# Designing an Incentive-compatible Reward Scheme for Algorand

by

Maizi Liao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

Founded in 2017, Algorand is the first carbon-negative blockchain protocol inspired by proof of stake. Algorand uses a Byzantine agreement protocol to add new blocks to the blockchain. The protocol can tolerate malicious users as long as a supermajority of the stake is controlled by non-malicious users. The protocol achieves about 100x more throughput that Bitcoin and can be easily scaled to millions of nodes. Despite its impressive features, Algorand lacks a reward-distribution scheme to incentivize nodes to participate in the protocol. In this work, we study the incentive issue in Algorand through the lens of game theory. We model the Algorand protocol as a Bayesian game and propose a novel reward scheme to address the incentive issue in Algorand. Through rigorous analysis, we derive necessary conditions to ensure that participation in the protocol is a Bayesian Nash equilibrium even in the presence of a malicious adversary. In addition, we propose a referral mechanism to ensure that malicious nodes cannot earn more rewards in expectation compared to non-malicious nodes.

**Acknowledgements**

I would like to thank Prof. Seyed Majid Zahedi and Prof. Wojciech Golab for supervising my research. I would also like to thank Prof. Kate Larson and Prof. Mahesh Tripunitara for reviewing my thesis.

## Dedication

This is dedicated to the one I love.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The concept of blockchain is first popularized by Bitcoin [57] as a tamper-resistant distributed transaction ledger. The ledger consists of a chain of blocks. To add a new block to the chain, nodes in the system run a consensus protocol, which ensures that nodes are in agreement on the new block. Consensus protocols are divided into two main categories: permissioned and permissionless. Permissioned consensus protocols consider a system that comprises a fixed and known set of authenticated nodes [23, 50, 59]. In contrast, permissionless consensus protocols allow nodes to join the system without any permission [44, 55].

When permission is not required, the system could become prone to a Sybil attack[1]. To mitigate the Sybil attack threat, permissionless consensus protocols often use additional mechanisms. For example, Bitcoin uses proof of work (PoW), which requires nodes to solve a computationally intensive puzzle. Winners earn the right to add blocks to the blockchain and collect rewards for their computational effort. PoW suffers from high energy and computational costs [2]. Proof of stake (PoS) has been proposed to mitigate these costs [7, 47]. In most of PoS consensus protocols, nodes stake their cryptocurrency assets to gain rights to add blocks and earn rewards.

Algorand is a randomized, committee-based consensus protocol inspired by proof of stake [40, 24]. The core of Algorand is a Byzantine agreement protocol that allows nodes to reach consensus on a new block in the presence of Byzantine faults[2]. Nodes are selected

[1] In a Sybil attack, the attacker creates a large number of pseudonymous identities to gain disproportionate control and/or influence over the system.

[2] In a Byzantine fault, a node may inconsistently appear both failed and functioning to other nodes. The term is taken from the "Byzantine general problem" [51].

randomly to participate in the Byzantine agreement protocol as committee members. The random selection is conducted by each node locally, but committee memberships are verifiable by other nodes. The original reward scheme of Algorand[3] rewards all nodes proportionally to their account balance. Although simple, this reward scheme suffers for the free-rider problem: nodes have no incentive to participate in the protocol as doing so imposes computational and communication costs. This can be seen by tracking the number of nodes that actively participate in Algorand. According to [1], in May 2022, only about 1.6 billion units of Algorand's cryptocurrency were registered to participate while a total of about 8.4 billion units of Algorand's cryptocurrency were available. Moreover, while there were more than 1.7 million active accounts, only 361 unique accounts had recently participated in Algorand's consensus protocol. Since the safety and liveness of Algorand depend mainly on high participation of nodes, the lack of participation poses a serious threat to Algorand by making it prone to attacks from malicious participants.

The incentive problem in Bitcoin-like blockchains has been studied extensively in recent years [34, 63, 60, 52, 26]. However, the results do not apply to Algorand due to its unique consensus protocol. Amoussou-Guenou et al. [12, 13] analyze incentives in a committee-based consensus protocol. They present a game-theoretic model of the protocol and study its safety and liveness under equilibrium strategies of participating nodes. Although related, their analysis is not applicable to Algorand as their studied protocol has major differences with the consensus protocol in Algorand. One of the main differences is that they consider the membership of nodes in the committee to be common knowledge. In Algorand, however, nodes' membership in the committee is determined locally by each node and is private information.

In a closely related work, Fooladgar et al. [38] propose a role-based reward scheme to mitigate the free-rider problem in Algorand. They model Algorand as a static non-cooperative game and study equilibrium strategies under their proposed reward scheme. Although all nodes that are selected as committee members are incentivized to participate, their proposed reward scheme only incentivizes a limited fraction of non-committee nodes. Moreover, their analysis only considers well-behaved nodes and does not apply to systems with Byzantine nodes corrupted by a malicious adversary.

In this paper, we propose IRS, an incentive-compatible reward scheme for Algorand. We model Algorand as a Bayesian game and study nodes' strategies under our proposed reward scheme. Our analysis considers an adversary that can corrupt nodes in a probabilistic

[3]Algorand is moving from its original reward scheme to a new reward scheme called the Governance Rewards [6]. Under the new reward scheme, only agents who commit to participate in the governance of the Algorand ecosystem will be rewarded. Agents have to prove their commitment by locking their cryptocurrency assets for a potentially long term. This new scheme is in line with proof-of-stake protocols.

manner. We show that if certain conditions are met, all nodes are incentivized to participate in the protocol regardless of being selected as committee members. We further propose a referral reward mechanism to ensure that the malicious adversary does not gain more expected rewards per corrupted node compared to non-corrupted nodes. In summary, we make the following contributions.

- We present a detailed cost model for nodes' participation in Algorand (§3).
- We model the Algorand protocol as a Bayesian game (§4).
- We propose IRS, a novel incentive-compatible reward scheme to address the free-rider problem (§5).
- We study equilibrium strategies under IRS and derive necessary conditions to ensure participation in equilibrium (§5).
- We propose a novel referral mechanism to ensure that corrupted nodes receive less than or equal expected reward compared to non-corrupted nodes (§6).
- We present detailed implementation requirements for real-world deployment of IRS (§7).

# Chapter 2

# Background

In this chapter, we provide some background materials about Algorand. We first introduce the common communication and adversary models used in distributed systems. Next we describe the consensus problem, which is fundamental to distributed systems like blockchains. Then we categorize blockchains based on the consensus protocols.

## 2.1 Communication Model

In distributed systems, the underlying network usually suffers from uncertainty such that the messages could be delayed for some period. The communication model defines the limit of such message delays.

There are three common communication models in distributed system literature: the Synchronous model, the Asynchronous model, and the Partially Synchronous model. The following are the assumptions made for each model:

- **Synchronous Model**: Any message sent will be received within a known time bound $t'$.
- **Asynchronous Model**: Any message sent will be received within an unknown time bound.
- **Partially Synchronous Model** [31]: Any message sent at time $t$ will be received within $t' + max(t, GST)$ where $GST$ (Global Stabilization Time) is unknown.

One way to reason about the Partially Synchronous model is that the communication is asynchronous before GST and becomes synchronous after that.

## 2.2 Adversary Model

The nodes in a distributed system may suffer from some kinds of failures. For example, a power shortage may cause a node to crash and a software bug could lead to undefined behaviour from the nodes. Moreover, a malicious attacker may take full control of some nodes and make the nodes behave arbitrarily. Theses failures are usually captured by assuming the existence of an adversary which can corrupt $f$ out of $n$ nodes in the system.

The adversaries are usually classified by the types of failure it can inflict on the corrupted nodes. There are three common kinds of failure in distributed systems [31]:

- **Crash Failure**: The corrupted nodes stop sending or receiving any messages.
- **Omission Failure**: The corrupted nodes fail to send or receive messages when they should.
- **Byzantine Failure** [51]: The corrupted nodes could send erroneous messages.

Note that the Byzantine failure is a superclass of the crash failure and the omission failure since a Byzantine node can pretend to be a one of these types. There are many other failures but most of them can be subsumed by the Byzantine failure. In a permissionless setting where everyone can join and leave the system at any time, such as Bitcoin [57], the protocols designers usually focus on the Byzantine failure since it is the most general failure mode. In addition, malicious users can join the system and it is hard to predict what they will do.

Another important assumption about the adversary is the inability to break the cryptographic primitives used in the distributed systems. Distributed systems usually assume a public key infrastructure (PKI) where each node has a public-private key pair $(pk, sk)$. When a node wants to send a message $m$ to other nodes, it should send the signed version of the message $m_{sk}$ instead. Other nodes can verify the signature using the sender's public key $pk$. Many consensus protocols [23, 42, 68, 40] are based on the assumption that the adversary cannot forge the signature of correct nodes. This prevents impossibility results caused by the issue where the adversary can simulate correct nodes [36].

## 2.3 The Consensus Problem

The consensus problem is one of the fundamental problems in distributed systems like blockchains. It occurs when multiple nodes in the system need to make an agreement on a value. In blockchains, the nodes in the system want to reach an agreement on the order and the content of the blocks, which contains an ordered set of transactions. The formal definition of the consensus problem is defined as the following:

> In a system with $n$ nodes indexed by $1, \ldots, n$, there are $f$ out of $n$ nodes with failures. Each node $i$ has an input value $v_i \in V$. The nodes must decide a value among these inputs such that the following properties are satisfied [31]:

- **Safety**: If a correct node decides a value $v$, all other correct nodes decide on the same value $v$. The decided value $v$ should satisfy the application-specific validity conditions.
- **Liveness**: All the correct nodes will eventually decide a value $v$.

A seminal work [37] from Fischer, Lynch and Paterson proved that it is impossible to satisfy both properties under asynchronous communication model when there exists an adversary who can only crash one of the nodes. However, as noted in the paper [37], this does not mean the consensus problem is not solvable under asynchronous network models; rather, the impossibility results indicate that more refined models or less strict requirements on the solutions are needed. There are mainly two possible workarounds to circumvent the impossibility result. The first method is by finding a middle ground between the synchronous model and the asynchronous model, such as the partially synchronous model introduced by [31]. Another method is to lighten the constraint on the liveness property by randomization such that all the correct nodes eventually decide a value $v$ with probability 1 [17, 20, 19].

The solutions to the consensus problem, usually called consensus protocols, are said to be $t$-resilient if they can tolerate up to $t$ faulty nodes. The maximum number of corrupted nodes a consensus protocol can tolerate is called the optimal resilience of the protocol. As shown in table 2.1, it has been proved in [31, 20] that the optimal resilience of any asynchronous or semi-synchronous Byzantine consensus protocols is $\lfloor \frac{n-1}{3} \rfloor$ where $n$ is the total number of nodes. If we assume the communication is synchronous, then the optimal resilience is increased to $\lfloor \frac{n-1}{2} \rfloor$.

| Communication Model | Optimal Resilience |
|---|---|
| Synchronous | $\lfloor \frac{n-1}{2} \rfloor$ |
| Asynchronous | $\lfloor \frac{n-1}{3} \rfloor$ |
| Semi-synchronous | $\lfloor \frac{n-1}{3} \rfloor$ |

Table 2.1: Optimal resilience of Byzantine consensus protocols under different communication models

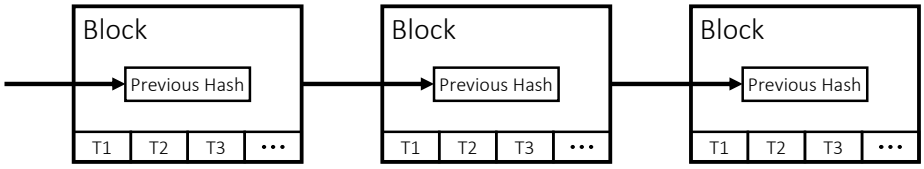


Figure 2.1: Blockchain data structure

These optimal resilience thresholds are based on the assumption that nodes in the system are equivalent such that they have the same voting power. In an open system like Bitcoin [57] where nodes can join and leave freely, making such an assumption is prone to the Sybil attack [30] where an attacker can create identities in the system cheaply and gain more voting power. Once the voting power of the attacker exceeds the optimal resilience, it can break the safety property of the system. In this scenario, the voting power of nodes should depend on some resource which cannot be easily replicated. Section 2.2 discusses how to prevent Sybil attacks under an open environment in details.

## 2.4 Blockchain

As shown in Fig. 2.1, each block in a blockchain contains a set of ordered transactions and the hash of the previous block. To reduce the block size, some blockchains such as Bitcoin [57] will use a data structure called Merkle Tree to store the transactions. Each block also includes the hash of the previous block, forming a chain of blocks. Some additional metadata could be added to a block such as the identity of the block proposer and the timestamp of the block.

Another thing people can store in a blockchain is smart contract [65, 67]. A smart contract is a collection of functions and the data. The functions could be written in Solidity [9] or other general purpose programming languages [14]. Once a smart contract is deployed

on the blockchain, it can not be changed since it is also finalized by the consensus protocol when the nodes in the system are trying to reach an agreement on the block. Any node in the system can then invoke the functions in the smart contract. In blockchains, smart contracts are usually used to ensure that the transactions recorded in blockchain will only take effect when the required conditions are met. Bitcoin implements a script which can also be considered as a kind of smart contract. However, Bitcoin script does not support looping and thus is not a Turing-complete [8].

Blockchains could be classified into two categories: permissioned and permissionless. Permissioned blockchains, also known as private blockchains, use an access control mechanism to restrict unauthorized users to access the ledger. Examples include HyperLedger Fabric [14] and Libra (now called Diem) [4]. In contrast, permissionless blockchains do not have any access restrictions. Examples include Bitcoin [57] and Ethereum [67]. To secure the ledger, permissionless blockchains usually utilize a proof-of-work or proof-of-stake mechanism. In permissionless blockchains, nodes are usually not treated equally, meaning that nodes with more voting power will have greater influence on the blockchains.

## 2.5 Proof-of-work Protocols

Proof of work is originally proposed to deter spammers and discourage junk mail [32]. Bitcoin is the first cryptocurrency that adopted PoW to prevent Sybil attacks [57]. The main idea is that to generate a new block, a node is required to solve a computationally intensive cryptographic puzzle. The puzzle is designed in such a way that the solution cannot be found faster than by brute-force search. To solve the puzzle, a node needs to find a *nonce*. When combining the *nonce* with the *payload* in a block, such as the hash of previous block and the hash of the root of the merkle tree, as the input to a public known hash function $H(\cdot)$ (such as SHA-256), the output should be less than a small number $D$ which is a difficulty parameter [57].

$$H(nonce, payload) < D.$$

The voting power in the PoW system is essentially the computation power owned by the node. In addition, Bitcoin will only reward the block proposers of the blocks which are on the longest chain. For a Bitcoin miner to increase its portion of blocks on the longest chain, it needs to invest more computation power into solving the puzzles. This reward-distribution scheme instills competition among the miners and aggravates energy consumption. Nowadays, the total power consumption by Bitcoin is about 135 TWh a year, comparable to the annual power consumption of Sweden [2].

