

On Incentive Compatible Role-based Reward Distribution in Algorand

Mehdi Fooladgar^{†*}, Mohammad Hossein Manshaei^{†‡*}, Murtuza Jadliwala[◇], and Mohammad Ashiqur Rahman[‡]

[†]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran

[◇]Department of Computer Science, University of Texas at San Antonio, USA

[‡]Department of Electrical and Computer Engineering, Florida International University, USA

Abstract—Algorand is a recent, open-source public or permissionless blockchain system that employs a novel proof-of-stake Byzantine consensus protocol to efficiently scale the distributed transaction agreement problem to billions of users. Despite its promise, one relatively under-studied aspect of this protocol has been the incentive compatibility of its reward sharing approach, without which cooperation among rational network users cannot be guaranteed, resulting in protocol failure. This paper is the first attempt to address this problem. By carefully modeling the participation costs and rewards received within a strategic interaction scenario in Algorand, we first show that even a small number of non-participating users (due to insufficiency of the expected incentives) can result in the network failing to append new transaction blocks. We further show that this effect, which was observed in simulations, can be formalized by means of a game-theoretic model that realistically captures the strategic interactions between users in Algorand. Specifically, we formally prove that mutual cooperation under the currently proposed reward sharing approach in Algorand is not a Nash equilibrium. To remedy this, we propose a novel reward sharing approach for Algorand and formally show that it is incentive-compatible, i.e., it can guarantee cooperation within a group of selfish users. Extensive numerical and Algorand simulation results further confirm our analytical findings. Moreover, these results show that for a given distribution of stakes in the network, our reward sharing approach can guarantee cooperation with a significantly smaller reward per round.

Index Terms—Blockchain, Algorand, Incentive Compatibility, Game Theory, Reward Sharing.

I. INTRODUCTION

A *blockchain* is an immutable distributed database that records a time-sequenced history of facts called transactions. This record is maintained by constructing consistent copies of the cryptographic hash-chain of transaction blocks in a distributed fashion. One key aspect of any blockchain protocol is the *consensus* algorithm which enables agreement among a distributed network of autonomous *nodes* or *users*¹ (a.k.a. *miners* in certain protocols) on the state of the blockchain, under the assumption that a fraction of them could be malicious or faulty. Blockchains could be further categorized as *permissioned* or *permissionless* depending on whether a trusted infrastructure exists or not to establish verifiable identities for network nodes. In *Bitcoin* [1], a popular permissionless blockchain protocol, consensus is achieved by the network selecting a *leader* in an unbiased fashion once every 10 minutes

on an average (called a *round*). The selected leader gets the right to commit or append a new block onto the blockchain. The network then implicitly accepts this block by adding on top of it or reject it by appending on some other block. Bitcoin uses a *Proof-of-Work* (PoW) mechanism to select the leader in each round, where nodes or miners compete with each other by solving a hash puzzle. The node that wins this competition gets the right to propose the next block. As PoW involves significant computation, Bitcoin includes a reward mechanism to incentivize miners. Besides Bitcoin, several other permissionless systems (e.g., *Ethereum* [2] and other altcoins [3]) also employ a PoW-based consensus algorithm and a reward model to ensure honest participation.

Bitcoin's PoW-based consensus mechanism, however, has several shortcomings. In particular, the hash puzzle-based PoW approach is wasteful in terms of energy [4], it does not prevent forking and results in mining centralization [5]. Also, it does not scale well with the number of transactions and network users [6]. Several platform-specific efforts, such as BIP 102 [7] and Bitcoin-NG [8], have been proposed to improve Bitcoin's transaction throughput. Alternatively, platform-agnostic solutions aimed to improve the scalability-related shortcomings of PoW-based consensus by employing a committee or sharding approach [9], payment networks [10], [11], and side-chains [12] have also been proposed. Several other approaches have tried to either improve the existing version of PoW [13] or have proposed alternatives such as Proof-of-Stake (PoS) [14]–[18], Proof-of-Burn (PoB) [19], Proof-of-Elapsed Time (PoET) [20] and Proof-of-Personhood (PoP) [21].

Of all these improvements, the Algorand protocol [18], [22] has garnered the most attention within the permissionless blockchain community, primarily because of its innovative PoS-based consensus protocol that is not only computationally (and energy) efficient, but also provides strong security guarantees against forking in a network comprising of faulty and malicious users or nodes. Algorand eliminates the possibility of hash power centralization by removing the difference between normal network users and miners and scales pretty well. In fact, Algorand can commit about 750 MBytes of transactions per hour, which is 125 times of Bitcoin's throughput [22]. These security and performance guarantees of Algorand's consensus design has resulted in a lot of optimism within the blockchain community. However, one critical issue has not received much attention, if any: *does the currently proposed reward distribution approach in Algorand promote*

*M. Fooladgar and M. H. Manshaei are equally contributing authors.

¹The term users and nodes are used interchangeably. Typically, *users* control *nodes* which are computational systems that are part of the Algorand peer-to-peer network and execute the reference software.

participation or cooperation among rational users to complete all the required protocol tasks?

Since the inception of Bitcoin, a significant effort has been spent by the research community towards understanding the incentive-compatibility of its reward distribution approach [23]–[25], towards characterizing the strategic behavior of rational miners in mining pools [24], [26]–[30], and towards designing new incentive-compatible PoW-based cryptocurrencies [4] and scalability solutions [31], [32]. However, no such analysis for Algorand exists yet, and this paper attempts to fill this research gap. We make the first attempt to formally analyze Algorand’s reward distribution strategy by employing well-established game-theoretic tools and techniques. More specifically, by modeling a single round of Algorand’s consensus protocol as a single-stage non-cooperative multi-player game, we show that without an efficient reward sharing protocol, rational nodes would have more incentive to deviate from cooperation and behave selfishly. To overcome this problem, we propose a new reward distribution approach for Algorand, which, in addition to the stake possessed by users, considers their roles during consensus or Byzantine agreement for distributing the per-round rewards. We further show that our proposed role-based reward distribution approach is able to converge to a Nash equilibrium (NE) where a certain subset of nodes will cooperate. We conduct an extensive empirical evaluation of the proposed reward distribution approach using both numerical and Algorand protocol simulations. Our evaluations further confirm our analytical results by showing that we can distribute significantly smaller rewards among users while enforcing cooperation in Algorand. The Algorand Foundation can use our results to keep track of the network state and adapt the distributed rewards accordingly. To the best of our knowledge, this paper is the first to provide a systematic analysis of incentive design in Algorand.

II. ALGORAND SYSTEM MODEL

In this section, we first summarize the Algorand protocol. This description is intended to provide readers with the main concepts of Algorand. Interested readers are referred to [1] and [18] for more technical details on Bitcoin and Algorand.

A. Contrasting Consensus in PoW with PoS Blockchains

We begin by first contrasting PoW consensus approach of Bitcoin with the more recent PoS approach of Algorand. In this direction, we highlight some of the most significant shortcomings of Bitcoin’s PoW approach and discuss how they are overcome by Algorand’s PoS consensus.

- PoW wastes significant amount of computation, and by relation, the electrical energy used to achieve it. PoW schemes also assume that a majority of the nodes contributing to the network’s hash or computational power are honest, i.e., at least 51% of the network’s hash power comes from honest users.

- PoW-based consensus eventually leads to concentration or centralization of power, where entities in the network eventually monopolize computational power to control new block addition (e.g., Bitcoin Mining Pools) [27]–[29].

- PoW allows the possibility of forking, where two different hash chains could reach the same length and neither one supersedes the other [24], [26]. Efforts to mitigate the impact of forking in existing solutions have resulted in the block inter-arrival and transaction confirmation times to become impractically high (e.g., current Bitcoin block inter-arrival time is 10 minutes while transaction confirmation time is 1 hour). As a result, current PoW blockchain solutions do not scale well with the number of transactions and users. To overcome these shortcomings, Algorand proposes a novel PoS based consensus protocol. Similar to Bitcoin, Algorand is fully decentralized and maintains a public, immutable ledger of transactions by reaching consensus on the order of transactions in the ledger. However in Algorand all users are “equal”, i.e., there is no distinction between users (miners) who can add new blocks and those who just create and receive transactions. Moreover, as each user/node runs the same computationally efficient function to achieve consensus (as opposed to PoW-based systems where users compete for the right to add the next block), Algorand does not waste computations, and thus electricity. Lastly, the design of Algorand’s consensus protocol guarantees that there is no forking with an overwhelmingly high probability. A side-effect of this is that Algorand scales extremely well with the number of users/nodes and transactions, compared to classical PoW-based systems [22].

