The Halt Game: Sometimes Rewards Cannot Cover Expenses In The PoW-Based Blockchain

Junjie Hu, *Student Member, IEEE,* Huan Yan, *Student Member, IEEE,*
Na Ruan, *Senior Member, IEEE* and Zhen Xiao, *Senior Member, IEEE*

***Abstract*—Proof-of-work (PoW) blockchain relies on incentive mechanisms to ensure the security and correctness of its underlying consensus protocol. Most research about it, based on a static model only considering coin-base rewards and transaction fee rewards, fails to accurately describe the complex real-world blockchain ecosystem. We propose a generic selfish mining attack applicable to arbitrary PoW blockchain systems and introduce a dynamic PoW blockchain incentive model. This model takes into account static basic rewards, dynamic whale rewards related to network protocol, and expenditures tied to players' strategies. Unlike traditional incentive models that assume players continuously mine by default, we find players prefer to halt mining at the beginning of each mining cycle to reduce operational expenses and then resume mining at an appropriate time to enhance their rewards. We further proof players' optimal strategy exists and it is determined by reward parameters. We implement a modified PoW blockchain system simulator and comprehensively validate these results using 256 full nodes in it of three mainstream PoW blockchains: Bitcoin, Ethereum 1.x, and Bitcoin Cash. We finally discuss the impact of different parameters on the security of PoW blockchain systems and propose practical mitigating measures for the mining halt.**

***Index Terms*—Proof-of-work, Blockchain, the Halt Game, Incentive Mechanism, Mining Game.**

I. INTRODUCTION

PROOF-of-work (PoW) blockchain systems, exemplified by Bitcoin [1] and Ethereum 1.x [2], have facilitated the creation of cryptocurrencies with market valuations amounting to billions of dollars [3]. These blockchain systems are grounded in decentralized blockchain protocols and leverage PoW mechanisms to guarantee their security. The blockchain is maintained and operated by participants, termed *miners*, who possess and control a certain quantity of computational resources. Miners acquire accounting rights by solving intricate cryptographic puzzles. To incentivize their participation, the blockchain system offers rewards to miners who successfully generate valid blocks. Ideally, the rewards allocated to miners are relatively equitable, reflecting the proportion of computational resources they invest.

This research has received support from National Key R&D Program of China (2023YFB2704700), National Natural Science Foundation of China (62472276), Shanghai Committee of Science and Technology, China (23511101000, 24BC3200400), Science and Technology Project of the State Grid Corporation of China (5700-202321603A-3-2-ZN).

Junjie Hu and Na Ruan are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: nakamoto@sjtu.edu.cn, naruan@sjtu.edu.cn). Zhen Xiao is with the Department of Computer Science, Peking University, Beijing 100871, China (e-mail: xiaozhen@pku.edu.cn).

Corresponding author: Na Ruan.

However, PoW blockchain is susceptible to selfish mining attacks [4], where miners can strategically withhold and release private blocks to gain disproportionate advantages. Such attacks undermine the stability of the system, compromising its security and liveness. Several studies [5], [6] have analyzed the minimum security threshold necessary to initiate selfish mining, emphasizing the importance of setting this security threshold as high as possible to deter miners from engaging in such attacks. To our knowledge, these studies on selfish mining have primarily focused on a model that solely considers the rewards associated with successfully generating blocks, neglecting the expenses involved in renting, operating mining machines, and other intricate transaction fee incentives. In real-world blockchain systems, however, these objectively existing rewards and expenses cannot be overlooked, as they significantly influence the mining strategies adopted by participants [7]–[9].

We propose a dynamic PoW blockchain incentive model that incorporates both rewards and expenses. Our model expands upon the traditional PoW-based blockchain incentive framework by encompassing coin-base rewards, transaction fee rewards, whale transaction incentives, capital expenditures, and operational expenses. To apply this dynamic PoW incentive model to different types of PoW blockchain systems, we further propose a generic selfish mining applicable to any PoW blockchain. We focus on determining the optimal mining strategy for generic selfish players within this comprehensive blockchain incentive model. To this end, we develop a modified PoW blockchain system simulator to simulate the construction process of the PoW blockchain. Additionally, we implement an equilibrium search tool to identify the optimal initiation times for selfish and honest participants when the system attains a Nash equilibrium.

