adversarial potential winners, the adversary can choose to broadcast the one that optimizes its future rewards, which can be estimated by computing hypothetical seeds for the rounds following round t (see bullet 4 from Definition 6). The adversary can also choose to sacrifice the current round and remain silent if the future rewards from the honest wallet B getting elected more than compensates for losing round t.

Suppose a seed $Q_{\tau+1}$ is realized for which the adversary has not computed any hypothetical future seeds. At the instant in which $Q_{\tau+1}$ is realized, the adversary is indifferent between $Q_{\tau+1}$ and a fresh draw from $U[0,1]$ and thus, $Q_{\tau+1}$ is unbiased. Now, consider a round τ in which the smallest score either belongs to B or C. The first time the adversary learns the honest credential with the smallest score, the adversary would have pre-computed neither the credential nor any hypothetical future credentials following the honest credential since computing them would require the secret key of the honest wallet with the smallest score. The adversary cannot thwart $\textsc{Cred}_\tau^B$ or $\textsc{Cred}_\tau^C$ from becoming the seed $Q_{\tau+1}$. Thus, the seed $Q_{\tau+1}$ is unbiased and round τ is a stopping time. We call such stopping times as forced stopping times. Forced stopping times are easy to identify since we only have to ensure that the smallest score does not belong to an adversarial wallet.

Definition 9 (Forced Stopping Time)**.** *Let i be the wallet with the smallest score in round τ. τ is a forced stopping time if $i \notin A$.*

Lemma 4 (Lemma 4.2 from Ferreira et al., 2022)**.** *If τ is a forced stopping time, τ is a stopping time.*

We will use τ to denote the first forced stopping time of the adversary. We will only consider forced stopping times (and not any 'unforced' stopping times) for the remainder of the paper. Because of this and for convenience, we abuse notation and refer to forced stopping times plainly as stopping times.

3.3 The Omniscient Adversary

As a warm up, we look at the omniscient adversary studied by Ferreira et al. [2022] that is stronger than the adversary in CSSPA(α,β) in the following aspects:

- $\beta = 1$. Further, for any $r \geq 0$ and $(i_t, i_{t+1}, \ldots, i_{t+r}) \in \left(A \cup \{B\}\right)^{r+1}$, the adversary pre-computes the credentials $\textsc{Cred}_{t+r'}^{i_{t+r'}}$ for $1 \leq r' \leq r$ assuming $i_{t+\hat{r}}$ is elected to lead in round $t+\hat{r}$ for all $0 \leq \hat{r} < r'$. In other words, the omniscient adversary can precompute hypothetical future credentials even when B is elected to be the leader.
- $X_t = 1$ for all rounds $t < \tau$, i.e., the omniscient adversary is rewarded to delay the first stopping time, even if it entails being elected only for a very small fraction of rounds. By Lemma 3, the omniscient reward

$$\text{Rew}^{\text{OMNI}}(\pi) = \frac{\mathbb{E}[\sum_{t=1}^{\tau} X_t(\pi)]}{\mathbb{E}[\tau]} = \frac{\mathbb{E}[\tau - 1]}{\mathbb{E}[\tau]} = 1 - \frac{1}{\mathbb{E}[\tau]}$$

Similar to the adversary in CSSPA(α,β), we define τ to be a stopping time for the omniscient adversary if the set of adversarial potential winners $\hat{W}$ for round τ is empty. We defer our discussions on the omniscient adversary to the full version. We summarize our findings in Theorem 2.

Theorem 2. *For the omniscient adversary with stake α, there exists a constant $\kappa \approx 0.38$ such that,*

(1) *for $\alpha > \kappa$, there exists a strategy π such that $\mathbb{E}[\tau]$ is unbounded and $\text{Rew}^{\text{OMNI}}(\pi) = 1$, and,*
(2) *for $\alpha \leq \kappa$ and any strategy π, $\mathbb{E}[\tau] \leq \frac{1-3\alpha+3\alpha^2-3\alpha^3}{(1-3\alpha+\alpha^2)(1-\alpha+\alpha^2)}$ and $\text{Rew}^{\text{OMNI}}(\pi) \leq \alpha \cdot \left(\frac{1-2\alpha+\alpha^2-\alpha^3}{1-3\alpha+3\alpha^2-3\alpha^3}\right)$.*

We also show a non-closed form upper bound on the optimal omniscient rewards that can be made tight up to an arbitrarily small additive error.

The following results on the size of potential winners and first stopping time of the omniscient adversary would be bootstrapped further to get upper bounds on the optimal rewards of the actual adversary in CSSPA(α, β).

Lemma 5 (Corollary 4.1 from Ferreira et al., 2022)**.** *For a random seed $Q_t \sim U[0,1]$ and $i^* \geq 0$, the probability $Pr(|W_t(Q_t)| = i^* + 1)$ of the adversary having exactly $i^* + 1$ potential winners equals $\alpha^{i^*}(1-\alpha)$.*

Lemma 6. *For the omniscient adversary with stake $\alpha < \kappa$, $Pr(\tau > 0) = 1$ and $Pr(\tau > 1) = \alpha$. For $T \geq 2$,*

$$Pr(\tau > T) \leq \alpha^2 \cdot \tfrac{2-2\alpha+\alpha^2}{1-\alpha+\alpha^2} \cdot \left(\alpha \cdot \tfrac{2-\alpha}{1-\alpha}\right)^{T-2}.$$

4 Estimating the Optimal Adversarial Reward

We proceed to designing simulations that estimate the expected adversarial reward from playing a strategy π in CSSPA(α, β). We find the optimal adversarial strategy in CSSPA(α, β) quite complex to describe. We reformulate CSSPA(α, β) such that a succinct description of the optimal adversarial strategy becomes possible. Then, we propose a simulation that computes the adversary's optimal reward precisely but requires an infinite run-time. We describe a sequence of modifications to the simulation that trades off run-time for precision to get provable bounds on the adversary's optimal rewards.

4.1 A Linear Version of CSSPA(α, β)

Optimizing the adversarial reward $\text{Rew}(\pi) = \frac{\mathbb{E}[\sum_{t=1}^{\tau} X_t(\pi)]}{\mathbb{E}[\tau]}$ depends on maintaining a balance between getting a myopic gain in the reward by winning the election in the current round and a long-term gain through delaying the first stopping time. This inherent trade-off between the short-term and long-term gains of actions in CSSPA(α, β) makes both describing the optimal adversarial strategy and simulating it hard. We reformulate CSSPA(α, β) through an approach similar to Sapirshtein et al. [2016] that allows the adversary to myopically optimize its reward without having to worry about long-term consequences.

We introduce an entry fee λ in CSSPA(α, β) that the adversary is charged to participate in each round and consider the total adversarial reward instead of the rate at which the adversary is elected.

Definition 10 (LinearCSSPA(α, β, λ))**.** *In* LinearCSSPA(α, β, λ)*, the network consists of three players- the adversary with stake α, two honest players B and C with stakes $\beta(1-\alpha)$ and $(1-\beta)(1-\alpha)$ respectively. Prior to round 1, the adversary learns the values of α, β, the entry fee λ and that B and C are honest. For $n \to \infty$, the adversary distributes its stake into a set A of n wallets, each containing a stake $\frac{\alpha}{n}$. The adversary makes the following decisions in round t:*

1. *The adversary pays an entry fee λ.*
2. *The adversary learns the seed Q_t of round t.*
3. *The adversary computes the credentials $\textsc{Cred}_t^i$ for all wallets $i \in A$. Further, the adversary learns the credentials $\textsc{Cred}_t^B$. The adversary knows that the credential of player C is drawn from $\exp((1-\beta)(1-\alpha))$ but does not learn $\textsc{Cred}_t^C$.*
4. *For any $r \geq 0$ and $(i_t, i_{t+1}, \ldots, i_{t+r}) \in (A \cup \{B\}) \times A^r$, the adversary precomputes the credentials $\textsc{Cred}_{t+r'}^{i_{t+r'}}$ for $1 \leq r' \leq r$ assuming $i_{t+\hat{r}}$ is elected to lead in round $t+\hat{r}$ for all $0 \leq \hat{r} < r'$.*
5. *The adversary either remains silent or broadcasts the credential of a wallet $i \in A$.*
6. *The game terminates if either B or C have scores smaller than all wallets $i \in A$, i.e, a stopping time is reached.*

Definition 11 (Reward of a Strategy). *For a strategy π played by the adversary in* LinearCSSPA(α, β, λ), *let the Bernoulli random variable $X_t(\pi)$ be* 1 *if the adversary is elected in round t and* 0 *otherwise. The adversary earns an expected reward*

$$\mathrm{Rew}^{\mathrm{Lin}}(\pi) = \mathbb{E}\Big[\sum_{t=1}^{\tau}(X_t(\pi) - \lambda)\Big]$$

We conclude the discussion by relating the rewards $\mathrm{Rew}(\pi)$ in CSSPA(α, β) and $\mathrm{Rew}^{\mathrm{Lin}}(\pi)$ in LinearCSSPA(α, β, λ).

