Blockchains Cannot Rely on Honesty

Jakub Sliwinski & Roger Wattenhofer

ABSTRACT

This work proposes a novel blockchain with an incentive scheme such that all players following the protocol constitute a strict, strong Nash equilibrium. In other words, following the protocol is guaranteed to be the optimal strategy. Our blockchain takes the form of a directed acyclic graph, resulting in improvements with respect to throughput and speed.

More importantly, for our blockchain to function, it is not expected that the miners conform to some presupposed protocol in the interest of the system's operability. Instead, our system works if miners act selfishly, trying to get the maximum possible rewards, with no consideration for the overall health of the blockchain.

1 INTRODUCTION

A decade ago, Satoshi Nakamoto presented his now famous Bitcoin protocol [10]. Nakamoto assembled some stimulating techniques in an attractive package, such that the result was more than just the sum of its parts.

The Bitcoin blockchain promises to order and store transactions meticulously, despite being anarchistic, “without a trusted party". Literally anybody can participate, as long as “honest nodes collectively control more CPU power than any cooperating group of attacker nodes." [10]

In Section 6 of his seminal paper, Nakamoto argues that it is rational to be honest thanks to block rewards and fees. However, it turns out that Nakamoto was overly optimistic, and rational does not imply honest. If a miner has a fast network *and/or* a significant fraction of the hashing power, the miner may be better off by not being honest, holding blocks back instead of immediately broadcasting them to the network [2].

If the material costs and payoffs of mining are low, one can argue that the majority of miners will want to remain honest. After all, if too many miners stop conforming to the protocol, the system will break down. However, the costs and payoffs of participation vary over time, and majority of miners remaining altruistic is never guaranteed. Strategies outperforming the protocol may or may not be discovered for different blockchain incentive designs. However, as long as it is not proven that no such sophisticated strategy exists, the system remains in jeopardy.

1.1 Blockchain Game

Typical blockchains, such as Bitcoin's, take the form of a rooted tree of blocks. During the execution of the protocol, players continually create new blocks that are appended to the tree as new leaves. Creating blocks is computationally intensive, so that the network creates a specific number of blocks in a given time period, such as one block every ten minutes on average in Bitcoin. One path of blocks, such as the longest path, is distinguished as the main chain and keeps being extended by addition of new leaves. The network's participants want to create blocks that remain incorporated into the main chain, as these blocks are rewarded. Ideally, the leaves would be added in sequence, each appended to the previous one. However, by chance or malice, it is inevitable that some leaves are appended to the same block and create a "fork". Then, it is uncertain which one will end up extending the main chain. According to typical solutions, one of the leaves is eventually chosen as the one extending the chain, and the creator of the other one misses out on block rewards. This approach introduces some unwanted incentives in the potential to punish other players. Even worse, some factors such as network connectivity start to play a role and might influence the behaviour of players.

1.2 Our Contribution

We propose a blockchain design with an incentive scheme guaranteeing that deviating from the protocol strictly reduces the overall share and amount of rewards. All players following the protocol constitute a strict, strong Nash equilibrium. Our approach is to ensure that creating a fork will always be detrimental to all parties involved. Our design allows blocks to reference more than one previous block; in other words, the blocks form a directed acyclic graph (DAG). We prove that miners creating a new block have an incentive to always reference all previously unreferenced blocks. Hence, all blocks are recorded in the blockchain and no blocks are discarded.

2 MODEL AND PRELIMINARIES

2.1 Rounds

Communication between players (miners) is divided into rounds. Each round consists of each player: 1) computing (mining) new blocks, 2) sending newly found blocks to all other players, 3) receiving all messages before the next round commences. The length of one round can be thought of as the network delay.

2.2 Players

To avoid confusion in how we build on previous work, we stick to the usual terminology of *honest* players and an *adversary*. The players that conform to the protocol are called honest. A coalition of all parties that considers deviating from the protocol is controlled by an adversary. We gradually introduce new elements, and eventually show that by deviating from the protocol, the adversary reduces its share and amount of rewards. Hence, rational becomes synonymous with honest.

The adversary constitutes a minority as described in Section 2.5, otherwise the adversary can take over the blockchain by simply ignoring all actions by honest players.

The adversary is also more powerful than honest players. First of all, we consider the adversary as a single entity. The adversary

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand

https://doi.org/doi

does not have to send messages to itself, so the mine/send/receive order within a round does not apply to the adversary. Moreover, the adversary gets to see all messages sent by honest players in round r before deciding its strategy of round r. After seeing the honest messages, the adversary is not allowed to create new blocks again in this round. Moreover, the adversary controls the order that messages arrive to each player.

2.3 Blocks

Blocks are the messages that the players exchange, and a basic unit of the blockchain. Formally, a block B is a tuple $B = \langle \mathcal{T}_B, \mathcal{R}_B, c, \eta \rangle$, where:

- $\mathcal{T}_B$ is the content of the block
- $\mathcal{R}_B$ is a set of references (hashes) to previously existing blocks, i.e. $\mathcal{R}_B = \{h(B_1), \dots, h(B_m)\}$
- c is a public key of the player that created the block
- η is the proof-of-work nonce, i.e., a number such that for a hash function h and difficulty parameter D, $h(B) < D$ holds.