We formulate and analyze a game, termed the *halt game*, within the context of three distinct PoW blockchain platforms: Bitcoin (BTC), Ethereum 1.x (ETH), and Bitcoin Cash (BCH). We aim at optimizing the profits of generic selfish players, with honest players adhering to the prescribed protocols. Specifically, the *halt game* entails a strategy where mining machines cease mining at the commencement of each mining cycle, with the option to resume mining at some intermediate point or to refrain from mining entirely, thereby minimizing expenses and maximizing utilities. The decision-making process in this game is contingent upon various parameters, including rewards and expenses.

To accomplish our objectives, we initially develop a realistic PoW blockchain incentive model for the mining process,

accounting for varying transaction fee rewards and expenses (in Section III). Subsequently, we conduct a rigorous game-theoretic analysis of the blockchain system and examine the variations in system parameters across three distinct scenarios. The experimental outcomes corroborate our theoretical analysis (in Section IV). We define the mining strategies for honest and generic selfish participants, and then establish a Markov model [10] for generic selfish mining applicable to any PoW blockchain. Notably, unlike previous studies, the state transition rate in our Markov model is dynamic, adapting to the different mining strategies employed by players within different Markov states. Based on this Markov model, we derive the expected total utility (the difference between expected rewards and expected expenses) for both generic selfish and honest players. To validate our theoretical findings, we implement a modified PoW blockchain system simulator and an equilibrium search tool. Utilizing the equilibrium search tool, we can derive the Nash equilibrium strategies for honest players and the optimal generic selfish players. With the modified PoW blockchain system simulator, we can obtain the long-term total utilities for both types of players. By integrating these two tools, we ultimately achieve the optimal generic selfish mining strategy under the Nash equilibrium, along with the corresponding long-term total utilities. (in Section V). Furthermore, we conduct an in-depth investigation into the generic selfish mining strategies and uncover several intriguing insights (in Section VI). Finally, we discuss potential measures to mitigate the halt game within our incentive model and provide guidance for the design of innovative blockchain systems (in Section VII).

In summary, our principal contributions are outlined as follows:

- We present a dynamic PoW blockchain incentive model that comprehensively incorporates coin-base rewards, transaction fee rewards, whale transaction incentives, capital expenditures, and operational expenses.
- We propose a generic selfish mining applicable to any PoW blockchain, and subsequently establish a Markov model for it that is applicable to any PoW blockchain.
- We derive the expected total utility for both generic selfish and honest miners based on the Markov model, which is a dynamic function contingent upon the miners' strategies and system parameters.
- We implement a modified PoW blockchain system simulator and an equilibrium search tool, which enable us to ascertain the long-term total utility of each player and the optimal mining strategy under Nash equilibrium, respectively.
- We identify the occurrence of the halt game within the system, conduct an extensive analysis across various scenarios, and propose effective countermeasures to mitigate the aforementioned dilemmas.

II. PRELIMINARIES

Proof-of-work blockchain. PoW [11], [12] blockchain protocol accounts for approximately 80% of the total market value of cryptocurrencies, such as Bitcoin [1] and Ethereum 1.x [2]. Participants engaged in mining within the blockchain system, commonly referred to as miners, aggregate and store users' transactions in the form of merkle trees within blocks, sequentially ordering these blocks to form a blockchain. The blockchain represents a decentralized system that allows anyone to join at any time and append new blocks by demonstrating the completion of a specific computational task, thereby serving as proof-of-work.