Theorem 3. *For an entry fee λ and a strategy π, $\mathrm{Rew}^{\mathrm{Lin}}(\pi) > 0$ (resp. $\mathrm{Rew}^{\mathrm{Lin}}(\pi) < 0$) if and only if $\lambda < \mathrm{Rew}(\pi)$ (resp. $\lambda > \mathrm{Rew}(\pi)$). Further, $\mathrm{Rew}^{\mathrm{Lin}}(\pi) = 0$ when $\lambda = \mathrm{Rew}(\pi)$.*

A standard binary search would locate the value of λ such that $\mathrm{Rew}^{\mathrm{Lin}}(\pi) = 0$, in which case, $\mathrm{Rew}(\pi) = \lambda$. We defer the proof to the full version.

4.2 The Ideal Simulation

LinearCSSPA(α, β, λ) has a recursive structure that we exploit while designing simulations to estimate $\mathrm{Rew}^{\mathrm{Lin}}(\pi)$ of a strategy π. In some round t, by broadcasting the credential $\textsc{Cred}_t^i$ of a wallet $i \in A$ and winning the election (or remaining silent and letting B with credential $\textsc{Cred}_t^B$ win), the adversary induces an instance of LinearCSSPA(α, β, λ) with an initial seed $Q_0 = \textsc{Cred}_t^i$. If the adversary recursively "knew" the expected future reward r_i that it would get from broadcasting $\textsc{Cred}_t^i$ for each adversarial potential winner i and the reward r_0 from letting B win, the adversary can decide its actions just based on $\vec{r} = (r_i)_{i \in \mathbb{N} \cup \{0\}}$ and the scores $\vec{c} = (c_i)_{i \in \mathbb{N} \cup \{0\}}$ of the wallets in $A \cup \{B\}$[11]. Of course, the adversary always runs the risk of the current round being a stopping time, in which case, the total future rewards earned by the adversary equals zero.

Let D^π be the distribution of rewards the adversary achieves by playing the strategy π in LinearCSSPA(α, β, λ). We are interested in estimating the expected reward $\mathbb{E}_{s \sim \mathrm{D}^\pi}[s]$. We do so by constructing the *CDF* of D^π in AddLayer$(\alpha, \beta, \lambda, \pi, \mathrm{D}^\pi)$ by sampling from D^π infinitely many times. This can be done by setting up infinitely many 'induced-instances' of LinearCSSPA(α, β, λ). For each of these induced-instances:

(1) For each $i \geq 1$, we sample the i^{th} smallest score c_i amongst all adversarial wallets using Lemma 2:
$$c_1 \leftarrow \exp(\alpha), c_i \leftarrow c_{i-1} + \exp(\alpha)$$
(2) For each $i \geq 1$, we sample the reward r_i earned from broadcasting the credential of the adversarial wallet with the i^{th} smallest score from the distribution D^π.
(3) We compute the adversarial reward in expectation over the reward r_0 from letting B win, the scores c_0 and c_{-1} of B's and C's wallets respectively by simulating the behaviour of π for scores $\vec{c} = (c_i)_{i \in \mathbb{N} \cup \{0\}}$ and rewards $\vec{r} = (r_i)_{i \in \mathbb{N} \cup \{0\}}$.

Estimating the adversarial reward reduces to finding a fixed-point D^π to AddLayer$(\alpha, \beta, \lambda, \pi, \mathrm{D}^\pi)$.

AddLayer$(\alpha, \beta, \lambda, \pi, \mathrm{D}^\pi)$:
(1) For $1 \leq \ell \leq n_t = \infty$.
  (a) DrawAdv(α, D^π):
    - Sample $\vec{r}_{-0}$: Draw $k = \infty$ rewards $r_1, r_2, \ldots, r_k$ i.i.d from D^π.

[11] This is not entirely true. The adversary can still play strategies based on the credentials and rewards from previous rounds. However, given that the adversary's goal is to optimize the total rewards earned across rounds, such strategies can be safely ignored. As we will see, the optimal strategy can be codified in this language.

- Sample $\vec{c}_{-0}$: Draw $k = \infty$ scores $c_1, c_2, \ldots, c_k$ of adversarial wallet as follows. Draw $c_1 \leftarrow \exp(\alpha)$ and $c_{i+1} \leftarrow c_i + \exp(\alpha)$ (a fresh sample for each i) for $1 \le i \le k-1$. For convenience, set $c_{k+1} = \infty$.
- Return $(\vec{r}_{-0}, \vec{c}_{-0})$.

(b) sample$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi, \mathrm{D}^{\pi})$: Simulate the action of the strategy π in the current round given D^{π}, $\vec{r}_{-0}$ and $\vec{c}_{-0}$. Return the reward s_ℓ in expectation over the reward r_0 from letting B win, B's score c_0 and C's score c_{-1}.

(2) Return D^{π} to be the uniform distribution over $\{s_1, s_2, \ldots, s_{n_t}\}$.

Simulate$(\alpha, \beta, \lambda, \pi)$:

(1) Compute a fixed-point D^{π} to AddLayer$(\alpha, \beta, \lambda, \pi, \mathrm{D}^{\pi})$.

(2) Return $\mathbb{E}_{s\sim\mathrm{D}^{\pi}}[s]$.

See Appendix D.5 for a summary of the notations and functions used in the simulations.

4.3 The Optimal Solution

We describe the optimal strategy π^{OPT} in the recursive language introduced in Section 4.2. Let $\mathrm{D}^{\mathrm{OPT}} = \mathrm{D}^{\pi^{\mathrm{OPT}}}$ be the distribution of rewards from playing π^{OPT}. At a start of a round t, the adversary learns the scores $\vec{c}$ of wallets in $A \cup \{B\}$ and rewards $\vec{r}$ from remaining silent and from broadcasting the credential of each wallet in A. We use $\vec{c}_{-0}$ and $\vec{r}_{-0}$ to denote the scores $(c_i)_{i\in\mathbb{N}}$ and rewards $(r_i)_{i\in\mathbb{N}}$ associated with the wallets in A. Re-index the adversary's wallets (and therefore, $\vec{c}_{-0}$ and $\vec{r}_{-0}$) in increasing order of its scores. Let $i^*(\vec{c}) := |\{i > 0 | c_i \le c_0\}|$ be the number of adversarial potential winners.

We compare the expected future rewards of all possible actions the adversary can take at the start of round t.

(1) Suppose the adversary abstains from broadcasting. It earns a reward r_0 unless a stopping time is reached, which happens either when $i^*(\vec{c}) = 0$ or when C has a score $c_{-1} \sim \exp((1-\beta)(1-\alpha))$ smaller than the score c_0 of B. The probability of C having a larger score than c_0 equals $e^{-c_0(1-\beta)(1-\alpha)}$. Thus, the expected reward from remaining silent equals $e^{-c_0(1-\beta)(1-\alpha)} r_0 \cdot \mathbb{1}(i^*(\vec{c}) \neq 0)$. We define

$$h(c_0, r_0) := e^{-c_0(1-\beta)(1-\alpha)} r_0$$

(2) From broadcasting the credential of an adversarial potential winner i with score c_i and future reward r_i, the adversary earns a reward 1 from getting elected in the current round and thus, a total reward $(1 + r_i)$. This, once again, is subject to the current round not being a stopping time. The current round is not a forced stopping time if C has a score larger than c_i, which happens with probability $e^{c_i(1-\beta)(1-\alpha)}$, and if $i^*(\vec{c}) \neq 0$. The adversary has a potential winner i and $i^*(\vec{c})$ is at least 1 as a consequence. Hence, the expected reward from broadcasting the credential of i equals $e^{c_i(1-\beta)(1-\alpha)}(1 + r_i)$. The adversary can broadcast the credential of the wallet that maximizes its reward to earn

$$g(c_0, \vec{c}_{-0}, \vec{r}_{-0}) = \max_{i \le i^*(\vec{c})} \{e^{-c_i(1-\beta)(1-\alpha)}(1 + r_i)\}$$

The adversary also pays an entry fee λ. Between remaining silent and broadcasting its best credential, the adversary wins

$$\begin{aligned}
&\max\{h(c_0, r_0)\mathbb{1}(i^*(\vec{c}) \neq 0), g(c_0, \vec{c}_{-0}, \vec{r}_{-0})\} - \lambda \\
&\quad = \max\{h(c_0, r_0)\mathbb{1}(i^*(\vec{c}) \neq 0), g(c_0, \vec{c}_{-0}, \vec{r}_{-0})\mathbb{1}(i^*(\vec{c}) \neq 0)\} - \lambda \\
&\quad = \max\{h(c_0, r_0), g(c_0, \vec{c}_{-0}, \vec{r}_{-0})\}\mathbb{1}(i^*(\vec{c}) \neq 0) - \lambda
\end{aligned}$$

While using the future rewards from round t to compute the optimal action to take in round $t-1$, the adversary will not know the values c_0 and r_0 since B does not broadcast $\textsc{Cred}_t^B$ until the start of round t. The adversary can only compute the future rewards from playing an action in expectation over c_0 and r_0. With this in mind, we construct $\mathrm{D}^{\mathrm{OPT}}$ to be the distribution of (future) rewards in expectation over c_0 and r_0. Given $\vec{c}_{-0}$ and $\vec{r}_{-0}$, we implement a sampling procedure $\mathrm{sample}(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi^{\mathrm{OPT}}, \mathrm{D}^{\mathrm{OPT}})$ by setting the ℓ^{th} sample s_ℓ to be

$$\mathbb{E}_{c_0 \sim \exp((1-\beta)\,(1-\alpha)), r_0 \sim \mathrm{D}^{\mathrm{OPT}}}[\max\{h(c_0, r_0), g(c_0, \vec{c}_{-0}, \vec{r}_{-0})\}\,\mathbb{1}(i^*(\vec{c}) \neq 0)] - \lambda$$

Finding a fixed-point $\mathrm{D}^{\mathrm{OPT}}$ for $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \mathrm{D}^{\mathrm{OPT}})$ seems intractable and we resort to heuristic methods instead. One natural heuristic would be to begin at the point-mass distribution $\mathrm{D}_0^{\mathrm{OPT}}$ at 0 and iterate infinitely many times to get the sequence $\left(\mathrm{D}_t^{\mathrm{OPT}}\right)_{t \in \mathbb{N} \cup \{0\}}$ of distributions satisfying $\mathrm{D}_{t+1}^{\mathrm{OPT}} = \mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \mathrm{D}_t^{\mathrm{OPT}})$. We end this section by summarizing the challenges in executing the above heuristic. The pseudo-code for the optimal strategy is in Appendix D.1.