B. Summary of Algorand

Next, we summarize the creation, distribution and agreement of transaction blocks in Algorand, as shown in Fig. 1.

- 1) *Assumed Adversary Model*: In addition to standard cryptographic assumptions, Algorand assumes that *honest users* always run bug-free reference software and follow all protocol steps. As is standard in PoS systems, Algorand assumes that the fraction of money held by honest users is above some threshold h (a constant greater than $\frac{2}{3}$). An adversary can participate in Algorand by creating multiple sybil nodes/users and owning some money or stake in the system. An adversary in Algorand can arbitrarily corrupt honest users, provided that the amount of money held by honest, non-compromised users remain above h . However, an adversary cannot compromise the keys of honest non-compromised users. Algorand assumes that most honest users receive messages sent by most other honest users within some known time bound in order to continue to make progress on adding blocks to the blockchain (i.e., *liveness goal*). This is the *strong synchrony* assumption. Algorand can achieve consensus or agreement on blocks (i.e., *safety goal*) even if the network is *asynchronous* (or controlled by the adversary) for a long but bounded period of time, provided it is strongly synchronous for a period of time after.

- 2) *Network and Communication Protocol*: The Algorand network is a *peer-to-peer* network of honest and faulty or malicious nodes, where each node is represented by a public/private key pair (see Fig. 1-(a)). The number of malicious or faulty nodes is bounded by the honest stake ownership condition outlined earlier. Nodes in the network communicate in a peer-to-peer fashion using unique TCP connections. Commu-

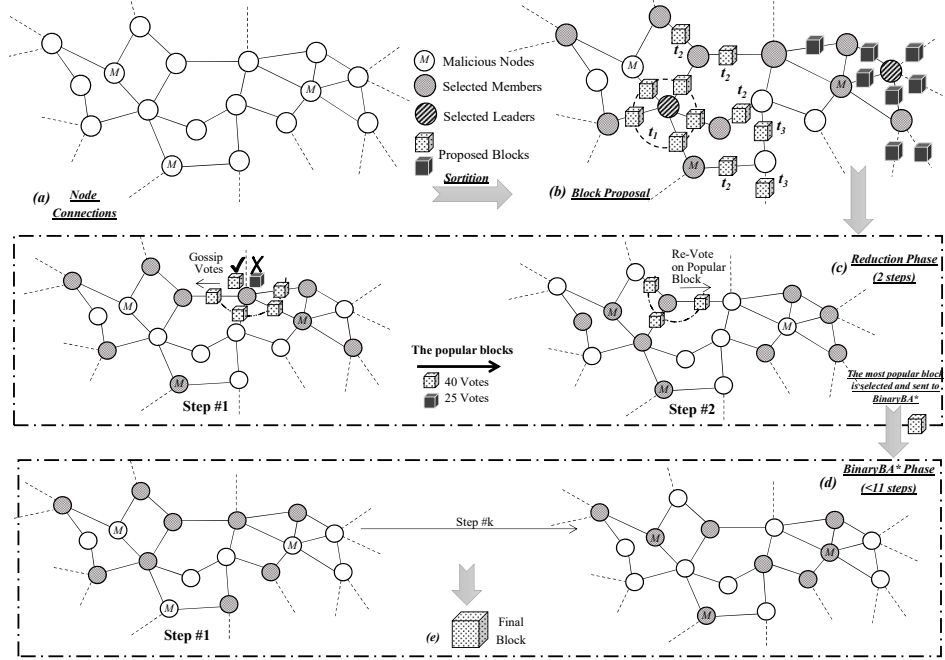


Fig. 1: Algorand System Model. (a) Algorand nodes build a peer-to-peer network. (b) Each user executes cryptographic *sortition* algorithm to determine his role. At the time t_1 each leader sends his proposed block to all first-hop neighbors. Consequently, all nodes forward their received blocks to their neighbors. (c) The Reduction phase reduces consensus problem to an agreement on one or two options. In *step#1*, committee members vote for the highest priority block they received. In *step#2*, new committee members count last step votes and re-transmit popular blocks as their vote in the second step. (d) BinaryBA \star phase reach agreement on a proposed block from the reduction phase or an empty block. This phase can be followed up (on average for 11 steps, i.e., $k = 11$) to ensure that each node agrees on the same consensus. In each step committee members votes for their observation of the reduction phase. (e) the final block would be added to the chain.

nications happen by means of a standard *gossip protocol* where each node broadcasts his message to all his peers, who in turn relay it to their neighbors. The Algorand communication protocol defines four types of *messages*:

1. Transaction: This message transfers a certain amount of *Algos* (currency unit in Algorand) from a sender to a receiver (identified with their public-keys) and signed by the sender (with its private key), which is referred to as a transaction. Multiple transactions are organized into a *block*. An Algorand block is either a set of transactions or an empty block. In addition, each block contains a pre-determined random seed (described later) and the hash of the previous consensus or agreed block it is extending.

2. Voting: This message contains a signed vote by the sender along with the *sortition proof* (described below). Each sortition proof is associated with a priority value which is computed in a deterministic fashion.

3. Block proposal: This message contains a new block (to be added), along with the signed hash of the block and a sortition proof establishing the role of the sender as a *leader*.

4. Credential: This message contains the sortition proof of the leader, which is generally broadcast at the beginning of each round by the leader using the gossip protocol. Peer nodes

employ the priority values extracted from sortition proofs in the credential messages to avoid relaying block proposals with low priorities. This helps preventing congestion in the network due to a significantly large number of block proposals.

3) Consensus or Byzantine Agreement: Algorand's consensus or Byzantine Agreement (denoted as BA \star) protocol operates in *rounds*, where in each round all nodes attempt to reach agreement on a new block of transactions. At the beginning of a round, each node employs cryptographic sortition to privately determine if it is a block proposer or leader, i.e., has the right to propose a block for that round. To propose a block, each leader node assembles the pending and validated transactions inside a block proposal and gossips it together with its sortition proof of being elected a leader (see Fig. 1-(b)). After block proposals are broadcast, each node collects incoming block proposals for a fixed duration, selecting and retaining the one valid block with the highest priority sortition proof.

Each user then (asynchronously) initializes the BA \star protocol with the highest-priority block they have received. The BA \star protocol enables all nodes in the network to reach consensus on a single block. The BA \star protocol comprises of two sequential phases, the *Reduction phase* (Fig. 1-(c)) followed by the *BinaryBA \star phase* (Fig. 1-(d)), with each phase

consisting of a sequence of *steps*. At a high level, in each step first a random or unpredictable group of nodes referred to as *committee members* is elected. Then the elected committee members *vote* on a specific block, based on votes received from the previous step, and broadcast their new votes in voting messages. Readers should recall that all voting messages also contain a sortition proof which proves the validity of the broadcaster as a committee member.

- *Reduction Phase* (Fig. 1-(c)): This first phase of the BA \star protocol comprises of *exactly two* steps. In the first step, each committee member votes on (the hash of) the blocks proposed for consideration. In the second step, committee members vote for the block hash that received votes over a certain threshold. If no block hash receives enough votes, committee members vote for the hash of the default empty block. Reduction phase concludes with either at most one non-empty block hash (the one that received the maximum number of votes above the threshold) or hash of an empty block (if no block hashes received enough votes). This output of the reduction phase is passed as input to the BinaryBA \star phase.

- *BinaryBA \star Phase* (Fig. 1-(d)): The goal of the BinaryBA \star Phase is to reach agreement or consensus on the majority voted non-empty block (hash) from the reduction phase or, in case there is no consensus, on the empty (default) block. In the common case, when the network is strongly synchronous and the block proposer or leader is honest, BinaryBA \star phase will start with the same block hash for most users, and will reach consensus in the first step, since most committee members vote for the same block hash value. If the network is not strongly synchronous, BinaryBA \star may return consensus on two different blocks (i.e., the block received from the reduction phase and the empty block). The outcome of the BinaryBA \star phase is used by the BA \star algorithm to arrive at either a *final* or a *tentative* consensus. Final consensus means that BA \star will not reach consensus on any other block for that round, while tentative consensus means that BA \star is unable to guarantee the safety goal in this round, either because of network asynchrony or due to a malicious block proposer.

