Cryptocurrency Mining Games with Economic Discount and Decreasing Rewards

Marcelo Arenas
PUC & IMFD Chile
marenas@ing.puc.cl

Juan Reutter
PUC & IMFD Chile
jreutter@ing.puc.cl

Etienne Toussaint
University of Edinburgh
etienne.toussaint@ed.ac.uk

Martín Ugarte
PUC & IMFD Chile
martin@martinugarte.com

Francisco Vial
ProtonMail & IMFD Chile
fvial@pm.me

Domagoj Vrgoč
PUC & IMFD Chile
dvrgoc@ing.puc.cl

Abstract

In the consensus protocols used in most cryptocurrencies, participants called *miners* must find *valid blocks* of transactions, appending them to a shared tree-like data structure. Ideally, the rules of the protocol should ensure that miners maximize their gains if they follow a default strategy, which consists on appending blocks only to the longest branch of the tree, called the *blockchain*. Our goal is to understand under which circumstances are miners encouraged to follow the default strategy. However, most of the existing models work with simplified payoff functions, without considering the possibility that rewards decrease over time because of the game rules (like in Bitcoin), nor integrating the fact that a miner naturally prefers to be paid earlier than later (the economic concept of discount). In order to integrate these factors, we consider a more general model where issues such as economic discount and decreasing rewards can be set as parameters of an infinite stochastic game in which players always try to produce valid blocks. In this model, we study the limit situation in which a miner does not receive a full reward for a block if it stops being in the blockchain. We show that if rewards are not decreasing, then miners do not have incentives to create new branches, no matter how high their hash power is. On the other hand, when working with decreasing rewards similar to those in Bitcoin, we show that miners have an incentive to create such branches; however, the minimal proportion of hash power for which it happens is close to half.

2012 ACM Subject Classification Algorithms and data structures

Keywords and phrases cryptocurrency, game theory, cryptomining, economic discount, decreasing rewards

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.50

© Marcelo Arenas, Juan Reutter, Etienne Toussaint, Martín Ugarte, Francisco Vial, Domagoj Vrgoč; licensed under Creative Commons License CC-BY

1 Introduction

The Bitcoin Protocol [14, 15, 16], or Nakamoto Protocol, introduces a novel decentralized network-consensus mechanism that is trustless and open for anyone connected to the Internet. This open and dynamic topology is supported by means of an underlying currency (a so-called *cryptocurrency* [16]), to encourage/discourage participants to/from taking certain actions. The largest network running this protocol at the time of writing is the Bitcoin network, and its underlying cryptocurrency is Bitcoin (BTC). The success of Bitcoin lead the way for several other cryptocurrencies; some of them are replicas of Bitcoin with slight modifications (e.g. Litecoin [27] or Bitcoin Cash [25]), while others introduce more involved modifications (e.g. Ethereum [26, 22] or Monero [28]).

The data structure used in these protocols is an append-only record of transactions, which are assembled into *blocks*, and appended to the record once they are marked as valid. The incentive to generate valid new blocks is an amount of currency, which is known as the *block reward*. In order to give value to the currencies, the proof-of-work framework mandates that participants generating new blocks are required to solve some computationally hard problem per each new block. This is known as *mining*, and the number of problems per second that a miner can solve is referred to as her *hash power*. Agents who participate in the generation of blocks are called *miners*. In Bitcoin, for example, the hard problem corresponds to finding blocks with a hash value that, when interpreted as a number, is less than a certain threshold. Since hash functions are pseudo-random, the only way to generate a valid block is to try with several different blocks, until one of them has a hash value below the established threshold.

Miners are not told where to append the new blocks they produce. The only requirement is that new blocks must include a pointer to a previous block in the data structure, which then naturally forms a tree of blocks. The consensus data structure is generally defined as the longest branch of such a tree, also known as the *blockchain*. In terms of cryptocurrencies, this means that the only valid currency should be the one that originates from a block in the blockchain. Miners looking to maximise their rewards may then attempt to create new branches out of the blockchain, to produce a longer branch that contains more of their blocks (and earn more block rewards) or to produce a branch that contains less blocks of a user they are trying to harm. This opens up several interesting questions: under what circumstances are miners encouraged to produce a new branch in the blockchain? What is the optimal strategy of miners assuming they have a rational behaviour? Finally, how can we design new protocols where miners do not have incentives to deviate from the main branch?

Our goal is to provide a model of mining that can incorporate different types of block rewards (including the decreasing rewards used in e.g. Bitcoin, where rewards for block decrease after a certain amount of time), as well as the economic concept of *discount*, i.e. the fact that miners prefer to be rewarded sooner than later, and that can help in answering the previous questions. Since mining protocols vary with each cryptocurrency, distilling a clean model that can answer these questions while simultaneously covering all practical nuances of currencies is far from trivial [10]. Instead, we abstract from these rules and focus on the limit situation in which a miner does not receive the full reward for a block if it stops being in the blockchain. More precisely, the reward for a block b is divided into an infinite number of payments, and the miner loses some of them whenever b does not belong to the blockchain. This limit situation represents miners with a strong incentive to put–and maintain–their blocks in the blockchain, and is relevant when studying cryptocurrencies as a closed system, where miners do not wish to spend money right away but rather be able to cash-out their wealth at any point in time. In terms of how mining is performed, we consider these two simple rules: each player i is associated a fixed value h_i specifying her proportion of the hash power against the total hash power, and she tries in each step to append a new block somewhere in the tree of blocks, being h_i her probability of succeeding.

The last two rules mentioned above are the standard way of formalizing mining in a cryptocurrency On the other hand, the way a miner is rewarded for a block in our model takes us on a different path from most of current literature, wherein agents typically mine with the objective of cashing-out as soon as possible or after an amount of time chosen a priori [10, 2]. Far from being orthogonal, our framework is complementary with these studies, as it allows to validate some of the assumptions and results obtained in these articles with miners who have stronger motives to mine and keep their blocks in the blockchain.

Contributions. Our first contribution is a model for blockchain mining, given as an infinite stochastic game in which maximising the utility corresponds to both putting blocks in the blockchain and maintaining them there for as long as possible. A benefit of our model is that using few basic design parameters we can accommodate different cryptocurrencies, and not focus solely on Bitcoin, while also allowing us to account for fundamental factors such as deflation, or discount in the block reward. The second contribution of our work is a set of results about strategies in two different scenarios. First, we study mining under the assumption that block rewards are constant (as it will eventually be in cryptocurrencies with tail-emission such as Monero or Ethereum), and secondly, assuming that per-block reward decreases over time (a continuous approximation to Bitcoin rewards).