(1) The iterated-point heuristic does not guarantee convergence. Even if the iteration converges, there could be a multitude of fixed-points and the iteration could converge to a distribution that is not the optimal reward.
(2) We run $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \cdot)$ $T = \infty$ many times. In each execution of $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \cdot)$, the adversary can pick one of $k = \infty$ actions– one each for broadcasting credentials of wallets $i \in A$ and one for staying silent. π^{OPT} compares the rewards of each of these actions before making a decision.
(3) Given a distribution D_t, we compute $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \mathrm{D}_t)$ by constructing $n_t = \infty$ samples. As we will see in Section 4.4.3, the simulation is not even an unbiased estimator of the reward $\mathbb{E}_{s \sim \mathrm{D}_T^{\mathrm{OPT}}}[s]$ once we constrain n_t to be finite.
(4) The sample s_ℓ is constructed by computing the reward in expectation over B's score c_0 and reward r_0 from remaining silent. This involves calculating a double integral. The integrals can be calculated in finite-time by approximating them by a Riemann sum. However, even a polynomial run-time would not be practical due to the sheer number of samples we construct. We require a linear run-time.

4.4 Moving from Ideal to Practical

4.4.1 Convergence of the Iterated-Point Heuristic. We discuss the natural variant $\mathrm{LinearCSSPA}(\alpha, \beta, \lambda, T)$ that terminates after T rounds if a stopping time has not been reached yet. We will argue that the distribution of optimal rewards $\left(\mathrm{D}_T^{\mathrm{OPT}}\right)_{T \in \mathbb{N} \cup \{0\}}$ satisfies the same recursion as the iterated-point heuristic on $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \cdot)$ and converges to $\mathrm{D}^{\mathrm{OPT}}$ as $T \to \infty$.

When $T = 0$, the game terminates even before it starts and the adversary gets a total reward zero. Thus, $\mathrm{D}_0^{\mathrm{OPT}}$ is the point-mass on zero, identical to the initial point of the iterated-point heuristic. t rounds before termination, by broadcasting the credential $\textsc{Cred}_{-t}^i$ of a wallet $i \in A$ ($i = B$ if the adversary remains silent), the adversary induces an instance of $\mathrm{LinearCSSPA}(\alpha, \beta, \lambda, t-1)$ with an initial seed $Q_0 = \textsc{Cred}_{-t}^i$. Thus, if the adversary recursively knew the rewards $r_i \sim \mathrm{D}_{t-1}^{\mathrm{OPT}}$ from each potential winner i, the adversary would broadcast the credential (or stay silent) that would maximize its reward from the last $t-1$ rounds. This is the same operation performed by $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \cdot)$ on $\mathrm{D}_{t-1}^{\mathrm{OPT}}$. By induction, the distribution of rewards $\mathrm{D}_t^{\mathrm{OPT}}$ equals the reward distribution $\mathrm{AddLayer}(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \mathrm{D}_{t-1}^{\mathrm{OPT}})$ output by the t^{th} iteration of the iterated-point method.

As $T \to \infty$, the distribution of rewards T rounds before termination and $T-1$ rounds before termination are identical. This is equivalent to claiming $\mathrm{D}_T^{\mathrm{OPT}}$ as $T \to \infty$ approaches the reward distribution $\mathrm{D}^{\mathrm{OPT}}$. Thus, the iterated-point method converges and converges to the correct distribution of rewards.

4.4.2 Infinite Rounds and Credentials. Let LinearCSSPA$(\alpha, \beta, \lambda, T)$ be the variant of LinearCSSPA(α, β, λ) terminating after round T. Simulate$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}})$ loops infinitely to construct the distribution of adversarial rewards in LinearCSSPA$(\alpha, \beta, \lambda, T)$ as $T \to \infty$. Further, for each round of the simulation, DrawAdv$(\alpha, \mathrm{D}_t^{\mathrm{OPT}})$ samples a score and a reward for each of the infinite wallets the adversary operates. We revisit CSSPA(α, β) and argue that terminating CSSPA(α, β) after T rounds and constraining the adversary to broadcasting the credentials of a wallet only if it is amongst the k smallest scores in A does not cause a significant drop in the adversary's optimal reward. Once established, we can estimate the reward from playing $\pi_{T,k}^{\mathrm{OPT}}$, the optimal strategy in CSSPA(α, β) that terminates after T rounds and never uses a score outside the k smallest scores in A, instead of estimating the reward from π^{OPT}.

We abuse notation to describe the optimal strategy of the adversary in CSSPA(α, β) as π^{OPT} and its reward distribution by $\mathrm{D}^{\mathrm{OPT}}$. We say the adversary is k-scored if the adversary is constrained to either stay silent or broadcast a credential amongst its wallets with the k smallest scores.

Theorem 4. *For $\alpha \leq 0.29$ and a k-scored adversary, the difference in the expected rewards between playing π^{OPT} and $\pi_{T,k}^{\mathrm{OPT}}$ in* CSSPA(α, β) *satisfies*

$$0 \leq |\,\mathrm{Rew}(\pi^{\mathrm{OPT}}) - \mathrm{Rew}(\pi_{T,k}^{\mathrm{OPT}})| \leq \alpha^2 \cdot \tfrac{2-2\alpha+\alpha^2}{1-\alpha+\alpha^2} \cdot [\alpha \cdot \tfrac{2-\alpha}{1-\alpha}]^{T-2} + \alpha^k$$

We defer the proof to the full version of the paper.

We estimate the k-scored adversary's optimal reward in LinearCSSPA$(\alpha, \beta, \lambda, T)$ and binary search over λ to estimate the adversarial reward $\mathrm{Rew}(\pi_{T,k}^{\mathrm{OPT}})$ in CSSPA(α, β). We can then upper bound and lower bound the optimal reward $\mathrm{Rew}(\pi^{\mathrm{OPT}})$ by

$$\mathrm{Rew}(\pi_{T,k}^{\mathrm{OPT}}) \leq \mathrm{Rew}(\pi^{\mathrm{OPT}}) \leq \mathrm{Rew}(\pi_{T,k}^{\mathrm{OPT}}) + \alpha^2 \cdot \tfrac{2-2\alpha+\alpha^2}{1-\alpha+\alpha^2} \cdot [\alpha \cdot \tfrac{2-\alpha}{1-\alpha}]^{T-2} + \alpha^k$$

We abuse notation as usual and call the k-scored adversary's optimal strategy in LinearCSSPA$(\alpha, \beta, \lambda, T)$ as $\pi_{T,k}^{\mathrm{OPT}}$. Let $\mathrm{D}_{T,k}^{\mathrm{OPT}}$ be our estimate of the distribution of rewards from playing $\pi_{T,k}^{\mathrm{OPT}}$. We modify the simulation to terminate after T rounds and bake the k-scored adversary into AddLayer$(\alpha, \beta, \lambda, \pi_{T,k}^{\mathrm{OPT}}, \cdot)$. The modified pseudo-code can be found in Appendix D.2.

4.4.3 Constructing Infinitely many Samples for AddLayer$(\alpha, \beta, \lambda, k, \pi_{T,k}^{\mathrm{OPT}}, \cdot)$. We address the infinite run-time for AddLayer$(\alpha, \beta, \lambda, k, \pi_{T,k}^{\mathrm{OPT}}, \cdot)$ from constructing infinitely many samples to perfectly describe the *CDF* of its output. For an input distribution D_0, we want to approximate the sequence of distributions $\left(\mathrm{D}_t\right)_{0 \leq t \leq T}$ such that $\mathrm{D}_t := \mathrm{AddLayer}(\alpha, \beta, \lambda, k, \pi_{T,k}^{\mathrm{OPT}}, \mathrm{D}_{t-1})$ while constructing only finitely many samples for each of them. We dedicate a section in the appendix of the full version to discuss the challenges from using the most natural technique to bound the error from estimation in our simulations– Chernoff bounds or McDiarmid's inequality followed by a union bound. Even more importantly, we also find that the estimator that arises from constructing finite number of samples might not even be unbiased (Appendix B).