4) *Cryptographic Sortition*: Each node in the network employs a cryptographic sortition algorithm to determine if it is selected as a leader (or block proposer) at the beginning of each round, and later, if it is selected as a committee member at the beginning of each step (of both the Reduction and BinaryBA \star phases). The sortition algorithm is implemented using *Verifiable Random Functions (VRF)* [33] which allow users to produce verifiable proofs using their private keys that can be publicly verified using the corresponding public key. Specifically, in order to generate a sortition proof for step s in round r , a user i computes $\text{sig}_i(r, s, Q^{r-1})$, where sig_i is a digital signature computed using the user i 's private key, and Q^{r-1} is a random seed (predetermined at the end of the previous round, i.e., $r - 1$). This sortition proof is included by the nodes in their block proposals and voting messages in order to prove their roles as leader and committee members, respectively. The recipients of these messages first validate the signature (using the public key) and then compute the hash of

TABLE I: List of Symbols in Algorand Analysis

Symbol	Definition
R_i	Foundation reward in round t_i
F_i	Summation of transaction fees in round t_i
P_i^F	Reward pool level in round t_i
B_i	The shared rewards in round t_i
α	Fraction of rewards shared between leaders
β	Fraction of rewards shared between committee members
γ	Fraction of rewards shared between remaining online nodes
c^{fix}	Common costs of Algorand nodes
c^L	Costs for Algorand leaders
c^M	Costs for committee members
c^K	Costs for Algorand remaining online nodes
r_i^L	Rewards per each unit of stake for a leader
r_i^M	Rewards per each unit of stake for a committee member
r_i^K	Rewards per unit of stake for a remaining online node
s_j	Stake of node $j \in \{l_j, m_j, k_j\}$; s_{l_j} is reward for leader l_j
S_L	Summation of all stakes for leaders; i.e. $S_L = \sum_{j \in L} s_{l_j}$
S_M	Summation of all stakes for committee members
S_K	Summation of all stakes for other nodes
S_N	Summation of all stakes, i.e., $S_N = S_L + S_M + S_K$
$u_{l_j}^{t_j}$	Payoff for leader l_j in round t_i
$u_{m_j}^{t_j}$	Payoff for committee member m_j in round t_i
$u_{k_j}^{t_j}$	Payoff for remaining node k_j in round t_i

the sortition proof to verify a certain sortition condition that determines the validity of the claimed role. The possibility that the condition is verified is directly proportional to the stake of the node (to which the proof belongs to) and depends on a constant role parameter fixed in the protocol. Due to space restrictions, we will not further elaborate on this and interested readers can find more technical details in [18], [22], [33].

III. PROBLEM FORMULATION AND MOTIVATION

Similar to any other permissionless blockchain protocol, Algorand must also provide enough incentives to foster cooperation among its participants, whether they are leaders, committee members, or online nodes, in order to enable effective consensus (on the set of transactions), i.e., to achieve the safety and liveness goals. In this section, we first summarize all the processing costs in the Algorand consensus protocol, followed by a discussion of how rewards could be distributed among the various protocol participants. Next, we empirically show that if rewards are not appropriately distributed, rational participants have an incentive to not cooperate (in the consensus protocol tasks), resulting in no new blocks being added. Our goal here is to highlight the need for designing an *incentive-compatible reward sharing mechanism* for achieving cooperation in Algorand. Table I presents the notations used throughout the paper.

A. Algorand Costs

From the operational details of Algorand, as outlined earlier, it is clear that each participant or user, irrespective of their role, is expected to perform some processing and communication tasks during each phase of the protocol which incurs some measurable cost, say, in terms of consumed energy. These costs for each processing task can be further quantified using monetary values (e.g., *Dollars* or *Algos*) by using the current energy costs. Below we present a brief description of each of these tasks that incur some significant cost, which are also

TABLE II: Algorand tasks and costs given the role of nodes.

Task	Symbol	Leader l_j	Committee m_j	Others k_j
Transaction Verification	c^{ve}	✓	✓	✓
Seed Generation	c^{se}	✓	✓	✓
Sortition Algorithm	c^{so}	✓	✓	✓
Verify Sortition Proof	c^{vs}	✓	✓	✓
Block Proposition	c^{bl}	✓		
Gossiping	c^{go}	✓	✓	✓
Block Selection	c^{bs}		✓	
Vote	c^{vo}		✓	
Vote Counting	c^{vc}	✓	✓	✓

summarized in Table II. We would like to stress that our goal here is not to precisely quantify the cost associated with each task - we simply argue that each of these tasks incurs a significant cost which cannot be ignored by the user. Rather than precisely quantifying each of these costs, we abstract each as a fixed parameter within our model.

Transaction Verification Cost (c^{ve}): This cost is incurred by an Algorand node to check the validity of a transaction. For each transaction validity check, the node must verify the signature and also check whether the sending user has enough *Algos* in its account for a successful transaction. A leader assembles a set of transactions into a block, and all Algorand nodes check the validity of transactions inside a block.

Seed Generation Cost (c^{se}): Algorand requires a random and publicly known seed as an input to the sortition algorithm. Thus, a new seed is published in each round of Algorand. This seed is a random number generated by VRF [33] from the previous seed value and the current round number. Also, for security reasons Algorand refreshes the seed every R rounds [22]. We parameterize the cumulative cost of generating a new seed in each round as c^{se} .

Sortition Algorithm Cost (c^{so}): As outlined in the previous section, the sortition algorithm employs a VRF function [33] to generate a membership proof which is included by leaders and committee members in their messages to prove their role (as a leader or committee member). The per-node cost of executing the sortition algorithm is parameterized as c^{so} .

Block proposition Cost (c^{bl}): The cost of assembling a set of outstanding, but valid, transactions (including the sortition proof) into a block and broadcasting it to the neighboring nodes in the network is borne by each (selected) leader node in each round. We parameterize this cost as c^{bl} .

Gossiping Cost (c^{go}): During each round, each node in the Algorand network supports the network by forwarding (gossiping) network messages, including transactions, blocks, and votes. This expected gossiping cost for each round for each node is parameterized as c^{go} .

Block Selection Cost (c^{bs}): In each round of the Algorand protocol, the sortition algorithm will select multiple (up to 70) nodes as leaders, with each leader proposing its own block. Each committee member in each round, specifically, during the reduction phase of the BinaryBA* protocol, needs to select (and vote) for the block with the highest priority. This block selection cost borne by a subset of committee members in each round, which includes the verification of sortition proofs, is

parameterized as c^{bs} .

Vote Cost (c^{vo}): Each selected committee member during each step of the BinaryBA* protocol should validate and check incoming messages (including, votes from previous steps) before submitting its own vote in that step. This cost, which also includes the cost to append the sortition proof to the outgoing vote and broadcast to neighbors, is parameterized as c^{vo} . The timeout to submit a vote is defined by Algorand and is equal to 20 *seconds*. It should be noted that c^{vo} does not include the cost of vote counting and is outlined next.

Vote Counting Cost (c^{vc}): After all committee members have submitted their votes, each Algorand node should validate voting messages by checking their sortition proofs. c^{vc} represents all associated costs of sortition proofs and signature verifications incurred when counting and tallying the votes inside each received vote message.

Given the above task-specific costs, let us now outline the overall costs incurred by each node. Each node incurs a cumulation of two types of costs: (i) a fixed cost, and (ii) a role-based cost. The fixed cost (c^{fix}) represents the required costs borne by each node irrespective of its role and is equal to $c^{ve} + c^{se} + c^{so} + c^{go} + c^{vs} + c^{vc}$. In addition to the fixed cost c^{fix} in each round, each node incurs a cost based on its role(s) (i.e., Leader, Committee Member or None) in that round and is represented as follows:

$$c^j = \begin{cases} c^{fix} + c^{bl} & j \in \mathbb{L} \\ c^{fix} + c^{bs} + c^{vo} & j \in \mathbb{M} \\ c^{fix} & j \in \mathbb{K}, \end{cases} \quad (1)$$

where \mathbb{L} , \mathbb{M} , and \mathbb{K} are the sets of leaders, committee members and all other users without particular role, in round i .