The content of the block $\mathcal{T}_B$ depends on the application. In general, $\mathcal{T}_B$ contains some information that the block creator wishes to record in the blockchain for all participants to see. We consider blockchain properties independently of the content $\mathcal{T}_B$. The content $\mathcal{T}_B$ is briefly discussed in Section 5.4.

The creator of B holds the private key corresponding to c. The creator can later use the key to withdraw the reward for creating B. The amount of reward is automatically determined by the protocol, and at the core of our contribution in Section 5.

2.4 DAG

$\mathcal{R}_B$ includes at least one hash of a previous block, which might be the hash of a special *genesis* block $\langle \emptyset, \emptyset, \perp, 0 \rangle$. The hash function is pre-image resistant, i.e. it is infeasible to find a message given its hash. If a block B' includes a reference to another block B, B' must include $h(B)$, and hence has to be created after B.

A directed cycle of blocks is impossible, as the block which was created earliest in such a cycle cannot include a hash to the other blocks that were created later. Consequently, the blocks always form a directed acyclic graph (*DAG*) with the genesis block as the only root (block without any parent) of this DAG.

2.5 Mining

Creating a new block is achieved by varying η to find a hash value that is smaller than the difficulty parameter $\mathcal{D}$, i.e., $h(\langle \mathcal{T}_B, \mathcal{R}_B, c, \eta \rangle) < D$. Creating blocks in this way is called *mining*. Blocks are called honest if mined by an honest player, or adversarial if mined by the adversary.

By varying D, the protocol designer can set the probability of mining a block with a single hashing query arbitrarily. The difficulty D could also change during the execution of the protocol to adjust the rate at which blocks are created. For simplicity and clarity we leave the details of changing D to future work, and assume D to be constant.

The honest players control the computational power to mine α blocks in expectation in one round. The computational power of the adversary is such that the expected number of blocks the adversary can mine in one round is equal to β. The adversary does not experience a delay in communication with itself, so the adversary might mine multiple blocks forming a chain in one round.

Assumptions. The following assumptions are made in order to satisfy the prerequisites of Lemma 4.2 from [5], which links our work to traditional blockchains. Intuitively, Lemma 4.2 states that a traditional blockchain works with respect to the most basic requirement. If one believes a blockchain to function in this basic way under some other assumptions, those assumptions can be used instead, and our results would apply in the same way.

Because of the delay in communication, the effective computational power of the honest players corresponds to the probability $\alpha' \approx \alpha e^{-\alpha}$ [5] that in a given round exactly one honest player mines a block.

(1) The honest players have more mining power: $\alpha' \geq \beta(1 + \epsilon)$ for a constant $\epsilon > 0$.
(2) The difficulty D is set such that the expected number of blocks mined within one round is less than one: $\alpha + \beta < 1$.

3 THE PROTOCOL

The protocol by which the honest players construct the block DAG is quite natural:

- Every round, attempt to mine new blocks.
- Reference in $\mathcal{R}_B$ all unreferenced blocks observed.
- Broadcast newly mined blocks to all other players immediately.[1]

4 THE BLOCK DAG

Each player stores the DAG formed by all blocks known to the player. For each block B, one of the referenced blocks B_i is the parent $B_i = P(B)$, and B is the child of $P(B)$. The parent is automatically determined based on the DAG structure. The parent-child edges induce the *parent tree* from the DAG.

The players use Algorithm 1 by [15] to select a chain of blocks going from the genesis block to a leaf in the parent tree. The selected chain represents the current state of the blockchain; it is called the *main* chain. The main chain of a player changes from round to round. Players adopt main chains that may be different from each other, depending on the blocks observed.

Algorithm 1: Main chain selection algorithm.

```
Input: a block tree $T$
Output: block $B$ - the end of the selected chain
$B \leftarrow$ genesis        // start at the genesis block.
while $B$ has a child in $T$ do
    $B \leftarrow$ heaviest child of $B$
                    // continue with the child of B
                 // with most nodes in its subtree.
return $B$
```

Let *past*(B) denote the set of blocks reachable by references from B and the DAG formed by those blocks. The protocol dictates referencing all blocks that otherwise would not be included in *past*(B).

[1] Similarly to other works in the area we assume the network supports a message diffusion mechanism that delivers messages in each round, similarly to the Bitcoin's network.

Then, by creating a new block B, the creator communicates only being aware of blocks in $past(B)$. Based on $past(B)$, we determine $P(B)$ as the end of the main chain (Algorithm 1) of the DAG of the player when creating a new block B [6].

Definition 4.1. A block B is the child of the block returned by Algorithm 1 in the parent tree of $past(B)$.

Lemma 4.2 by [5], encapsulates the notion that a blockchain (represented by the parent tree in our description) functions properly with respect to a basic requirement. Intuitively, it states that from any point in time, the longer one waits, the more probable it becomes that some honest block mined after that point in time is contained in a main chain of each honest player. The probability of the contrary decreases exponentially with time.