Although Bitcoin is designed with the vision to be a decentralized system, it becomes more centralized as the total computation power in Bitcoin increases [39, 26]. The same phenomenon happens to Ethereum [67] as well, which is another popular PoW blockchain. About 90% of mining power is controlled by only 16 distinct mining entity in Bitcoin and only 11 distinct mining entity in Ethereum. One possible explanation is that most small miners are risk-averse, which means their utility is a strictly concave function of the reward. It has been shown by [63, 52, 26] that the reward scheme used by Bitcoin and Ethereum is not collusion-proof in the sense that miners are incentivized to form mining pools to reduce reward variance. Another potential reason is that the cost of mining (such as electricity cost) is different among the miners and [15] shows that this asymmetric cost can lead to centralized mining.

## 2.6 Proof-of-stake Protocols

Proof-of-stake (PoS) is proposed to mitigate the energy waste problem caused by PoW. The idea of proof-of-stake is first discussed in a Bitcoin forum [7] and is first utilized in PPCoin [47]. In proof-of-stake blockchains, nodes that hold a greater balance of the assets will have more voting power in the system.

There are two ways for nodes to provide proof of stake to be a block proposer [28]. They can either provide a public computation result similar to the nonce in proof-of-work blockchains, which can be computed repeatedly by other nodes in the system to verify the result, or a private computation result secured by the node's private key through a cryptographic mechanism, which can only be verified by using the node's public key.

The public computation method was first used by PPCoin [47] and then by a series of following works, such as Ouroboros [46] and Snow White [27]. For example, PPCoin [47] makes use of coin age, which is the product of the amount of the coins and the time period of holding the coins, to play the role of hashing power in proof-of-work blockchains. The node in PPCoin will use a hash function similar to the one in Bitcoin to check if it is selected as a block proposer. Once selected, the coin age of the node will be consumed (reset the time period of holding the coins to 0). One important difference between the hashing process of PPCoin and proof-of-work blockchains is that the node is restricted to try only a certain amount of hashes per second (proportionally to the amount of stakes holding by the node). This can be achieved by requiring nodes to add time as one of the input to the hash function and will mitigate the computation competition between the nodes.

The private computation method was used in blockchains including Algorand [40] and Ouroboros Praos [28]. In Algorand, a node $x$ will feed the role (such as block proposers) with a seed as an input to a verifiable random function (VRF). The VRF also needs the private key of the node and outputs a hash and a proof. The hash can then be used to determine the eligibility of the node as the role. Other nodes, who know the public key of the node $x$ in the system can verify the eligibility of the node $x$ by using the proof to check that the hash is indeed computed by the VRF with the corresponding inputs.

Some PoS blockchains [21, 67] require nodes to stake part of their account balance. The stake of the nodes cannot be spent for a certain time. During this time, the stake of the nodes will be the voting power of the nodes. If a node is found to be malicious, its stake will be expropriated as a punishment. A potential problem of such a punishment mechanism is that nodes with a small account balance may not be willing to stake their account balance, and only nodes with a large account balance will stake their assets to participate in the protocol [24].

Delegated Proof-of-Stake (DPoS) [3, 5] is another popular variant of PoS. In DPoS, nodes will delegate their voting power to a fixed set of validators. The validators will be treated equally and the DPoS blockchains can use traditional consensus protocols to reach agreement on the blocks. Delegates found to be malicious can be expelled by the nodes through voting mechanisms. However, DPoS is inherently permissioned and centralized. As a result, it is prone to attacks by adversaries, such as denial of service attacks or Eclipse attacks [43].

## 2.7 Finality

One of the problems of blockchains is that forking may happen due to network delays and the randomness from the selection mechanisms. For example, multiple Bitcoin miners may solve the puzzles for the blocks at the same height around the same time and nodes may receive different blocks at the same height. This leads to a divergent view of the order and the content of the blocks.

Bitcoin [57] and many other permissionless blockchains use the longest-chain rule to resolve forks. The longest-chain rule requires the nodes to follow the longest chain when facing forks and break ties in favour of the fork which was discovered first. Protocols adopting the longest-chain rule usually ask nodes to confirm a block when there are $k$ more blocks after it, where $k$ is a safety parameter. A larger $k$ means a higher probability that the block will not be reverted, but also a longer confirmation latency. Another problem

caused by the forks and the longest-chain rule is selfish mining [34]. When a node possesses more than $\frac{1}{3}$ of the voting power, it is incentivized to hide the block created by itself to maximize utility.

By contrast, committee-based consensus protocols [40, 21] using certificates of votes can provide nearly instant finality of a new block. As soon as a node receives enough votes for a block, it can mark the block as finalized since no other blocks at the same height will also receive enough votes. In this case, forks will never happen or will only happen with a negligible probability, and the protocols will not suffer from selfish mining.

Table 2.2 below shows different categories of blockchains and examples of them.

| Openness | Consensus Protocol | Example |
|---|---|---|
| Permissioned | Quorum Certificate | HyperLedger Fabric[14], Diem[68, 4] |
| Permissionless | PoW + Longest-Chain | Bitcoin[57], Ethereum[67] |
| | PoW + Quorum Certificate | ByzCoin[48], Solida[10] |
| | PoS + Longest-Chain | PPCoin[47], Ouroboros[46], Snow White[27] |
| | PoS + Quorum Certificate | Algorand[40], Tendermint[21] |

Table 2.2: Blockchain classifications

## 2.8 The Algorand Protocol

The Algorand protocol maintains a public, permissionless blockchain. Adding a new block to the blockchain requires multiple steps. Algorithm (1) provides high-level pseudocode of the Algorand protocol[1]. At each step, all nodes wait for the messages from the previous step for a fixed amount of time. Each node then validates and propagates received messages to its neighbors. The protocol can terminate at specific steps (i.e., $k > 4$ where $k \not\equiv 1 \pmod 3$) if a termination condition is met (i.e., enough votes are received or the final step, $\mathrm{K}_{\max}$, is reached). At each non-terminal step, a random committee of self-selected nodes is formed. Committee members generate and propagate a message according to the protocol. The message is a block proposal if $k = 1$, and its a vote on a proposed block if $k > 1$.

**Cryptographic sortition.** In Algorand, nodes are assumed to have access to a unique signature scheme (e.g., [56]). As shown in Algorithm (2), at step $k$, given a publicly

[1]For more details, see Appendices B.1–B.3.

**Algorithm 1:** High-level pseudocode of the Algorand protocol

```
for k = 1, ..., K_max do
    if k > 1 then Validate and gossip step-(k - 1) messages for a fixed time period;
    if k > 4 and k ≢ 1 (mod 3) then Exit if termination condition for step k is
     met;
    (x_k, σ_k) ← Sortition(k);
    if .x_k ≤ p_c then
        message ← Generate a message;
        Propagate(message, σ_k);
    end
end
```

**Algorithm 2:** Sortition($k$) procedure given seed $Q$

```
procedure Sortition(k) begin
    σ ← SIG(k, Q);
    x ← H(σ);
    return (x, σ);
end
```

known random seed, $Q$, each node privately computes a hashlen-bit-long random string $x_k = H(\text{SIG}(k, Q))$ by digitally signing $(k, Q)$ and then hashing it using a random oracle $H$ [41]. The string $x_k$ is interpreted as a binary expansion of a number between 0 and 1, denoted by $.x_k = x_k/2^{\text{hashlen}}$. If this number is less than a known threshold, $p_c$, then the node is a member of the committee at step $k$. The threshold is set such that the expected size of the committee is $\tau$ (i.e., $p_c = \tau/n$, where $n$ is the number of nodes in the system). $.x_k$ is also used to represent the priority of the node; the smaller $.x_k$ is the higher the priority of the node will be. For nodes in the committee, $\sigma_k = \text{SIG}(k, Q)$ is the committee credential. Committee members propagate their credential alongside their generated message.

**Adversary model.** The committee is guaranteed to reach Byzantine agreement [61, 29, 35] in the presence of an adversary that can corrupt nodes and control their actions. The Algorand protocol is resilient to such adversary as long as it cannot corrupt more than 1/3 of the nodes. This is achieved by setting the expected size of the committee, $\tau$, such that, with high probability, at least 2/3 of the committee members are non-Byzantine nodes. In this paper, however, we consider a slightly different adversary model. In particular, we

assume that the adversary corrupts each node with a fixed probability $p_b < 1/3$. Under our probabilistic adversary model, it is possible for the adversary to corrupt more than 1/3 of nodes ex post. However, we show in §5 that for large systems, under our adversary model, non-Byzantine nodes still constitute more than 2/3 of the committee with high probability. Consequently, Algorand protocol is guaranteed to reach Byzantine agreement with high probability.

**Non-Byzantine majority.** Since Algorand is a permissionless blockchain, the adversary can easily introduce as many new nodes as it wishes. Therefore, instead of assuming that the system has at least a 2/3 majority of non-Byzantine nodes, it is often more meaningful to assume that at least 2/3 of the cryptocurrency assets are controlled by non-Byzantine nodes. In other words, instead of assuming that the adversary can corrupt up to 1/3 of the nodes, it is often assumed that the adversary can control up to 1/3 of the assets in the blockchain. Algorand achieves this by assigning *sub-nodes* to each node in proportion to the balance of its account. The cryptographic sortition algorithm then randomly selects each sub-node as a committee member. In this paper, we present our analysis under the simpler assumption that each node has a single sub-node. We then show how to modify the Algorand protocol and our analysis to consider the more realistic assumption that each node controls multiple sub-nodes.

**Gossip network and protocol.** In Algorand, each node is provided with an *address-book* file containing the IP address and the port number of other nodes. Nodes form a gossip network by selecting a subset of $n_{rp}$ random peers to gossip messages to. The parameter $n_{rp}$ depends on the number of nodes, and it is set such that the gossip network is strongly connected. Messages are disseminated on the gossip network using a gossip protocol. The message dissemination is initiated by committee members at each step. Each committee member propagates their generated message to their randomly selected peers. Those peers then forward the message to their own peers. And this process continues until the message is received by all the nodes in the network. To avoid forwarding loops, nodes do not propagate the same message twice.

**Timing guarantee.** In this paper, we assume that the gossip network is strongly synchronous. This is a widely adopted network assumption [40, 21, 12, 13] which states that all messages propagated initially by non-Byzantine nodes are received by all other non-Byzantine nodes within a known time period. We further assume that the network remains strongly synchronous if a majority of nodes run the gossip protocol. This means that for large systems, the adversary cannot launch an Eclipse attack [43] with high probability.

# Chapter 3

# Proposed Cost Model

In this section, we provide our proposed cost model. Nodes running the Algorand protocol incur processing and communication costs at each step of the protocol. These costs are measurable in quantitative terms (e.g., energy consumption) and can be expressed in monetary values (e.g., cryptocurrency or Dollar).

## 3.1 Operational Costs

Below, we provide a brief description of the different operational costs incurred by nodes running the Algorand protocol.

**Cryptographic sortition.** At every non-terminal step, nodes invoke the cryptographic sortition to privately identify if they are a committee member for that step. The sortition algorithm produces a digital signature and runs a hash function to generate a random string and a credential. We use $c_{cs}$ to denote the cost of running the sortition algorithm per step.

**Block generation.** At step 1, each node in the committee generates a block proposal. This requires the committee members to first validate a set of pending transactions that they have heard about. Next, validated transactions are assembled in a new block proposal The block is then digitally signed by the committee member. We use $c_{bg}$ to represent all the costs associated with generating a new block proposal.

**Vote generation.** At step $k \geq 2$, selected committee members generate a vote based on the Algorand protocol. This requires three steps: (i) computing the value of the vote,

(ii) constructing the vote, and (iii) digitally signing the vote. We let $c_{vg}(k)$ denote the total cost of generating a vote at step $k \geq 2$. This cost depends on step number as the amount of processing needed to compute the value of the vote is not the same across different steps.

**Block validation.** At step 2, all nodes validate all unique block proposals they receive before gossiping them to their peers. This requires nodes to first verify the signature of the block proposer for each received block proposal. Next, the sortition of the block proposer is verified and the priority of the block proposal is calculated. Finally, nodes validate the content of the block (i.e., all pending transactions) and its hash. Note that in the gossip protocol, a node could receive the same message from different peers. We assume that nodes can detect duplicates with negligible cost to avoid validating the same block proposal multiple times. For example, nodes can cache the hashes of blocks. We use $c_{bv}$ to denote the total cost of validating a received block proposal.

**Vote validation.** At step $k \geq 3$, all nodes validate all unique votes they receive before propagating them in the gossip network. For each vote, nodes verify the signature of the committee member that initiated the vote. Nodes then verify the sortition for the committee member using the committee credential that is included with the message. Similarly to block validation, we assume that nodes validate each unique vote once even if they receive the vote multiple times from different peers. We let $c_{vv}$ represent the costs incurred by each node for validating each vote.

**Message propagation.** All nodes gossip unique validated messages that they receive to their peers. Committee members additionally propagate their generated messages to their peers. A message is either a block proposal or a vote. The size of a block proposal is usually larger than that of a vote. Therefore, gossiping a block proposal has a higher communication cost compared to gossiping a vote. We use $c_{bp}$ to denote the cost of sending a block proposal and $c_{vp}$ to denote the cost of sending a vote to each peer.

## 3.2 Total Cost Model

The total cost incurred by any node $i$ at step $k$ is denoted by $C_i(k)$, and it has two components: (a) baseline costs, $C_i^b(k)$, and (b) committee costs, $C^c(k)$. We model $C_i(k)$ as follows.

$$C_i(k) = C_i^b(k) + C^c(k) \times \mathbb{1}(.x_{i,k} < p_c)^1. \tag{3.1}$$

**Baseline costs.** $C_i^b(k)$ represents the baseline costs that do not depend on whether node $i$ is selected as a committee member at step $k$. We model $C_i^b(k)$ as follows.

$$C_i^b(k) = \begin{cases} c_{cs} & \text{if } k = 1, \\ c_{cs} + m_i(k) \cdot c_{bv} + \bar{m}_i(k) \cdot n_{rp} \cdot c_{bp} & \text{if } k = 2, \\ c_{cs} + m_i(k) \cdot c_{vv} + \bar{m}_i(k) \cdot n_{rp} \cdot c_{vp} & \text{if } k \geq 3. \end{cases}$$

In this formula, $n_{rp}$ is the number of each node's peers (i.e., $|N_i| = n_{rp}$, where $N_i$ is the set of node $i$'s randomly selected peers). $m_i(k)$ is the number of unique messages (block proposals or votes) received by node $i$ at step $k$. And $\bar{m}_i(k)$ is the number of unique validated messages received by node $i$ at step $k$, which depends on the size of the committee at step $k$.