In the first scenario of constant rewards, we show that the default strategy of always mining on the latest block of the blockchain is indeed a Nash equilibrium and, in fact, provides the highest possible utility for all players. Therefore with constant reward, we prove that long forks should not happen, as it is not an optimal strategy. On the other hand, if block reward decreases over time, we prove that strategies that involve forking the blockchain can be a better option than the default strategy, and thus we study what is the best strategy for miners when assuming everyone else is playing the default strategy. We provide different strategies that involve branching at certain points of the blockchain, and show how to compute their utility. When we analyse which one of these strategies is the best, we see that the choice depends on the hash power, the rate at which block rewards decrease over time, and the usual financial discount rate. We confirm the commonly held belief that players should start deviating from the default strategy when they approach 50% of the network's hash power (also known as 51% attack), but we go further: there are more complex strategies that prove better than default even with less than 50% of the hash power. Further investigation is needed but these results complement and improve our current understanding of mining strategies and tend to show that even with decreasing reward long forks should not happen if no miner is holding close to 50% of the hash power, therefore validate the assumption used in most previous works (see e.g. [10, 2]).

Related work. Our framework takes us on a different path that most of current literature offering a game-theoretic characterisation for blockchain mining [10, 2, 11], which typically model the reward of players as the proportion of their blocks with respect to the total number of blocks (we pay for each block). Each choice has its own benefits; our choice allows us to analyse different forms of rewards and also introduce a discount factor on the utility, which we view as one of the main advantages of our model. It is also common to introduce assumptions that limit the set of strategies. For instance, Kiayias et.al. [10] assume that only one block per depth generates reward, which is natural in their framework but limits the set of valid strategies they consider. Moreover, Biais et.al.[2] assume that the reward of a block depends on the proportion of hash-power dedicated to blockchains containing it at a time chosen a priori. These assumptions do not take into account every potential forking strategies, or the fact that a miner may want to adapt his cash-out strategy based on the situation. Lastly, our framework cannot deal with strategies that feature a tactical release of blocks often referred as selfish mining, in which miners opt not to release new blocks in hope that these will give them a future advantage [19, 6, 8, 18, 17]. Our model can be extended to account for most of those strategies, for example by defining states as a tuple of trees, one for each player. However work studying precise problems and taking into account the intrinsic cost of mining like electricity [23, 2] cannot easily be added to our framework, because it requires a continuous time-based model for mining.

Among other works that approach cryptocurrency mining from a game-theoretical point of view, we mention [12, 3], noting that these differ from our work either in the choice of a reward function, the space of mining strategies considered, or both. As far as we are aware, our work is the first to provide a model that can account for multiple choices in the reward function (say, constant reward or decreasing reward), and without any assumption on the set of strategies. Recently, the perks of adding new functionalities to bitcoin's mining protocol have been studied: In [11], it is shown that a pay-forward option would ensure optimality of the default behaviour, even when miner rewards are mainly given as transaction fees. There is also interesting work regarding mining strategies in multi-cryptocurrency markets [5, 20], and a number of articles on network properties of the Bitcoin protocol, as well as technical considerations regarding its security and privacy (see e.g. the survey by Conti et al. [4]). Interestingly, some network settings can inflict undesired mining behaviour [1, 9, 24].

Proviso. Due to the lack of space, some proofs are deferred to the full version.

2 A Game-theoretic Formalisation of Crypto-Mining

The mining game is played by a set $\mathbf{P} = \{0, 1, , \ldots, m-1\}$ of players, with $m \geq 2$. In this game, each player gains some reward depending on the number of blocks she owns. Every block must point to a previous block, except for the first block which is called the *genesis block*. Thus, the game defines a tree of blocks. Each block is put by one player, called the *owner* of this block. Each such tree is called a *state of the game*, or just *state*, and represents the knowledge that each player has about the blocks that have been mined thus far.

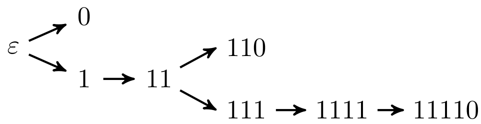
The key question for each player is, then, where do I put my next block? The general rule in cryptocurrencies is that miners are only allowed to spend their reward if their blocks belongs to the *blockchain*, which in this paper is simply the longest chain of blocks in the current state (the model is general enough to consider other forms of blockchain such as Ethereum's notion, but some of the results may change with this other definition). Thus, players face essentially two possibilities: put their blocks right after the end of the blockchain, or try to *fork*, betting that a smaller chain will eventually become the blockchain. As the likelihood of mining the next block is directly related to the comparative hash power of a player, we model mining as an infinite stochastic game, in which the probability of executing the action of a player p is given by her comparative hash power.

In what follows we define the components of the game considered in this paper. Our formalisation is similar to others in the literature [10, 11], except for the way in which miners are rewarded and the way in which these rewards are accumulated in the utility function. As these elements are fundamental for our model, we analyse them in detail in Section 2.1.

Blocks, states and the notion of blockchain. In a game played by m players, a block is defined as a string b over the alphabet $\{0, 1, \ldots, m-1\}$. We denote by $\mathbf{B}$ the set of all blocks, that is, $\mathbf{B} = \{0, 1, \ldots, m-1\}^*$. Each block apart from ε has a unique owner, defined by the function owner : $(\mathbf{B} \smallsetminus \{\varepsilon\}) \to \{0, 1, \ldots, m-1\}$ such that owner(b) is equal to the last symbol of b. As in [10], a state of the game is defined as a tree of blocks. More precisely, a state of the game, or just state, is a finite and nonempty set of blocks $q \subseteq \mathbf{B}$ that is prefix closed. That is, q is a set of strings over the alphabet $\{0, 1, \ldots, m-1\}$ such that if $b \in q$, then every prefix of b (including the empty word ε) also belongs to q. Note that a prefix closed subset of $\mathbf{B}$ uniquely defines a tree with ε as the root. The intuition here is that each element of q corresponds to a block that was put into the state q by some player. The genesis block corresponds to ε. When a player p decides to mine on top of a block b, she puts another block into the state defined by the string $b \cdot p$, where we use notation $b_1 \cdot b_2$ for the concatenation of two strings b_1 and b_2. Notice that with this terminology, given $b_1, b_2 \in q$, we have that b_2 is a descendant of b_1 in q if b_1 is a prefix of b_2, which is denoted by $b_1 \preceq b_2$. Moreover, a path in q is a nonempty set π of blocks from q for which there exist blocks b_1, b_2 such that $\pi = \{b \mid b_1 \preceq b \text{ and } b \preceq b_2\}$; in particular, b_2 is a descendant of b_1 and π is said to be a path from b_1 to b_2. Finally, let $\mathbf{Q}$ be the set of all possible states in a game played by m players, and for a state $q \in \mathbf{Q}$, let $|q|$ be its size, measured as the cardinality of the set q of strings (or blocks).