Lemma 4.2 (Fresh Block Lemma). *For all $r, \Delta \in \mathbb{N}$, with probability $1 - e^{-\Omega(\Delta)}$, there exists a block mined by an honest player on or after round r that is contained in the main chain of each honest player on and after round $r + \Delta$.*

Lemma 4.2 can be proved with respect to other chain selection rules, for instance picking the child with the longest chain instead of the heaviest child as in Algorithm 1. Our work can be applied equally well using such chain selection rules.

If the protocol designer has control over some factor x, probability of the form $e^{-\Omega(x)}$ can be set arbitrarily low with relatively small variation of x. Probability of the form $e^{-\Omega(x)}$ is called negligible.[2]

4.1 Block Order

We will now explain, how all blocks reachable by references will be ordered, following the algorithm of [6]. According to the resulting order, the contents of blocks that fall outside of the main chain can be processed, as if all blocks formed one chain.

Definition 4.3. Each player processes blocks in the order $Order(B)$, where B is the last block of the main chain.

```
Algorithm 2: $Order(B)$: a total order of blocks in $past(B)$.
  Input: a block $B$
  Output: a total order of all blocks in $past(B)$
1 On the first invocation, $visited(\cdot)$ is initialized to $false$ for
    each block.
2 if $visited(B)$ then return $\emptyset$
3 $visited(B) \leftarrow true$            // Blocks are visited
    depth-first.
4 if $B = genesis$ then return $(B)$
5 $O \leftarrow Order(P(B))$
                // Get the order of $P(B)$ recursively.
6 for $i = 1, \ldots, m$ do
7     $O \leftarrow O.append(Order(B_i))$
                  // Append newly included blocks.
8 $O \leftarrow O.append(B)$              // Append $B$ at the end.
9 return $O$
```

Note the order of executing the FOR loop in line 6 of the Algorithm 2 has to be the same for each player for them to receive consistent orders of blocks. Algorithm 2 processes B_i's in the order of inclusion in $\mathcal{R}_B$, but the order could be alphabetical or induced by the chain selection rule.

Based on lines numbered 5-8 we can state Corollary 4.4.

Corollary 4.4. *$Order(B)$ extends $Order(P(B))$ by appending all newly reachable blocks not included yet in $Order(P(B))$.*

Lemma 4.5. *Any announced block becomes referenced by a block contained in the main chain of any honest player after Δ rounds with probability $1 - e^{-\Omega(\Delta)}$.*

Proof. Suppose a block B is announced at round r. By Lemma 4.2, some honest block A mined in the following Δ rounds is contained in the main chains adopted by honest players after round $r + \Delta$. Since A is honest, $B \in past(A)$. □

Corollary 4.6. *All announced blocks are eventually referenced in the main chains of honest players.*

4.2 Stale Blocks

We now introduce a mechanism to distinguish blocks that were announced within a reasonable number of rounds from blocks that where withheld by the miner for an extended period of time. Such withheld blocks are called *stale*. Honest miners broadcast their blocks immediately, so stale blocks can be attributed to the adversary. In our incentive scheme, stale blocks will not receive any rewards and will also be ignored for the purpose of determining other block rewards. Thus we ensure that it is pointless for the adversary to wait too long before broadcasting its blocks.

The basic definition of whether a block A is stale is termed with respect to some other block B. We are only interested in blocks B that form the main chain. When the main chain is extended, the sets of stale and non-stale blocks are preserved (and extended). Hence, stale-ness is determined by the eventual main chain.

Definition 4.7. Given a block B, the set sets of blocks S_B is computed by Algorithm 3. Then, $\bar{S}_B = past(B) \setminus S_B$. If $A \in S_B$ we call A *stale*.

The constant p of Algorithm 3 is chosen by the protocol designer. Intuitively, given a main chain ending with block B that references another block A, we judge A by the distance one needs to backtrack along the main chain to find an ancestor of A. If the distance exceeds p, A is stale.

We call $P^i(B)$ the i^{th} *ancestor of* B and B is a *descendant* of $P^i(B)$.[3] By $LCA(B_1, B_2)$ we denote the block that is an ancestor of B_1 and an ancestor of B_2, such that none of its children are simultaneously an ancestor of B_1 and an ancestor of B_2.

For blocks A and B, $D(A, B)$ is the distance between A and B in the parent tree, i.e. $D(A, P(A)) = 1$, $D(A, P(P(A))) = 2$, etc.

Corollary 4.8 shows that when the main chain is extended, the stale-ness of previously seen blocks is preserved.

Corollary 4.8. *If $A \in past(P(B))$ then $A \in S_B \iff A \in S_{P(B)}$.*

[2] Probabilities of this form are often disregarded completely in proofs [13].

[3] Note that ancestors and descendants are defined based on the parent tree and not based on other non-parent references building up the DAG.