We tackle the above challenges by maintaining two distributions, $\overline{\mathrm{D}}_t$ that dominates D_t and $\underline{\mathrm{D}}_t$ that is dominated by D_t. We sketch our method for constructing an empirical distribution that is dominated by the true distribution. Suppose for a sufficiently large number of samples, we can guarantee that, for all values r, the quantile $\tilde{q}$ in the empirical distribution constructed by sampling n times and the quantile q in the true distribution satisfy $\tilde{q} \in [q-\delta, q+\delta]$. Dropping the δ smallest

(strongest) quantiles and replacing them with δn samples of the infimum of the true distribution would give us a new estimated distribution that is dominated by the true distribution. We call this process deflation. We will compute $\underline{\mathrm{D}}_t$ by first computing $\mathrm{AddLayer}(\alpha, \beta, \lambda, k, \pi^{\mathrm{OPT}}_{T,k}, \underline{\mathrm{D}}_{t-1})$ and then deflating the outcome by a suitable parameter δ. By a straightforward induction, $\underline{\mathrm{D}}_t$ is dominated by D_t.

We can construct $\overline{\mathrm{D}}_t$ from $\overline{\mathrm{D}}_{t-1}$ through an analogous inflation procedure. However, the error in the estimated reward due to inflation is much larger than the error due to deflation. An inflated reward with a quantile $\tilde{q} \in [0, \delta]$ is much more likely to be chosen by an adversary, since the adversary picks its optimal future rewards). This leaves a bigger impact on the estimated reward and influences the rewards in successive rounds too. This differs from deflate since a deflated reward with a quantile $\tilde{q} \in [1-\delta, 1]$ is very likely to be ignored by the adversary since the reward from a different wallet is likely to be higher. To mitigate the strong credentials from drifting the estimated reward far way from $\mathbb{E}_{s\sim \mathrm{D}_t}[s]$, we perform a more nuanced inflation procedure.

$\mathrm{Deflate}(n, \gamma, \mathrm{D}, t)$: Given an input D drawn uniformly from n samples,

(1) Delete the largest $n \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$ samples from D

(2) Append $n \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$ copies of $-\lambda$ to D

$\mathrm{Inflate}(n, \gamma, \omega, \mathrm{D}, t)$: Given an input D drawn uniformly from $\left(s_\ell\right)_{1\le \ell \le n}$ (in descending order),

(1) Delete the smallest $n \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$ samples from D

(2) Append ωn copies of $t\ (1-\lambda)$ to D

(3) For $1 \le \ell < \frac{n}{\omega\, n} \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$:

- Append ωn copies of s_ℓ

Theorem 5. *Let* $\mathrm{TruncatedSimulate}(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega)$ *output an upper bound* $\overline{\mathrm{D}}_{T,k}$ *and a lower bound* $\underline{\mathrm{D}}_{T,k}$. *Then,*

(1) *With probability at least* $1 - T\left(\gamma + \frac{e^{-\omega n}}{\omega}\sqrt{\frac{\ln \gamma^{-1}}{2n}}\right)$, $\mathbb{E}_{s\sim \overline{\mathrm{D}}_{T,k}}[s] \ge \mathbb{E}_{s\sim \mathrm{D}^{\mathrm{OPT}}_{T,k}}[s]$.

(2) *With probability at least* $1 - T\,\gamma$, $\mathbb{E}_{s\sim \underline{\mathrm{D}}_{T,k}}[s] \le \mathbb{E}_{s\sim \mathrm{D}^{\mathrm{OPT}}_{T,k}}[s]$.

We defer the proof to the full version.

Our estimate is not precisely equal to $\mathrm{D}^{\mathrm{OPT}}_{T,k}$ and to differentiate the two, we denote our estimation of $\mathrm{D}^{\mathrm{OPT}}_{T,k}$ by $\hat{\mathrm{D}}^{\mathrm{OPT}}_{T,k}$. We update the simulations to contain inflate and deflate in Appendix D.3.

4.4.4 Computing Expectations. In this section, we tackle the final challenge of needing to compute integrals accurately while constructing the sample s_ℓ. This involves computing a double integral, one over the score c_0 of B and the reward r_0 from staying silent. Naively integrating over the reward distribution from the previous round described by n samples would result in a run-time of $\Omega(n)$ for each of the n samples, and consequently $\Omega(n^2)$ for simulating one round. Even though this is $\mathrm{Poly}(n)$, it still turns out to be intractable to run for the extremely large number of samples we expect to handle each round. Instead, we aim to reduce the run-time to $\tilde{O}(n)$.

To begin with, observe that for each sample s_ℓ, we are drawing k scores for the wallets of the k-scored adversary. Thus, we end up with a run-time of $\Omega(k \cdot n)$ no matter what we do. Any additional compute that we perform for each sample is only going to increase the order of the run-time. Thus, our aim is to run as much pre-compute as possible before even constructing the first sample, and minimize the number of fresh computations needed to be performed with each sample. We get a practical run-time through a combination of pre-computes and by computing

the integrals involved with taking expectations over r_0 and c_0 as a discrete sum. We defer the details to the full version. Note that we introduce two parameters ϵ and η that reflect the precision to which we discretize the distributions constructed during the simulations and the precision to which we approximate the two integrals. We do all our pre-computations through the function Precompute$(\hat{\mathrm{D}}, \epsilon, \eta, t)$.

Lemma 7. *Precompute$(\hat{\mathrm{D}}, \epsilon, \eta, t)$ terminates in $O(\frac{t}{\epsilon\,\eta})$ time.*

TruncatedSimulate$(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, \epsilon, \eta)$ in Appendix D.4 reflects the updates in terms of the precomputations. Appendix D.4 also compiles the changes made to the simulations across various stages and presents a summary.

Lemma 8. *A single execution of TruncatedSimulate$(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, \epsilon, \eta)$ terminates in time $O(Tkn + \frac{T^2}{\epsilon\,\eta})$.*

4.5 Locating the Optimal Expected Reward and the Optimal Adversarial Strategy

Remember that the end goal of the simulations is to compute the optimal reward for an adversary playing CSSPA(α, β). We denote the optimal k-scored adversarial strategy in LinearCSSPA$(\alpha, \beta, \lambda, T)$ by $\pi^{\mathrm{OPT}}_{T,k}(\lambda)$ and in CSSPA(α, β) that terminates in T rounds by $\pi^{\mathrm{OPT}}_{T,k}$. Let $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda}(\pi)$ be the expected reward from playing π in LinearCSSPA$(\alpha, \beta, \lambda, T)$. Remember that $\mathrm{Rew}(T)$ is the expected reward from playing π in CSSPA(α, β). By Theorem 3, $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda}(\pi^{\mathrm{OPT}}_{T,k}) \geq 0$ iff $\lambda \leq \mathrm{Rew}(\pi^{\mathrm{OPT}}_{T,k})$ with equality holding precisely when $\lambda = \mathrm{Rew}(\pi^{\mathrm{OPT}}_{T,k})$. To estimate $\mathrm{Rew}(\pi^{\mathrm{OPT}}_{T,k})$, we run TruncatedSimulate$(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k}(\lambda), \gamma, \omega, \epsilon, \eta)$ and binary search over λ until the expected reward output by the simulation is approximately zero. More precisely, we search for λ such that the expected values of the upper bound $\overline{\mathrm{D}}_{T,k}(\lambda)$ and the lower bound $\underline{\mathrm{D}}_{T,k}(\lambda)$ are slightly larger and slightly smaller than zero. However, the binary search could potentially output any λ such that $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda}(\pi^{\mathrm{OPT}}_{T,k}(\lambda)) \approx 0$ even if λ is far off from $\mathrm{Rew}(\pi^{\mathrm{OPT}}_{T,k})$. Such an error will become all the more likely given that the simulations only produce an interval $\left[\mathbb{E}_{s\sim\overline{\mathrm{D}}_{T,k}(\lambda_1)}[s], \mathbb{E}_{s\sim\underline{\mathrm{D}}_{T,k}(\lambda_1)}[s]\right]$ such that $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda}(\pi^{\mathrm{OPT}}_{T,k}(\lambda))$ lies in this interval, instead of exactly computing the rewards. The following theorem rules out such scenarios.

Theorem 6. *Let TruncatedSimulate$(\alpha, \beta, \lambda_1, T, k, \pi^{\mathrm{OPT}}_{T,k}(\lambda_1), \gamma, \omega, \epsilon, \eta)$ output the upper bound and lower bound distributions $\overline{\mathrm{D}}_{T,k}(\lambda_1)$ and $\underline{\mathrm{D}}_{T,k}(\lambda_1)$ respectively, such that $\mathbb{E}_{s\sim\overline{\mathrm{D}}_{T,k}(\lambda_1)}[s] - \mathbb{E}_{s\sim\underline{\mathrm{D}}_{T,k}(\lambda_1)}[s] \leq \delta$. Suppose for some $r \in \left[\mathbb{E}_{s\sim\overline{\mathrm{D}}_{T,k}(\lambda_1)}[s], \mathbb{E}_{s\sim\underline{\mathrm{D}}_{T,k}(\lambda_1)}[s]\right]$, $|r - \mathrm{Rew}^{\mathrm{Lin}}_{\lambda_2}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_2))| \leq \zeta$. Then, $|\lambda_1 - \lambda_2| \leq \zeta + \delta$ with probability at least $1 - \left(2\,T\,\gamma + T\,\frac{e^{-\omega n}}{\omega}\sqrt{\frac{\ln\gamma^{-1}}{2n}}\right)$.*

We defer the proof to Appendix C.1. By locating the optimal adversarial reward (approximately), we also uncover a very succinct description of the adversary's (near) optimal strategy, which we describe in Appendix C.2.