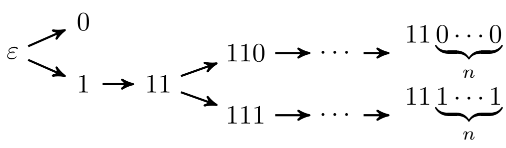
The *blockchain* of a state q, denoted by bc(q), is the path π in q of largest length, in the case this path is unique. If two or more paths in q are tied for the longest, then we say that the blockchain in q does not exist, and we assume that bc(q) is not defined (so that bc$(\cdot)$ is a partial function).

▶ **Example 2.1.** Consider the following state q of the game with players $\mathbf{P} = \{0, 1\}$:



In this case, we have that $q = \{\varepsilon, 0, 1, 11, 110, 111, 1111, 11110\}$, so q is a finite and prefix-closed subset of $\mathbf{B} = \{0, 1\}^*$. The owner of each block $b \in q \smallsetminus \{\varepsilon\}$ is given by the the last symbol of b; for instance, we have that owner$(11) = 1$ and owner$(11110) = 0$. Moreover, the longest path in q is $\pi = \{\varepsilon, 1, 11, 111, 1111, 11110\}$, so that the blockchain of q is π (in symbols, bc$(q) = \pi$). Finally, $|q| = 8$, as q is a set consisting of eight blocks (including the genesis block ε).

Assume now that q' is the following state of the game:



We have that bc(q') is not defined since the paths $\pi_1 = \{\varepsilon, 1, 11, 110, \cdots, 110^n\}$ and $\pi_2 = \{\varepsilon, 1, 11, 111, \cdots, 111^n\}$ are tied for the longest path in q'. ◀

Actions of a miner. On each step, each miner chooses a block in the current state, and attempts to mine on top of this block. Thus, in each turn, each of the players race to place the next block in the state, and only one of them succeeds. The probability of succeeding is directly related to the comparative amount of hash power available to this player, the more hash power the more likely it is that she will mine the next block. Once a player places a block, this block is added to the current state, obtaining a different state from which the game continues.

Let $p \in \mathbf{P}$. Given a block $b \in \mathbf{B}$ and a state $q \in \mathbf{Q}$, we denote by $\text{mine}(p, b, q)$ an action in the mining game in which player p mines on top of block b. Such an action $\text{mine}(p, b, q)$ is considered to be valid if $b \in q$ and $b \cdot p \notin q$. The set of valid actions for player p is collected in the set:

$$\mathbf{A}_p = \{\text{mine}(p, b, q) \mid b \in \mathbf{B}, q \in \mathbf{Q} \text{ and } \text{mine}(p, b, q) \text{ is a valid action}\}.$$

Moreover, given $a \in \mathbf{A}_p$ with $a = \text{mine}(p, b, q)$, the result of applying a to q, denoted by $a(q)$, is defined as the state $q \cup \{b \cdot p\}$. Finally, we denote by $\mathbf{A}$ the set of combined actions for the m players, that is, $\mathbf{A} = \mathbf{A}_0 \times \mathbf{A}_1 \times \cdots \times \mathbf{A}_{m-1}$.

Miner's Pay-off. Most cryptocurrencies follow these rules for miner's payment: (1) Miners receive a possibly delayed one-time reward per each block they mine. (2) The only blocks that are valid are those in the blockchain; if a block is not in the blockchain then the reward given for mining this block cannot be spent.

The second rule enforces that we cannot just give miners the full block reward when they put a block at the top of the current blockchain or after a delay, because blocks out of the blockchain may eventually give the same reward as valid ones. To illustrate this, consider the state q' in Example 2.1, where we have two paths (π_1 and π_2) competing to be the blockchain, and consider π_2 to be the latest path to be mined upon to reach q'. If player 0 had already been fully paid for any blocks 110^i, where $i \leq n$, then if π_2 win the race and becomes the blockchain, such block 110^i would not be part of the blockchain anymore but still has given full reward to player 0. To the best of our knowledge other attempts to formalize mining, especially bitcoin's mining partially emancipate from this rule, as only the first block to be confirmed will be paid, and artificially nullify the incentive to engage in long races.

In the following sections we will show how different reward functions can be used to understand different mining scenarios that arise in different cryptocurrencies. For now we assume, for each player $p \in \mathbf{P}$, the existence of a reward function $r_p : \mathbf{Q} \to \mathbb{R}$ such that the reward of p in a state q is given by $r_p(q)$. Moreover, the combined reward function of the game is $\mathbf{R} = (r_0, r_1, \ldots, r_{m-1})$. In Section 2.1 we provide a detailed explanation of how our pay-off model can be used to pay for blocks and at the same time to ensure that players try to maintain their blocks in the blockchain.

Transition probability function. As a last component of the game, we assume that $\mathbf{Pr}$: $\mathbf{Q} \times \mathbf{A} \times \mathbf{Q} \to [0, 1]$ is a transition probability function satisfying that for every state $q \in \mathbf{Q}$ and combined action $\mathbf{a} = (a_0, a_1, \ldots, a_{m-1})$ in $\mathbf{A}$, we have that $\sum_{p=0}^{m-1} \mathbf{Pr}(q, \mathbf{a}, a_p(q)) = 1$.

Notice that if p_1 and p_2 are two different players, then for every action $a_1 \in \mathbf{A}_{p_1}$, every action $a_2 \in \mathbf{A}_{p_2}$ and every state $q \in \mathbf{Q}$, it holds that $a_1(q) \neq a_2(q)$. Thus, we can think of $\mathbf{Pr}(q, \mathbf{a}, a_p(q))$ as the probability that player p places the next block, which will generate the state $a_p(q)$. As we have mentioned, such a probability is directly related with the hash power of player p, the more hash power the likely it is that action a_p is executed and p mines the next block before the rest of the players. In what follows, we assume that the hash power of each player does not change during the mining game, which is captured by the following condition: for each player $p \in \mathbf{P}$, we have that $\mathbf{Pr}(q, \mathbf{a}, a_p(q)) = h_p$ for every $q \in \mathbf{Q}$ and $\mathbf{a} \in \mathbf{A}$ with $\mathbf{a} = (a_0, a_1, \ldots, a_{m-1})$. We refer to such a fixed value h_p as the hash power of player p. Moreover, we assume that $h_p > 0$ for every player $p \in \mathbf{P}$, as if this is not the case then p can be removed from the game.