Algorithm 3: Compute S_B.

```
  Input: a block $B$
  Output: a set $S_B$
1 if $B = genesis$ then return $\emptyset$
2 $S \leftarrow S_{P(B)}$   // Copy $S_{P(B)}$ for blocks in past($P(B)$).
3 for $A \in past(B) \setminus past(P(B))$ do
4     $X = LCA(A, B)$
5     $Age = D(X, B)$  // age = distance from $B$ to LCA.
6     if $Age > p$ then
7         $S = S \cup \{A\}$
              // $A$ is stale iff age is bigger than $p$
8 return $S$
```

Proof. Line 5 in Algorithm 3 sets S_B as the same as $S_{P(B)}$, while the following FOR loop adds only blocks $A \notin past(P(B))$. □

Theorem 4.9 establishes the most important property of staleness. The probability that the adversary can successfully make an honest block stale decreases exponentially with p, and is negligible.

Theorem 4.9 (Honest Blocks are Not Stale). *Let B be an honest block mined on round r. With probability $1 - e^{-\Omega(p)}$, after round $r + O(p)$ each honest player H adopts a main chain ending with a block B_H such that $B \in \bar{S}_{B_H}$.*

Proof. Let $\Delta = \lfloor \frac{p}{2(\alpha+\beta)(1+\epsilon)} - \frac{1}{2} \rfloor = O(p)$. By Lemma 4.2, with probability $1 - e^{-\Omega(\Delta)}$, on and after round r, honest players have adopted main chains containing a block C mined between rounds $r - \Delta$ and r (or the genesis block if $r - \Delta < 1$). Hence C is an ancestor of B. By Lemma 4.2, let D be the honest block mined between rounds $r + 1$ and $r + \Delta + 1$ that honest players adopted in the main chain on and after round $r + \Delta + 1$, again with probability $1 - e^{-\Omega(\Delta)}$. D is honest and mined after round r, so $B \in past(D)$.

Since C was mined on or after round $r - \Delta$, and D was mined on or before round $r + \Delta + 1$, $D(C, D)$ is at most the number Y of blocks mined between rounds $r - \Delta$ and $r + \Delta + 1$. By the Chernoff bound:

$$e^{-\frac{\epsilon^2(2(\alpha+\beta)\Delta)}{3}} \geq \Pr[Y \geq (1+\epsilon)(\alpha+\beta)(2\Delta+1)] \geq \Pr[Y \geq p]$$

Since C is an ancestor of D, C is an ancestor of $LCA(B, D)$, and $D(C, D) \geq D(LCA(B, D), D)$. By Algorithm 3:

$$D(C, D) < p \implies B \in \bar{S}_D.$$

By union bound, the probability that such C and D exist and that $B \in \bar{S}_D$ is at least equal

$$1 - 2e^{-\Omega(\Delta)} - e^{-\frac{\epsilon^2(2(\alpha+\beta)\Delta)}{3}} = 1 - e^{-\Omega(p)}.$$

By Corollary 4.8 and induction, with probability $1 - e^{-\Omega(p)}$, after round $r + \Delta$ all honest players adopt only chains ending with blocks X such that $B \in \bar{S}_X$. □

5 THE REWARD SCHEME

Consider coupling the presented protocol with a reward mechanism $\mathcal{R}^0$ that, intuitively speaking, grants some flat amount b of reward to all non-stale blocks, and 0 reward to stale blocks. $\mathcal{R}^0$ is a special case of the reward scheme properly defined in Definition 5.3.

Corollary 5.1. *Under the reward scheme $\mathcal{R}^0$, honest players are rewarded proportionally to the number of blocks they mine, except with negligible probability.*

Proof. By Theorem 4.9 honest blocks are not stale, so honest miners receive rewards linear in the number of blocks they mined. The adversary might only decrease its rewards by producing stale blocks, otherwise the adversary is rewarded in the same way. □

Note that $\mathcal{R}^0$ achieves the same fairness guarantee as the Fruitchains protocol to be discussed in Section 6.3 — honest blocks are incorporated into the blockchain as non-stale, while withholding a block for too long makes it lose its reward potential. Both protocols rely on the honest majority of participants to guarantee this fairness.

The Fruitchains protocol relies critically on merged-mining [11] (also called 2-for-1 POW [3]) fruits and blocks. While fruits are mined for the rewards, blocks are supposed to be mined entirely voluntarily with negligible extra cost. The reward scheme $\mathcal{R}^0$ avoids this complication.

Granting flat amount of reward for each non-stale block leaves a lot of room for deviation that goes unpunished. In the case of the Fruitchains protocol, mining blocks does not contribute rewards in any way. Hence, any deviation with respect to mining blocks (which decide the order of contents) is free of any cost for the adversary. In the context of cryptocurrency transactions, a rational adversary should always attempt to double-spend.

In the case of $\mathcal{R}^0$, the adversary can refrain from referencing some recent blocks, and suffer no penalty. However, attempting to manipulate the order of older blocks would render the adversary's new block stale, and hence penalize. Thus, we view even the base case $\mathcal{R}^0$ of the presented reward scheme as a strict improvement over the Fruitchains protocol.

5.1 Penalizing Deviations

Central to our design is the approach to treating forks i.e. blocks that "compete" by referencing the same parent block and not each other. Typically, blockchain schemes specify that one of the blocks eventually 'loses' and the creator misses out on some rewards, hence discouraging the competition. However, there are ways of manipulating this process to one's advantage, and the uncertainty of which block will win the competition introduces unneeded incentives. We penalize all parties involved in creating a fork.