5 Simulation Results

Below, we summarize the results from our simulations. In Figure 1 (Appendix E), we compare the bounds from previous works against our results. We see that our bounds on the adversarial rewards of both the omniscient adversary (blue) and the actual adversary in CSSPA(α, β) (green) is significantly tighter than the bound for the omniscient adversary from Ferreira et al. [2022] (orange). The bounds plotted in Figure 1 is empirical, and are not provably correct, since we did not inflate samples when constructing the reward distributions. However, they are fairly representative of the

scale of the marginal rewards the adversary achieves from being strategic. For instance, even with an extremely large stake of 0.2, we get the marginal rewards to be in the range $[0.0068, 0.0078]$ (for a simulation with deflated and inflated sampling), which is not considerable. Further, observe that the rewards from the 1-lookahead strategy (red), which is much more tractable than the optimal strategy to describe and compute, is already close to the optimal adversarial reward. In Figure 2 in Appendix E, we plot the marginal rewards of the adversary against its stake for various values of β. Observe that the network connectivity β plays an important role in the rewards and the adversary has significant gains for larger values of β. For instance, at $\alpha = 0.2$ the marginal utility when $\beta = 1$ is at least 0.0068 (from a simulation constructed from deflated sampling) and at most 0.0021 (from a simulation constructed from inflated sampling) when $\beta = 0$. Finally, Figure 3 (also in Appendix E) compares the adversary's marginal rewards as a function of β for a stake $\alpha = 0.25$, once again, highlighting the role of connectivity in strategic manipulation in blockchain protocols.

Acknowledgments

This work is supported by a Ripple UBRI grant, and an NSF CAREER Award CCF-1942497. The authors also thank anonymous reviewers for valuable feedback during the review process.

References

Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. 2012. On bitcoin and red balloons. In *Proceedings of the 13th ACM Conference on Electronic Commerce, EC 2012, Valencia, Spain, June 4-8, 2012*, Boi Faltings, Kevin Leyton-Brown, and Panos Ipeirotis (Eds.). ACM, 56–73. https://doi.org/10.1145/2229012.2229022

Maryam Bahrani and S. Matthew Weinberg. 2023. Undetectable Selfish Mining. *CoRR* abs/2309.06847 (2023). https://doi.org/10.48550/ARXIV.2309.06847 arXiv:2309.06847

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

Linda Cai, Jingyi Liu, S Matthew Weinberg, and Chenghan Zhao. 2024. Profitable Manipulations of Cryptographic Self-Sortition are Statistically Detectable. In *In submission to the 25th ACM Conference on Economics and Computation*.

Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. 2016. 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*. 154–167. https://doi.org/10.1145/2976749.2978408

Jing Chen and Silvio Micali. 2019. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.* 777 (2019), 155–183. https://doi.org/10.1016/J.TCS.2019.02.001

Phil Daian, Rafael Pass, and Elaine Shi. 2016. Snow White: Provably Secure Proofs of Stake. *IACR Cryptology ePrint Archive* 2016 (2016), 919. http://eprint.iacr.org/2016/919

Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. 1956. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics* (1956), 642–669.

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

Matheus V. X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. 2022. Optimal Strategic Mining Against Cryptographic Self-Selection in Proof-of-Stake. In *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, David M. Pennock, Ilya Segal, and Sven Seuken (Eds.). ACM, 89–114. https://doi.org/10.1145/3490486.3538337

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

Matheus Xavier Ferreira, S. Matthew Weinberg, Danny Yuxing Huang, Nick Feamster, and Tithi Chattopadhyay. 2019. Selling a Single Item with Negative Externalities. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. 196–206. https://doi.org/10.1145/3308558.3313692

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

Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China,*

October 28-31, 2017. ACM, 51–68. https://doi.org/10.1145/3132747.3132757

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

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

Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*. 357–388. https://doi.org/10.1007/978-3-319-63688-7_12

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

Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. 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*. 515–532. https://doi.org/10.1007/978-3-662-54970-4_30

Itay Tsabary and Ittay Eyal. 2018. The Gap Game. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 713–728. https://doi.org/10.1145/3243734.3243737

Aviv Yaish, Gilad Stern, and Aviv Zohar. 2023. Uncle Maker: (Time)Stamping Out The Competition in Ethereum. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda (Eds.). ACM, 135–149. https://doi.org/10.1145/3576915.3616674

Aviv Yaish, Saar Tochner, and Aviv Zohar. 2022. Blockchain Stretching & Squeezing: Manipulating Time for Your Best Interest. In *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, David M. Pennock, Ilya Segal, and Sven Seuken (Eds.). ACM, 65–88. https://doi.org/10.1145/3490486.3538250

Roi Bar Zur, Ittay Eyal, and Aviv Tamar. 2020. Efficient MDP analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 113–131.

A A More General CSSPA(α, β)

In this section, we review the general version of CSSPA(α, β) defined in Ferreira et al. [2022].

Definition 12 (CSSPA$(\alpha, \vec{\alpha}, \beta)$)**.** *In* CSSPA$(\alpha, \vec{\alpha}, \beta)$*, the network consists of the adversary with stake α and honest players with stakes given by $\vec{\alpha}$. Prior to round* 1*, the adversary learns the values of $\alpha, \vec{\alpha}$ and β and that the network apart from the adversary is honest. For a choice $n \geq 1$, the adversary distributes its stake arbitrarily over a set A of n wallets. The adversary makes the following decisions in round t:*

1. *The adversary learns the seed Q_t of round t.*
2. *The adversary computes the credentials $\textsc{Cred}_t^i$ for all wallets $i \in A$. The adversary chooses a subset of honest players B with total stake at most $\beta(1-\alpha)$ and learns the credentials $\textsc{Cred}_t^i$ for all $i \in B$. For all wallets $i \in C$, the adversary knows that $\textsc{Cred}_t^i$ will be drawn independently from* $\exp(\alpha_i)$*.*
3. *For any $r \geq 0$ and $(i_t, i_{t+1}, \ldots, i_{t+r}) \in (A \cup B) \times A^r$, the adversary precomputes the credentials $\textsc{Cred}_{t+r'}^{i_{t+r'}}$ for $1 \leq r' \leq r$ assuming $i_{t+\hat{r}}$ is elected to lead in round $t+\hat{r}$ for all $0 \leq \hat{r} < r'$.*
4. *The adversary either remains silent or broadcasts the credentials of some subset A_t of the adversarial wallets A.*

From the above definition of CSSPA$(\alpha, \vec{\alpha}, \beta)$, Ferreira et al. [2022] make a series of refinements that lead to CSSPA(α, β) without compromising on the adversarial reward.

Lemma 9 (Observation 3.2 from Ferreira et al., 2022)**.** *For any $\alpha, \vec{\alpha}, \beta$, define $\vec{\alpha}'$ to have two honest players with stakes $\alpha_1 = \beta(1-\alpha)$ and $\alpha_2 = (1-\beta)(1-\alpha)$ respectively. For any strategy π in* CSSPA$(\alpha, \vec{\alpha}, \beta)$*, there exists a strategy π' in* CSSPA$(\alpha, (\alpha_1, \alpha_2), \beta)$ *such that* $\text{Rew}(\alpha, (\alpha_1, \alpha_2), \beta; \pi') = \text{Rew}(\alpha, \vec{\alpha}, \beta; \pi)$*.*

Lemma 10 (Lemma 3.1 from Ferreira et al., 2022)**.** *Let π be a strategy in* $\mathrm{CSSPA}(\alpha, (\alpha_1, \alpha_2), \beta)$ *where the adversary splits its stake into n wallets. Then there exists a strategy π' such that the adversary divides its stake into $2n$ wallets and* $\mathrm{Rew}(\pi') \geq \mathrm{Rew}(\pi)$.

Lemma 11 (Observation 3.1 from Ferreira et al., 2022)**.** *For any strategy π in* $\mathrm{CSSPA}(\alpha, (\alpha_1, \alpha_2), \beta)$ *that distributes the adversarial stake across n wallets, there exists an adversarial strategy π' that also distributes the stake across the same number of wallets, broadcasts the credential of at most one wallet in A and results in exactly the same leaders as π, thereby getting the same reward as π.*

The following conclusion is straightforward from the above lemmas.