The mining game: definition, strategy and utility. Putting together all the components, a mining game is a tuple $(\mathbf{P}, \mathbf{Q}, \mathbf{A}, \mathbf{R}, \mathbf{Pr})$, where $\mathbf{P}$ is the set of players, $\mathbf{Q}$ is the set of states, $\mathbf{A}$ is the set of combined actions, $\mathbf{R}$ is the combined pay-off function and $\mathbf{Pr}$ is the transition probability function.

A strategy for a player $p \in \mathbf{P}$ is a function $s : \mathbf{Q} \to \mathbf{A}_p$. We define $\mathbf{S}_p$ as the set of all strategies for player p, and $\mathbf{S} = \mathbf{S}_0 \times \mathbf{S}_1 \times \cdots \times \mathbf{S}_{m-1}$ as the set of combined strategies for the game (recall that $\mathbf{P} = \{0, \ldots, m-1\}$ is the set of players). To define the notions of utility and equilibrium, we need some additional notation. Let $\mathbf{s} = (s_0, \ldots, s_{m-1})$ be a combined strategy. Then given $q \in \mathbf{Q}$, define $\mathbf{s}(q)$ as the combined action $(s_0(q), \ldots, s_{m-1}(q))$. Moreover, given an initial state $q_0 \in \mathbf{Q}$, the probability of reaching state $q \in \mathbf{Q}$, denoted by $\mathbf{Pr}^{\mathbf{s}}(q \mid q_0)$, is defined as 0 if $q_0 \not\subseteq q$ (that is, if q is not reachable from q_0), and otherwise it is recursively defined as follows: if $q = q_0$, then $\mathbf{Pr}^{\mathbf{s}}(q \mid q_0) = 1$; otherwise, we have that $|q| - |q_0| = k$, with $k \geq 1$, and

$$\mathbf{Pr}^{\mathbf{s}}(q \mid q_0) = \sum_{\substack{q' \in \mathbf{Q}\,: \\ q_0 \subseteq q' \text{ and } |q'| - |q_0| = k-1}} \mathbf{Pr}^{\mathbf{s}}(q' \mid q_0) \cdot \mathbf{Pr}(q', \mathbf{s}(q'), q).$$

In this definition, if for a player p we have that $s_p(q') = a$ and $a(q') = q$, then $\mathbf{Pr}(q', \mathbf{s}(q'), q) = h_p$. Otherwise, we have that $\mathbf{Pr}(q', \mathbf{s}(q'), q) = 0$ (this is well defined since there can be at most one player p whose action in the state q' leads us to the state q). For readability we write $\mathbf{Pr}^{\mathbf{s}}(q)$ instead of $\mathbf{Pr}^{\mathbf{s}}(q \mid \{\varepsilon\})$ to denote the probability of reaching state q from the intial state $\{\varepsilon\}$ that contains only the genesis block ε. The framework just described corresponds to a Markov Decision Process [13], but we do not explore this connection in this paper because we are not interested in the steady distributions of these processes.

Finally, we define the utility of players given a particular strategy. As is common when looking at personal utilities, we define it as the summation of the expected rewards, where future rewards are discounted by a factor of $\beta \in (0, 1)$ which is used to model the fact that money in the present is worth more than money in the future.

▶ **Definition 2.1.** *The β–discounted utility of a player p for a strategy $\mathbf{s}$ from a state q_0 in the mining game, denoted by $u_p(\mathbf{s} \mid q_0)$, is defined as:*

$$u_p(\mathbf{s} \mid q_0) \quad = \quad (1 - \beta) \cdot \sum_{q \in \mathbf{Q}\,:\, q_0 \subseteq q} \beta^{|q| - |q_0|} \cdot r_p(q) \cdot \mathbf{Pr}^{\mathbf{s}}(q \mid q_0).$$

Notice that the value $u_p(\mathbf{s} \mid q_0)$ may not be defined if this series diverges. To avoid this problem, from now on we assume that for every pay-off function $\mathbf{R} = (r_0, \ldots, r_{m-1})$, there exists a polynomial P such that $|r_p(q)| \leq P(|q|)$ for every player $p \in \mathbf{P}$ and state $q \in \mathbf{Q}$. Under this simple yet general condition, which is satisfied by the pay-off functions considered in this paper and in other game-theoretical formalisation's of Bitcoin mining [10], we can show that $u_p(\mathbf{s} \mid q_0)$ is a real number. Moreover, as for the definition of the probability of reaching a state from the initial state $\{\varepsilon\}$, we use notation $u_p(\mathbf{s})$ for the β–discounted utility of player p for the strategy $\mathbf{s}$ from $\{\varepsilon\}$, instead of $u_p(\mathbf{s} \mid \{\varepsilon\})$.

2.1 On the pay-off and utility of a miner

As mentioned earlier, we design our pay-off model with the goal of incentivising players to mine in the blockchain, and to keep their blocks in the blockchain. In this sense, the payment of a miner for a block b should be proportional to the amount of time b has been in the blockchain; in particular, the miner should be penalised if b ceases to be in the blockchain, and this penalty should decrease with time. In what follows, we explain how our pay-off model meets this goal.

Given a player p and a state q, for every block $b \in q$ assume that the reward obtained by p for the block b in q is given by $r_p(b, q)$, so that $r_p(q) = \sum_{b \in q} r_p(b, q)$. This decomposition can be done in a natural and straightforward way for the pay-off functions considered in this paper and in other game-theoretical formalisations of cryptomining [10, 11]. The reward for a mined block b is not granted immediately according to Definition 2.1, instead, a portion of $r_p(b, q)$ is paid in each state q where b is in. In other words, if a miner owns a block, then she will be rewarded for this block in every state where this block is part of the blockchain, in which case $r_p(b, q) > 0$.