The cost of running the cryptographic sortition is incurred at every step as a baseline cost. At every step $k \geq 2$, nodes validate every unique message they receive. They then propagate all validated messages to their randomly selected peers. At step $k = 2$, the messages are block proposals, and at step $k \geq 3$, the messages are committee votes. In general, $m_i(k)$ could be greater than $\bar{m}_i(k)$ as Byzantine nodes might send invalid messages to their peers. Non-Byzantine nodes do not propagate invalid messages and stop accepting any further messages from nodes that propagated invalid messages to them in previous steps. Therefore, we have the following inequality at any step $k'$.

$$\sum_{k=1}^{k'} m_i(k) \leq |\hat{N}_i| + \sum_{k=1}^{k'} \bar{m}_i(k),$$

where $\hat{N}_i$ is the set of nodes of which node $i$ is a randomly selected peer. Since invalid messages do not impact the protocol, in this paper, we assume that Byzantine nodes do not send such messages (i.e., $m_i(k) = \bar{m}_i(k)$). The expected number of unique valid messages per step is equal to the expected number of committee members. Therefore, the average baseline cost for agent $i$ at step $k$, $\bar{C}^b(k)$, can be derived by setting $m_i(k) = \bar{m}_i(k) = \tau$.

**Committee costs.** $C^c$ represents the committee costs incurred by a node when it is selected as a committee member at step $k$. We model $C^c(k)$ as follows.

$$C^c(k) = \begin{cases} c_{bg} + n_{rp} \cdot c_{bp} & k = 1, \\ c_{vg}(k) + n_{rp} \cdot c_{vp} & k \geq 2. \end{cases}$$

At every step $k \geq 1$, nodes that are selected as a committee member have to generate a message and broadcast it to all their peers. At step $k = 1$, the message is a block proposal, and at step $k \geq 2$, the message is a committee vote.

[1] $\mathbb{1}(\cdot)$ is the indicator function which returns 1 if the condition is true, and 0 otherwise.

# Chapter 4

# Proposed Game Model

To study nodes' incentives, we model the participation of nodes in the Algorand protocol as a Bayesian game. We formally define the Algorand game and describe nodes' strategies and utilities. We then discuss solution concepts for our proposed game.

## 4.1 The Algorand Game

To study nodes' incentives for participation in the Algorand protocol, we use Bayesian games. A Bayesian game consists of a set of agents. Each agent has a type and a set of available actions. Agents do not know their types before the start of the game. They, however, know a common prior probability distribution over types. At the beginning of the game, each agent privately observes its own type. Agents then simultaneously take their actions without knowing each others' types. Finally, agents receive a real-valued utility (also known as payoff) given their joint types and actions. A Bayesian game is formally defined as follows.

**Definition 1** (Bayesian Game [64])**.** *A Bayesian game is represented by a tuple* $(N, A, \Theta, p, u)$ *where:*

- $N = \{1, \ldots, n\}$ *is a set of agents;*
- $A = A_1 \times \cdots \times A_n$*, where* $A_i$ *is a set of actions available to agent* $i$*;*
- $\Theta = \Theta_1 \times \cdots \times \Theta_n$*, where* $\Theta_i$ *is the type space of agent* $i$*;*

- $P : \Theta \mapsto [0, 1]$ *is a common prior probability distribution over types; and*
- $u = (u_1, \ldots, u_n)$*, where* $u_i : A \times \Theta \mapsto \mathbb{R}$ *is the utility function for agent* $i$*.*

**Agents and actions.** In our setting, agents represent Algorand nodes. Agents are assumed to be rational in the sense that they selfishly choose an action to maximize their utility function. We consider three actions: (i) cooperate, $C$, (ii) defect, $D$, (iii) misbehave, $M$. A cooperative agent fully runs the Algorand protocol's code and consequently incurs all the processing and communication costs associated with it. A defective agent does not run any code (e.g., logs off from the system) and incurs no costs. A misbehaving agent runs a malicious code to sabotage the system. Node that the malicious code can imitate the behaviour of cooperating or defecting. Non-Byzantine agents that are not corrupted by the adversary do not misbehave. They only choose between cooperating and defecting. They cooperate if and only if their expected rewards exceed their expected costs. Byzantine agents, however, always misbehave (i.e., they run the adversary's malicious code).

Formally, we denote the action of agent $i$ by $a_i \in A_i = \{C, D, M\}$. A vector of actions $a = (a_1, \ldots, a_n) \in A$ is called an action profile. An action profile $a$ can be written as $(a_i, a_{-i})$, where $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$ is an action profile without agent $i$'s action[1].

**Types.** We define the type of each agent $i$ to be $\theta_i = (\theta_{i,0}, \theta_{i,1}, \ldots, \theta_{i,\mathrm{K_{max}}})$, where $\theta_{i,0} \in \{0, 1\}$ indicates if agent $i$ is corrupted by the adversary (1 if $i$ is Byzantine and zero otherwise), and $\theta_{i,k}$ is a number between 0 and 1 represented in binary by the hash result of the sortition algorithm run by agent $i$ at step $k = 1, \ldots, \mathrm{K_{max}}$ ($\theta_{i,k} = .x_{i,k}$, where $x_{i,k}$ is returned by $\texttt{Sortition}_i(k)$). The type space of agent $i$ is denoted as $\Theta_i$. At the beginning of the Algorand game, each agent observers whether it is corrupted by the adversary. Agents also receive a random seed for the cryptographic sortition algorithm. Given the random seed, agents can run the sortition algorithm for all steps to know what exactly their type vector is.

**Prior probabilities.** The probability that agent $i$ is corrupted by the adversary is $\mathbb{P}(\theta_{i,0} = 1) = p_b$. For $k = 1, \ldots, \mathrm{K_{max}}$, $\theta_{i,k}$'s are drawn independently for a uniform distribution between 0 and 1. Therefore, the probability that agent $i$ is selected as a committee member is $\mathbb{P}(\theta_{i,k} \leq p_c) = p_c$. The adversary corrupts agents independently. Agents also are selected as committee members at any given step independently.

**Utility functions.** For agent $i$, the utility function, $u_i$, maps action profiles, $a = (a_1, \ldots, a_n)$, and type vectors, $\theta = (\theta_1, \ldots, \theta_n)$, to real-valued payoffs. If $\theta_{i,0} = 1$, then

[1]Throughout the paper, we use $-i$ to denote all agents except agent $i$.

we assume that $u_i(a, \theta)$ is $-B$ if $a_i \in \{C, D\}$ and 0 if $a_i = M$ where $B$ is a large real number. Under this assumption, regardless of their type, Byzantine agents always prefer $M$ to $C$ and $D$. To model non-Byzantine agents preferences, we assume that if $\theta_{i,0} = 0$, then $u_i(a, \theta) = -B$ if $a_i = M$. For $a_i \in \{C, D\}$, the utility of non-Byzantine agents is equal to the rewards they receive minus the costs they incur. We analyze utility functions in more details in §5.

Table A.1 in Appendix A provides detailed description of all the parameters used in our game model and analysis.

## 4.2 Strategies and Equilibria

A strategy defines a description of how a game would be played in every contingency. In a Bayesian game, a strategy prescribes a distribution over actions for every type that an agent could have. Let $\Delta(A_i)$ be the set of all probability distributions over $A_i$. For agent $i$, a strategy $s_i : \Theta_i \mapsto \Delta(A_i)$ is a mapping from agent $i$'s types to distributions over agent $i$'s actions. The set of all strategies for agent $i$ is denoted by $S_i$. By $s_i(a_i \mid \theta_i)$, we indicate the probability that agent $i$ takes action $a_i$ under $s_i$ given that agent $i$'s type is $\theta_i$. Similar to action profiles, a strategy profile $s = (s_1, \ldots, s_n) \in S$ is a vector of strategies, where $S = S_1 \times \cdots \times S_n$ is a set of all possible strategy profiles.

**Expected utilities.** In Bayesian games, there are two main sources of uncertainty: (i) types and (ii) actions. Types are drawn from the prior probability distribution, $P$, and actions are taken based on agents' strategies. To capture both sources of uncertainty, the ex ante expected utility of agent $i$ is modeled as follows.

$$EU_i(s) = \mathbb{E}_{a,\theta}[u_i(a, \theta)] = p_b \cdot \mathbb{E}_{a,\theta}[u_i(a, \theta) \mid \theta_{i,0} = 1] + (1 - p_b) \cdot \mathbb{E}_{a,\theta}[u_i(a, \theta) \mid \theta_{i,0} = 0].$$

The expectation is taken with respect to $\theta$ and $a \sim s(\cdot \mid \theta)$. This formula models the expected utility of agent $i$ before the start of the game and before the agent observes its type.

Given the defined expected utility model, we can define the set of agent $i$'s best responses to strategy profile $s_{-i}$ as:

$$BR_i(s_{-i}) = \arg\max_{s_i \in S_i} \; EU_i((s_i, s_{-i})).$$

Intuitively, a best response is a strategy which provides the highest expected utility given the strategy of others. Note that there may be more than one strategy that maximizes agent $i$'s expected utility for a given $s_{-i}$.

**Bayesian Nash equilibrium.** As discussed before, a strategy is a full contingency plan. Agents **simultaneously** choose their strategies **before** the start of the game and do not change their adopted strategies during the game. Once the game starts, each agent observes its type and acts as prescribed by its strategy. Agents strategies form a Bayesian Nash equilibrium (BNE) when the strategy of each agent is a best response to the strategies adopted by other agents. Formally, a strategy profile $s^*$ is a BNE if and only if $s_i^* \in BR_i(s_{-i}^*)$, for all $i$. Informally, in a BNE, agent $i$ does not have any incentive to unilaterally change its strategy from $s_i^*$ if it knows that other agents have fixed their strategies to $s_{-i}^*$.

# Chapter 5

# Incentive Analysis in Algorand Game

In this section, we formulate agents' utilities and study their BNE strategies. We first consider Algorand's original reward scheme. We show that under this reward scheme, cooperation is not a BNE strategy. We then propose a novel reward scheme and show that under certain conditions, our proposed reward scheme incentivizes all non-Byzantine agents to cooperate regardless of their type.

## 5.1 Algorand's Original Reward Scheme

Algorand's original reward scheme is called Participation Rewards [6]. Under this reward scheme, *The Algorand Foundation* distributes a fixed amount of cryptocurrency assets as a reward among all agents. Agents are assigned sub-nodes in proportion to the balance of their accounts. Under Algorand's original reward scheme, the fixed reward, $R$, is distributed equally among all sub-nodes. If agent $i$ is assigned $w_i$ sub-nodes, then agent $i$'s reward, $R_i$, is equal to $R \cdot w_i/W$, where $W$ is the total number of sub-nodes in the system. The first advantage of this reward scheme is its simplicity: it is easy to implement, and it is easy to explain to agents how they are rewarded. The most important advantage of Algorand's original reward scheme is that it provides proportional rewards.

**Definition 2.** *Proportional rewards. Let $R_i$ denote the expected reward of agent $i$. A reward scheme provides proportional rewards if for any agent $i$ and $j$, $R_i/R_j = w_i/w_j$.*

The proportional rewards property ensures that the expected fraction of assets controlled by the adversary does not increase by the action of the reward scheme.

Although Algorand's original reward scheme is simple and provides proportional rewards, it suffers from a key drawback: it fails to prevent free riding. Agents do not have any incentive to cooperate as they receive their rewards irrespective of their cooperation. To formally analyze this, let us define $s^*$ and $a^*$ as follows.

**Theorem 3.** *Let $s^*$ be a strategy profile where for each agent $i$, if $\theta_{i,0} = 0$, then $s_i^*(C \mid \theta_i) = 1$, and otherwise, $s_i^*(M \mid \theta_i) = 1$. Under Algorand's original reward scheme, $s^*$ is not a BNE.*

*Proof.* Suppose agent $i$ is a non-Byzantine agent (i.e., $\theta_{i,0} = 0$). If agent $i$ defects, it receives its rewards without incurring any costs. Formally, $u_i(a, \theta) = R/n$ for all $\theta \in \Theta$ if $a_i = D$[1]. Let $s_i'$ be a strategy that chooses $D$ regardless of the type (i.e., $s_i'(D \mid \theta_i) = 1$ for all $\theta_i \in \Theta_i$). It can be easily shown that $EU_i((s_i', s_{-i}^*)) = (1 - p_b) \cdot R/n$. If a non-Byzantine agent $i$ cooperates, it receives its reward but incurs some strictly positive costs. This means that $u_i(a, \theta) < R/n$ if $a_i = C$. Let $a^*(\theta)$ be an action profile where $a_i^*$ is $C$ if $\theta_{i,0} = 0$ and $M$ otherwise. The expected utility of agent $i$ for $s^*$ can be written as:

$$\begin{aligned} EU_i(s^*) &= (1 - p_b) \cdot \mathbb{E}_\theta[u_i(a^*(\theta), \theta) \mid \theta_{i,0} = 0] \\ &< (1 - p_b) \cdot R/n = EU_i((s_i', s_{-i}^*)). \end{aligned}$$

This implies that $s_i^*$ is not a best response to $s_{-i}^*$, which in turn means that $s^*$ is not a BNE under Algorand's original reward distribution. □

## 5.2 Incentive-compatible Reward Scheme

To address the free-rider problem, we propose IRS, a novel incentive-compatible reward scheme for Algorand. Under IRS, the Algorand Foundation distributes rewards among agents based on their cooperation. The cooperation of committee members can be easily tracked as their messages are guaranteed to reach all other agents. However, tracking the cooperation of the agents that are not selected as a committee member is challenging as they do not initiate any messages. To address this challenge, our proposed reward scheme requires committee members at step $k = 2, \ldots, \mathrm{K_{max}}$ to include with their vote the identities of the agents from which they have received a valid message at step $k - 1$.

Let $R^c(k)$ and $R^b(k)$ denote a fixed baseline reward and a fixed committee reward at step $k$, respectively. Under IRS, a cooperating agent $i$ receives a committee reward of

[1]We present the proof for the case where all agents are assigned a single sub-node. Our proof easily extends to the case where agents are assigned different number of sub-nodes.

$R^c(k)$ if it is selected as a committee member at step $k$. Additionally, if agent $i$'s identity is included in the vote generated by agent $j \in N_i$ at step $k$, then agent $i$ receives a baseline reward of $R^b(k)/n_{rs}$. We assume that Byzantine agents do not include the identity of non-Byzantine agents in their vote when they are selected as committee members. In other words, non-Byzantine agents do not receive any baseline reward for propagating messages to their Byzantine peers. Given this assumption, the total reward of a cooperating agent $i$ at step $k$ given an action profile $a$ and a type vector $\theta$ can be formulated as:

$$R_i(a, \theta, k) = (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{1}(\theta_{j,k} \leq p_c, a_j = C) + R^c(k) \cdot \mathbb{1}(\theta_{i,k} \leq p_c). \tag{5.1}$$

To show that our reward scheme prevents the free-rider problem, we first prove that any non-Byzantine agent cannot unilaterally change the outcome of each step by defecting. Given that, we formulate the expected utility of each non-Byzantine agent assuming that all other non-Byzantine agents cooperate regardless of their type. We then prove that if certain conditions are met, cooperation is a best response for a non-Byzantine agent when other non-Byzantine agents cooperate. This then shows that cooperation is a BNE strategy for non-Byzantine agents under our proposed reward scheme.