Theorem 7. *The optimal adversarial reward in* $\mathrm{CSSPA}(\alpha, \vec{\alpha}, \beta)$ *is at most the optimal adversarial reward in* $\mathrm{CSSPA}(\alpha, \beta)$ *for all $\alpha, \vec{\alpha}$ and β.*

B Bias from Finite Sampling

We revisit the computation of the adversary's reward when its k smallest scores are $\vec{c}_{-0}$ and the corresponding rewards are $\vec{r}_{-0}$. The reward from broadcasting the credential i (or remaining silent) equals $e^{-c_i\,(1-\beta)\,(1-\alpha)}\,(r_i + \mathbb{1}(i \neq 0))$ (assuming a stopping time is not reached). The adversary chooses the action that maximizes its reward.

Consider a toy version of the adversarial game over just two rounds. In the first round, a coin with heads probability $\frac{1}{k}$ is tossed n times. Let the empirically observed probability of heads be p_1. For each trial in the second round, toss k coins each with heads probability p_1. The outcome is 1 even if one of k tosses turns out heads (we take the maximum amongst k Bernoulli trials, similar to the adversarial game). Repeat the trial n times to observe an empirical probability p_2.

We verify that p_2 does not remain constant for $k = 2$ over different choices of n. If $n = \infty$, $p_1 = \frac{1}{k} = \frac{1}{2}$ and $p_2 = 1 - (1 - \frac{1}{k})^2 = \frac{3}{4}$. However, for $n = 10$ samples, $p_1 = \frac{m}{n}$ with probability $\binom{n}{m} \cdot 2^{-n}$ and $\mathbb{E}[p_2] = \sum_{m=0}^{n} \left[1 - \left(1 - \frac{m}{n}\right)^2\right] \times \binom{n}{m} \cdot 2^{-n} \approx 0.784 \neq 0.75$

A similar bias creeps into the adversarial game. The effect of the bias on the estimated rewards worsens with the number of rounds T and becomes better with the number of samples n.

C Locating λ and Describing the Optimal Adversarial Strategy

C.1 Proof of Theorem 6

We first begin with a toy version that says if the rewards from $\mathrm{LinearCSSPA}(\alpha, \beta, \lambda_1, T, k)$ and $\mathrm{LinearCSSPA}(\alpha, \beta, \lambda_2, T, k)$ are close, then $\lambda_1 - \lambda_2$ is small.

Lemma 12. *Suppose that λ_1 and λ_2 are such that*

$$|\,\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_1}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_1)) - \mathrm{Rew}^{\mathrm{Lin}}_{\lambda_2}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_2))| \leq \zeta.$$

Then, $|\lambda_1 - \lambda_2| \leq \zeta$.

Proof. Without loss of generality, assume that $\lambda_1 \leq \lambda_2$. Assume for contradiction that $\lambda_2 - \lambda_1 > \zeta$.

As a thought experiment, let the adversary ape $\pi^{\mathrm{OPT}}_{T,k}(\lambda_2)$ in $\mathrm{LinearCSSPA}(\alpha, \beta, \lambda_1, T, k)$. Since the rewards are linear, the adversary earns $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_2}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_2))$ plus the savings in entry fee from paying just $\lambda_1 < \lambda_2$. Since the adversary participates in at least one round, the adversary saves more than $(\lambda_2 - \lambda_1)$ in entry fee, and thus, the total reward is at least $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_2}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_2)) + (\lambda_2 - \lambda_1)$. However, $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_1}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_1)) - \mathrm{Rew}^{\mathrm{Lin}}_{\lambda_2}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_2)) \leq \zeta$ and $(\lambda_2 - \lambda_1) \geq \zeta$. The adversary makes strictly more than $\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_1}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_1))$ in $\mathrm{LinearCSSPA}(\alpha, \beta, \lambda_1, T, k)$. This is a contradiction, since $\pi^{\mathrm{OPT}}_{T,k}(\lambda_1)$ is the optimal strategy. □

In Theorem 6, we show that even if we estimate LinearCSSPA$(\alpha, \beta, \lambda_1, T, k)$ up to an error of δ then, the true reward in LinearCSSPA$(\alpha, \beta, \lambda_2, T, k)$ and our estimated rewards can still not be close unless $\lambda_1 - \lambda_2$ is small.

Proof of Theorem 6. From Theorem 5, with probability at least $1 - \left(2\,T\,\gamma + T\,\frac{e^{-\omega n}}{\omega}\,\sqrt{\frac{\ln \gamma^{-1}}{2n}}\right)$,

$$\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_1}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_1)) \in \left[\mathbb{E}_{s\sim \overline{\mathrm{D}}_{T,k}(\lambda_1)}[s], \mathbb{E}_{s\sim \underline{\mathrm{D}}_{T,k}(\lambda_1)}[s]\right]$$

Thus, $|\mathrm{Rew}^{\mathrm{Lin}}_{\lambda_1}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_1)) - r| \le \delta$ and $|r - \mathrm{Rew}^{\mathrm{Lin}}_{\lambda_2}(\pi^{\mathrm{OPT}}_{T,k}(\lambda_2))| \le \zeta$. The claim follows from the triangle inequality and Lemma 12. □

C.2 The Optimal Adversarial Strategy

Once we (approximately) locate the optimal reward λ^* of the k-scored adversary in CSSPA(α, β) that terminates in T rounds, the description of a near optimal adversarial strategy becomes succinct. The adversary pretends that it is k-scored and is participating in LinearCSSPA$(\alpha, \beta, \lambda, T)$ and plays the actions recommended by $\pi^{\mathrm{OPT}}_{T,k}(\lambda^*)$. In particular, the adversary resets CSSPA(α, β) with frequency at least T and behaves like a k-scored adversary. The adversary gets a zero reward in LinearCSSPA$(\alpha, \beta, \lambda, T)$, which corresponds to a reward λ^* in CSSPA(α, β).

The reward from the above strategy approaches the optimal reward as the error in locating λ^* reduces. The loss in the reward from being k-scored and from resetting CSSPA(α, β) every T rounds would approach zero as $T \to \infty$ and $k \to \infty$.

D Summary

D.1 Simulating the Optimal Strategy

AddLayer$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \mathrm{D}^{\mathrm{OPT}}_{t-1})$:

(1) For $1 \le \ell \le n_t = \infty$.
  (a) DrawAdv$(\alpha, \mathrm{D}^{\mathrm{OPT}}_{t-1})$:
    - Sample $\vec{r}_{-0}$: Draw $k = \infty$ rewards $r_1, r_2, \ldots, r_k$ i.i.d from $\mathrm{D}^{\mathrm{OPT}}_{t-1}$.
    - Sample $\vec{c}_{-0}$: Draw $k = \infty$ scores $c_1, c_2, \ldots, c_k$ of adversarial wallet as follows. Draw $c_1 \leftarrow \exp(\alpha)$ and $c_{i+1} \leftarrow c_i + \exp(\alpha)$ (a fresh sample for each i) for $1 \le i \le k-1$. For convenience, set $c_{k+1} = \infty$.
    - Return $(\vec{r}_{-0}, \vec{c}_{-0})$.
  (b) sample$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi^{\mathrm{OPT}}, \mathrm{D}^{\mathrm{OPT}}_{t-1})$:
  Return sample s_ℓ equal to

$$\mathbb{E}_{c_0\sim\exp(\beta(1-\alpha)), r_0\sim \mathrm{D}^{\mathrm{OPT}}_{t-1}}\Big[\max_{0\le i\le i^*(\vec{c})}\{e^{-c_i\cdot(1-\beta)\cdot(1-\alpha)}\,(r_i + \mathbb{1}(i\neq 0))\}\cdot \mathbb{1}(i^*(\vec{c})\neq 0)\Big] - \lambda$$

(2) Return $\mathrm{D}^{\mathrm{OPT}}_t$ to be the uniform distribution over $\{s_1, s_2, \ldots, s_{n_t}\}$.

Simulate$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}})$:

(1) Initialize $\mathrm{D}^{\mathrm{OPT}}_0$ to be the point-mass distribution at 0.
(2) For $1 \le t \le T = \infty$:
  (a) $\mathrm{D}^{\mathrm{OPT}}_t$ = AddLayer$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}, \mathrm{D}^{\mathrm{OPT}}_{t-1})$.
(3) Return $\mathbb{E}_{s\sim \mathrm{D}^{\mathrm{OPT}}_T}[s]$.

D.2 Summary of Simulation After Truncating to a Finite Number of Rounds and Adversarial Wallets

AddLayer$(\alpha, \beta, \lambda, k, \pi^{\mathrm{OPT}}_{T,k}, \mathrm{D}^{\mathrm{OPT}}_{t-1,k})$:

(1) For $1 \le \ell \le n_t = \infty$.
  (a) DrawAdv$(\alpha, \mathrm{D}^{\mathrm{OPT}}_{t-1,k})$:
    - Sample $\vec{r}_{-0}$: Draw k rewards $r_1, r_2, \ldots, r_k$ i.i.d from $\mathrm{D}^{\mathrm{OPT}}_{t-1}$.
    - Sample $\vec{c}_{-0}$: Draw k scores $c_1, c_2, \ldots, c_k$ of adversarial wallet as follows. Draw $c_1 \leftarrow \exp(\alpha)$ and $c_{i+1} \leftarrow c_i + \exp(\alpha)$ (a fresh sample for each i) for $1 \le i \le k-1$. For convenience, set $c_{k+1} = \infty$.
    - Return $(\vec{r}_{-0}, \vec{c}_{-0})$.
  (b) sample$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi^{\mathrm{OPT}}_{T,k}, \mathrm{D}^{\mathrm{OPT}}_{t-1,k})$: Return sample s_ℓ
(2) Return $\hat{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$ to be the uniform distribution over $\{s_1, s_2, \ldots, s_{n_t}\}$.