Specifically, miners engage in an iterative process to deduce the solution to the cryptographic puzzle necessary for generating a valid block. This process can be formalized as a Bernoulli trial: miners randomly propose an answer, yielding a "true" outcome if the answer aligns with the criteria (successfully finding a valid block) or a "false" outcome if it does not (prompting another random guess). A series of independent Bernoulli trials collectively constitute a Bernoulli process. Upon observing a sequence of such trials, the number of attempts required to achieve a successful result conforms to a geometric distribution. Consequently, the time taken to successfully discover a valid block is distributed according to an exponential distribution. Notably, both the geometric and exponential distributions exhibit memorylessness, thereby imparting a memoryless characteristic to the mining process. This implies that the success rate for each trial remains constant and is governed by the difficulty parameter of the protocol aforementioned. A miner's likelihood of discovering a valid solution remains unaffected by their previous failures. Therefore, upon a miner successfully acquiring a valid block, by integrating it into their local blockchain and initiating the mining process anew, their chances of mining a subsequent block remain unchanged, a property also known as progress-free. Thus the success rate of a miner in completing this computational task is directly proportional to their normalized computational power. Ideally, the proportion of a miner's computational power is expected to equate to their rate of discovering new blocks. However, in practice, attackers with a substantial majority of computational power, exceeding 51%, may deviate from the system's protocols. This underscores the notion that increased mining power participating in the blockchain system enhances its robustness and security. In the absence of centralized enforcement, the security of the PoW blockchain is heavily contingent upon each miner's adherence to the prescribed protocol, as this constitutes their optimal response.

Mining process. The process of mining a new block can be delineated as follows. A miner first compiles a collection of pending transactions intended for inclusion in the next valid block. Subsequently, the miner endeavors to solve a cryptographic puzzle, which is formulated based on the selected pending transactions, the most recently appended block, and the underlying blockchain protocol. Upon discovering a valid solution, the miner disseminates the block across the network, prompting other miners to ascertain its validity. If the block is confirmed as valid, the miners collectively agree to incorporate it into their respective local blockchain copies and proceed to restart the mining process, with the newly validated block serving as the tip block [4], [13].

TABLE I
MINING REWARDS IN DIFFERENT POW BLOCKCHAIN SYSTEMS.

Reward Type	ETH	BTC	BCH	Purpose
Coin-base reward	√	√	√	To reward the players who solve the puzzle and motivate them to continue mining.
Transaction fee (Gas fee) rewards	√	√	√	To execute pending transactions.
Uncle reward	√	×	×	To compensate the players whose blocks have not been selected as the main chain and reduce centralization trend of mining.
Nephew reward	√	×	×	To stimulate the players to merge the forked blockchain.
Whale transaction reward	√	√	√	To prioritize the inclusion of the whale transaction into a pending-published block.

Difficulty parameter adjustment. Blockchain allows miners to join or depart the network at any time, which can result in fluctuations in block interval times-a phenomenon that is undesirable. To mitigate this issue, the blockchain protocol incorporates difficulty parameters that undergo periodic adjustments to maintain a consistent block time interval [14]. As the aggregate computational power engaged in the protocol escalates, adjusting the cryptographic puzzle to have a smaller target becomes pivotal in preserving the stability of the block time interval. Conversely, an increase in the difficulty parameter also diminishes the likelihood of miners with equivalent computational power discovering a new block, thereby influencing the mining rate [15].

Reward and expense. To incentivize participants to engage in mining and contribute to the network, the system grants rewards to the miner who successfully solves the cryptographic puzzle, typically in the form of cryptocurrency. The composition of these rewards varies across different blockchain systems. For instance, in the Bitcoin network, rewards consist of newly minted coin-base rewards for publishing a valid block, along with transaction fee rewards and whale transaction rewards paid by the transactions incorporated within the block. In contrast, in Ethereum 1.x, the rewards encompass not only coin-base rewards and gas fee rewards (analogous to transaction fee rewards) and whale transaction rewards, but also uncle rewards and nephew rewards, which are unique to the Ethereum ecosystem, as described in Table I. The mining process also entails associated expenses. Specifically, mining machines utilized in this process necessitate capital expenditures for their acquisition and rental, as well as operational expenses for their maintenance and functioning [8], [9].

To facilitate a deeper understanding of the composition of mining rewards, we employ an illustrative example to delineate the distinctions among various types of rewards. As depicted in Figure 1, solid white blocks signify regular blocks that constitute the longest valid chain; their creators are entitled to corresponding coin-base rewards, gas fee rewards, and whale transaction rewards. Solid gray blocks represent uncle blocks, which are the initial blocks linking a regular block on a non-longest valid chain. When an uncle block is referenced by a subsequent regular block, its miners are eligible for uncle rewards. Solid blue blocks encompass both regular and nephew blocks, situated on the longest valid chain and referencing other uncle blocks; their miners receive coin-base rewards, gas fees, and nephew rewards. Conversely, dashed white blocks symbolize orphan blocks, for which their creators receive no rewards, serving as a penalty for deviant mining practices.