**Lemma 4.** *Let $NR_k$ and $NB_k$ be random variables indicating the number of non-Byzantine and the number of Byzantine agents selected as committee members at any step $k$, respectively. Let $\mu_r = \mathbb{E}(NR_k)$, $\mu_b = \mathbb{E}(NB_k)$, $\mu = \mu_b + \mu_r/2$, $\delta_r = 1 - T/(1 - p_b)$, and $\delta = 2 \times T/(1 + p_b) - 1$.*

- $\mathbb{P}(NR_k \leq T \cdot \tau) \leq e^{-\mu_r \cdot \delta_r^2/2}$, *and*
- $\mathbb{P}(NB_k + NR_k/2 \geq T \cdot \tau) \leq e^{-\mu \cdot \delta^2/(2+\delta)}$.

*Proof.* Let $X_{i,k}$ be a random variable that takes value 1 if agent $i$ is selected as a committee member and agent $i$ is not corrupted by the adversary, and takes value 0 otherwise. Similarly, let $Y_{i,k}$ be a random variable that takes value 1 if agent $i$ is selected as a committee member and agent $i$ is corrupted by the adversary, and takes 0 otherwise. We can write $NR_k = \sum_i X_{i,k}$, and $NB_k = \sum_i Y_{i,k}$. We have $\mathbb{E}(X_{i,k}) = (1 - p_b) \cdot p_c$, and $\mathbb{E}(Y_{i,k}) = p_b \cdot p_c$. Therefore, $\mu_r = \mathbb{E}(NR_k) = \sum_i \mathbb{E}(X_{i,k}) = (1-p_b) \cdot \tau$, and $\mu_b = \mathbb{E}(NB_k) = \sum_i \mathbb{E}(Y_{i,k}) = p_b \cdot \tau$.

Since $\delta_r \geq 0$, according the the Chernoff bound, the following inequality holds for any $k$.

$$\mathbb{P}(NR_k \leq T \cdot \tau) = \mathbb{P}(NR_k \leq (1 - \delta_r) \cdot (1 - p_b) \cdot \tau) = \mathbb{P}(NR_k \leq (1 - \delta_r) \cdot \mu_r) \leq e^{-\mu_r \cdot \delta_r^2/2}.$$

Similarly, since $\delta \geq 0$, we have the following inequality for any $k$.

$$\begin{aligned}\mathbb{P}(NB_k + NR_k/2 \geq T \cdot \tau) &= \mathbb{P}(NB_k + NR_k/2 \geq (1+\delta) \cdot (1+p_b) \cdot \tau/2) \\ &= \mathbb{P}(NB_k + NR_k/2 \geq (1+\delta) \cdot \mu) \leq e^{-\mu \cdot \delta^2/(2+\delta)}.\end{aligned}$$

□

We assume that $p_b$, $\tau$, and $T$ are set such that $NR_k > T \cdot \tau$ and $NB_k + NR_k/2 < T \cdot \tau$ are true with overwhelming probability. For example, using Lemma (4), with $p_b = 0.2$, $\tau = 4000$ and $T = 0.7$, the probability that $NR_k \leq T \cdot \tau$ is less than $10^{-10}$ and the probability that $NB_k + NR_k/2 \geq T \cdot \tau$ is less than $10^{-13}$. A main implication of the two inequalities is that more than 2/3 of the selected committee members at each step are non-Byzantine agents with overwhelming probability.

**Proposition 5.** *Consider any agent $i$. Suppose that the strategy profile of all agents except agent $i$ is $s^*_{-i}$ where for each agent $j \in N \setminus i$, if $\theta_{j,0} = 0$, then $s^*_j(C \mid \theta_j) = 1$, and $s^*_j(M \mid \theta_j) = 1$ otherwise. In a large system (i.e., $n \to \infty$), the safety and liveness guarantees of the Algorand protocol are met with high probability regardless of the strategy of agent $i$.*

*Proof.* Assume that $p_b$, $\tau$, and $T$ are set such that $NR_k > T \cdot \tau$ and $NB_k + NR_k/2 < T \cdot \tau$ with overwhelming probability. Consider a new system consisting of all agents except agent $i$. For large systems, it is easy to modify Lemma (4) to show that the two inequalities still hold for the new system with the same $p_b$, $\tau$, and $T$. Moreover, the Chernoff bound can be applied to show that with $p_b \leq 1/3$, the network is synchronous with high probability regardless of agent $i$'s cooperation. Given that the network is synchronous, and the two inequalities hold with high probability, Theorem 1 from [24] can be applied to the new system to guarantee safety and liveness of the system. □

We next formulate the expected utility of each agents. The Algorand protocol implements a randomized algorithm. As discussed in §2.8, the algorithm runs in multiple steps. We use $K(a, \theta)$ to denote the total number of steps it takes the algorithm to complete as a function of agents' types and their actions. Given $a$ and $\theta$, the total utility of agent $i$ with $\theta_{i,0} = 0$ can be formulated as:

$$u_i(a, \theta) = \sum_{k=1}^{K(a,\theta)} u_i(a, \theta, k),$$

where $u_i(a, \theta, k)$ is $R_i(a, \theta, k) - C_i(a, \theta, k)$[2] if $a_i = C$ and 0 otherwise.

**Lemma 6.** *Consider any agent $i$. Suppose that the strategy profile of all agents except agent $i$ is $s^*_{-i}$ where for each agent $j \in N \setminus i$, if $\theta_{j,0} = 0$, then $s^*_j(C \mid \theta_j) = 1$, and $s^*_j(M \mid \theta_j) = 1$ otherwise. Suppose further that agent $i$ adopts strategy $s_i$ which plays $M$ if $\theta_{i,0} = 1$ and plays $C$ with probability $s_c$ and $D$ with probability $1 - s_c$ if $\theta_{i,0} = 0$. Let $a^*(\theta)$ be an action profile where $a^*_i$ is $C$ if $\theta_{i,0} = 0$ and $M$ otherwise. Define $K(\theta) = K(a^*(\theta), \theta)$. Under IRS, the expected utility of agent $i$ is:*

$$EU_i((s_i, s^*_{-i})) = s_c \cdot (1 - p_b) \cdot \sum_{\ell=1}^{\mathrm{K_{max}}} \mathbb{P}(K(\theta) = \ell) \sum_{k=1}^{\ell} \bar{u}(k),$$

*where $\bar{u}(k) = R^b(k) \cdot p_c \cdot (1 - p_b) - \bar{C}^b(k) + (R^c(k) - C^c(k)) \cdot p_c$.*

*Proof.* If agent $i$ is corrupted by the adversary, it strictly prefers action $M$, which leads to a payoff of zero as defined in §4.1. If agent $i$ is not corrupted, it takes action $D$ with probability $1 - s_c$ for a payoff of zero. We can write the expected utility of agent $i$ for $(s_i, s^*_{-i})$ as:

$$EU_i((s_i, s^*_{-i})) = s_c \cdot (1 - p_b) \cdot \mathbb{E}_\theta \left[ \sum_{k=1}^{K(\theta)} u_i(a^*(\theta), \theta, k) \;\middle|\; \theta_{i,0} = 0 \right].$$

We can write the expectation in the above formula as follows.

$$\begin{aligned}
&\mathbb{E}_\theta \left[ \sum_{k=1}^{K(\theta)} u_i(a^*(\theta), \theta, k) \;\middle|\; \theta_{i,0} = 0 \right] \\
&= \sum_{\ell=1}^{\mathrm{K_{max}}} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ \sum_{k=1}^{\ell} u_i(a^*(\theta), \theta, k) \;\middle|\; \theta_{i,0} = 0, K(\theta) = \ell \right] \\
&= \sum_{\ell=1}^{\mathrm{K_{max}}} \mathbb{P}(K(\theta) = \ell) \sum_{k=1}^{\ell} \mathbb{E}_\theta \left[ u_i(a^*(\theta), \theta, k) \mid \theta_{i,0} = 0, K(\theta) = \ell \right] \\
&= \sum_{\ell=1}^{\mathrm{K_{max}}} \sum_{k=1}^{\ell} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ R_i(a^*(\theta), \theta, k) - C_i(a^*(\theta), \theta, k) \mid \theta_{i,0} = 0, K(\theta) = \ell \right].
\end{aligned}$$

[2] In Equation (3.1), $C_i$ depends on $\theta$ through $.x_{i,k}$, and it is formulated assuming that agent $i$ cooperates.

The second equality holds because the expected value of the sum of random variables is equal to the sum of their expectations. Given Equation (5.1), we have the following for $R_i$.

$$
\begin{aligned}
&\mathbb{P}(K(\theta)=\ell)\cdot\mathbb{E}_\theta\left[R_i(a^*(\theta),\theta,k)\mid\theta_{i,0}=0,K(\theta)=\ell\right]\\
&\quad=(R^b(k)/n_{rs})\sum_{j\in N_i}\mathbb{P}(K(\theta)=\ell)\cdot\mathbb{P}(\theta_{j,k}\le p_c,a_j^*(\theta)=C\mid\theta_{i,0}=0,K(\theta)=\ell)\\
&\qquad+R^c(k)\cdot\mathbb{P}(K(\theta)=\ell)\cdot\mathbb{P}(\theta_{i,k}\le p_c\mid\theta_{i,0}=0,K(\theta)=\ell)\\
&\quad=(R^b(k)/n_{rs})\sum_{j\in N_i}\mathbb{P}(K(\theta)=\ell)\cdot\mathbb{P}(\theta_{j,k}\le p_c,\theta_{j,0}=0\mid K(\theta)=\ell)\\
&\qquad+R^c(k)\cdot\mathbb{P}(K(\theta)=\ell)\cdot\mathbb{P}(\theta_{i,k}\le p_c\mid K(\theta)=\ell)\\
&\quad=(R^b(k)/n_{rs})\sum_{j\in N_i}\mathbb{P}(\theta_{j,k}\le p_c,\theta_{j,k}=0)\cdot\mathbb{P}(K(\theta)=\ell\mid\theta_{j,k}\le p_c,\theta_{j,0}=0)\\
&\qquad+R^c(k)\cdot\mathbb{P}(\theta_{i,k}\le p_c)\cdot\mathbb{P}(K(\theta)=\ell\mid\theta_{i,k}\le p_c)\\
&\quad=(R^b(k)/n_{rs})\sum_{j\in N_i}p_c\cdot(1-p_b)\cdot\mathbb{P}(K(\theta)=\ell\mid\theta_{j,k}\le p_c,\theta_{j,0}=0)\\
&\qquad+R^c(k)\cdot p_c\cdot\mathbb{P}(K(\theta)=\ell\mid\theta_{i,k}\le p_c).
\end{aligned}
$$

The second equality holds since there is no dependency between any two elements of $\theta$, and according to the definition of $a^*$, $a_j^*(\theta)=C$ if and only if $\theta_{j,0}=0$. The third equality follows simply from Bayes' rule.

The number of steps needed to complete the Algorand protocol depends on whether the highest-priority committee member at steps $k\in\{1,7,10,\ldots,\mathrm{K_{max}}\}$ is corrupted (see Theorem 1 in [24]). Let $E_j$ be a binary random variable that takes value 1 if agent $j$ is not the highest-priority committee member at any of steps $k\in\{1,7,10,\ldots,\mathrm{K_{max}}\}$ and takes value 0 otherwise. If $E_j=1$, then agent $j$'s action and type do not affect the number of steps it takes the Algorand protocol to complete.

Agent $j$ becomes the highest-priority committee member at any step $k$ if $\theta_{j,k}<\theta_{j',k}$ for all $j'\in N$. $\theta_{j,k}$ is uniformly distributed between 0 and 1. Therefore, the probability that an agent is the highest-priority committee member at any steps is $1/n$. Since $\theta_{j,k}$'s are independent and identically distributed, we have $\mathbb{P}(E_j=0)\le\mathrm{K_{max}}/3n$, and we can

write:

$$\begin{aligned}
&\mathbb{P}(K(\theta)=\ell \mid \theta_{j,k}\le p_c, \theta_{j,k}=0)\\
&\quad=\mathbb{P}(K(\theta)=\ell \mid \theta_{j,k}\le p_c, \theta_{j,k}=0, E_j=0)\cdot\mathbb{P}(E_j=0)\\
&\qquad+\mathbb{P}(K(\theta)=\ell \mid \theta_{j,k}\le p_c, \theta_{j,k}=0, E_j=1)\cdot\mathbb{P}(E_j=1)\\
&\quad\simeq\mathbb{P}(K(\theta)=\ell \mid \theta_{j,k}\le p_c, \theta_{j,k}=0, E_j=1)\\
&\quad=\mathbb{P}(K(\theta)=\ell \mid E_j=1)\\
&\quad=\mathbb{P}(K(\theta)=\ell). \qquad (5.2)
\end{aligned}$$

The first equality follows from the law of total probability. The second equality holds asymptotically since $\mathbb{P}(E_j=0)$ goes to zero as $n$ goes to infinity. The third equality holds since given $E_j=1$, the number of steps needed to complete the protocol does not depend on agent $j$'s type. Finally, the fourth equality holds since $E_j=1$ and $K(\theta)=\ell$ are independent events. Knowing that agent $j$ is not the highest-priority agent at some steps does not reveal any information about the number of steps needed to complete the protocol. The same argument can be applied to show that $\mathbb{P}(K(\theta)=\ell \mid \theta_{i,k}\le p_c)=\mathbb{P}(K(\theta)=\ell)$.

Given Equation (3.1), we have the following for $C_i$.

$$\begin{aligned}
&\mathbb{P}(K(\theta)=\ell)\cdot\mathbb{E}_\theta\left[C_i(a^*(\theta),\theta,k)\mid\theta_{i,0}=0,K(\theta)=\ell\right]\\
&\quad=\mathbb{P}(K(\theta)=\ell)\cdot\left(\bar{C}^b(k)+C^c(k)\cdot\mathbb{P}(\theta_{i,k}\le p_c\mid\theta_{i,0}=0,K(\theta)=\ell)\right)\\
&\quad=\mathbb{P}(K(\theta)=\ell)\cdot\bar{C}^b(k)+C^c(k)\cdot\mathbb{P}(\theta_{i,k}\le p_c)\cdot\mathbb{P}(K(\theta)=\ell\mid\theta_{i,k}\le p_c)\\
&\quad=\mathbb{P}(K(\theta)=\ell)\cdot\bar{C}^b(k)+C^c(k)\cdot p_c\cdot\mathbb{P}(K(\theta)=\ell). \qquad (5.3)
\end{aligned}$$

The second equality follows from Bayes' rule, and the third equality holds as shown for $R_i$. Note that for this equality equality, we also use the fact that there is no dependency between $\theta_{i,0}$ and $\theta_{i,k}$. This completes the proof of the lemma. □

Next, we prove that $s^*$ is a BNE under certain conditions.