The *conflict set* introduced in Definition 5.2 contains the blocks that "compete" with a given block. Stale blocks are excluded, as we ignore them for the purpose of computing rewards. Like stale-ness, the conflict set is defined with respect to some other block A. Again, we are only interested in blocks A that form the main chain, and the conflict set indicated by the eventual main chain.

The conflict set of a non-stale block B contains all non-stale blocks X that are not reachable by references from B, and B is not reachable by references from X.

Definition 5.2 (Conflict Set). For blocks A and B where $B \in \bar{S}_A$,

$$X_A(B) = \{X : X \in past(A) \land X \in \bar{S}_A \land X \notin past(B) \land B \notin past(X)\}.$$

Intuitively, the scheme we propose awards every block some amount of reward b decreased by a penalty c multiplied by the size of the conflict set. The ultimate purpose of the properties we

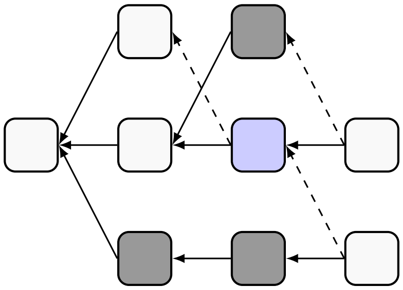

Figure 1: An example of a conflict set. The gray blocks constitute the conflict set of the blue block. The dashed arrows are references and the solid arrows are parent references.

establish is to make sure that rational miners want to minimize the conflict set of the blocks they create, following the protocol as a consequence.

Definition 5.3 (Rewards). A reward scheme $\mathcal{R}^{c,b}$ is such that given the main chain ending with a block A, each block $B \in past(A)$ is granted $\mathcal{R}^{c,b}_A(B)$ amount of reward:

$$\mathcal{R}^{c,b}_A(B) = \begin{cases} 0, & \text{if } B \in S_A \text{ or } D(A, LCA(A,B)) \le 2p. \\ b - c|X_A(B)|, & \text{otherwise.} \end{cases}$$

We write $\mathcal{R}^c$ for $\mathcal{R}^{c,b}$ if b is clear from context, or just $\mathcal{R}$ if c is clear from context.

In our reward scheme, the reward associated with a given block are decreased linearly with the size of the block's conflict set. We need to ensure that no block reward is negative, otherwise the reward scheme would break down. Lemma 5.4 shows that it is only possible for the conflict set to reach certain size; the probability that the conflict set of a block is bigger than linear in p is negligible. Intuitively, it is due to the fact that stale blocks cannot be part of a conflict set, and after enough time has passed from broadcasting some block B, new blocks either have to reference B or are stale.

As a consequence, we establish in Corollary 5.5 that the rewards are non-negative.

Lemma 5.4. *Let $x \ge p$ and B be a block. The probability that any honest player adopts a main chain ending with a block A such that $|X_A(B)| > xp$ is $e^{-\Omega(x)}$.*

Proof. Let r be the round B was announced. Let P_i, $i \in \{1, \dots, 2p\}$ (respectively F_i, $i \in \{1, \dots, p\}$), be an honest block mined between rounds $r - \frac{xi}{4} - 1$ and $r - \frac{x(i-1)}{4} - 1$ (resp. $r + \frac{x(i-1)}{4} + 1$ and $r + \frac{xi}{4} + 1$) contained in the main chain of every honest player on and after round $r + \frac{xp}{4} + 1$; by Lemma 4.2 and union bound such blocks exist with probability $1 - e^{-\Omega(x)}$.

Since F_1 is honest, $B \in past(F_1)$. By Algorithm 3, if $P_p \notin past(B)$, then $B \in S_{F_1}$ and $X_A(B)$ remains undefined for honest players. Otherwise, assume $P_p \in past(B)$.

Let Z be a block such that $Z \notin past(B) \wedge B \notin past(Z)$. Since $Z \notin past(B)$, $Z \notin past(P_p)$. By Algorithm 3, either $P_{2p} \in past(Z)$, or Z becomes stale in the main chains of honest players from round $r - \frac{x(p-1)}{4} - 1$ on. Assume $P_{2p} \in past(Z)$, and hence Z is mined on or after round $r - \frac{2xp}{4} - 1$.

Since $B \notin past(Z)$, $F_1 \notin past(Z)$. Then, either Z is announced before round $r + \frac{xp}{4} + 1$, or by Algorithm 3, Z becomes stale in the main chains of honest players afterwards. Assume Z is announced before round $r + \frac{xp}{4} + 1$.