B. Reward Sharing in Algorand

Given these costs, it is clear that rational users (which we assume our users are) will fully participate in Algorand's distributed consensus protocol if and only if they have enough incentive to do it. As Algorand is a cryptocurrency, the mechanism for providing incentives is straightforward - pay users in *Algos* for their participation efforts and costs. For instance, Bitcoin has an incentive model where the winner of the PoW puzzle receives incentives/rewards in the form of block rewards and transaction fees (paid out as Bitcoins) to be engaged in the PoW and block addition process. A similar question arises in Algorand: *which users should be paid, and how much, in order to enable their continued participation in the distributed consensus process?*

Recently the Algorand Foundation, dedicated to fulfilling the global promise of blockchain technology by leveraging the Algorand protocol, has suggested a tentative version of reward sharing in their protocol [34], [35], as shown in Fig. 2. The proposed mechanism assumes creation and maintenance of two reward pools: (i) *Foundation Reward Pool*, and (ii) *Transaction Reward Pool*. These pools are nothing but public keys controlled by the Foundation. These public keys act as a central (foundation-controlled) storage where reward

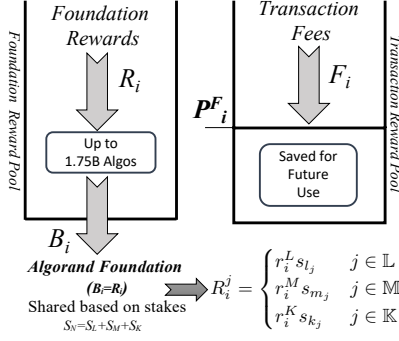


Fig. 2: Algorand Reward sharing mechanism.

distribution related funds (*Algos*) are deposited. All rewards for each round of the Algorand protocol are expected to be disbursed (or transferred) from this public key. To bootstrap the new cryptocurrency, the Algorand foundation implemented a ceiling of 1.75 billion *Algos* to be disbursed from the *Foundation Reward Pool*. Per the foundation, in each round R_i *Algos* are added to the Foundation Reward Pool until the ceiling of 1.75 billion is reached. The projected total rewards for the first 12 periods are suggested by the Algorand Foundation [34] as 10 (period 1), 13, 16, 19, 22, ..., 38, and 38 Millions of *Algos* (period 12), respectively [35]. Each reward period spans 500 thousands blocks. For example, in the first reward period, 10 millions *Algos* would be distributed, which is equal to approximately 20 *Algos* for each round, if in each round a block could be successfully added to the ledger.

The reward sharing proposal suggests that in each round this reward R_i should be distributed among Algorand users in proportion to their current system stake, irrespective of their roles (i.e., leaders or committee members). In other words, users with higher stake receive a larger portion of the allocated foundation reward R_i in each round. The transaction fees accumulated from the transactions in the added blocks during the bootstrap phase are saved or deposited into the *Transaction Fee Pool*. This pool is not planned to be used for reward disbursement until the 1.75 billion *Algos* ceiling of the Foundation Reward pool is met. In summary, currently only the *Foundation Reward Pool* is being used for the per-round reward (or incentive disbursement). Out of the R_i *Algos* disbursed to the *Foundation Reward Pool* per round i , let us assume that B_i (where, $B_i \leq R_i$) *Algos* are actually disbursed among the system users. Initially B_i is expected to be equal to R_i . Let us assume that the total value of stake in the system is S_N . Thus, $S_N = S_L + S_M + S_K$. Here, S_L , S_M , and S_K are the total stake values of leaders, committee members, and all other online nodes in round i , respectively. These values are changing in each round, but for the sake of presentation we write S_N instead of $S_N(i)$. Then, the rewards assigned to a leader node l_j in round i , R_i^j , would be $\frac{B_i s_{l_j}}{S_N}$, where s_{l_j} is the stake of leader l_j . In summary, the reward distribution is:

$$R_i^j = \begin{cases} r_i^L s_{l_j} & j \in \mathbb{L} \\ r_i^M s_{m_j} & j \in \mathbb{M} \\ r_i^K s_{k_j} & j \in \mathbb{K}, \end{cases} \quad (2)$$

where $r_i^L = r_i^M = r_i^K = r_i = \frac{B_i}{S_N}$. We now analyze if the incentives provided by this mechanism is enough to guarantee cooperation (in the consensus process) by rational nodes.

C. To be Cooperative or Not: Problem Motivation

Let us assume that an Algorand node is *cooperative* when it plays its role, i.e., performing all the assigned tasks, and consequently, accepting all the associated costs. In contrast, a *defecting* node only remains online but does not perform any of its assigned tasks, except sortition computation to join the network (i.e., paying cost c^{so}). In this case, if appropriate countermeasures (e.g., punishment mechanisms) are not deployed, the defecting nodes may end up earning rewards by simply relying on other nodes to honestly perform their tasks and not contributing anything towards the block proposal, verification, and consensus tasks. Considering this definition for cooperative and defecting behavior, we can divide Algorand node behaviors into the following four categories:

- **Honest nodes:** These nodes always cooperate. They are also altruistic and cooperate even when the reward is not more than the cost of cooperation.
- **Honest but Selfish nodes:** These nodes cooperate and defect depending on the amount of received incentives versus the cost for their actions. Hence, they are selfish and will cooperate if and only if the reward is more than the cost of cooperation.
- **Malicious nodes:** They arbitrarily cooperate or defect. In addition to this, they may inject malicious transactions and blocks, or arbitrarily compromise other nodes.
- **Faulty nodes:** These nodes are offline due to system malfunction (and not by choice) and do not contribute anything to the network operation.

In this paper, we assume that all network nodes behave in an *honest but selfish manner*. Moreover, in this preliminary work, we assume that nodes do not arbitrarily behave maliciously or become faulty. In other words, nodes make a strategic decision to cooperate (participate) or defect (not participate) solely by maximizing their own interests/incentives. They neither make any arbitrary protocol participation decision, nor maliciously modify the protocol to maximize their interests/incentives. To get an insight into the robustness of the proposed Algorand reward sharing approach against selfish (or rational) node/user behavior, we conduct preliminary simulation experiments.

Our simulator, written in Python, is based on the Algorand discrete event simulator by Deka et al. [36] and implements all Algorand protocol modules, including, Sortition, Reduction, and BinaryBA*. We are also able to simulate network delays and various synchrony conditions, as well as, customize different network parameters such as total number of nodes and the distribution of network message delays in our simulator. Within this simulation framework, we also implemented the reward sharing protocol proposed by the Algorand Foundation (described earlier), which computes a per-round reward to be shared among the nodes. We simulate each round of the Algorand block proposal and consensus protocol, as outlined in Section II, and execute the reward sharing algorithm at the end of each round to compute and distribute rewards.

We simulate the Algorand with 100 different seed numbers and let the protocol operate for 100 rounds in each simulation. In each simulation instance, we randomly select defecting nodes (i.e., honest and selfish nodes who chose to defect given their payoff) by means of a uniform distribution. We consider the total number of defecting nodes in the network in steps of 5%, 15%, and 25% of all the nodes in the Algorand network. Moreover, we distribute the stakes among all nodes with a uniform distribution between 1 to 50 *Algos*. Note that we compute trimmed mean which ignores 20% top and bottom data to compute the mean values of these 100 simulations. In our simulation each node sends the messages to 5 other nodes that are randomly selected from the network. We first analyze the impact of defecting nodes on the block creation process. The corresponding number of nodes who extracted final, tentative, or no blocks from the network messages (i.e., votes) are plotted in Fig. 3. As Fig. 3-(a) shows, even with a low defection rate of 5% the number of tentative blocks is increasing in the network. Moreover, about 7% of nodes do not receive any block. When the number of defecting nodes is increasing the Algorand network fails to inform most of the nodes about the final blocks.

For example, as shown in Fig. 3-(b), with 15% defection rate, most of the nodes do not reach any consensus on a final block after round #30. In other words, with 15% defection the network may transition to a weak synchrony or even an asynchrony state in some rounds and it prevents some nodes from receiving network messages (e.g., votes and block proposals). However, by reaching a strong synchrony state after a long period of asynchrony (i.e. weak synchrony assumption), nodes who have extracted tentative blocks can finalize their blocks. This effect has been highlighted in Fig. 3-(b) in the proximity of rounds 17 through 20. As shown in the figure, in round #17 the asynchrony of the network has caused an increase in the number of nodes that have extracted tentative blocks from the network. But in round #18, network becomes synchronous again and consequently a majority of the Algorand nodes are able to extract the finalized blocks. We also need to clarify that these defecting nodes may control more than threshold h (i.e. Algorand honest assumption as defined in Section II-B1) of stakes in the network. This happens if there are more nodes with high values of stakes in the list of defecting nodes. Defection of these nodes can amplify the network synchrony problem in the Algorand network and consequently the block creation process. Finally, the results show that even with 25% defection the network fails in the first few rounds.