**Theorem 7.** *Let $s^*$ be a strategy profile where for each agent $i$, if $\theta_{i,0}=0$, then $s_i^*(C\mid\theta_i)=1$, and otherwise, $s_i^*(M\mid\theta_i)=1$. Given $p_c$, $p_b$, $\bar{C}^b(k)$, and $C^c(k)$ for $k=1,\dots,\mathrm{K}_{\max}$, $s^*$ is a BNE under IRS if:*

$$R^b(k)(1-p_b)+R^c(k)\ge\bar{C}^b(k)/p_c+C^c(k),\quad\forall k=1,\dots,\mathrm{K}_{\max}.$$

*Proof.* To prove that $s^*$ is a BNE, it suffices to show that $s_i^*$ is a best response to $s_{-i}^*$ for all agent $i\in N$. Recall that $s_i^*$ is a best response to $s_{-i}^*$ when $EU_i((s_i^*,s_{-i}^*))\ge EU_i((s_i,s_{-i}^*))$

for all $s_i \in S_i$. If $R^b(k) \cdot (1 - p_b) + R^c(k) \geq \bar{C}^b(k)/p_c + C^c(k)$ for all $k = 1, \ldots, \mathrm{K_{max}}$, then $\bar{u}(k)$ in Lemma (6) is greater than or equal to zero for all $k = 1, \ldots, \mathrm{K_{max}}$. Consequently, $EU_i((s_i, s^*_{-i}))$ is maximized when $s_c = 1$. In this case, we have $s = s^*$ which means $s^*$ is a BNE. □

# Chapter 6

# Expected Rewards under IRS

In §5, we showed that cooperating is a BNE strategy for non-Byzantine agents under the IRS reward scheme. In this section, we study expected rewards gained by Byzantine and non-Byzantine nodes. If there is no adversary (i.e., $p_b = 0$), then in equilibrium, all agents receive the same expected rewards under IRS according to Lemma (6) and Theorem (7). However, in the presence of an adversary, IRS could fail to provide proportional rewards. This is mainly because IRS relies on selected agents to refer cooperation of other agents. Non-Byzantine agents refer both Byzantine and non-Byzantine agents. Byzantine agents, however, do not refer non-Byzantine agents. If we assume that Byzantine agents do refer other Byzantine agents, then the adversary has the opportunity to receive a higher expected reward per Byzantine agent than non-Byzantine agents[1]. This could pose a risk to Algorand as the expected proportion of assets controlled by the adversary could increase under IRS.

## 6.1 Referral Mechanism

To prevent corrupted nodes to receive higher expected rewards than non-corrupted nodes, we propose a referral reward mechanism. Under this mechanism, selected committee members receive extra rewards for every agent that they refer in their vote. More specifically, for every agent that is referred in a committee member's vote, the committee member receives a referral reward of $R^r(k)/|\hat{N}_i|$, where $R^r(k)$ is a fixed referral reward at step $k$. The addition of referral rewards poses a dilemma for the adversary. If a Byzantine agent

[1] This could be achieved when, for example, the adversary code fully mimics the protocol except referring the cooperation of the other agents.

propagates messages to its non-Byzantine peers, it gains expected baseline rewards but enables non-Byzantine agents to claim referral rewards. Similarly, if a Byzantine committee member refers its non-Byzantine neighbors, it gains referral rewards, but that allows non-Byzantine agents to gain baseline rewards. In the remainder of this section, we consider four cases for the adversary code: (I) does not propagate messages to non-Byzantine peers but refers all neighbors, (II) does not propagate messages to non-Byzantine peers and only refers Byzantine neighbors, (III) propagates messages to all peers but only refers Byzantine neighbors, and (IV) propagates messages to all peers and refers all neighbors.

The total reward of agent $i$ given action profile $a$, and type vector $\theta$ under IRS with referral rewards can be formulated as follows.

$$R_i(a,\theta) = \sum_{k=1}^{K(a,\theta)} R_i^b(a,\theta,k) + R_i^c(a,\theta,k) + R_i^r(a,\theta,k).$$

In this formula, $R_i^c(a,\theta,k) = R^c(k) \cdot \mathbb{1}(\theta_{i,k} \leq p_c)$ for all four cases. However, the definition of $R_i^b(a,\theta,k)$ depends on each case as follows.

$$R_i^b(a,\theta,k) = \begin{cases} (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{1}(\theta_{j,k} \leq p_c, a_j \in \{C,M\}) & \text{for (I)/(IV) if } \theta_{i,0} = 0, \\ (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{1}(\theta_{j,k} \leq p_c, a_j = C) & \text{for (II)/(III) if } \theta_{i,0} = 0, \\ (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{1}(\theta_{j,k} \leq p_c, a_j \in \{C,M\}) & \text{for (III)/(IV) if } \theta_{i,0} = 1, \\ (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{1}(\theta_{j,k} \leq p_c, a_j = M) & \text{for (I)/(II) if } \theta_{i,0} = 1. \end{cases}$$

For cases (I) and (IV), Byzantine committee members refer all their neighbors. Therefore, if $\theta_{i,0} = 0$, then agent $i$ receives baseline rewards for being referred by any cooperating or misbehaving peer that is selected as a committee member. For cases (II) and (III), Byzantine committee members only refer their Byzantine neighbors. This means that if $\theta_{i,0} = 0$, then agent $i$ receives baseline rewards only for being referred by cooperating committee members among its peers. For cases (III) and (IV), Byzantine agents propagate messages to all their peers. Therefore, if $\theta_{i,0} = 1$, agent $i$ receives baseline rewards for being referred by any cooperating or misbehaving peer that is selected as a committee member. Finally, for cases (I) and (II), Byzantine agents do not propagate messages to their non-Byzantine peers. For these two cases, if $\theta_{i,0} = 1$, agent $i$ receives baseline rewards for being referred by misbehaving committee members among its neighbors.

Similarly, $R_i^r(a, \theta, k)$ is defined as follows.

$$R_i^r(a,\theta,k) = \begin{cases} (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{1}(\theta_{i,k} \leq p_c, a_j = C) & \text{for (I)/(II) if } \theta_{i,0} = 0, \\ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{1}(\theta_{i,k} \leq p_c, a_j \in \{C, M\}) & \text{for (III)/(IV) if } \theta_{i,0} = 0, \\ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{1}(\theta_{i,k} \leq p_c) & \text{for (I)/(IV) if } \theta_{i,0} = 1, \\ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{1}(\theta_{i,k} \leq p_c, \theta_{j,0} = 1) & \text{for (II)/(III) if } \theta_{i,0} = 1. \end{cases}$$

For cases (I) and (II), Byzantine agents only propagate messages to their Byzantine neighbors. Therefore, if $\theta_{i,0} = 0$, then committee member $i$ can only claim referral rewards for cooperating peers. For cases (III) and (IV), Byzantine agents propagate messages to all their neighbors. This means that if $\theta_{i,0} = 0$, then committee member $i$ receives referral rewards for referring any cooperating or misbehaving peer. For cases (I) and (IV), Byzantine agents refer all their neighbors. Therefore, if $\theta_{i,0} = 1$, committee member $i$ receives referral rewards for all its neighbors. Finally, for cases (II) and (III), Byzantine agents only refer their Byzantine neighbors. For these two cases, if $\theta_{i,0} = 1$, committee member $i$ only claims referral rewards for its Byzantine neighbors.

We next formulate conditional expectation of rewards of an agent given that agent being a non-Byzantine or a Byzantine agent.

**Lemma 8.** *Consider any agent $i$. Suppose that the strategy profile of all agents except agent $i$ is $s^*_{-i}$ where for each agent $j \in N \setminus i$, if $\theta_{j,0} = 0$, then $s^*_j(C \mid \theta_j) = 1$, and $s^*_j(M \mid \theta_j) = 1$ otherwise. Suppose further that agent $i$ adopts strategy $s_i$ which plays $M$ if $\theta_{i,0} = 1$ and plays $C$ with probability $s_c$ and $D$ with probability $1 - s_c$ if $\theta_{i,0} = 0$. Let $a^*(\theta)$ be an action profile where $a^*_i$ is $C$ if $\theta_{i,0} = 0$ and $M$ otherwise. Define $K(\theta) = K(a^*(\theta), \theta)$. Under IRS with referral rewards, the conditional expectation of agent $i$'s reward given $\theta_{i,0} = 0$ is:*

$$\mathbb{E}_{a,\theta}[R_i(a,\theta) \mid \theta_{i,0} = 0] = s_c \cdot \sum_{\ell=1}^{\mathrm{K_{max}}} \mathbb{P}(K(\theta) = \ell) \sum_{k=1}^{\ell} \bar{R}(k),$$

*where $\bar{R}(k)$ is defined as:*

$$\bar{R}(k) = \begin{cases} R^b(k) \cdot p_c + R^c(k) \cdot p_c + R^r(k) \cdot p_c \cdot (1 - p_b) & \textit{for (I)}, \\ R^b(k) \cdot p_c \cdot (1 - p_b) + R^c(k) \cdot p_c + R^r(k) \cdot p_c \cdot (1 - p_b) & \textit{for (II)}, \\ R^b(k) \cdot p_c \cdot (1 - p_b) + R^c(k) \cdot p_c + R^r(k) \cdot p_c & \textit{for (III)}, \\ R^b(k) \cdot p_c + R^c(k) \cdot p_c + R^r(k) \cdot p_c & \textit{for (IV)}. \end{cases}$$

*Similarly, the conditional expectation of agent $i$'s reward given $\theta_{i,0} = 1$ is:*

$$\mathbb{E}_{a,\theta}[R_i(a,\theta) \mid \theta_{i,0} = 1] = \sum_{\ell=1}^{\mathrm{K_{max}}} \mathbb{P}(K(\theta) = \ell) \sum_{k=1}^{\ell} \bar{R}_b(k),$$

*where* $\bar{R}_b(k)$ *is defined as:*

$$\bar{R}_b(k) = \begin{cases} R^b(k) \cdot p_c \cdot p_b + R^c(k) \cdot p_c + R^r(k) \cdot p_c & \textit{for (I)}, \\ R^b(k) \cdot p_c \cdot p_b + R^c(k) \cdot p_c + R^r(k) \cdot p_c \cdot p_b & \textit{for (II)}, \\ R^b(k) \cdot p_c + R^c(k) \cdot p_c + R^r(k) \cdot p_c \cdot p_b & \textit{for (III)}, \\ R^b(k) \cdot p_c + R^c(k) \cdot p_c + R^r(k) \cdot p_c & \textit{for (IV)}. \end{cases}$$

*Proof.* Using the same techniques in the proof of Lemma (6), we can write:

$$\begin{aligned} &\mathbb{E}_{a,\theta}[R_i(a^*(\theta), \theta) \mid \theta_{i,0} = 0] \\ &= s_c \cdot \mathbb{E}_\theta \left[ \sum_{k=1}^{K(\theta)} R_i(a^*(\theta), \theta) \,\middle|\, \theta_{i,0} = 0 \right] \\ &= s_c \cdot \sum_{\ell=1}^{\mathrm{K_{max}}} \sum_{k=1}^{\ell} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ R_i(a^*(\theta), \theta) \mid \theta_{i,0} = 0, K(\theta) = \ell \right]. \end{aligned}$$

For the remaining of the proof, we focus on case (I) in the definition of $R_i(a, \theta)$. The proof easily extends to other three cases. We first consider agent $i$ with $\theta_{i,0} = 0$.

$$\begin{aligned} &\mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ R_i(a^*(\theta), \theta) \mid \theta_{i,0} = 0, K(\theta) = \ell \right] \\ &\quad = (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{j,k} \le p_c, a_j^*(\theta) \in \{C, M\} \mid \theta_{i,0} = 0, K(\theta) = \ell) \\ &\qquad + R^c(k) \cdot \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{i,k} \le p_c \mid \theta_{i,0} = 0, K(\theta) = \ell) \\ &\qquad + (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{j,k} \le p_c, a_j^*(\theta) = C \mid \theta_{i,0} = 0, K(\theta) = \ell) \\ &\quad = (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{j,k} \le p_c \mid K(\theta) = \ell) \\ &\qquad + R^c(k) \cdot \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{i,k} \le p_c \mid K(\theta) = \ell) \\ &\qquad + (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{j,k} \le p_c, \theta_{j,0} = 0 \mid K(\theta) = \ell) \end{aligned}$$

$$
\begin{aligned}
&= (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{P}(\theta_{j,k} \le p_c) \cdot \mathbb{P}(K(\theta) = \ell \mid \theta_{j,k} \le p_c) \\
&\quad + R^c(k) \cdot \mathbb{P}(\theta_{i,k} \le p_c) \cdot \mathbb{P}(K(\theta) = \ell \mid \theta_{i,k} \le p_c) \\
&\quad + (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{P}(\theta_{j,k} \le p_c, \theta_{j,0} = 0) \cdot \mathbb{P}(K(\theta) = \ell \mid \theta_{j,k} \le p_c, \theta_{j,0} = 0) \\
&= (R^b(k)/n_{rs}) \sum_{j \in N_i} p_c \cdot \mathbb{P}(K(\theta) = \ell) \\
&\quad + R^c(k) \cdot p_c \cdot \mathbb{P}(K(\theta) = \ell) \\
&\quad + (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} p_c \cdot (1 - p_b) \cdot \mathbb{P}(K(\theta) = \ell) \\
&= (R^b(k) \cdot p_c + R^c(k) \cdot p_c + R^r(k) \cdot p_c \cdot (1 - p_b)) \cdot \mathbb{P}(K(\theta) = \ell).
\end{aligned}
$$

The second equality holds because there is no dependency between any two elements of $\theta$. Moreover, given the definition of $a^*$, $a_j^*(\theta) \in \{C, M\}$ for all $\theta$, and $a_j^*(\theta) = C$ if and only if $\theta_{j,0} = 0$. The third equality follows from Bayes' rule. The fourth equality holds due to Equation (5.2) and the fact that $K(\theta) = \ell$ is independent from $\theta_{i,k} \le p_c$ for all $i$.