Hence, in our model, a miner is payed a portion of a block's reward each time it is included in the blockchain, and even though she gets payed infinitely many times for each block, the discount factor in the definition of utility ensures that there is no overpay. In other words, when a player mines a new block, she will receive the full amount for this block only if she manages to maintain the block in the blockchain up to infinity. Otherwise, if this block ceases to be in the blockchain, the miner receives only a fraction of the full amount and, thus, is penalised. Formally, given a combined strategy $\mathbf{s}$, we can define the utility of a block b for a player p, denoted by $u_p^b(\mathbf{s})$, as follows:

$$u_p^b(\mathbf{s}) \quad = \quad (1-\beta) \cdot \sum_{q \in \mathbf{Q}\,:\, b \in \mathrm{bc}(q)} \beta^{|q|-1} \cdot r_p(q, b) \cdot \mathbf{Pr}^{\mathbf{s}}(q).$$

For simplicity, here we assume that the game starts in the genesis block ε, and not in an arbitrary state q_0. The discount factor in this case is $\beta^{|q|-1}$, since $|\{\varepsilon\}| = 1$.

To see that we pay the correct amount for each block, assume that there is a maximum value for the reward of a block b for player p, which is denoted by $M_p(b)$. Thus, we have that there exists $q_1 \in \mathbf{Q}$ such that $b \in q_1$ and $M_p(b) = r_p(b, q_1)$, and for every $q_2 \in \mathbf{Q}$ such that $b \in q_2$, it holds that $r_p(b, q_2) \leq M_p(b)$. Again, such an assumption is satisfied by most currently circulating cryptocurrencies, by the pay-off functions considered in this paper, and by other game-theoretical formalisations of cryptomining [10, 11]. Then we have that:

▶ **Proposition 2.2.** *For every player $p \in \mathbf{P}$, block $b \in \mathbf{B}$ and combined strategy $\mathbf{s} \in \mathbf{S}$, it holds that:* $u_p^b(\mathbf{s}) \leq \beta^{|b|} \cdot M_p(b)$.

Thus, the utility obtained by player p for a block b is at most $\beta^{|b|} \cdot M_p(b)$, that is, the maximum reward that she can obtained for the block b in a state multiplied by the discount factor $\beta^{|b|}$, where $|b|$ is the minimum number of steps that has to be performed to reach a state containing b from the initial state $\{\varepsilon\}$. Moreover, a miner can only aspire to get the maximum utility for a block b if once b is included in the blockchain, it stays in the blockchain in every future state. This tell us that our framework puts a strong incentive for each player in maintaining her blocks in the blockchain.

3 Equilibria with constant reward

The first version of the game we analyse is when the reward function $r_p(q)$ pays each block in the blockchain the same amount c. This is important for understanding what happens when currencies such as Ethereum or Monero switch to tail-emission, changing from a decreased reward scheme to a constant reward scheme. Further, it also helps us to establish the main techniques we use.

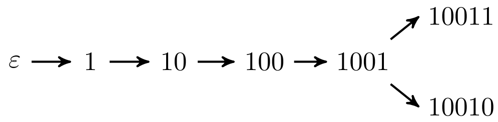


Figure 1 Although two paths are competing to become the blockchain, the blocks up to 1001 will contribute to the reward in both paths.

3.1 Defining constant reward

When considering the constant reward c for each block, $r_p(q)$ will equal c times the number of blocks owned by p in the blockchain $\mathrm{bc}(q)$ of q, when the latter is defined. On the other hand, when $\mathrm{bc}(q)$ is not defined it might seem tempting to simply define $r_p(q) = 0$. However, even if there is more than one longest path from the root of q to its leaves, it is often the case that all such paths share a common subpath (for instance, when two competing blocks are produced with a small time delay). While in this situation the blockchain is not defined, the miners know that they will at least be able to collect their reward on the portion of the state these two paths agree on. Figure 1 illustrates this situation.

Recall that a block b is a string over the alphabet $\mathbf{P}$, and we use notation $|b|$ for the length of b as a string. Moreover, given blocks b_1, b_2, we use $b_1 \preceq b_2$ to indicate that b_1 is a prefix of b_2 when considered as strings. Then we define:

$$\begin{aligned} \mathrm{longest}(q) &= \{b \in q \mid \text{for every } b' \in q : |b'| \leq |b|\} \\ \mathrm{meet}(q) &= \{b \in q \mid \text{for every } b' \in \mathrm{longest}(q) : b \preceq b'\}. \end{aligned}$$

Intuitively, $\mathrm{longest}(q)$ contains the leaves of all paths in the state q that are currently competing to be the blockchain, and $\mathrm{meet}(q)$ is the path from the genesis block to the last block for which all these paths agree on. For instance, if q is the state from Figure 1, then we have that $\mathrm{longest}(q) = \{10011, 10010\}$, and $\mathrm{meet}(q) = \{\varepsilon, 1, 10, 100, 1001\}$. Notice that $\mathrm{meet}(q)$ is well defined as $\preceq$ is a linear order on the finite and non-empty set $\{b \in q \mid \text{for every } b' \in \mathrm{longest}(q) : b \preceq b'\}$. Also notice that $\mathrm{meet}(q) = \mathrm{bc}(q)$, whenever $\mathrm{bc}(q)$ is defined.

The reward function we consider in this section, which is called **constant reward**, is then defined for a player p as follows :

$$r_p(q) = c \cdot \sum_{b \in \mathrm{meet}(q)} \chi_p(b),$$

where c is a positive real number, $\chi_p(b) = 1$ if $\mathrm{owner}(b) = p$, and $\chi(p) = 0$ otherwise. Notice that this function is well defined since $\mathrm{meet}(q)$ always exists. Moreover, if q has a blockchain, then we have that $\mathrm{meet}(q) = \mathrm{bc}(q)$ and, hence, the reward function is defined for the blockchain of q.

3.2 The default strategy maximizes the utility

Let us start with analysing the simplest strategy, which we call the *default* strategy: regardless of what everyone else does, keep mining on the blockchain. More precisely, a player following the default strategy tries to mine upon the final block that appears in the blockchain of a state q. If the blockchain in q does not exist, meaning that there are at least two longest paths from the genesis block, then the player tries to mine on the final block of the path that maximizes her reward, which in the case of constant reward corresponds to the path containing the largest number of blocks belonging to her (if there is more than one of these paths, then between the final blocks of these paths she chooses the first according to a lexicographic order on the strings in $\{0,\ldots,m-1\}^*$). Notice that this is called the default strategy as it reflects the desired behaviour of the miners participating in the Bitcoin network. For a player p, let us denote this strategy by DF_p, and consider the combined strategy $\mathbf{DF} = (\mathrm{DF}_0, \mathrm{DF}_1, \ldots, \mathrm{DF}_{m-1})$.