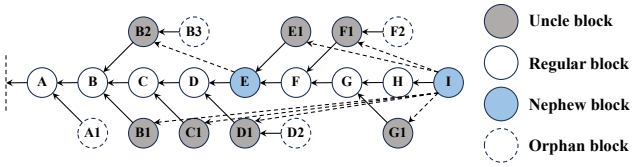


Fig. 1. Different block types in Ethereum 1.x.

The quantities of coin-base rewards, transaction fee rewards, whale transaction rewards, uncle rewards, and orphan penalties are dictated by the blockchain protocol. It is worth noting that each pending transaction carries different transaction fees, and miners have the autonomy to select which transactions to include within a block.

In the two most prominent blockchain systems currently, namely Bitcoin and Ethereum, rewards are primarily composed of coin-base rewards. In Bitcoin, the expected daily subsidy amounts to 3.125 BTCs×6×24=450 BTCs, given that the average block interval is 10 minutes and each coin-base reward is 3.125 BTCs. Meanwhile, the daily transaction fees paid average approximately a few hundred BTCs per day [16]. In Ethereum 1.x, approximately 20,000 new Ethers are minted daily [17], while transaction fees amount to roughly 2,000 Ethers daily [18]. Notice that Ethereum 2.0 has undergone a transition from PoW to PoS, resulting in the elimination of mining.

Fork. Blockchain, as a decentralized system, inherently faces the risk of forking due to the absence of centralized control. A fork that arises from disagreements among different participants regarding blockchain system protocols is termed a *protocol fork* or *hard fork* [19] (e.g., pay-to-script-hash, or P2SH [20]). Conversely, a fork resulting from disagreements over the current state of the blockchain system is referred to as a *state fork* or *soft fork* [19], which can occur either unintentionally or deliberately. For instance, when a miner discovers a valid block, they broadcast it to other participants; however, due to communication delays in network transmission, this can lead to inconsistent local views among participants. If another block is generated concurrently, an unintentional state fork inevitably occurs [21]. Furthermore, participants may intentionally deviate from the protocol and launch malicious attacks to gain disproportionate advantages, resulting in deliberate state forks [4]. It is important to note that all mining attacks discussed in this paper constitute deliberate state forks.

III. MODEL

Assumption. We make the following assumptions to simplify our analysis, which are grounded in the assumptions of other mining attacks, such as selfish mining attack [4], [22], block withholding attack [4], [23]–[25], and fork after withholding attack [26], [27].

- Assume all mining machines have the same computational power and formalize participants' computational power as the number of mining machines they own. Hence the relative computational power of a miner is equal to the ratio of the number of mining machines they own to the total number of mining machines in the network.
- Formalize coin-base rewards as λ_0 instead of 3.125BTCs or 3Ethers. Additionally, other rewards and expenses are formalized as functions about λ_0. It makes our results apply to Bitcoin, Ethereum 1.x and other incentive schemes.
- Neglect network transmission time. This assumption ensures that the system will not generate unintentional state forks. It is rational because the probability of unintentional forks occurring in the Bitcoin system can be negligible, approximately 0.41% [21], [26].
- Honest participants are profit-driven but honest. They mine following the protocol, yet they can choose whether and when to turn their mining machines on autonomously.
- Participants include all the pending transactions in the network into the block by default. This is reasonable because players are profit-driven, and collecting more transactions is more profitable.
- Attackers can only conduct the behavior of the generic selfish mining strategy defined in this paper, without any other malicious behavior. That is, attackers cannot perform other mining attacks.