Next, we consider agent $i$ with $\theta_{i,0} = 1$. When agent $i$ is corrupted by the adversary, it strictly prefers action $M$. We can write the conditional expectation of agent $i$'s reward for $(s_i, s_{-i}^*)$ as:

$$
\begin{aligned}
&\mathbb{E}_{a,\theta}[R_i(a^*(\theta), \theta) \mid \theta_{i,0} = 1] \\
&\quad = \mathbb{E}_\theta \left[ \sum_{k=1}^{K(\theta)} R_i(a^*(\theta), \theta) \;\middle|\; \theta_{i,0} = 1 \right] \\
&\quad = \sum_{\ell=1}^{\mathrm{K_{max}}} \sum_{k=1}^{\ell} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ R_i(a^*(\theta), \theta) \mid \theta_{i,0} = 1, K(\theta) = \ell \right].
\end{aligned}
$$

Given the definition of $R_i(a, \theta)$, we have the following for case (I). The derivation of equations is analogous to the above case when $\theta_{i,0} = 0$.

$$
\begin{aligned}
&\mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ R_i(a^*(\theta), \theta) \mid \theta_{i,0} = 1, K(\theta) = \ell \right] \\
&\quad = (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{j,k} \le p_c, a_j^*(\theta) = M \mid \theta_{i,0} = 1, K(\theta) = \ell)
\end{aligned}
$$

$$
\begin{aligned}
&+ R^c(k) \cdot \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{i,k} \leq p_c \mid \theta_{i,0} = 1, K(\theta) = \ell) \\
&+ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{i,k} \leq p_c \mid \theta_{i,0} = 1, K(\theta) = \ell) \\
&= (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{j,k} \leq p_c, \theta_{j,k} = 1 \mid K(\theta) = \ell) \\
&+ R^c(k) \cdot \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{i,k} \leq p_c \mid K(\theta) = \ell) \\
&+ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{P}(\theta_{i,k} \leq p_c \mid K(\theta) = \ell) \\
&= (R^b(k)/n_{rs}) \sum_{j \in N_i} \mathbb{P}(K(\theta_{j,k} \leq p_c, \theta_{j,k} = 1) \cdot \mathbb{P}(K(\theta) = \ell \mid \theta_{j,k} \leq p_c, \theta_{j,k} = 1) \\
&+ R^c(k) \cdot \mathbb{P}(\theta_{i,k} \leq p_c) \cdot \mathbb{P}(K(\theta) = \ell \mid \theta_{i,k} \leq p_c) \\
&+ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} \mathbb{P}(\theta_{i,k} \leq p_c) \cdot \mathbb{P}(K(\theta) = \ell \mid \theta_{i,k} \leq p_c) \\
&= (R^b(k)/n_{rs}) \sum_{j \in N_i} p_c \cdot p_b \cdot \mathbb{P}(K(\theta) = \ell) \\
&+ R^c(k) \cdot p_c \cdot \mathbb{P}(K(\theta) = \ell) \\
&+ (R^r(k)/|\hat{N}_i|) \sum_{j \in \hat{N}_i} p_c \cdot \mathbb{P}(K(\theta) = \ell) \\
&= (R^b(k) \cdot p_c \cdot p_b + R^c(k) \cdot p_c + R^r(k) \cdot p_c) \cdot \mathbb{P}(K(\theta) = \ell).
\end{aligned}
$$

□

Given the expected rewards, we formulate the expected utility of each agent in the following lemma.

**Lemma 9.** *Consider any agent $i$. Suppose that the strategy profile of all agents except agent $i$ is $s^*_{-i}$ where for each agent $j \in N \setminus i$, if $\theta_{j,0} = 0$, then $s^*_j(C \mid \theta_j) = 1$, and $s^*_j(M \mid \theta_j) = 1$ otherwise. Suppose further that agent $i$ adopts strategy $s_i$ which plays $M$ if $\theta_{i,0} = 1$ and plays $C$ with probability $s_c$ and $D$ with probability $1 - s_c$ if $\theta_{i,0} = 0$. Let $a^*(\theta)$ be an action profile where $a^*_i$ is $C$ if $\theta_{i,0} = 0$ and $M$ otherwise. Define $K(\theta) = K(a^*(\theta), \theta)$. Under IRS with referral rewards, the expected utility of agent $i$ is:*

$$
EU_i((s_i, s^*_{-i})) = s_c \cdot (1 - p_b) \cdot \sum_{\ell=1}^{\mathrm{K_{max}}} \mathbb{P}(K(\theta) = \ell) \sum_{k=1}^{\ell} \bar{R}(k) - \bar{C}^b(k) - C^c(k) \cdot p_c,
$$

*where $\bar{R}(k)$ is defined similar to Lemma (8).*

*Proof.* Using the same techniques in the proof of Lemma (6), we can write:

$$EU_i((s_i, s^*_{-i})) = \mathbb{E}_\theta \left[ \sum_{k=1}^{K(\theta)} u_i(a^*(\theta), \theta, k) \;\middle|\; \theta_{i,0} = 0 \right]$$
$$= \sum_{\ell=1}^{\mathrm{K_{max}}} \sum_{k=1}^{\ell} \mathbb{P}(K(\theta) = \ell) \cdot \mathbb{E}_\theta \left[ R_i(a^*(\theta), \theta, k) - C_i(a^*(\theta), \theta, k) \mid \theta_{i,0} = 0, K(\theta) = \ell \right].$$

Substituting conditional expectation of rewards from Lemma (8) and conditional expectation of costs from Equation (5.3) completes the proof. □

Given the two lemmas, we next prove that cooperation is a BNE strategy for non-Byzantine agents under IRS with referral rewards under certain conditions.

**Theorem 10.** *Let $s^*$ be a strategy profile where for each agent $i$, if $\theta_{i,0} = 0$, then $s^*_i(C|\theta_i) = 1$, and otherwise, $s^*_i(M \mid \theta_i) = 1$. Given $p_c$, $p_b$, $\bar{C}^b(k)$, and $C^c(k)$ for $k = 1, \ldots, \mathrm{K_{max}}$, $s^*$ is a BNE under IRS with referral rewards if:*

$$R^b(k) \cdot (1 - p_b) + R^c(k) + R^r(k) \cdot (1 - p_b) \geq \bar{C}^b(k)/p_c + C^c(k), \quad \forall k = 1, \ldots, \mathrm{K_{max}}.$$

*Proof.* For $\bar{R}(k)$ defined in Lemma (8), it is easy to show that for all four cases:

$$\bar{R}(k) \geq R^b(k) \cdot p_c \cdot (1 - p_b) + R^c(k) \cdot p_c + R^r(k) \cdot p_c \cdot (1 - p_b), \quad \forall k = 1, \ldots, \mathrm{K_{max}}.$$

Therefore, the condition of the theorem implies:

$$\bar{R}(k) - \bar{C}^b(k) + C^c(k) \cdot p_c \geq 0.$$

Consequently, $EU_i((s_i, s^*_{-i}))$ is maximized when $s_c = 1$. In this case, we have $s = s^*$ which means $s^*$ is a BNE. □

We conclude this section by showing that Byzantine agents cannot gain more expected rewards under IRS with referral rewards than non-Byzantine agents for all four cases when certain conditions on referral and baseline rewards are met.

**Theorem 11.** *Let $s^*$ be a strategy profile where for each agent $i$, if $\theta_{i,0} = 0$, then $s_i^*(C|\theta_i) = 1$, and otherwise, $s_i^*(M \mid \theta_i) = 1$. Under IRS, with the referral reward mechanism, the expected reward of each Byzantine agent for strategy profile $s^*$ is less than or equal to that of non-Byzantine agents if:*

$$R^b(k) \cdot (1 - p_b) \geq R^r(k) \cdot p_b \text{ and } R^r(k) \cdot (1 - p_b) \geq R^b(k) \cdot p_b, \quad \forall k = 1, \ldots, \mathrm{K}_{\max}\,.$$

*Proof.* Given Lemma (8), to ensure that the expected reward of any Byzantine agent is less than or equal to that of non-Byzantine agents (i.e. $\mathbb{E}_{a,\theta}[R_i(a,\theta) \mid \theta_{i,0} = 0] \geq \mathbb{E}_{a,\theta}[R_i(a,\theta) \mid \theta_{i,0} = 1]$), the following inequalities must hold for all $k = 1, \ldots, \mathrm{K}_{\max}$.

$$\begin{aligned}
&R^b(k) + R^r(k) \cdot (1 - p_b) \geq R^b(k) \cdot p_b + R^r(k),\\
&R^b(k) \cdot (1 - p_b) + R^r(k) \cdot (1 - p_b) \geq R^b(k) \cdot p_b + R^r(k) \cdot p_b,\\
&R^b(k) \cdot (1 - p_b) + R^r(k) \geq R^b(k) + R^r(k) \cdot p_b,\\
&R^b(k) + R^r(k) \geq R^b(k) + R^r(k).
\end{aligned}$$

The first inequality (corresponding to case (I)) can be simplified as $R^b(k)(1-p_b) \geq R^r(k)p_b$. The second inequality (corresponding to case (II)) always holds since $0 \leq p_b < 1/3$. The third inequality (corresponding to case (III)) can be written as $R^r(k)(1 - p_b) \geq R^b(k)p_b$. Finally, the fourth inequality (corresponding to case (IV)) always holds. □

# Chapter 7

# Implementation Details

In this section, we briefly discuss the detailed considerations required to implement IRS.

## 7.1 Gossip Protocol in IRS

To implement IRS, three main modifications to Algorand's default gossip protocol are needed. First, we require the randomness of the peer-selection mechanism to be verifiable (e.g., through verifiable pseudo-random peer selection [53]). This requirement prevents the adversary from gaining unauthorized baseline and referral awards. Byzantine nodes cannot be rewarded for propagating messages to nodes that are not among their randomly selected peers. Byzantine committee members also cannot refer other Byzantine nodes that are not randomly connected to them.

Second, we require nodes to disable selective propagation. Selective propagation is an optimization technique that prevents nodes from propagating low-priority block proposals [40, 24]. Although this technique reduces network congestion, it prevents low-priority block proposals from reaching all nodes in the gossip network. This in turn prevents the Algorand Foundation from tracking cooperation of some committee members at step 1. One optimization that could be implemented to replace selective propagation is to only send the committee member's credential for the low-priority block proposals without sending the entire block proposal.

Third, we require nodes to track the identity of all nodes from which they have received a valid message, even if the message is a duplicate message. For example, suppose that agent $i$ receives message $m$ at step $k$ first from agent $j$ and later from agent $j'$. Agent $i$

propagates message $m$ to its peers only the first time it receives it from agent $j$. However, it saves the identity of both agents $j$ and $j'$ as propagators for $m$ at step $k$. If agent $i$ becomes a committee member at step $k+1$, it includes the identity of both agents $j$ and $j'$ in its generated vote. This allows the Algorand Foundation to not only track the cooperation of the committee members but also the cooperation of their peers. Since committee members are selected randomly, their cooperating peers can be considered as random samples of the non-voting agents that cooperate.

## 7.2 Consideration of Assets in IRS

**Cryptographic sortition.** Extending the sortition algorithm to the case where agents have more than one sub-node is straightforward. Assume agent $i$ has $w_i$ sub-nodes, it can run the sortition algorithm on each of its sub-nodes by adding the index of the sub-node as an input when generating the credential $\sigma$. However, this method is computationally intensive. One optimization is to use the inverse transform sampling method, which is used by Algorand [24]. In this method, the interval of $[0, 1)$ is partitioned into consecutive intervals of the following form.

$$I^M_{w_i,p} = \big[B(M; w_i, p), B(M+1; w_i, p)\big), \qquad \forall M \in \{0, 1, \ldots, w_i - 1\}.$$

If $.x_i$ falls in the $I^M_{w_i,p}$ interval, then node has exactly $M$ selected sub-users.

**Adversary model.** Instead of assuming that the adversary corrupts each node with probability $p_b < 1/3$, we assume each sub-node of an agent is corrupted with probability $p_b$. Again, the results in §5–§6 still hold because non-Byzantine agents constitute more than 2/3 of the committee with high probability.

**Non-Byzantine majority of assets.** Each agent $i$ has $w_i$ sub-nodes. The number of sub-nodes for each agent is known to all agents. The total number of sub-nodes for all agents is $W = \sum_{i \in N} w_i$. W assume that the adversary can corrupt each sub-node with probability $p_b < 1/3$. We further define $W_{-i} = \sum_{j \in N \setminus i} w_j$ for all agents $i$ and require $W_{-i}$ to go to infinity as $n$ goes to infinity. We also require that $w_i/W < (1 - 3 \times p_b)$ for all agents $i$. This requirement ensures that if any single agent is removed from the set of agents, in expectation, the adversary would not control more than one-third of the remaining sub-nodes. For example, if $p_b = 0.3$, then no single agent should have greater than or equal to one-tenth of all sub-nodes.

**Gossip network.** When each agent has more than one sub-node, we assume the gossip network is constructed among the sub-nodes. This means agents need to select a subset

of random peers for each of their sub-nodes. This can be achieved by adding the indexes of the sub-nodes as part of the inputs when conducting the verifiable pseudo-random peer selection [53].

**Gossip protocol.** When a sub-node propagates a message to another sub-node, it is required to include the identities of both the sender and the receiver sub-nodes in the message. When an agent receives a message, it will verify if the sender sub-node and the receiver sub-node are indeed connected.

**IRS.** The simplest method to extend IRS to consider the case where agents have more than one sub-node is to reward each sub-node independently. In this way, the total reward of an agent is the sum of the rewards of its sub-nodes.

# Chapter 8

# Related Works

The incentive compatibility of the reward-distribution scheme is crucial to the security of the blockchain. Nakamoto [57] analyzes the incentive mechanism of Bitcoin informally. Bitcoin provides incentive for the miners by providing the block creator with two kinds of rewards. First, the block creator will get a newly mined coin (called block rewards). Second, when the input value is larger than the output value in a transaction, the difference (called transaction fees) will be awarded to the block creator. These rewards incentivize the miners to put more hashing power to the system and thus maintain the safety of Bitcoin. It is believed that the Bitcoin protocol is incentive compatible if majority (more than 50%) of the hashing power is possessed by the honest miners [57]. However, if an attacker has more than 50% of hashing power in Bitcoin, it can successfully double-spend its bitcoins and violates the safety property of the system (this is known as 51% attack).