We can easily calculate the utility of player p under $\mathbf{DF}$. Intuitively, a player p will receive a fraction h_p of the next block that is being placed in the blockchain, corresponding to her hash power. Therefore, at stage i of the mining game, the blockchain defined by the game will have i blocks, and the expected amount of blocks owned by the player p will be $h_p \cdot i$. The total utility for player p is then

$$u_p(\mathbf{DF}) \quad = \quad (1-\beta) \cdot h_p \cdot c \cdot \sum_{i=0}^{\infty} i \cdot \beta^i \quad = \quad h_p \cdot c \cdot \frac{\beta}{(1-\beta)}.$$

The question then is: can any player do better? As we show in the following theorem, the answer is no, as the default strategy maximizes the utility.

▶ **Theorem 3.1.** *Let p be a player, β be a discount factor in $(0,1)$ and u_p be the utility function defined in terms of β. Then for every combined strategy $\mathbf{s}$:* $\quad u_p(\mathbf{s}) \leq u_p(\mathbf{DF})$.

The proof of this theorem relies on the fact that, under constant rewards, forking becomes less profitable because all blocks are worth the same amount of money, regardless of their position. This fact, combined with the economic discount, provides little incentives for players to sacrifice some time in order to fight for a longer blockchain: their reward is higher if instead of fighting they just keep mining on the blockchain.

A strategy $\mathbf{s}$ is a Nash equilibrium from a state q_0 in the mining game for m players if for every player $p \in \mathbf{P}$ and every strategy s for player p ($s \in \mathbf{S}_p$), it holds that $u_p(\mathbf{s} \mid q_0) \geq u_p((\mathbf{s}_{-p}, s) \mid q_0)$ (here as usual we use $(\mathbf{s}_{-p}, s)$ to denote the strategy $(s_0, s_1, \ldots s_{p-1}, s, s_{p+1}, \ldots, s_{m-1})$). As a corollary of Theorem 3.1, we obtain

▶ **Corollary 3.2.** *For every $\beta \in (0,1)$, the strategy $\mathbf{DF}$ is a Nash equilibrium.*

Hence, miners looking to maximise their wealth are better off with the default strategy. Especially this results prove that long fork should not happen and therefore validate the underlying assumption of other models [10]. Interestingly, previous work shows that under a setting in which miners are rewarded for the fraction of blocks they own against the total number of blocks, and no financial discount is assumed, then default strategy may not be an optimal strategy [10]. This suggests that miner's behaviour can really deviate depending on what are their short and long term goals, and we believe this is an interesting direction for future work.

4 Decreasing Reward

Miner's fees in many cryptocurrencies, including Bitcoin and Monero, are not constant, but decrease over time. We model such fees as a constant factor $\alpha \in [0,1]$ that is lowered after every new block in the blockchain. That is, we use the following reward function r_p for all players $p \in \mathbf{P}$, denoted as the **α-discounted reward**:

$$r_p(q) \quad = \quad c \cdot \sum_{b \in \mathrm{meet}(q)} \alpha^{|b|} \cdot \chi_p(b).$$

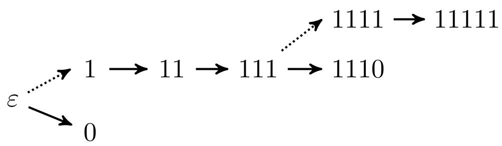


Figure 2 Dashed arrows indicate when player 1 does a fork. The first block (block 0) is mined by player 0. At this point, player 1 decides to fork (mining the block 1), and successfully mines the blocks 11 and 111 on this branch. When player 0 mines the block 1110, player 1 decides to fork again, mining the blocks 1111 and 11111.

In this section, we show that forking can be a good strategy when miner's fees decrease over time. Not only we confirm the folklore fact that it is profitable to fork with more than half of the hash power, but our exploration gives us a concrete strategy that beats the default with less than half of the hash power.

4.1 When is forking a good strategy?

To calculate when forking is a viable option, we consider a scenario when one of our m players decides to deviate from the default strategy, while the remaining players all follow the default strategy. In this case we can reduce the m player game to a two player game, where all the players following the default strategy are represented by a single player with the combined hash power of all these players. Therefore in this section we will consider that the mining game is played by two players 0 and 1, where 0 represents the miners behaving according to the default strategy, and 1 the miner trying to determine whether forking is economically more viable than mining on the existing blockchain. We always assume that player 1 has hash power h, while player 0 has hash power $1 - h$.

Let us first show the utility for player 1 when she uses the default strategy $\mathbf{DF} = (\mathrm{DF}_0, \mathrm{DF}_1)$.

▶ **Lemma 4.1.** *If h is the hash power of player 1, then*

$$u_1(\mathbf{DF}) \quad = \quad h \cdot c \cdot \frac{\alpha \cdot \beta}{(1 - \alpha \cdot \beta)}.$$

As in the case of constant reward, this corresponds to h times the utility of winning all the blocks in the single blockchain generated by the default strategy.

Now suppose that player 1 deviates from the default strategy, and considers a strategy based on forking the blockchain once player 0 mines a block. How would this new strategy look? In this section we consider the strategy AF (for *always fork*), where player 1 forks as soon as player 0 mines a block in the blockchain, and she continues mining on the new branch until it becomes the blockchain. Here player 1 is willing to fork every time player 0 produces a block in the blockchain. In other words, in AF, player 1 tries to have all the blocks in the blockchain. This strategy is depicted in Figure 2.

The utility of always forking. We want to answer two questions. On the one hand, we want to know whether AF is a better strategy than DF_1 for player 1, under the assumption that player 0 uses DF_0, and under some specific values of α, β and h. On the other hand, and perhaps more interestingly, we can also answer a more analytical question: given realistic values of α and β, how much hash power does player 1 need to consider following AF instead of DF_1? Answering both questions requires us to compute the utility for the strategy $\mathbf{AF} = (\mathrm{DF}_0, \mathrm{AF})$.