TruncatedSimulate$(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k})$:

(1) Initialize $\mathrm{D}^{\mathrm{OPT}}_{0,k}$ to be the point-mass distribution at 0.
(2) For $1 \le t \le T$:
  (a) $\mathrm{D}^{\mathrm{OPT}}_{t,k} = \mathrm{AddLayer}(\alpha, \beta, \lambda, k, \pi^{\mathrm{OPT}}_{T,k}, \mathrm{D}^{\mathrm{OPT}}_{t-1,k})$.
(3) Return $\mathbb{E}_{s \sim \mathrm{D}^{\mathrm{OPT}}_{T,k}}[s]$.

D.3 Summary of Simulation After Truncating to a Finite Number of Rounds and Adversarial Wallets

FiniteSampleAddLayer$(\alpha, \beta, \lambda, k, t, n, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, \hat{\mathrm{D}}^{\mathrm{OPT}}_{t-1,k})$:

(1) For $1 \le \ell \le n$.
  (a) DrawAdv$(\alpha, \hat{\mathrm{D}}^{\mathrm{OPT}}_{t-1,k})$.
  (b) sample$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi^{\mathrm{OPT}}_{T,k}, \hat{\mathrm{D}}^{\mathrm{OPT}}_{t-1,k})$: Return sample s_ℓ
(2) $\tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$ be the uniform distribution over $(s_\ell)_{1 \le \ell \le n}$ (in descending order)
(3) Inflate while computing the upper bound and deflate while computing the lower bound.
  - Deflate$(n, \gamma, \tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}, t)$:
    (a) Delete the largest $n \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$ samples from $\tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$
    (b) Append $n \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$ copies of $-\lambda$ to $\tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$
  - Inflate$(n, \gamma, \omega, \tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}, t)$:
    (a) Delete the smallest $n \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$ samples from $\tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$
    (b) Append ωn copies of $t\,(1-\lambda)$ to $\tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$
    (c) For $1 \le \ell < \frac{n}{\omega n} \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$:
      – Append ωn copies of s_ℓ
(4) Return $\hat{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$ to be the uniform distribution over $\{s_1, s_2, \ldots, s_{n_t}\}$.

TruncatedSimulate$(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega)$:

(1) Initialize $\hat{\mathrm{D}}^{\mathrm{OPT}}_{0,k}$ to be the point-mass distribution at 0.
(2) For $1 \le t \le T$:
  (a) $\hat{\mathrm{D}}^{\mathrm{OPT}}_{t,k} = \mathrm{FiniteSampleAddLayer}(\alpha, \beta, \lambda, k, t, n, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, \hat{\mathrm{D}}^{\mathrm{OPT}}_{t-1,k})$.
(3) Return $\mathbb{E}_{s \sim \hat{\mathrm{D}}^{\mathrm{OPT}}_{T,k}}[s]$.

D.4 A Summary of the Simulation

FiniteSampleAddLayer$(\alpha, \beta, \lambda, k, t, n, \pi^{\text{OPT}}_{T,k}, \gamma, \omega, \epsilon, \eta, \hat{\text{D}}^{\text{OPT}}_{t-1,k})$:

(1) Precompute$(\hat{\text{D}}^{\text{OPT}}_{t-1,k}, \epsilon, \eta, t)$:

(a) Construct the pdf d of $\hat{\text{D}}^{\text{OPT}}_{t-1,k}$ up to a discretization error ϵ.

(b) Construct the cdf $\hat{\text{D}}^{\text{OPT}}_{t-1,k}$ recursively using the following recursion up to a discretization error ϵ.

$$\hat{\text{D}}^{\text{OPT}}_{t-1,k}(\theta-\epsilon) = Pr(r_0 \le \theta-\epsilon) = \hat{\text{D}}^{\text{OPT}}_{t-1,k}(\theta) - d(\theta)$$

(c) Construct the right tail of the expectation $E(\theta) = \mathbb{E}_{r_0\sim D}[r_0 \times \mathbb{1}(r_0 > \theta)]$ recursively by

$$E(\theta-\epsilon) = E(\theta) + (\theta-\epsilon) \times d(\theta-\epsilon)$$

(d) Compute $E_{\max}(\theta) = \theta\hat{\text{D}}^{\text{OPT}}_{t-1,k} + E(\theta)$ for all $-t\lambda \le \theta \le t(1-\lambda)$ up to a discretization error ϵ

(e) If $\beta \neq 1, 0$: For $-t\lambda \le g(i^*, \vec{c}_{-0}, \vec{r}_{-0}) \le t(1-\lambda)$ in steps of ϵ:

(i) For $0 \le c \le 1$ in steps of η

- $G(c, g(i^*, \vec{c}_{-0}, \vec{r}_{-0})) = \eta \times \sum_{\zeta=0, \text{ in steps of } \eta}^{c} \zeta^{\frac{1-\beta}{\beta}} E_{\max}(\frac{g(i^*, \vec{c}_{-0}, \vec{r}_{-0})}{\zeta^{\frac{1-\beta}{\beta}}})$

(2) For $1 \le \ell \le n$.

(a) DrawAdv$(\alpha, \hat{\text{D}}^{\text{OPT}}_{t-1,k})$:

- Sample $\vec{r}_{-0}$: Draw k rewards $r_1, r_2, \ldots, r_k$ i.i.d from $\text{D}^{\text{OPT}}_{t-1}$.
- Sample $\vec{c}_{-0}$: Draw k scores $c_1, c_2, \ldots, c_k$ of adversarial wallet as follows. Draw $c_1 \leftarrow \exp(\alpha)$ and $c_{i+1} \leftarrow c_i + \exp(\alpha)$ (a fresh sample for each i) for $1 \le i \le k-1$. For convenience, set $c_{k+1} = \infty$.
- Return $(\vec{r}_{-0}, \vec{c}_{-0})$.

(b) SampleFromPrecompute$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi^{\text{OPT}}_{T,k}, G, \hat{\text{D}}^{\text{OPT}}_{t-1,k})$:

(i) For $1 \le i^* \le k$:

- Compute $g(i^*, \vec{c}_{-0}, \vec{r}_{-0}) = \max_{1\le i\le i^*} e^{-c_i\cdot(1-\beta)(1-\alpha)}(1+r_i)$

(ii) If $\beta = 0$:

- return $s_\ell = g(k, \vec{c}_{-0}, \vec{r}_{-0}) - \lambda$

(iii) Else if $\beta = 1$:

- For $1 \le i^* \le k$: $f(i^*, \vec{c}_{-0}, \vec{r}_{-0}) = E_{\max}(g(i^*, \vec{c}_{-0}, \vec{r}_{-0}))\left[e^{-c_{i^*}(1-\alpha)} - e^{-c_{i^*+1}(1-\alpha)}\right]$
- return $s_\ell = \sum_{i^*=1}^{k} f(i^*, \vec{c}_{-0}, \vec{r}_{-0}) - \lambda$

(iv) $\beta \neq 0, 1$:

- For $1 \le i^* \le k$: $f(i^*, \vec{c}_{-0}, \vec{r}_{-0}) = G(c_{i^*+1}, g(i^*, \vec{c}_{-0}, \vec{r}_{-0})) - G(c_{i^*}, g(i^*, \vec{c}_{-0}, \vec{r}_{-0}))$
- return $s_\ell = \sum_{i^*=1}^{k} f(i^*, \vec{c}_{-0}, \vec{r}_{-0}) - \lambda$

(3) $\tilde{\text{D}}^{\text{OPT}}_{t,k}$ be the uniform distribution over $(s_\ell)_{1\le\ell\le n}$ (in descending order)

(4) Inflate while computing the upper bound and deflate while computing the lower bound.

- Deflate$(n, \gamma, \tilde{\text{D}}^{\text{OPT}}_{t,k}, t)$:

(a) Delete the largest $n \cdot \sqrt{\frac{\ln\gamma^{-1}}{2n}}$ samples from $\tilde{\text{D}}^{\text{OPT}}_{t,k}$

(b) Append $n \cdot \sqrt{\frac{\ln\gamma^{-1}}{2n}}$ copies of $-\lambda$ to $\tilde{\text{D}}^{\text{OPT}}_{t,k}$

- Inflate$(n, \gamma, \omega, \tilde{\text{D}}^{\text{OPT}}_{t,k}, t)$:

(a) Delete the smallest $n \cdot \sqrt{\frac{\ln\gamma^{-1}}{2n}}$ samples from $\tilde{\text{D}}^{\text{OPT}}_{t,k}$

(b) Append ωn copies of $t\ (1-\lambda)$ to $\tilde{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$

(c) For $1 \le \ell < \frac{n}{\omega\, n} \cdot \sqrt{\frac{\ln \gamma^{-1}}{2n}}$:
- Append ωn copies of s_ℓ

(5) Return $\hat{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$ to be the uniform distribution over $\{s_1, s_2, \ldots, s_{n_t}\}$.