In summary, the above simulation results show that without an incentive-compatible reward sharing approach that fosters cooperation, rational nodes will be inclined to defect from the block creation and consensus process resulting in an asynchrony state, thereby failing to add new blocks. In the following section, we outline a game-theoretic model to analyze the effect of defection in the Algorand network and propose a solution to prevent defection in the network.

IV. GAME MODEL AND INCENTIVE ANALYSIS

To obtain insight into the strategic behavior of nodes in Algorand, we model their interaction using a static non-cooperative game. We first focus on the interaction between nodes that are supposed to interact and create blocks in each round. Let us assume that each round i of Algorand is a static game \mathcal{G}^{Al} where network nodes are players. We assume that all strategies are hard-wired in each node. In other words, each node does not change his chosen strategy during round i of the game. They also choose their strategies simultaneously.

In our Algorand game \mathcal{G}^{Al} users must decide whether to cooperate and contribute to make a new block or not. The game \mathcal{G}^{Al} is formally defined as a triplet $(\mathbb{P}, \mathbb{S}, \mathbb{U})$, where \mathbb{P} is the set of players, \mathbb{S} is the set of strategies and \mathbb{U} is the set of payoff values. The set of players \mathbb{P} includes leaders \mathbb{L} , committee members \mathbb{M} , and all other users \mathbb{K} , i.e., $\mathbb{P} = \mathbb{L} \cup \mathbb{M} \cup \mathbb{K}$. An Algorand node can take an action (s_i) from the set $\mathbb{S} = \{C, D, O\}$, where C , D , and O represent (i) *Cooperate*, (ii) *Defect*, and (iii) *Offline*, respectively. As we discussed in previous section, cooperative nodes follow all defined tasks, while defecting nodes are only online but do not perform their assign tasks. Moreover, a node can play *offline* in round i (i.e., plays O), in which it runs sortition computing but it becomes offline and does not receive any reward. Given the above assumption, the following lemma shows that the O strategy is always strictly dominated by D strategy.

Lemma 1. *In \mathcal{G}^{Al} , strategy O is strictly dominated by playing defection (D).*

Proof. A user always obtains greater payoffs by playing D instead of O , for all possible strategy profiles of other users (i.e., opponents). In fact, a user can obtain the reward by playing D in the current version of Algorand, but its payoff would be $-c^{so}$, if it plays O . \square

Given the result in Lemma 1, we are not going to consider strategy O in our analysis as it will not be chosen by any rational player. In the following section, we present our results for the analysis of \mathcal{G}^{Al} , based on the proposed reward sharing by the Algorand Foundation [34].

A. Analysis of \mathcal{G}^{Al}

In \mathcal{G}^{Al} , we define strategy profiles $All - D$ and $All - C$, where all nodes choose to play C and D , respectively. We apply the Nash Equilibrium (NE) concept to analyze \mathcal{G}^{Al} . At the NE profile no player can unilaterally change his strategy to increase his utility. The following theorem shows the existence of an all defection strategy ($All - D$) NE for \mathcal{G}^{Al} .

Theorem 1. *In each round i of \mathcal{G}^{Al} with N players (n_L leaders, n_M committee members, and n_K remaining nodes), all-defection strategy profile ($All - D$) is a Nash equilibrium.*

Proof. Let us consider a strategy profile where all Algorand nodes defect, where there is no incurred costs such as c^L , c^M , or c^K for all nodes. Hence, the payoff for each node would

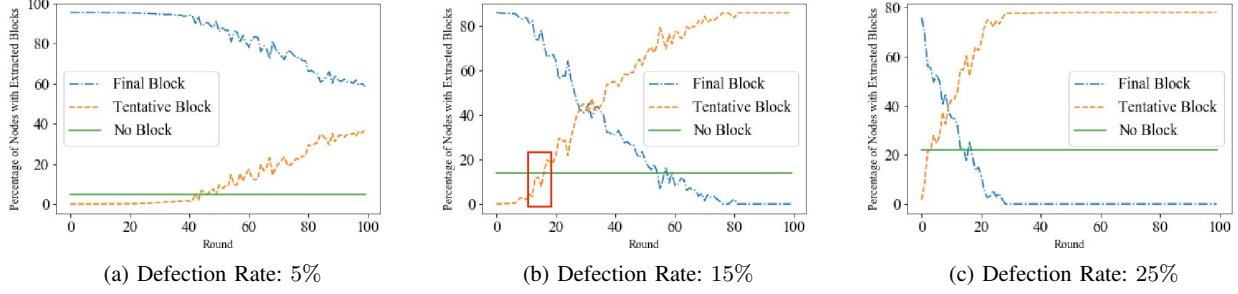


Fig. 3: The percentage of nodes who extracted the tentative and final blocks with different rate of defection. In each scenario, defecting nodes will not cooperate if the benefit is not more than the cost of cooperation for them.

be $u_i = -c^{so}$ as there is no block added to the chain and they cannot earn any *Algo*. In this case:

1) None of the Algorand leaders l_j can increase their payoff unilaterally by changing their strategies. Because, the cooperative leader can not gain any reward without the contribution of at least S_{STEP} committee members in each step of BA* protocol and S_{FINAL} members for the final committee, as discussed in Section II. In other words, the payoff of a leader who deviates from D to C would be $u_i^{l_j}(C) = -c^L$, which is always smaller than his defecting payoff (i.e., $u_i^{l_j}(D) = -c^{so}$).

2) Similarly, a cooperative committee member m_j cannot obtain any reward without the contribution of leaders and sufficient number of committee members. In this case, payoff of a committee member who has deviated is $u_i^{m_j} = -c^M$.

3) With similar justification, we can prove that all other online nodes k_j will not be able to increase their payoffs unilaterally by deviating from D to C , as its payoff would be decreased to $u_i^{k_j}(C) = -c^K$. Hence, *All-D* strategy profile is a NE in \mathcal{G}^{Al} . \square

In fact, in such distributed protocols one would like to enforce *All-C* strategy profiles as a Nash equilibrium. But the following theorem shows that the current Algorand incentive mechanism cannot enforce cooperation among all nodes.

Theorem 2. *In each round i of \mathcal{G}^{Al} with N players ($n_L > 1$ leaders, n_M committee members, and n_K remaining nodes), if rewards are shared solely based on the current values of the stakes as shown in Equation (2), i.e., the proposed Algorand Foundation mechanism, we cannot establish all-cooperation strategy profile (*All-C*) as a Nash equilibrium strategy profile.*

Proof. Let us assume that all Algorand nodes have already cooperated and paid the costs c^L , c^M , or c^K as leader, committee member, or other online nodes. Given Equations (1) and (2), the pay of of node j in round i is

$$u_i^j(C) = \begin{cases} r_i s_{l_j} - c^L & \text{Leader } l_j \\ r_i s_{m_j} - c^M & \text{Committee member } m_j \\ r_i s_{k_j} - c^K & \text{Online node } k_j. \end{cases} \quad (3)$$

Consequently, by comparing cooperative and defecting payoffs for each node similar to Theorem 1, and if we assume that

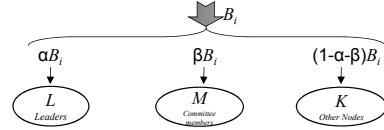


Fig. 4: Our proposed model shares the reward according to the roles of nodes as well as their stakes.

they deviate unilaterally, we can conclude that all nodes have incentive to deviate and increase their payoffs. Then, *All-C* strategy profile can never be an NE in \mathcal{G}^{Al} . \square

The results presented in Theorems 1 and 2 show that we cannot enforce cooperation in the current reward sharing approach for Algorand. In fact, if all users are rational they will try to only play D and the system remains in *All-D* Nash equilibrium. Intuitively, this occurs because different nodes receive the same (portion of the) rewards despite bearing different costs depending on their role in the round. To overcome this, we propose a novel incentive-compatible reward sharing mechanism that shares rewards by considering node roles.