▶ **Theorem 4.2.** *Let* $\mathbf{C}(x) = \frac{1-\sqrt{1-4x}}{2x}$ *denote the generating function of Catalan numbers. If h is the hash power of player 1, then*

$$u_1(\mathbf{AF}) = \frac{\Phi}{1-\Gamma}, \text{ where } \Phi \text{ and } \Gamma \text{ are defined as:}$$

$$\Phi = \frac{\alpha \cdot \beta \cdot h \cdot c}{(1-\alpha)} \cdot \left[\mathbf{C}(\beta^2 \cdot h \cdot (1-h)) - \alpha \cdot \mathbf{C}(\alpha \cdot \beta^2 \cdot h \cdot (1-h))\right],$$

$$\Gamma = \alpha \cdot \beta \cdot h \cdot \mathbf{C}(\alpha \cdot \beta^2 \cdot h \cdot (1-h))$$

Let us give some intuition on this result. Player 1, adopting the AF strategy, will always start the game mining on ε, regardless of how many blocks player 0 manages to append, and continues until her branch is the longest. Therefore, the only states that contribute to player 1's utility are those in where she made at least one successful fork (all others states give zero reward to her). Having player 1 achieved the longest branch once, say, at block b, both players will now mine on b and the situation repeats as if b were ε, with proper shifting in the reward and β-discount. In other words, we have $u_1(\mathbf{AF}) = \Phi + \Gamma \cdot u_1(\mathbf{AF})$, where Φ is the contribution of a single successful fork, and Γ is the shifting factor, from which we obtain the expression for $u_1(\mathbf{AF})$ given before.

Now, in order to quantify the contribution of Φ on successful forks, we need to sum over all possible moments of time in which this fork was finally made, weighted by the possibility that such a fork was actually made. However, this is not direct because there may be different paths leading to the same state, and therefore the probability of forking at a certain stage depends on the length and the form of the state. We quantify these by bringing out an analogy between Dyck words [21] and paths leading to states in which player 1 forks successfully for the first time. Then the theorem uses the fact that the number of Dyck words of length $2m$ is the m-th Catalan number.

When is AF better than DF? With the closed forms for $u_1(\mathbf{DF})$ and $u_1(\mathbf{AF})$, we can compare the utilities of these strategies for player 1 for fixed and realistic values of α and β, but varying her hash power. For α we calculate the compound version of the discount in Bitcoin, that is, a value of α that would divide the reward by half every 210.000 blocks, i.e. $\alpha = 0.9999966993$. For β we calculate the 10-minute rate that is equivalent to the US real interest rate in the last few years, which is approximately 2%. This gives us a value of $\beta = 0.9999996156$.

Figure 4a shows the value of the utility of player 1 for the combined strategies **AF** and **DF** (this figure also includes two other strategies that will be explained in the next section). The plot data was generated using GMP C++ multi precision library [7]. The point where the utility for **AF** and **DF** meet is $h = 0.499805 \pm 0.000001$, which means that player 1 should use **AF** as soon as she controls more than this proportion of the hash power (a similar result was obtained in [10], although in a model without discounted reward).

4.2 Giving up for more utility

By adding a little more flexibility to the strategy of always forking, we can identify approaches that make a fork profitable with less hash power. The families of strategies that we study in this section involve two parameters. The first parameter, denoted by k, regulates how far back the miner will fork, when confronted with a chain of blocks she does not own. The second parameter, called the *give-up* time, and denoted by ℓ, tells us the maximum number of blocks that the player's opponent is allowed to extend the current blockchain with before the player gives up mining on the forking branch. If the player does not manage to transform

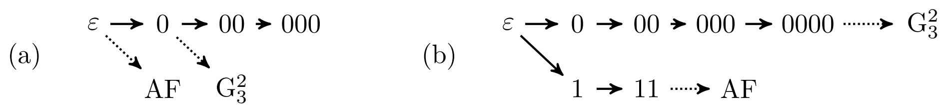
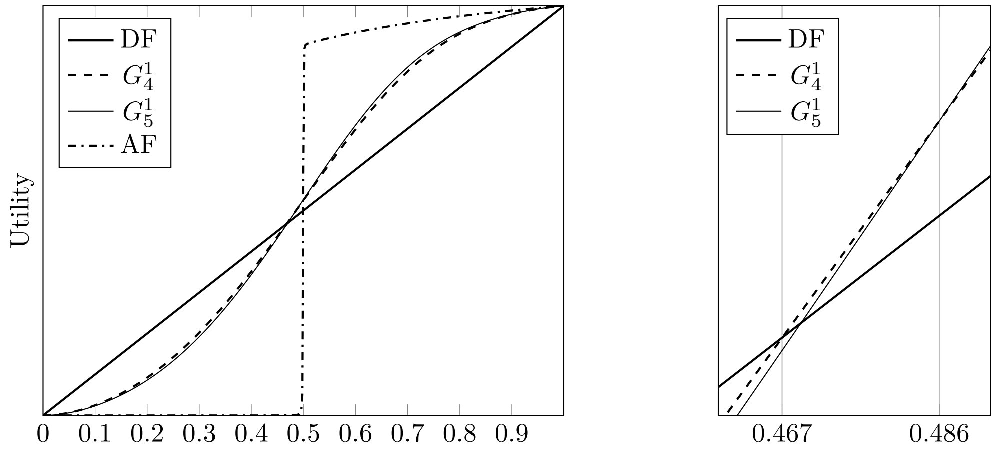


Figure 3 Difference between AF and G^2_3 in terms of actions in two states.



(a) Utilities for player 1 of combined strategies, **DF**, $\mathbf{G}^1_4$, $\mathbf{G}^1_5$ and **AF**

(b) Zoom in around the intersection

Figure 4 Comparing the utilities of three forking strategies against the default strategy

her fork into the new blockchain before her opponent mines more than ℓ blocks, she will restart the strategy treating the tail of the current blockchain as the new genesis block. We denote these strategies by G^k_ℓ.

▶ **Example 4.3.** Let us compare G^2_3 and AF. Since $k = 2$, both strategies take the same action when the state is $\{\varepsilon\}, \{\varepsilon, 0\}$, and $\{\varepsilon, 0, 00\}$, namely, mining at ε. Hence, assume that both strategies are facing a chain of three blocks owned by player 0, as shown in Figure 3(a). In this case, AF would again try to do a fork from the genesis block as no block belongs to player 1. On the other hand, G^2_3 would try to fork on the dotted line, that is, the second block that does not belong to her. The second difference is provided by the give-up time, which is shown in Figure 3(b). Normally, AF is willing to continue forking regardless of the hope of winning, therefore the move for the state in Figure 3(b) would still be to mine upon her own block 11. On the other hand, G^2_3 has now seen 4 blocks from the start of the fork (one more than the maximum $\ell = 3$), so with this strategy player 1 instead gives up and tries to mine upon 0000, rebooting the strategy as if 0000 was the genesis block. Note also that $\text{AF} = G^\infty_\infty$.