Therefore, $Z \in X_A(B)$ implies that Z is mined between rounds $r - \frac{2xp}{4} - 1$ and $r + \frac{xp}{4} + 1$. Let Y be the number of blocks mined between these rounds. By Chernoff bound:

$$\Pr[Y \ge xp] \le \Pr[Y \ge \frac{4}{3}(\alpha + \beta)(\frac{3xp}{4} + 2)] = e^{-\Omega(x)}.$$

Note the bound is appliable to any main chain of an honest player before round $r + \frac{xp}{4} + 1$ as well. The claim follows from the union bound. □

Corollary 5.5 (Rewards Are Non-Negative). *Let B be a block. The probability that any honest player adopts a main chain ending with a block A such that $\mathcal{R}^{c,b}_A(B) < 0$ is $e^{-\Omega(\frac{b}{cp})}$.*

Proof. Follows directly from Lemma 5.4. □

The conflict set of a block is determined based on the main chain. At some point, the reward needs to be determined and stay fixed. Lemma 5.6 shows that if the main chain has grown far enough from the block B, the new blocks A appended to the chain will not modify the conflict set of B.

Lemma 5.6. *If $D(P(A), LCA(P(A), B)) > 2p$ then $X_A(B) = X_{P(A)}(B)$*

Proof. From Definition 5.2, $X_{P(A)}(B) \subseteq X_A(B)$. Suppose for contradiction $\exists Y : Y \in X_A(B) \setminus X_{P(A)}(B)$. From Definition 5.2, $B \in \bar{S}_A$, therefore $B \in past(P^i(A))$. Hence, $P^i(A) \notin past(Y)$. Since $Y \notin past(P(A))$, $D(A, LCA(A, Y)) > p$ and $Y \in S_A$, a contradiction. □

The rewards in Definition 5.3 are only assigned as non-zero to blocks B such that $D(A, LCA(A, B)) > 2p$, where A is the block at the end of the main chain. By Corollary 5.7, these non-zero rewards are not modified by the blocks extending the main chain and remain fixed.

Corollary 5.7 (Rewards Are Final).

$$\forall B \in past(A) : \mathcal{R}_{P(A)}(B) \neq 0 \implies \mathcal{R}_A(B) = \mathcal{R}_{P(A)}(B).$$

Proof. $\mathcal{R}^{c,b}_A(B)$ is non-zero only if $D(A, LCA(A, B)) > 2p$. The corollary follows from Lemmas 4.8 and 5.6 and induction. □

The properties we have established so far culminate in Theorem 5.8.

Theorem 5.8. *Deviating from the protocol reduces the adversary's rewards and its proportion of rewards $\mathcal{R}^{c,b}$, except with negligible probability.*

Proof. Honest blocks are not-stale, except with negligible probability (Theorem 4.9). Block rewards are final and non-negative, except with negligible probability (Corollaries 5.7 and 5.5). Hence, eventual value of $\mathcal{R}^{c,b}(B)$ for honest blocks B depends only on $|X_A(B)|$. Since $Y \in X_A(Z) \iff Z \in X_A(Y)$, by increasing $|X_A(B)|$ of an honest block the adversary can only reduce the rewards of honest players (by $c|X_A(B)|$) if the adversary forfeits the same amount. Since the adversary constitutes a minority, its proportion of rewards decreases as well.

The adversary can also produce stale blocks, forfeiting the otherwise non-negative reward, while not changing the rewards of honest players.

Not referencing some known honest block directly increases $|X_A(B)|$. Withholding a block might only prevent some honest player from referencing it, thus increasing $|X_A(B)|$. Hence, any strategy effectively different from the protocol increases $|X_A(B)|$ of produced blocks, thus decreasing the amount and share of rewards of the adversary. □

5.2 Nash Equilibria

Discussing Nash equilibria with respect to the received rewards is problematic, since the game only continues as long as the blockchain functions. Hence, a strategy profile wherein a majority of miners do not follow the protocol might be meaningless, as it would often imply the lack of any main chain consistent among players, and lack of any globally defined rewards. Since we are only assured the blockchain operation continues if a majority of miners follows the protocol, we restrict our attention to strategy profiles where that is the case.

Theorem 5.8 shows that minimizing the conflict set of mined blocks is in the interest of the miner. Following the protocol i.e. not withholding blocks and referencing all other blocks is the unique strategy minimizing the conflict set of created blocks. There are negligibly improbable scenarios in which a player can increase its share of rewards by deviating, for example rendering some blocks of other players stale. However, committing to a strategy different from the protocol is associated with concrete punishments. Hence, by Theorem 5.8 the protocol designer can set the constants p, c, b, so that all players following the protocol constitute a strict, strong Nash equilibrium. In other words, all agents and all (minority) coalitions of agents strictly prefer to follow the protocol to any alternative strategy.

Corollary 5.9. *All players following the protocol constitute a strict, strong Nash equilibrium.*

However, there exist other Nash equilibria. Consider the scenario described in Example 5.10.

Example 5.10. Four blockchain players with equal hashing power each adopt the following strategies:

- Player 1 and 2: Follow the protocol.
- Player 3: Do not broadcast new blocks in the first round, otherwise follow the protocol.
- Player 4: If you receive three blocks of other players (child blocks of the genesis block) at the beginning of the second round, then induce penalties for yourself and other players as much as possible forever. Otherwise follow the protocol.

Since Player 3 refrains from broadcasting blocks in the first round, Player 4 can never receive three blocks of other players in the second round, and thus the strategy of Player 4 is identical to following the protocol.