B. Proposed Reward Sharing Mechanism

As shown in Fig. 4, we propose that the reward B_i must be divided into three parts, and then be distributed among the nodes given their stakes. In our model, we assume that αB_i , βB_i , and γB_i must be distributed among leaders, committee members, and other online nodes, where $\alpha \in (0, 1)$, $\beta \in (0, 1)$, and $\gamma \in (0, 1)$ should be chosen by the designer, such that Algorand Foundation can enforce the cooperation among users. Note that $\alpha + \beta + \gamma = 1$. Given this approach, one can provide different incentives to different types of users. Hence, the payoff would be calculated by

$$u_i^j(C) = \begin{cases} r_i^L s_{l_j} - c^L & \text{Leader } l_j \\ r_i^M s_{m_j} - c^M & \text{Committee member } m_j \\ r_i^K s_{k_j} - c^K & \text{Online node } k_j, \end{cases} \quad (4)$$

where $r_i^L = \frac{\alpha B_i}{S_L}$, $r_i^M = \frac{\beta B_i}{S_M}$, and $r_i^K = \frac{\gamma B_i}{S_K}$. Let us now define and analyze a new game \mathcal{G}^{Al+} , in which the payoffs are calculated by Equation (4).

C. Analysis of \mathcal{G}^{Al+}

In this section, we will first determine conditions under which user cooperation can be fostered. Then we will investigate the existence of NE in this game. The following lemma presents conditions under which network leaders and committee members have enough incentive to cooperate.

Lemma 2. *Considering \mathcal{G}^{Al+} with N players ($n_L > 1$ leaders, n_M committee members, and n_K remaining nodes), where reward B_i shares with ratios α , β , and $\gamma = 1 - \alpha - \beta$ between leaders, committee members, and remaining nodes. A selfish leader l_j or committee member m_j cannot deviate from C strategy unilaterally to increase its payoff, if and only if:*

$$B_i > \max\left\{\frac{c^L - c^{so}}{\left(\frac{\alpha}{S_L} - \frac{\gamma}{S_K + s_{l_j}^*}\right)s_{l_j}^*}, \frac{c^M - c^{so}}{\left(\frac{\beta}{S_M} - \frac{\gamma}{S_K + s_{m_j}^*}\right)s_{m_j}^*}\right\},$$

where $s_{l_j}^*$ and $s_{m_j}^*$ are the minimum values of stakes for the leaders and committee members in round i .

Proof. Let us consider that all leaders and committee members have cooperated in a given strategy profile. In this case, the payoff for any cooperative leader $l_j \in L$ would be equal to $u_{l_j}^{l_j}(C) = \frac{\alpha B_i}{S_L} s_{l_j} - c^L$. This payoff would be changed to $u_{l_j}^{l_j}(D) = \frac{\gamma B_i}{S_K + s_{l_j}} s_{l_j} - c^{so}$, if the leader l_j plays D and only keep its status online, without playing its role of a leader in Algorand. Hence, this leader has no incentive to defect if $u_{l_j}^{l_j}(C) > u_{l_j}^{l_j}(D)$. Consequently, we can show that under the following condition on B_i , the leader l_j has no incentive to deviate from C to D :

$$B_i > \frac{c^L - c^{so}}{\left(\frac{\alpha}{S_L} - \frac{\gamma}{S_K + s_{l_j}}\right)s_{l_j}}. \quad (5)$$

Similarly, any committee member $m_j \in M$ cannot increase his payoff unilaterally by defecting and play D if:

$$B_i > \frac{c^M - c^{so}}{\left(\frac{\beta}{S_M} - \frac{\gamma}{S_K + s_{m_j}}\right)s_{m_j}}. \quad (6)$$

Given two different bounds on the distributed rewards in Equations (5) and (6), and if we consider that $s_{l_j}^*$ and $s_{m_j}^*$ are the minimum values of stakes for leaders and committee members in round i , we can conclude that no leader or committee member can deviate in round i if

$$B_i > \max\left\{\frac{c^L - c^{so}}{\left(\frac{\alpha}{S_L} - \frac{\gamma}{S_K + s_{l_j}^*}\right)s_{l_j}^*}, \frac{c^M - c^{so}}{\left(\frac{\beta}{S_M} - \frac{\gamma}{S_K + s_{m_j}^*}\right)s_{m_j}^*}\right\}.$$

□

Lemma 2 shows that the Algorand Foundation must always distribute enough rewards to the leader and the committee members in each round to enforce cooperative behavior among them. The optimal reward B_i is a function of the cost of cooperation and the current state of stakes in this round. It also depends on the values of α , β , and γ , which must be selected by the administrator. We consider that these values would be announced at the beginning of each round. Another interesting fact is that if we assign more fraction of the reward B_i to the

leaders and the committee members (i.e., increasing the values of α and β), we can reduce the value of reward B_i , but have all leaders and committee members cooperate in a cooperative strategy profile. This will help the administrator to save more *Algos* for future use. Finally, giving more rewards to online nodes ($k_j \in K$) will increase the value of the required reward for cooperative behavior of leaders and committee members. Note that in Lemma 2, the following conditions must hold:

$$\frac{\alpha}{S_L} - \frac{\gamma}{S_K + s_{l_j}} > 0 \quad \& \quad \frac{\beta}{S_M} - \frac{\gamma}{S_K + s_{m_j}} > 0 \quad (7)$$

This can be easily proved given that the cost of cooperation for the leaders and the committee members (i.e., c^L and c^M) are always positive. Having established the required conditions for cooperation by leaders and committee members (Lemma 2), we can now establish conditions under which \mathcal{G}^{Al+} has a Nash equilibrium apart which is not the *All - D* strategy profile. In fact this new class of cooperative NE in \mathcal{G}^{Al+} will depend on the behavior of other online nodes and their roles in the Algorand network for any given round. Let us first review two important Algorand [22] concepts:

Definition 1. *In Algorand network, “strong synchrony” is a network state, where most honest Algorand nodes (e.g., 95%) can send messages that would be received by most of the other nodes (e.g., 95%) within a bounded time.*

Definition 2. *In Algorand network, with “weak synchrony” state, the network can be asynchronous for a long but bounded period of time. After this asynchrony period, network must be again strongly synchronous for a reasonably long time.*

Now we can form multiple sets of Algorand nodes which meet strongly synchronous network assumption together.

Definition 3. *“Strong synchrony set” is a list of Algorand nodes that together forms a strongly synchronous network.*

As Algorand protocol achieves liveness in strongly synchronous settings and safety with weak synchrony, the following theorem focuses on finding Nash equilibria for \mathcal{G}^{Al+} with the strong synchrony assumption.

Theorem 3. *In game \mathcal{G}^{Al+} with N players ($n_L > 1$ leaders, n_M committee members, and n_K remaining nodes), for each Algorand strong synchrony set \mathbb{Y} , a strategy profile s^* is a Nash equilibrium in round i , if in this strategy profile:*

- 1) All leaders cooperate,
- 2) All committee members cooperate,
- 3) All other nodes which are in \mathbb{Y} cooperate, and
- 4) Other online nodes defect

and the value of B_i is selected such that,

$$B_i > \max\left\{\frac{c^L - c^{so}}{\left(\frac{\alpha}{S_L} - \frac{\gamma}{S_K + s_{l_j}^*}\right)s_{l_j}^*}, \frac{c^M - c^{so}}{\left(\frac{\beta}{S_M} - \frac{\gamma}{S_K + s_{m_j}^*}\right)s_{m_j}^*}, \frac{(c^K - c^{so})S_K}{s_{k_j}^* \gamma}\right\},$$

where $s_{l_j}^*$, $s_{m_j}^*$, and $s_{k_j}^*$ are the minimum stakes for leaders, committee members, and other online nodes in \mathbb{Y} , in round i .