Define $\mathbf{G}^k_\ell$ as the combined strategy (DF_0, G^k_ℓ). We obtain an analytical form similar to that of Theorem 4.2, except in this case the set of paths leading to winning states has a more complex combinatorial nature, as expected when taking into account the parameters k and ℓ.

▶ **Theorem 4.4.** *For every pair of positive integers ℓ, k with $k < \ell$, we have that:*

$$u_1(\mathbf{G}^k_\ell) = \frac{\Phi_{\ell,k}}{1 - \Gamma_{\ell,k}},$$

where $\Phi_{\ell,k}$ and $\Gamma_{\ell,k}$ are rational functions of α, β and h.

In the proof of this theorem, we develop precise expressions for $\Phi_{\ell,k}, \Gamma_{\ell,k}$. The proof extends the techniques used to show Theorem 4.2, where we again look to compute the weighted sum of all states where player 1 manages to fork. This weighted sum, however, requires much more involved computation; we use a new combinatorial result that involve two sets of polynomials related to Dyck words.

We use Theorem 4.4 to analyse these strategies, plotting them, as we did before, for $\alpha = 0.9999966993$ and $\beta = 0.9999996156$. Figures 4a and 4b give interesting information about the advantages of these strategies. We fix $k = 1$, and plot in Figure 4a the utilities of combined strategies **DF**, $\mathbf{G}_4^1$, $\mathbf{G}_5^1$ and **AF**. In Figure 4b, we zoom in around the values of the hash power where **DF** intersects with $\mathbf{G}_4^1$ and $\mathbf{G}_5^1$. As we see in the figures, for a fork window $k = 1$, the optimal amount time player 1 should be willing to fight for a branch before giving up depends on the hash power. With little hash power the likelihood of winning a branch is small, so player 1 should give up as early as possible. However, the more hash power she obtains, the better it is to wait more. Interestingly, with more than 46.7% of the hash power, player 1 already should start using strategy G_4^1 to defeat the default strategy, and with more than 48.6% hash power, she should adopt G_5^1. We know that player 1 should use AF not before around $h = 0.499805$, so this gives us a lot of extra room to look for optimal strategies if we are willing to fork (especially considering that every percentage of hash power in popular cryptocurrencies may cost millions).

Plots for strategies with $k > 1$ present a similar behaviour: the more hash power we have, the more we should be willing to fight for our forks. The strategies we include in Figure 4 beat the default strategy under the least amount of hash power amongst any combination of values for k and ℓ with $k < \ell \leq 100$. The comparison is much less straightforward when looking at varying values of both k and ℓ, but in general, the more hash power the bigger the window of blocks one should aim to do a fork, and the more one should wait before giving up.

5 Concluding remarks

Our model of mining via a stochastic game allows for an intuitive representation of miners' actions as strategies, and gives us a way of understanding the rational behaviour of miners looking to accumulate cryptocurrency wealth. As it is the first model to provide payoff to miners for every branching strategy we can validate the commonly accepted assumption that long forks are not a viable strategy. In this respect, we would like to identify strategies that are a Nash equilibrium for the case of decreasing rewards. However, this has proven to be a difficult task. In particular, one can show that the default strategy can never be part of such an equilibrium, no matter how small the hash power is for one of the players, if the strategy of another player involves forks of any length. This means that one must look for much more complex strategies to find such an equilibrium.

One of the advantages of our model is its generality: it can be adapted to specify more complex actions, study other forms of reward and include cooperation between miners. For example, we are currently looking at strategies that involve withholding a mined block to the rest of the network, for which we need a slight extension of the notions of action and state. It would be very interesting how this model and previous work combine into a model where miner's behavior can deviate depending on both their short-and long-term goals. We would also like to study incentives under different models of cooperation between miners, and also other forms of equilibria in a dynamic setting.

References

1 Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). 2013. URL: http://arxiv.org/abs/1312.7013.

2 Bruno Biais, Christophe Bisiere, Matthieu Bouvard, and Catherine Casamatta. The blockchain folk theorem. 2018.

3 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*, 2016.

4 Mauro Conti, Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 2018.

5 Swapnil Dhamal, Tijani Chahed, Walid Ben-Ameur, Eitan Altman, Albert Sunny, and Sudheer Poojary. A stochastic game framework for analyzing computational investment strategies in distributed computing with application to blockchain mining. *arXiv preprint arXiv:1809.03143*, 2018.

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

7 GNU. The gnu multiple precision arithmetic library (v. 6.1.2). 2018. URL: https://gmplib.org/.

8 Ethan Heilman. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner (poster abstract). In *Financial Cryptography and Data Security*, 2014.

9 Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In *Financial Cryptography and Data Security*, 2014.

10 Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, pages 365–382, 2016.

11 Elias Koutsoupias, Philip Lazos, Foluso Ogunlana, and Paolo Serafino. Blockchain mining games with pay-forward. 2019. *To appear in The Web Conference.*

12 Joshua A. Kroll, Ian C. Davey, and Edward W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *The Twelfth Workshop on the Economics of Information Security (WEIS 2013)*, 2013.

13 Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

14 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

15 Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction.* Princeton University Press, 2016.

16 Arvind Narayanan and Jeremy Clark. Bitcoin's academic pedigree. *Commun. ACM*, 60(12):36–45, 2017.

17 Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. *IACR Cryptology ePrint Archive*, 2015:796, 2015.

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

19 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security*, pages 515–532, 2017.

20 Alexander Spiegelman, Idit Keidar, and Moshe Tennenholtz. Game of coins. *arXiv preprint arXiv:1805.08979*, 2018.

21 R.P. Stanley. *Catalan Numbers.* Cambridge University Press, 2015.

22 Christopher P Thompson. *Ethereum - Distributed Consensus (A Concise Ethereum History Book)*. CreateSpace Independent Publishing Platform, 2017.
23 Itay Tsabary and Ittay Eyal. The gap game. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 713–728. ACM, 2018.
24 Marie Vasek, Micah Thornton, and Tyler Moore. Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In *Financial Cryptography and Data Security*, 2014.
25 Bitcoin Cash Website. `https://www.bitcoincash.org`, 2018.
26 Ethereum Website. `https://ethereum.org`, 2018.
27 Litecoin Website. `https://litecoin.org`, 2018.
28 Monero Website. `https://getmonero.org`, 2018.