However, Player 3 deviates from the protocol. With some constant probability Players 1 and 2 broadcast a block each in the first round, so if Player 3 broadcast a block in the first round, Player 4 could receive three blocks. Then, this action *would* change the behaviour of Player 4 to cause penalties to herself and Player 3. Hence, the strategy profile is a Nash equilibrium.

The Nash equilibrium presented in Example 5.10 is based on a player threatening to induce penalties for other players by suffering penalties herself. Intuitively speaking, we suggest all Nash equilibria where some player does not follow the protocol are of this nature, but we do not formalize this concept. However, if the adversary wishes to spend resources solely to influence the behaviour of rational miners, there are always ways to achieve this outside the scope of any reward scheme, such as bribery (see Section 6.4).

5.3 Hurting Other Players

When designing a reward scheme, it might be seen as fair if each honest player is rewarded irrespectively of the strategies of other players. Such fairness principle is enjoyed by the Fruitchains protocol and our reward scheme $\mathcal{R}^0$. However, those schemes inevitably trivialize some aspect of the game and leave potential for deviation that goes unpunished. A relaxation of this principle is stated in Corollary 5.11 based on Theorem 5.8 and its proof.

Corollary 5.11. *Under the reward scheme $\mathcal{R}^{c,b}$, by deviating from the protocol the adversary can only reduce the rewards of other players by forfeiting at least the same amount.*

We observe that the property stated in Corollary 5.11 prevents the existence of selfish mining strategies such as those concerning Bitcoin and other traditional blockchains (see Section 6.1). Such strategies pose a threat since they enable forfeiting some rewards to penalize other players to an even bigger extent.

5.4 Block Content and Transaction Fees

Depending on the use of the blockchain, miners can be rewarded for including contents in their blocks in various ways. Typically, a transaction fee is awarded to only one miner that first includes the transaction in a block. As a result, the order of processing blocks is important for determining who collects the fees, as it indicates which block is the first. Problematic incentives are introduced with respect to manipulating the order.

Any particular fee-sharing scheme cannot be enforced, because the fee might be disguised as a regular transaction output paid to the miner directly. This can benefit both the transaction issuer and the miner, incentivizing the behavior.[4]

To be incentive compatible, it is not necessary that the fees are spread proportionally. What we want is that the miners never have an incentive to omit a reference to another block. As all blocks are assumed to eventually be included in the blockchain, it is enough to ensure that sufficiently small changes of the linearized order of the blocks have no effect on the miner rewards. This can be achieved by allowing multiple blocks to claim the same inclusion of contents, and having the fee be shared among the including blocks equally.

In other words, any player who wishes to include a transaction can do so within a certain window, without an effect on their incentives to reference other blocks. Crucially, sending the fee directly to

[4] If we disregard this vulnerability, the same fee-sharing approach as employed by the Fruitchains protocol can be applied to our work.

a miner as a transaction output removes the incentive for other miners to include the transaction, as well as the incentive to manipulate the place of the including block in the order.

The point of such a change would be to separate transaction inclusion from referencing blocks. Transaction inclusion is a complex game in itself, similar to the game studied by [6].

6 RELATED WORK

The model of round-based communication in the setting of blockchain was introduced in [3]. This paper formalizes and studies the security of Bitcoin.

6.1 Selfish Mining

Selfish mining is a branch of research studying a type of strategies increasing the proportion of rewards obtained by players in a Bitcoin-like system. Selfish mining exemplifies concerns stemming from the lack of proven incentive compatibility. Selfish mining was first described formally in [2], although the idea had been discussed earlier [9]. Selfish mining strategies have been improved [14] and generalized [12].

6.2 DAG

The way we order all blocks for the purpose of processing them was introduced in [6]. The authors consider an incentive scheme to accompany this modification. Their design relies on altruism, as referring extra blocks has no benefit, other than to creators of referred blocks. Hence, rational miners would never refer them, possibly degenerating the DAG to a blockchain similar to Bitcoin's. Some other shortcomings are discussed by the authors.

The authors of [7] contribute an experimental implementation of the directed acyclic graph structure and ordering of [6], in particular its advantages with respect to the throughput.

6.3 Fruitchains

Fruitchains [13] is probably the closest work to ours. Fruitchains is a protocol that gives a guarantee that miners are rewarded somewhat proportionally to their mining power. The objective might seem similar to ours, but there are fundamental differences. To achieve fairness, similarly to existing solutions, the Fruitchains protocol requires the majority of miners to cooperate without an incentive. In other words, in order to contribute to the common good of the system, players must put in altruistic work.

In contrast, we strive for a protocol such that any miner simply trying to maximize their share or amount of rewards will inadvertently conform to the protocol.

The Fruitchains protocol rewards mining of "fruits", which are a kind of blocks that do not contribute to the security of the system. The Fruitchains protocol relies on merged-mining [5] also called 2-for-1 PoW in [3]. In addition to fruits, the miners can mine "normal" blocks (containing the fruits) with minimal extra effort and for no reward. The functioning and security of the system depends only on mining normal blocks according to the protocol.