Proof. In order to prove that the strategy profile s^* defined by this theorem is a Nash equilibrium, we need to show that none of the users can increase their payoffs unilaterally, by changing the strategy. We divide the players into three groups, i.e., leaders, committee members, and other nodes. For the leader and committee members we apply our results presented in Lemma 2 to define s^* . In other words, the payoff of leaders and committee members cannot be increased unilaterally if the conditions of Lemma 2 are hold. Now, we will focus on remaining nodes, i.e., s_{k_j} . Given Definition 1, each remaining node could be a member of strong synchrony set \mathbb{Y} . Then two cases must be considered: (i) Online nodes who are not in \mathbb{Y} cannot increase their payoffs by deviating from D to C and accept the incurred cost of c^K , as the block would be made whether they cooperate or not. (ii) For each remaining node s_{k_j} which is the member of \mathbb{Y} , the payoff of cooperation would be $u_i^{k_j}(C) = \frac{\gamma B_i}{S_K} s_{k_j} - c^K$. But, the payoff of a defecting node would be $u_i^{k_j}(D) = -c^{so}$. In other words, if a member of \mathbb{Y} defects, no new final block would be created in this round given Definition 1. So, to prevent s_{k_j} from defecting the defined strategy in s^* , $u_i^{k_j}(C)$ must be greater than $u_i^{k_j}(D)$, hence, we can find another bound for B_i i.e., $B_i > \frac{(c^K - c^{so})S_K}{s_{k_j}\gamma}$. Given the above analysis, the defined strategy profiles which satisfies the condition form a Nash equilibrium strategy profile in game \mathcal{G}^{Al+} . \square

D. Proposed Reward Sharing Mechanism

Our next goal is to extend the current Algorand reward sharing method by considering the strategic behavior of users/nodes. In this case, we provide a solution for Algorand Foundation to foster cooperative behavior among all Algorand nodes. Moreover, our computed bounds in Theorem 3 shows that we can minimize the reward B_i by selecting suitable values for α , β , and γ . Our results presented in Section III-C showed that the Algorand Foundation needs to deploy an incentive-compatible mechanism to prevent nodes from selfish behavior (defecting) to unilaterally increase their payoff. We have proposed an algorithm based on Theorem 3 which provides enough incentive for Algorand nodes to cooperate. Our proposed Algorithm 1 proceeds as follows: at the end of each Algorand round, the Algorand Foundation extracts the list of the leaders, the committee members, and other online nodes as L , M , and K . These values can be simply computed by processing and verifying the list of all the sortition proofs that have been sent by the Algorand nodes in their votes (as committee members) or in the header of their block proposals (as leaders). Also, the Algorand foundation determines the set of other online nodes from their gossiped network messages. Then, the foundation will calculate the optimal values for α and β to minimize B_i , by using the defined bounds in Theorem 3. Next, the Algorand foundation computes the rewards for all nodes that have participated in this round based on their roles using the computed α , β , and B_i values. Finally, the foundation creates the reward transactions for each node and gossip it to the Algorand network. These transactions will

Algorithm 1 Incentive-Compatible Reward Sharing

```

1: procedure REWARD_SHARING( $i$ )
2:    $block_i \leftarrow$  Wait until final block created in round  $i$ 
3:   // Compute  $\alpha, \beta, B_i$  from Theorem 3 bounds
4:    $L, M, K, Stakes \leftarrow$  ExtractDataFromBlock( $block_i$ )
5:    $\alpha, \beta, B_i \leftarrow$  ComputeParameters( $L, M, K, Stakes$ )
6:   for all Node  $n \in block_i$  do
7:      $reward \leftarrow$  ComputeReward( $n, Role_i(n), \alpha, \beta, B_i$ )
8:     SendReward( $reward, n, i$ )
9:   end for
10:  RewardSharing( $i + 1$ )
11: end procedure

```

be verified by the Algorand network and put in the next blocks. We must clarify that as the stakes are computed at the end of each round, the B_i value is exactly equal to the minimum value of the reward that Algorand foundation should pay to the cooperative nodes. Thus, there is no incentive for selfish nodes to deviate from the reward sharing protocol.

V. EVALUATION

In order to evaluate our proposed mechanism, we first conduct a series of numerical analysis to obtain the best reward shares in our model (i.e., α and β). According to the results presented in Theorem 3, we can minimize the reward in each round such that it guarantees the cooperation of a subset of Algorand nodes. The optimal reward is ensured by choosing optimal reward shares for leaders and committee members, i.e., α and β . In our numerical analysis, we assume that the minimum acceptable values of stakes for each role are equal to $s_l^* = 1$, $s_m^* = 1$, and $s_k^* = 10$ *Algos*. In other words, by setting $s_k^* = 10$, we ignore any strong synchrony set containing nodes with stakes less than 10 *Algos* in this numerical analysis. We must highlight that in the real Algorand network nodes with low stakes do not choose to cooperate due to their negligible chance of getting elected as leaders or committee members and due to their low share of rewards [37]. Hence, setting $s_k^* = 10$ can results in a more realistic evaluation. We also assume that the cost of cooperation for the leaders, the committee members, and other nodes are $c^L = 16$, $c^M = 12$, $c^K = 6$, and $c^{so} = 5$ *micro Algos*. Our results show that for $(\alpha, \beta) = (0.02, 0.03)$, the minimum values of B_i would be about 5.2 *Algos* per round. Considering the value of S_K which is always much greater than S_L and S_M , the calculated bounds presented in Theorem 3 is usually a function of the third bound, i.e., $\frac{(c^K - c^{so})S_K}{s_{k_j}^*\gamma}$. Hence, to minimize the value of B_i we need to maximize γ and consequently minimize α and β (recall that $\gamma = 1 - \alpha - \beta$). In summary, our mechanism always considers enough share of the total reward for leaders and committee members, as shown in Equation (7). Moreover, it also provides enough rewards to all other online nodes considering the value B_i which is greater than $\frac{(c^K - c^{so})S_K}{s_{k_j}^*\gamma}$.

We then simulate an Algorand network containing 500,000 nodes, in which the amount of stakes for leaders and commit-

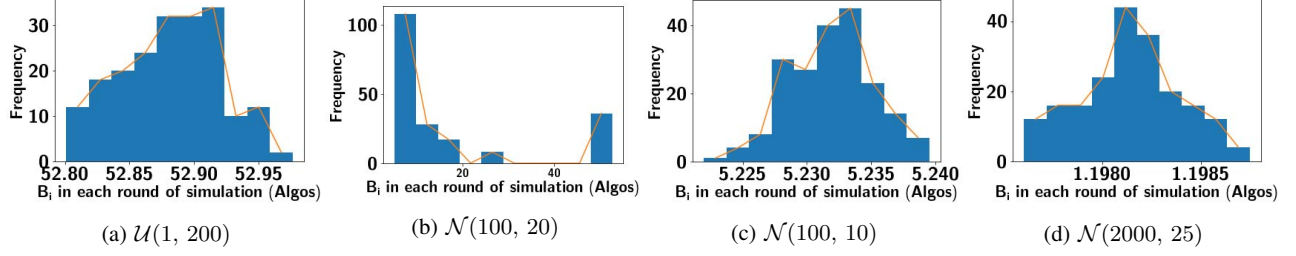


Fig. 5: Distribution of computed B_i values in each simulation by our proposed mechanism, for different distributions of stakes.

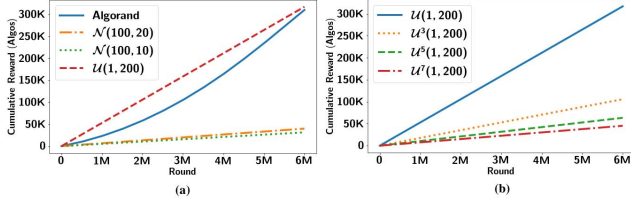


Fig. 6: (a) The cumulative rewards distributed among Algorand nodes by our proposed algorithm and Algorand Foundation. (b) The cumulative rewards when the Algorand nodes with less than 3 ($\mathcal{U}^3(1, 200)$), 5 ($\mathcal{U}^5(1, 200)$), and 7 ($\mathcal{U}^7(1, 200)$) stakes have been removed from the network.

tee members are $S_L = 26$ and $S_M = 13K$, respectively. We have chosen the number of nodes and the stake values from Gilad et al. simulation parameters in [22]. We run simulations with three different settings, where we distribute 50 million *Algos* among these 500K nodes using (i) uniform distribution of $\mathcal{U}(1, 200)$, (ii) normal distribution of $\mathcal{N}(100, 20)$, and (iii) $\mathcal{N}(100, 10)$. In each round, we also randomly choose 1000 nodes, in which nodes with higher stakes would be selected more often. Then we generate a series of random transactions for the selected nodes with a uniform distribution $\mathcal{U}(-4, 4)$. Negative values represent sending *Algos* while positive values represent receiving *Algos* in nodes. With these values we tried to emulate the real Algorand exchange system available at algoexplorer [37]. As for the reward sharing mechanism, we deploy both the Algorand Foundation proposal presented in Table 2 and our proposed mechanism presented in Algorithm 1. We run the simulation for 200 times with different distributions, where each instance executes for 10 rounds. Finally, we compute the average of total rewards.