TruncatedSimulate$(\alpha, \beta, \lambda, T, k, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, \epsilon, \eta)$:

(1) Initialize $\hat{\mathrm{D}}^{\mathrm{OPT}}_{0,k}$ to be the point-mass distribution at 0.
(2) For $1 \le t \le T$:
(a) $\hat{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$ = FiniteSampleAddLayer$(\alpha, \beta, \lambda, k, t, n, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, , \epsilon, \eta, \hat{\mathrm{D}}^{\mathrm{OPT}}_{t-1,k})$.
(3) Return $\mathbb{E}_{s \sim \hat{\mathrm{D}}^{\mathrm{OPT}}_{T,k}}[s]$.

D.5 A Summary of Notations and Functions in the Simulation

Notation	Description
α	The fraction of stake held by the adversary
β	The fraction of honest stake held by B
λ	The per-round entry fee the adversary has to pay to participate in LinearCSSPA(α, β, λ)
T	We simulate LinearCSSPA$(\alpha, \beta, \lambda, T)$
k	We simulate a k-scored adversary
$\pi^{\mathrm{OPT}}_{T,k}$	The optimal adversarial strategy in LinearCSSPA$(\alpha, \beta, \lambda, T)$ for a k-scored adversary
$\hat{\mathrm{D}}^{\mathrm{OPT}}_{t,k}$	The estimated distribution of optimal rewards in LinearCSSPA$(\alpha, \beta, \lambda, t)$ for a k-scored adversary
γ	Probability of estimation error from the DKW inequality in inflate/deflate
ω	A small quantile gets duplicated ωn times while inflating
ϵ	Discretization parameter of the reward distributions
η	Discretization parameter to compute Riemann sums

Table 1. A summary of notations

Function	Description
AddLayer$(\alpha, \beta, \lambda, \pi, D)$	Given a distribution D of rewards achieved by playing π in the last $t-1$ rounds, AddLayer$(\alpha, \beta, \lambda, \pi, D)$ computes the distribution of rewards won by playing π in the last t rounds of LinearCSSPA$(\alpha, \beta, \lambda, t)$
DrawAdv(α, D)	Given the reward distribution D, DrawAdv(α, D) samples rewards and scores for adversarial wallets
sample$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi, D)$	Conditional on the rewards and scores of the adversary's wallets, sample$(\alpha, \beta, \lambda, \vec{c}_{-0}, \vec{r}_{-0}, \pi, D)$ computes the reward of the adversary, in expectation over B and C's scores and the reward from letting B win.
Inflate$(n, \gamma, \omega, \mathrm{D}, t)$	For an input distribution D, Inflate$(n, \gamma, \mathrm{D}, t)$ deletes the smallest samples and replaces them with samples corresponding to a small quantile
Deflate$(n, \gamma, \mathrm{D}, t)$	For an input distribution D, Deflate$(n, \gamma, \mathrm{D}, t)$ deletes the largest samples and replaces them with the infimum of the distribution
Precompute$(\mathrm{D}, \epsilon, \eta, t)$	For an input distribution D, Precompute(D, ϵ, η, t) sets up the pre-computations required to speed up the sampling procedure
Simulate$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}_{T,k})$	Executes T iterations of AddLayer$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}_{T,k})$ and returns $\mathbb{E}_{s\sim \hat{\mathrm{D}}^{\mathrm{OPT}}_{T,k}}[s]$

Table 2. A summary of functions used in the simulation. Functions with similar names across different variants of the simulation have similar functions. For example, AddLayer$(\alpha, \beta, \lambda, \pi^{\mathrm{OPT}}_{T,k}, D)$ and FiniteSampleAddLayer$(\alpha, \beta, \lambda, k, t, n, \pi^{\mathrm{OPT}}_{T,k}, \gamma, \omega, D)$ have the sample functionality (adding an extra layer in the simulation), but in different variants.

E Results

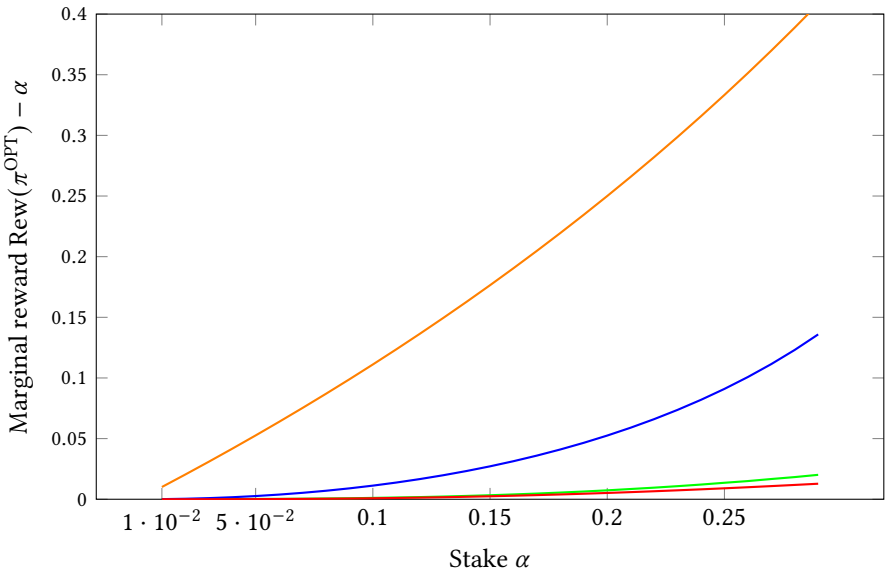

Fig. 1. Marginal reward vs adversarial stake. Legend: orange–upper bound from Ferreira et al., 2022; blue–tight upper bound for the omniscient adversary; green– un-inflated simulated upper bound for $\beta = 1$; red–reward from the 1-lookahead strategy in Ferreira et al., 2022.

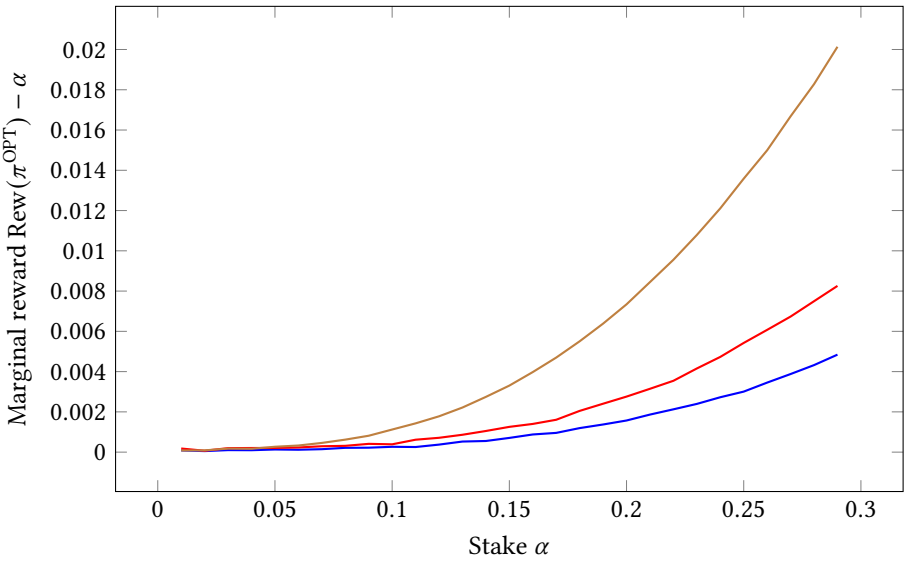

Fig. 2. Marginal reward vs adversarial stake. Legend: brown– un-inflated simulated upper bounds for $\beta = 1$; red– un-inflated simulated upper bounds for $\beta = 0.5$; blue– un-inflated simulated upper bounds for $\beta = 0$.

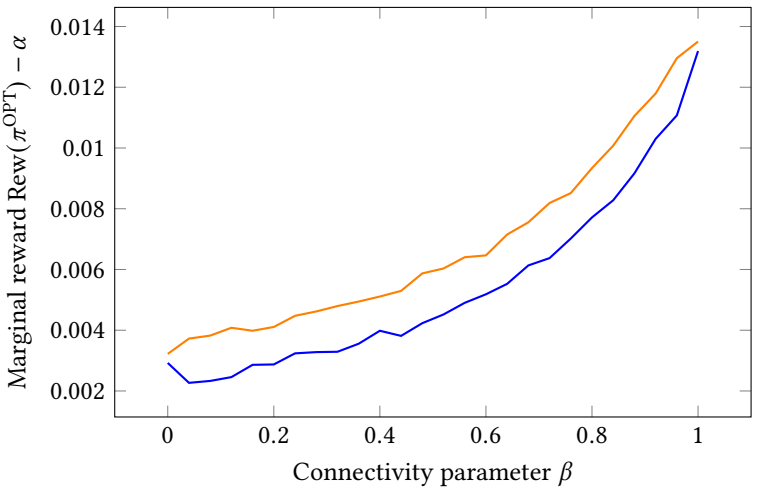

Fig. 3. Marginal reward vs network connectivity. Legend: orange– un-inflated simulated upper bound for $\alpha = 0.25$, blue– un-deflated simulated lower bound for $\alpha = 0.25$