## 8.1 Selfish Mining

A celebrated work [34] from Eyal and Sirer proves a negative result against the incentive compatibility of Bitcoin. They model the competitive mining among the miners as a strategic game and propose a novel mining strategy called selfish mining. While the original Bitcoin protocol requires the miners to broadcast the new block immediately, the selfish miner can actually gain more rewards in expectation by withholding the block and release it based on the difference between the public chain and the private chain of the selfish miner. For example, as shown in Fig 8.1, the selfish miner could hide the mined blocks until the honest miners are close to catch up (when the lead is only 1 block). This will invalidate the block mined by the honest players based on the Bitcoin chain selection

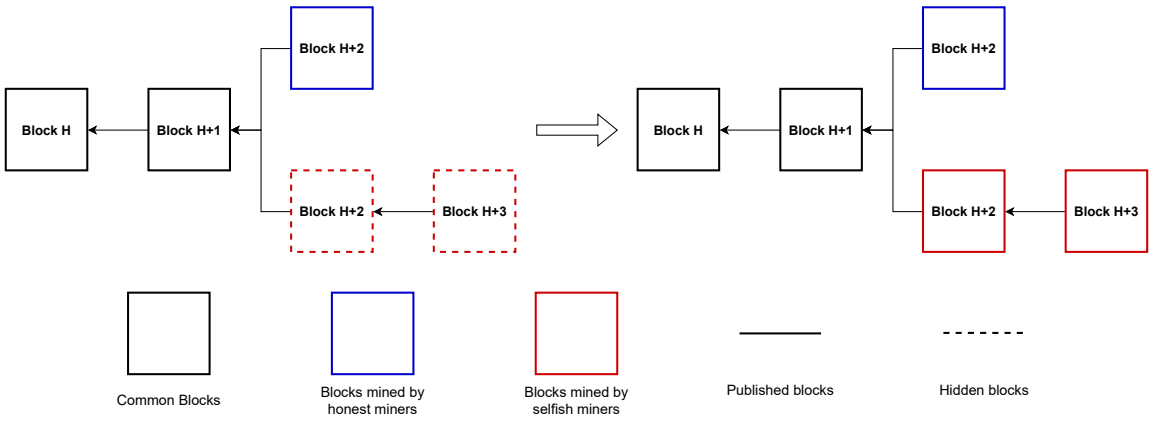


Figure 8.1: Selfish Mining

rule. The authors examine how much computation power the selfish miner needs to gain more revenue using the selfish mining threshold, and they call this amount the profitable threshold. In the extreme case, the profitable threshold of selfish mining is 0, which means a selfish miner with an arbitrarily small amount of the mining power can earn more profit by hiding the blocks instead of publishing them immediately. A modification to the Bitcoin chain selection protocol has been proposed by the authors to fix the profitable threshold of selfish mining strategies at $\frac{1}{4}$.

Since the publication of [34], there are a series of subsequent works on the optimality and the profitable threshold of selfish mining strategies. Sapirshtein, Sompolinsky and Zohar [62] propose an algorithm to find the optimal selfish mining strategies that are more profitable compared to the selfish mining strategy mentioned in [34]. They also show that the modification proposed by [34] cannot in fact guarantee a $\frac{1}{4}$ threshold. Attackers with strictly less than $\frac{1}{4}$ of computation resources can still be profitable from selfish mining under the modified chain selection rule. Nayak et al. [58] generalize the selfish mining strategy to the stubborn mining strategy by the intuition that the selfish miner should not give up easily when the public chain is longer than the private chain of the selfish miner. They demonstrate that stubborn mining earns the miner more revenue compared to selfish mining. Zhang, Zhang and Kemme [69] build a simulator based on the Markov Decision Process to analyze the profitable threshold of selfish mining when there exist two attackers. They claim that the profitable threshold will be dropped to 21% in this case. Recent work from Hou et al. [45] builds a general simulator by combining the Markov Decision Process with Reinforcement Learning. The simulation result suggests that the strategies found in

[62] are not optimal and that no profitable Nash Equilibrium exists when there are more than two competing selfish miners.

Pass and Shi [60] propose a new PoW blockchain protocol called the FruitChain protocol to avoid the potential selfish mining attacks. The FruitChain protocol is based on the original Bitcoin protocol. However, instead of recording transactions in the blocks, the miners put the transactions inside "fruits", which can be considered as another kind of block. The miners of FruitChain need to mine the fruits and the block at the same time. When a miner successfully resolve the puzzle of the fruit, it will add a link in the fruit to an existing recent block and broadcast the fruit. When a miner mined a new block, it will add all known fruits, which have not been added to any blocks yet, inside the block. The intuition behind is that even if a selfish miner fools the network to abandon a block mined by a honest player, a subsequent honest miner will include the fruits in its newly mined block. The block reward is evenly distributed among the miners of a certain number of previous blocks. The authors prove that in FruitChain protocol, the selfish miner can only get a small portion of extra profit than mining honestly thus honest mining is an $\epsilon$-Nash Equilibrium. Whether selfish mining can be completely avoided is still an open question.

## 8.2 Mining Gap

The works have been discussed so far mainly focus on the block rewards. This is reasonable since the block rewards is usually much larger than the transaction fees. However, the block rewards will keep decreasing and eventually becomes 0 [57]. After that, the only incentive will be the transaction fees. Carlsten et al. [22] create a simple game model where there are no block rewards and only the transaction fees matter. They build a simulator to run the game where the miners in the simulator perform no-regret learning to explore the strategy space. The result shows several potential deviations that could happen and harm the security of Bitcoin.

First, they predict that the mining gap, as shown in Fig. 8.2, will happen in this model. Without block rewards, rational miners will not start solving the puzzle until there are enough transaction fees to compensate the mining costs. This will reduce the hashing power of the whole system for a while and increase the proportion of the hashing power of the malicious miner.

Second, rational miners can change the default chain selection rule, which requires the miners to mine on the block it receives first, to mine on the block which left most transaction fees unclaimed. The successful miner can add the unclaimed transactions into

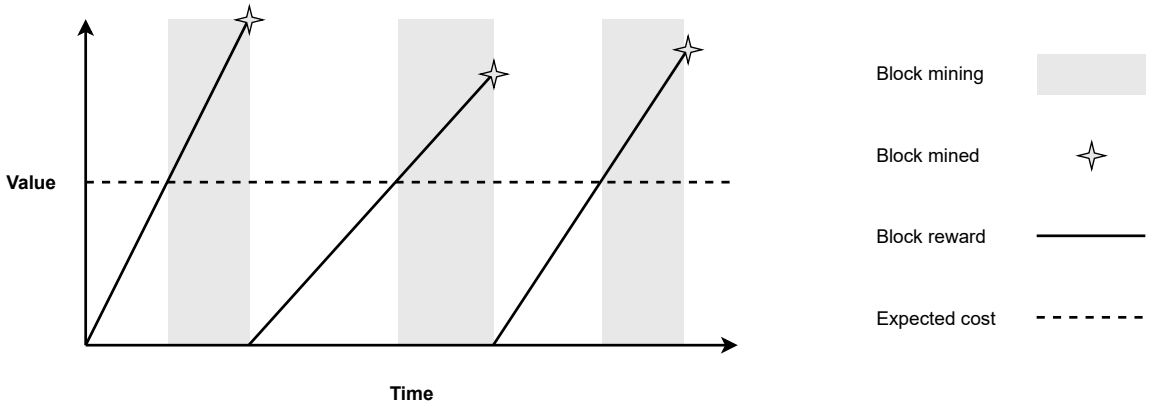


Figure 8.2: Mining Gaps

its own block and earn more rewards. The authors name this strategy PettyCompliant. Once there exists some PettyCompliant miners, a more harmful attack, called Undercutting attack, can be performed by intentionally creating a fork on the head of the chain which leaves a lot of transaction fees unclaimed. They showed that there exists a Nash equilibrium where miners only include a part of unclaimed transactions into the blocks. This will lead to a infinite large amount of unclaimed transactions as time grows to infinite. In addition, both the simulation and the analysis showed that even if 66% of the miners are honest, the undercutting attack is still profitable.

Third, they revisit selfish mining proposed by [34] in transaction-fee-only reward model and show that a modified selfish mining strategy performs better than the original selfish mining strategy. More importantly, this strategy works even when the hashing power of the selfish miner is arbitrarily small and the network connectivity of the selfish miner is poor.

Tsabary and Eyal [66] analyze the mining gap as a Gap Game in a realistic setting. The result shows that the mining gap will arise even before the block reward becomes 0 and the mining utilization will decrease by up to 90% in extreme case. This implies that block rewards are necessary for the blockchain security and they claim that the ratio between the block rewards and the transaction fees should be at least 6 to avoid the mining gaps.

## 8.3 Bribery Attacks

Bonneau [18] outlines three kinds of bribery attacks which are possible on Bitcoin and other cryptocurrencies using longest-chain proof-of-work consensus protocols. The basic ideas is that the attacker rents the hashing power from the miners instead of buying the computation resources. It is assumed that the attacker only cares about the short-term profit rather than the long-term health of Bitcoin. The first one is through out-of-band payment such that the attacker rents the hashing power from the miners by directly paying the miners with bitcoins or any other (crpto)currencies. The second attack is conducted by forming a negative-fee mining pool. Due to the probabilistic nature of Bitcoin, miners with little mining powers tend to collect their hashing power together and form a mining pool to reduce the variance of the expected revenue [25]. The block rewards won by the pool will be distributed among the small miners in the pool based on their hashing power [63]. These kinds of pools often charge some participation fees from the miners in the pools. A negative-fee mining pool can incentivize miners to join the pool and rent the hashing power to the attacker temporarily. The last kind of attack can be conducted through in-band payment on Bitcoin. The idea is that the attacker can create a fork itself and include a script to distribute bribe to miners of subsequent blocks of the fork.

Liao and Katz [54] proposed another kind of bribery attack which also use the in-band payment on Bitcoin. The idea is that the attacker can incentivize the miners to mine on its fork by publishing whale transactions which contain enormous amount of transaction fees.

## 8.4 Information Propagation

Most permissionless blockchains [57, 67, 40] rely on a peer-to-peer gossip protocol to propagate transactions. However, these protocols do not provide an incentive for the nodes in the blockchains to participate in the gossip of transactions. Moreover, Babaioff et al. [16] argue that these protocols provide an incentive for the nodes not to propagate transactions. Taking transaction propagation in Bitcoin as an example, every node who is aware of the transaction will try to include it in their own mined block. If a node that is aware of the transaction broadcasts it to other nodes, it increases the number of nodes who are aware of the transaction and thus decreases the probability of itself claiming the transaction fees. The authors of [16] propose almost-uniform reward schemes to solve this incentive issue. The result is a hybrid almost-uniform reward scheme that combines two almost-uniform reward schemes with different parameters. They prove that under the hybrid almost-uniform

reward scheme, the strategy where all nodes propagate transactions and do not create fake identities (Sybil attacks) is a Nash Equilibrium. In addition, the scheme guarantees that most of the nodes in the network will be aware of the transaction. The additional rewards required to implement the scheme are a constant in expectation and the user only needs to send the transaction to a small number of nodes in the beginning.

However, the model in [16] is highly restricted and not realistic in the real world. The authors assume the network is a forest of $d$-ary directed trees with height $H$ and each node has the same hashing power. Abraham et al. [10] propose a new consensus protocol called Solidus in which they design an incentive mechanism to encourage transaction propagation and puzzle propagation. The incentive mechanism in [10] has fewer restrictions on the network model. Moreover, they also discuss how to enforce the proposed reward scheme, which allows nodes to freely choose how much to pay the propagators of their transactions or puzzles. However, [10] does not consider Sybil attacks and the analysis of the incentive mechanism is limited to the competition among the nodes with same neighbors. Ersoy et al. [33] propose another incentive mechanism for transaction propagation under a network model with minimal restrictions. The resulting reward scheme encourages propagating without creating fake identities.

## 8.5 Algorand

Amoussou-Guenou et al. [12, 13] analyze a simplified committee-based consensus protocol. They propose to make some committee members pivotal such that they have incentives to participate. In this case, a rational node can unilaterally determine the result of the consensus protocol. If a pivotal node does not follow the protocol, consensus will not be reached and a penalty will be applied. As a result, all pivotal nodes participating in the protocol as required is a Nash equilibrium. Nevertheless, the solution is not practical in Algorand because the assumptions in [12, 13] do not hold. For example, the solution proposed by [13] assumes that all nodes are treated equally and have the same voting power, which is not the case in Algorand. Also, the solution assumes that all the committee members are ordered by publicly known indexes, while the membership is private in Algorand until nodes publish it. In addition, each node communicates with all other nodes directly in the simplified protocol while Algorand adopts a gossip protocol to disseminate messages.

Fooladgar et al. [38] analyze the Algorand protocol as a static non-cooperative game. They demonstrate that all nodes participating the protocol is not a Nash equilibrium under Algorand's original reward scheme. If all other nodes are participating the protocol, a node can free-ride to get its reward while not paying the cost of running an Algorand

node. They propose a new reward scheme where only participating nodes are rewarded. However, the analysis is limited to the rational-agent-only case. In addition, their method incentivizes all committee members but fails to incentivize message propagation for all nodes. Under their method, only nodes whose participation is necessary for the network to remain synchronous are incentivized to propagate messages. All other nodes are not incentivized to propagate messages in the network. In real-world deployments, however, no single node's participation should be necessary for keeping the network synchronous. In contrast, in this paper, in addition to incentivizing committee members, we incentivize all nodes to propagate messages. Moreover, our analysis considers Byzantine agents corrupted by an adversary.

In summary, compared to the previous work [38, 12, 13], our model captures some important features of the Algorand protocol, such as randomized membership selection and private membership information. Second, we propose a new reward scheme and prove that all nodes participating faithfully is a Bayesian Nash equilibrium under certain conditions for with Byzantine agents. Finally, our work is the first to consider fairness under a Byzantine setting: we propose a commission mechanism to ensure that malicious nodes cannot gain more than their fair share of reward.

# Chapter 9

# Conclusion & Future Works

In this thesis, we solve the problem of Algorand where nodes do not have enough incentive to participate the protocol. We first model the Algorand protocol as a Bayesian game which captures the facts that the selection is random and the results are private until nodes publishing them. Then, we describe our proposed reward-distribution scheme and derive necessary conditions such that all nodes participating in the protocol is a Bayesian Nash equilibrium. This means agents cannot increase their utility by unilaterally deviating from participating in the protocol. In the end, we propose another commission mechanism to ensure the fairness of the reward scheme such that the malicious nodes cannot earn more than their fair share of reward.

In terms of future work, there remain some open problems in designing incentive-compatible reward schemes for Algorand:

- Like Algorand's original reward scheme, our proposed reward scheme relies on a central authority to collect messages to identify the role of nodes and to verify the participation of nodes. A more decentralized reward scheme is preferred where nodes in the Algorand network determine how the rewards are distributed without a central authority.
- Although our model captures some important features of Algorand, a sequential model, which considers the strategies at the step level, and the corresponding analysis will be a interesting direction for future work.
- Our analysis is based on the assumption that the network is synchronous. Extending the analysis to a setting where the network is not synchronous would be a possible future work.

# References


[1] Algorand developer portal. https://metrics.algorand.org/.

[2] Bitcoin energy consumption index. https://digiconomist.net/bitcoin-energy-consumption/.

[3] BitShares technology – open-source blockchain-based software solutions. https://bitshares.org/.

[4] The Diem Association. https://www.diem.com/en-us/.

[5] EOS blockchain software & services. https://eos.io.

[6] Introducing Governance: Earn rewards for your participation in the decision making. https://algorand.foundation/governance.

[7] Proof of stake instead of proof of work. https://bitcointalk.org/index.php?topic=27787.0.

[8] Script - bitcoin wiki. https://en.bitcoin.it/wiki/Script.

[9] Solidity programming language. https://soliditylang.org/.

[10] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A blockchain protocol based on reconfigurable byzantine consensus. In *Proceedings of the 21st International Conference on Principles of Distributed Systems (OPODIS)*, page 25, 2018.

[11] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 45–58, 2005.

[12] Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Rational behaviors in committee-based blockchains. In *Proceedings of the 24th International Conference on Principles of Distributed Systems (OPODIS)*, pages 12:1–12:16, 2020.