Miners are asked to reference the fruits of other miners, benefiting others but not themselves, similarly to [inclusive]. The probability of not doing so having any effect is negligible, since majority of the miners are still assumed to reference said fruits.

The resulting system-wide cooperation guarantees fairness, inevitably removing many game-theoretic aspects from the resulting game. In particular, misbehaviour does not result in any punishment. It is common to analyze blockchain designs with respect to the expected cost of a double-spend attempt. In the case of Fruitchains, while the probability of double-spends being successful is similar to previous designs, the *cost* of attempting to double-spend is nullified. As a result, any miner might attempt to double-spend constantly at no cost, which we view as a serious jeopardy to the system.

In the absence of punishments, we also argue that not conforming to the protocol is often simpler. Since transaction fees are shared between miners, including transactions might be seen as pointless altogether. Mining only fruits with dummy, zero-fee transactions, while not including the fruits of others (or not mining for blocks altogether), would relieve the miner of a vast majority of the network communication.

Another game-theoretic issue of the Fruitchains protocol is that while it prescribes sharing of the transaction fees, miners might ask transaction issuers to disguise the fee as an additional transaction output, locking it to a specific miner, potentially benefiting both parties and disrupting the protocol.

As argued in Section 5, the reward scheme $\mathcal{R}^0$ is an improvement over Fruitchains in the same vein, achieving the same result while avoiding some of the complications.

In contrast to Fruitchains protocol, the approach of reward schemes $\mathcal{R}^{c,b}$ is to employ purely economic forces, clearly incentivizing desired behaviour while making sure that deviations are punished.

6.4 Bribery

Recently, there have been works highlighting the problems of bribery, e.g. [1, 8]. A bribing attacker might temporarily convince some otherwise honest players (either using threats or incentives) to join the adversary. Consequently, the adversary might gain more than half of the computational power, taking over the system temporarily.

Such bribery might be completely external to the reward scheme itself, for example the adversary might program a smart contract (perhaps in another blockchain) that provably offers rewards to miners that show they deviate from the protocol. Hence, no permissionless blockchain can be safe against this type of attack.

7 CONCLUSIONS

Mining is a risky business, as block rewards must pay for hardware investments, energy and other operation costs. At the time of this writing, the Bitcoin mining turnover alone is worth over $5 billion per year, which is without a doubt a serious market. Miners in this market are professionals, who will make sure that their investments pay off. Yet, many believe that a majority of miners will follow the protocol altruistically, in the best interests of everybody, the "greater good".

[5] One of the first mentions of merged-mining as used today is [11], although the general idea was mentioned as early as [4].

We argue that assuming altruistic miners is not strong enough to be a foundation for a reliable protocol. In this work, we introduced a blockchain incentive scheme such that following the protocol is guaranteed to be the optimal strategy.

We showed that our design is tolerant to miners acting rationally, trying to get the maximum possible rewards, with no consideration for the overall health of the blockchain.

To the best of our knowledge, our design is the first to provably allow for rational mining. Nakamoto [10] needed "honest nodes collectively control more CPU power than any cooperating group of attacker nodes". With our design it is possible to turn the word honest into the word rational.

REFERENCES

[1] Joseph Bonneau. 2016. Why Buy When You Can Rent? - Bribery Attacks on Bitcoin-Style Consensus. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*. 19–26.

[2] Ittay Eyal and Emin Gün Sirer. 2014. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *18th International Conference on Financial Cryptography and Data Security*. 436–454.

[3] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 281–310.

[4] Markus Jakobsson and Ari Juels. 1999. Proofs of work and bread pudding protocols. In *Secure Information Networks*. 258–272.

[5] Aggelos Kiayias and Georgios Panagiotakos. 2017. On Trees, Chains and Fast Transactions in the Blockchain. In *5th International Conference on Cryptology and Information Security in Latin America.*

[6] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. 2015. Inclusive Block Chain Protocols. In *19th International Conference on Financial Cryptography and Data Security*. 528–547.

[7] Chenxing Li, Peilun Li, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. 2018. Scaling Nakamoto Consensus to Thousands of Transactions per Second. *arXiv preprint arXiv:1805.03870* (2018).

[8] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. 2018. Smart Contracts for Bribing Miners. In *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*. 3–18.

[9] mtgox. 2010. https://bitcointalk.org/index.php?topic=2227.msg29606#msg29606. (2010).

[10] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf. (2008).

[11] Satoshi Nakamoto. 2010. https://bitcointalk.org/index.php?topic=1790.msg28696#msg28696. (2010).

[12] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2016. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *1st IEEE European Symposium on Security and Privacy*.

[13] Rafael Pass and Elaine Shi. 2017. Fruitchains: A Fair Blockchain. In *Symposium on Principles of Distributed Computing*. 315–324.

[14] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal selfish mining strategies in Bitcoin. In *20th International Conference on Financial Cryptography and Data Security*. 515–532.

[15] Yonatan Sompolinsky and Aviv Zohar. 2015. Secure high-rate transaction processing in Bitcoin. In *19th International Conference on Financial Cryptography and Data Security*. 507–527.