Our simulation results (Fig. 5) show that the calculated rewards for our proposed mechanism follows the distribution of stakes in the network. For example, we must distribute higher rewards (around 50 *Algos*) for uniform distribution of $\mathcal{U}(1, 200)$, as there exist many nodes with low stakes. But with a normal distribution $\mathcal{N}(100, 10)$, we need to only distribute smaller rewards, i.e., around 5 *Algos*. In fact $\mathcal{N}(100, 10)$ simulates the initial phase of Algorand, where around 50 millions *Algos* were in the network. Comparing the results presented in Fig. 5-(d) (which simulates current status of Algorand [37] with more than 1 billion *Algos*, by $\mathcal{N}(2000, 25)$

stake distribution) with Fig. 5-(c), we can also conclude that when the total stake of the network increases, we need smaller reward to enforce cooperation (around 1.2 *Algos*). The results show that the Foundation can adapt the rewards given the status of the network in terms of stakes, by using our model.

Fig. 6-(a) shows the cumulative calculated reward in each round with our proposed algorithm and Algorand Foundation mechanism. This result shows that our proposed mechanism distributes much smaller rewards among nodes, given the distribution of stakes. For example, in contrast to the reward sharing approach proposed by the Algorand Foundation which shares 20 *Algos* in each round for the first 500,000 rounds [34], our proposed reward sharing algorithm will share about 5.2 *Algos* for a normal distribution of stakes. More interestingly, our proposed solution will not increase the reward until 6 million blocks are generated, as it can guarantee cooperation without paying more *Algos*. Our approach only distributes more rewards when the distribution of stakes is $\mathcal{U}(1, 200)$. This is due to the fact that the number of nodes with small values of stakes are higher with this distribution. If we can remove the nodes with smaller stakes (e.g., nodes with up to 7 stakes) we can still keep the synchrony of the network and distribute much smaller rewards. This is shown in Fig. 6-(b).

VI. CONCLUSION

In this paper, we first introduced a system model to capture the main operational features of the Algorand. We then comprehensively studied the problem of node selfishness in Algorand and proposed a possible solution to overcome it by employing game-theoretic modeling and analysis. Our analytical results show that we can always enforce cooperation by carefully distributing the correct amount of rewards. Moreover, our numerical analysis validated that the proposed reward sharing mechanism outperforms the current proposal by the Algorand Foundation. We believe that this work is the first step towards a better understanding of the effects of selfish behavior in Algorand. Our mechanism can help the Algorand Foundation use the *Algos* wisely, as well as, adapt dynamically with the distribution of stakes in the network. In terms of future work, we can also get in touch with the Algorand Foundation to introduce our proposed mechanism for reward sharing in the initial phase (for 1.75 billion *Algos*), as well as the distribution of transaction fees as rewards in near the future.

REFERENCES

- [1] S. Nakamoto *et al.*, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
- [2] “Ethereum project.” <https://ethereum.org/>, July 2018.
- [3] “All Cryptocurrencies.” <https://coinmarketcap.com/all/views/all/>, July 2018.
- [4] K. J. O’Dwyer and D. Malone, “Bitcoin mining and its energy footprint,” in *25th IET Irish Signals Systems Conference and China-Ireland International Conference on Information and Communications Technologies (ISSC/CICT)*, pp. 280–285, June 2014.
- [5] M. Rosenfeld, “Analysis of bitcoin pooled mining reward systems,” *CoRR*, vol. abs/1112.4980, 2011.
- [6] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, “On scaling decentralized blockchains,” in *Financial Cryptography and Data Security* (J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, eds.), 2016.
- [7] J. Garzik, “Bitcoin Improvement Proposal 102.” <https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki>, 2015.
- [8] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-ng: A scalable blockchain protocol,” in *NSDI*, pp. 45–59, 2016.
- [9] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A secure sharding protocol for open blockchains,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30.
- [10] “Lightning Network.” <https://lightning.network/>, July 2018.
- [11] “The Raiden Network.” <https://raiden.network/>, July 2018.
- [12] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, “Enabling blockchain innovations with pegged sidechains.” <https://www.blockstream.com/sidechains.pdf>, 2014.
- [13] S. King, “Primecoin: Cryptocurrency with Prime Number Proof-of-Work.” <http://primecoin.io/bin/primecoin-paper.pdf>, 2013.
- [14] S. King and S. Nadal, “PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.” <https://decred.org/research/king2012.pdf>, 2012.
- [15] P. Vasin, “BlackCoins Proof-of-Stake Protocol v2.” <https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf>, 2014.
- [16] L. Ren, “Proof of Stake Velocity: Building the Social Currency of the Digital Age.” <https://www.reddcoin.com/papers/PoS.pdf>, 2014.
- [17] D. Pike, P. Nosker, D. Boehm, D. Grisham, S. Woods, and J. Marston, “Proof of Stake Time: A time-accepted periodic proof factor in a nonlinear distributed consensus.” <https://www.vericoin.info/downloads/VeriCoinPoSTWhitePaper10May2015.pdf>, 2015.
- [18] J. Chen and S. Micali, “Algorand,” *arXiv preprint arXiv:1607.01341*, 2016.
- [19] P4Titan, “Slimcoin. A Peer-to-Peer Cryptocurrency with Proof-of-Burn.” https://www.doc.ic.ac.uk/~ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf, 2014.
- [20] “Hyperledger sawtooth.” <https://www.hyperledger.org/projects/sawtooth>.
- [21] M. Borge, E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford, “Proof-of-personhood: Redemocratizing permissionless cryptocurrencies,” in *IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pp. 23–26, April 2017.
- [22] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, ACM, 2017.
- [23] B. Biais, C. Bisiere, M. Bouvard, and C. Casamatta, “The blockchain folk theorem,” *The Review of Financial Studies*, vol. 32, no. 5, pp. 1662–1715, 2019.
- [24] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, “Blockchain mining games,” in *Proceedings of the ACM Conference on Economics and Computation*, pp. 365–382, 2016.
- [25] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, “Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin,” in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 195–209, 2017.
- [26] N. T. Courtois and L. Bahack, “On subversive miner strategies and block withholding attack in bitcoin digital currency,” *arXiv preprint arXiv:1402.1718*, 2014.
- [27] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, “On power splitting games in distributed computation: The case of bitcoin pooled mining,” in *IEEE 28th Computer Security Foundations Symposium*, pp. 397–411, 2015.
- [28] I. Eyal, “The miner’s dilemma,” in *IEEE Symposium on Security and Privacy*, pp. 89–103, IEEE, 2015.
- [29] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosen-schein, “Bitcoin mining pools: A cooperative game theoretic analysis,” in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 919–927, Citeseer, 2015.
- [30] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden, “Incentive compatibility of bitcoin mining pool reward functions,” in *International Conference on Financial Cryptography and Data Security*, pp. 477–498, Springer, 2016.
- [31] M. H. Manshaei, M. Jadliwala, A. Maiti, and M. Fooladgar, “A game-theoretic analysis of shard-based permissionless blockchains,” *IEEE Access*, 2018.
- [32] S. Kim, “Two-phase cooperative bargaining game approach for shard-based blockchain consensus scheme,” *IEEE Access*, 2019.
- [33] S. Micali, M. Rabin, and S. Vadhan, “Verifiable random functions,” in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pp. 120–130, IEEE, 1999.
- [34] Algorand Foundation, “Token dynamics.” <https://algorand.foundation/token-dynamics>.
- [35] Algorand Foundation, “Rewards - an faq for users.” <https://algorand.foundation/rewards-faq>.
- [36] D. Deka, A. Singh, and P. Jain, “Algorand discrete event simulator.” GitHub, May 2019. Source is available at <https://github.com/ddeka0/CS620-Algorand-DES>.
- [37] “Algorand explorer.” <https://algoexplorer.io>.