[13] Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Rational vs byzantine players in consensus-based blockchains. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 43–51, 2020.

[14] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the 13th EuroSys Conference*, pages 1–15, 2018.

[15] Nick Arnosti and S Matthew Weinberg. Bitcoin: A natural oligopoly. *Management Science*, 2022.

[16] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On Bitcoin and red balloons. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, page 56, 2012.

[17] Michael Ben-Or. Another advantage of free choice (extended abstract) completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30, 1983.

[18] Joseph Bonneau. Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.

[19] Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocol. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 154–162, 1984.

[20] Gabriel Bracha and Sam Toueg. Resilient consensus protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 12–26, 1983.

[21] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *arXiv:1807.04938 [cs]*, 2019.

[22] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016.

[23] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, 1999.

[24] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

[25] Xi Chen, Christos Papadimitriou, and Tim Roughgarden. An axiomatic approach to block rewards. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 124–131, 2019.

[26] Lin William Cong, Zhiguo He, and Jiasun Li. Decentralized mining in centralized pools. *The Review of Financial Studies*, 34(3):1191–1235, 2021.

[27] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *International Conference on Financial Cryptography and Data Security*, pages 23–41. Springer, 2019.

[28] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[29] Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[30] John R Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-peer Systems (IPTPS)*, pages 251–260, 2002.

[31] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[32] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 139–147, 1993.

[33] Oguzhan Ersoy, Zhijie Ren, Zekeriya Erkin, and Reginald L. Lagendijk. Transaction propagation on permissionless blockchains: Incentive and routing mechanisms. In *Proceedings of the Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 20–30, 2018.

[34] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC)*, pages 436–454, 2014.

[35] Michael J Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Proceedings of the International Conference on Fundamentals of Computation Theory (FCT)*, pages 127–140, 1983.

[36] Michael J Fischer, Nancy A Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

[37] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[38] Mehdi Fooladgar, Mohammad Hossein Manshaei, Murtuza Jadliwala, and Mohammad Ashiqur Rahman. On incentive compatible role-based reward distribution in Algorand. In *Proceedings of the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 452–463, 2020.

[39] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. Decentralization in Bitcoin and Ethereum networks. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC)*, pages 439–457, 2018.

[40] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, pages 51–68, 2017.

[41] Oded Goldreich. *Foundations of cryptography: Volume 1, basic tools*. Cambridge university press, 2007.

[42] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. SBFT: A scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP*

*international conference on dependable systems and networks (DSN)*, pages 568–580, 2019.

[43] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security)*, pages 129–144, 2015.

[44] Christine V Helliar, Louise Crawford, Laura Rocca, Claudio Teodori, and Monica Veneziani. Permissionless and permissioned blockchain diffusion. *International Journal of Information Management*, 54:102–136, 2020.

[45] Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramèr, Giulia Fanti, and Ari Juels. SquirRL: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.

[46] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[47] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012.

[48] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th usenix security symposium (usenix security 16)*, pages 279–296, 2016.

[49] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, pages 133–169, 1998.

[50] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121)*, 2001.

[51] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, pages 382–401, 1982.

[52] Jacob Leshno and Philipp Strack. Bitcoin: An impossibility theorem for proof-of-work based protocols. *SSRN Electronic Journal*, 2019.

[53] Harry C. Li, Allen Clement, Edmund L. Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. BAR Gossip. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, page 191–204, 2006.

[54] Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. In *International conference on financial cryptography and data security*, pages 264–279. Springer, 2017.

[55] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. A survey on blockchain: A game theoretical perspective. *IEEE Access*, 7:47615–47643, 2019.

[56] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 120–130. IEEE, 1999.

[57] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[58] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an Eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320, 2016.

[59] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the USENIX Conference on USENIX Annual Technical Conference (USENIX ATC)*, pages 305–320, 2014.

[60] Rafael Pass and Elaine Shi. FruitChains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 315–324, 2017.

[61] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[62] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in Bitcoin. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC)*, pages 515–532, 2016.

[63] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of Bitcoin mining pool reward functions. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC)*, pages 477–498, 2016.

[64] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[65] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[66] Itay Tsabary and Ittay Eyal. The Gap Game. In *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*, pages 713–728, 2018.

[67] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[68] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.

[69] Shiquan Zhang, Kaiwen Zhang, and Bettina Kemme. A simulation-based analysis of multiplayer selfish mining. In *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5, 2020.

# APPENDICES

# Appendix A

# Notations

Table A.1: Notations

| Parameters | Description |
|---|---|
| $\sigma_k$ | Committee credential of a node |
| $x_k$ | Hash of $\sigma_k$ |
| $.x_k$ | Interpretation of $x_k$ as a number between 0 and 1 |
| $\tau$ | Expected number of sub-nodes selected at each step |
| $T$ | Fraction of $\tau$ that defines Algorand's voting threshold |
| $p_c$ | Probability that an agent is selected as a committee member |
| $p_b$ | Probability that an agent is corrupted by the adversary |
| $K_{\max}$ | Maximum number of steps to add a new block |
| $N_i$ | Set of agent $i$'s randomly selected peers |
| $\hat{N}_i$ | Set of agents of which agent $i$ is a randomly selected peer |
| $n_{rs}$ | Number of each node's randomly selected peers ($n_{rs} = \|N_i$) |
| $N$ | Set of all agents |
| $a_i$ | Action taken by agent $i$ |
| $A_i$ | Set of actions available to agent $i$ |
| $a$ | Joint action profile of all agents ($a = (a_1, \ldots, a_n)$) |
| $A$ | Set of possible joint action profiles |
| $\theta_{i,0}$ | Binary variable indicating whether agent $i$ is Byzantine or not |
| $\theta_{i,k}$ | Number of agent $i$'s sub-nodes selected at step $k > 0$ |
| $\theta_i$ | Type vector of agent $i$ ($\theta_i = (\theta_{i,1}, \ldots, \theta_{i,K_{\max}})$) |
| $\Theta_i$ | Type space of agent $i$ |

Continued on next page

**Table A.1 – continued from previous page**

| Parameters | Description |
|---|---|
| $\theta$ | Joint type vector of all agents ($\theta = (\theta_1, \ldots, \theta_n)$) |
| $\Theta$ | Set of possible joint type vectors |
| $s_i$ | Strategy of agent $i$ |
| $s_i(a_i \mid \theta_i)$ | Probability assigned to action $a_i$ by $s_i$ given $\theta_i$ |
| $S_i$ | Set of all strategies for agent $i$ |
| $s$ | Joint strategy profile for all agents |
| $S$ | Set of possible joint strategy profiles |
| $EU_i(s)$ | Expected utility of agent $i$ given strategy profile $s$ |
| $BR_i(s_{-i})$ | Best response of agent $i$ given $s_{-i}$ |
| $s^*$ | Desired strategy profile where all non-Byzantine agents cooperate |
| $C_i^b(k)$ | Baseline cost incurred by agent $i$ at step $k$ |
| $\bar{C}^b(k)$ | Average baseline cost incurred by any agent at step $k$ |
| $C^c(k)$ | Committee cost incurred by any committee member at step $k$ |
| $C_i(k)$ | Total cost incurred by agent $i$ at step $k$ |
| $R_i^b(a, \theta, k)$ | Baseline reward of cooperating agent $i$ at step $k$ |
| $R_i^c(a, \theta, k)$ | Committee reward of cooperating agent $i$ at step $k$ |
| $R_i^r(a, \theta, k)$ | Referral reward for referring a neighbour at step $k$ |
| $R_i(a, \theta, k)$ | Total reward of cooperating agent $i$ at step $k$ |
| $w_i$ | Number of sub-nodes of agent $i$ |
| $W$ | Total number of sub-nodes for all agents ($W = \sum_{i \in N} w_i$) |

# Appendix B

# Algorand

In this section we present a summary of the Algorand protocol [40, 24][1]. The protocol maintains a public, permissionless blockchain. Adding a new block to the blockchain requires multiple steps. At each step, a committee of randomly selected nodes is formed. Committee members play different roles at different steps. Each committee member proposes a new block at step 1 (see §B.1). At the subsequent steps, committee members vote to reach consensus on the block that should be added to the blockchain (see §B.2–§B.3).

## B.1 Step 1 (Block Proposal)

Algorithm (3) summarizes the procedure executed by all nodes at step one. First, nodes run the sortition algorithm to determine whether they are in the committee at step 1. A node is in the committee if at least one of its sub-nodes is selected by the sortition algorithm. A node that is in the committee generates a candidate block[2] and propagates the block to other nodes using a peer-to-peer gossip protocol[3]).

Nodes also propagate the output hash and the proof. This allows other nodes to verify the sortition of committee members. The hash is also used to prioritize committee

[1] We only present the details that are needed for our game-theoretic model. Interested readers could see [40, 24] for a detailed description and analysis of the protocol

[2] A candidate block contains a set of pending transactions that a node has heard about.

[3] A gossip protocol is a communication process that disseminates data to all nodes in the system. In the most common implementation, upon receiving a message, a node selects a small random set of peers to gossip the message to. To prevent loops, nodes do not relay the same message twice.

**Algorithm 3:** Step-1 procedure

```
(M, hash, proof) ← Sortition(1);
if M ≥ 1 then
    B ← Generate block proposal;
    Propagate(B, hash, proof);
end
```

members. The priority of each node in the committee is the highest priority of the node's selected sub-nodes. The priority of each selected sub-node is calculated by hashing the output hash concatenated with the sub-node's index. The smaller this hash is, the higher the priority of the sub-node will be. Prioritizing committee members enables nodes to reach consensus on a single block proposed by the highest-priority committee member.

## B.2 Step 2 & 3 (Graded Consensus)

At step 2, nodes run the first step of the Graded Consensus (GC) protocol. The main goal of the graded consensus (GC) protocol is to reduce the number of candidate blocks to one and to convert the problem of reaching consensus on an arbitrary value (the hash of a block) to reaching consensus on a single bit. Algorithm (4) shows the pseudo-code for step 2. Nodes wait a fixed time period to receive block proposals. While waiting, nodes validate any received blocks and gossip valid ones to their neighbors according to the gossip protocol. Once the time period expires, nodes invoke the sortition algorithm to check whether they are in the committee for this step. If at least one sub-node of a node is selected, then the node generates its vote according to the first step of the GC protocol. The node then propagates its vote. The node also propagates the hash and the proof to allow other nodes to verify its sortition. Note that if $M$ sub-nodes of a node are selected, then the node's vote is counted as $M$ *sub-votes*.

Algorithm (5) summarizes the procedure for step 3. In this step, nodes run the second step of the GC protocol. First, nodes validate and gossip any received votes from step 2 while waiting a fixed period of time. Nodes then run the sortition algorithm. If a node is in the committee for this step, then it generates its vote according to the second step of the GC protocol. The node then propagates its vote, hash, and proof.

**Algorithm 4:** Step-2 procedure

```
Validate and gossip received block proposals for a fixed time period;
(M, hash, proof) ← Sortition(2);
if M ≥ 1 then
    v ← Generate vote according to GC's 1st step;
    Propagate(v, hash, proof);
end
```

**Algorithm 5:** Step-3 procedure

```
Validate and gossip received step-2 votes for a fixed time period;
(M, hash, proof) ← Sortition(3);
if M ≥ 1 then
    v ← Generate vote according to GC's 2nd step;
    Propagate(v, hash, proof);
end
```

**Algorithm 6:** Step-4 procedure

```
Validate and gossip received step-3 votes for a fixed time period;
(v, b) ← Output of GC protocol;
(M, hash, proof) ← Sortition(4);
if M ≥ 1 then
    Propagate((v, b), hash, proof);
end
```

## B.3 Step 4 to $K_{max}$ (Binary Byzantine Agreement)

At step 4, nodes compute the output of the GC protocol and start the first step of the Binary Byzantine Agreement (BBA*) protocol. The main goal of the Binary Byzantine Agreement BBA* protocol is to reach consensus among all nodes on a binary value. The pseudo-code for step 4 is shown in Algorithm (6). Nodes first validate and gossip the received votes from step 3 for a fixed time period. After this period expires, each node computes the output of the GC protocol, $(v, b)$, where $v$ is a string and $b$ is a single bit. Nodes then start the first step of the BBA* protocol by invoking the sortition algorithm. If a node is selected as a committee member, it propagates its GC output.

**Algorithm 7:** Step-$k$ procedure for $5 \leq k \leq K_{\max}$ and $k \equiv 2 \pmod 3$

```
Validate and gossip received step-(k - 1) votes for a fixed time period;
if more than T · τ sub-votes are received for b = 0 then
    Terminate;
end
(M, hash, proof) ← Sortition(k);
if M ≥ 1 then
    v ← Get v from step 4;
    b ← Generate binary according to BBA*'s coin-fixed-to-0 step;
    Propagate((v, b), hash, proof);
end
```

**Algorithm 8:** Step-$k$ procedure for $6 \leq k \leq K_{\max}$ and $k \equiv 0 \pmod 3$

```
Validate and gossip received step-(k - 1) votes for a fixed time period;
if more than T · τ sub-votes are received for b = 1 then
    Terminate;
end
(M, hash, proof) ← Sortition(k);
if M ≥ 1 then
    v ← Get v from step 4;
    b ← Generate binary according to BBA*'s coin-fixed-to-1 step;
    Propagate((v, b), hash, proof);
end
```

Starting from step 5, until a termination condition is met, nodes iterate three steps of the BBA* protocol: (i) coin-fixed-to-0, (ii) coin-fixed-to-1, and (iii) coin-fixed-to-flip. In all three steps, nodes validate and gossip votes from the previous step for a fixed time period. In coin-fixed-to-0 steps, nodes terminate if they receive more than $T \cdot \tau$ sub-votes for $b = 0$, where $\frac{2}{3} < T < 1$ is a fraction of $\tau$ that defines Algorand's voting threshold. Otherwise, they run the sortition algorithm to check whether they are in the committee. Nodes in the committee then calculate their new $b$ and propagate it. The procedure is similar in coin-fixed-to-1 steps except that the termination condition is on $b = 1$. Algorithm (7) and Algorithm (8) show the procedure for these steps.

If nodes do not terminate in coin-fixed-to-0 and coin-fixed-to-1 steps, they run the coin-fixed-to-flip step. In this step, nodes that are selected in the committee calculate a new

**Algorithm 9:** Step-$k$ procedure for $7 \leq k \leq K_{\max}$ and $k \equiv 1 \pmod 3$

```
Validate and gossip received step-(k - 1) votes for a fixed time period;
(M, hash, proof) ← Sortition(k);
if M ≥ 1 then
    v ← Get v from step 4;
    b ← Generate binary according to BBA*'s coin-fixed-to-flip step;
    Propagate((v, b), hash, proof);
end
```

$b$ and propagate it. Algorithm (9) summarizes the procedure for coin-fixed-to-flip steps. These steps do not have a termination condition. Therefore, after a coin-fixed-to-flip step, nodes start running the next coin-fixed-to-0 step.