Model. We propose a realistic model of a generic selfish mining process with varying transaction fees and expenses throughout the rest of this work. Similar to what is commonly done in blockchain analysis [21], [28], we simulate the system in a quasi-static state [29]. This means miners maintain their behaviors, reaching an equilibrium in the system. To achieve this, in our model, we consider that the system consists of k mining machines $(m_1, m_2, \ldots, m_k)$ controlled by n players $(p_1, p_2, \ldots, p_n)$. The player p_i uses index $m_{\{i\}}$ to control the set of machines, satisfying $\forall m_{\{i\}} : m_{\{i\}} \neq \emptyset, \forall i \neq j : m_{\{i\}} \cap m_{\{j\}} = \emptyset$ and $\cup_{i=1}^{n} m_{\{i\}} = \{m_1, m_2, \ldots, m_k\}$. For simplicity, we assume that mining machines are identical. Let $S_n \in \{off, on\}$ denotes the state of m_i, with off being the default state and on. Each miner assigns a start time to each machine they control, which is the time when the machine is opened. The start time of each machine m_i is s_i, and we denote $a_i(t)$ as the set of machines that player p_i has turned on by time t and $a(t)$ as the set of active machines of all players, thus we have $a_i(t) = \{j | j \in m_{\{i\}} \wedge s_j < t\}$ and $a(t) = \cup_{i=1}^{n} a_i(t)$. We usually refer to $S_n = on$ as an active machine.

The relative mining power of p_i at time t is $\frac{a_i(t)}{\cup_{j=1}^{n} a_j(t)}$, which also means the probability of p_i finding the new block is $\frac{a_i(t)}{\cup_{j=1}^{n} a_j(t)}$ at time t. We denote $Block_{Interval}$ as the expected block time interval, thus the normalized start time of each machine m_i is $(\bar{s_j}) = \frac{s_j}{Block_{Interval}}$. We assume that no player can modify the active time of machines, which means $\bar{s_j}$ is a constant.

Once a machine is turned on, the time T_n to find a new block follows the exponential distribution $Exp(\mu(\bar{s}))$ with a rate $\mu(\bar{s})$, which is shared among $\cup_{i=1}^{n} m_{\{i\}} = \{m_1, m_2, \ldots, m_k\}$, where $\bar{s}$ denotes the vector of increasing orders of k machines' start time. The value of the parameter μ is explicitly determined by the underlying blockchain protocol $\prod$. Consequently, the expected block time interval, denoted as $E[T]$, is also a constant value that is solely determined by the protocol $\prod$. Furthermore, μ serves as an indicator of the complexity of the cryptographic puzzle associated with the blockchain, and the allocation of time to active mining machines by miners has a direct impact on its value. If the rate of the block is found to be too fast (too slow), $\prod$ will adjust μ to decrease (increase) the rate of each individual machine. In equilibrium, μ is fixed.

Note that the strategy space of each player p_i only includes turning the machine on, without turning it off, as it is an irrational behavior. The reason is that $T_n \sim Exp(\mu(\bar{s}))$, which is progress free (also referred to as memoryless). Furthermore, if no new blocks are found during an epoch, there will be cumulative pending transactions over time. Hence, if at a certain point in time, it is profitable to drive miners to turn the machine on, then this justification holds from the time until the subsequent block is found.

Once a block is selected as the longest valid chain subsequently, p_i who find and publish it will receive corresponding rewards. The value of corresponding reward is determined by $\prod$. In Bitcoin, specifically, total rewards R contain basic rewards R_b, which consist of coin-base rewards R_c, transaction fee rewards R_t and whale transaction rewards R_w. R_c is fixed at 3.125BTCs, and R_t is determined by the fees paid by transactions included in the block. In Ethereum 1.x, R consist of R_b, uncle rewards R_u as well as nephew rewards R_n. R_c is fixed at 3Ethers, and R_t is determined by the fees paid by transactions included in the block. The value of R_u and R_n will be defined later.

We now provide a formal definition for whale transaction block in Defination 1.

Definition 1 (Whale transaction block). *A block is determined to contain whale transactions if and only if the cumulative rate of transaction fees λ_i of the block exceeds the average cumulative rate of transaction fee rewards $\bar{\lambda}$ across a randomly selected set of 6,000 test blocks [30] by at least 0.01, where $\bar{\lambda}$ is calculated as the slope of a linear function representing the transaction fee rewards over time.*

The occurrence of whale transactions in a block is indicative of an elevated cumulative rate of transaction fees and a corresponding increase in the value of R within the same time period.

For the purpose of analyzing R_t and R_w, we randomly select 6,000 instances of actual Ethereum block data, referred to as measurement blocks, for examination. The findings reveal a general concordance between transaction fee rewards and

temporal variables, adhering to a linear regression model. To quantify the linear relationship between these variables, we employ the Pearson correlation coefficient as a metric to assess the correlation between R_t and the time variable. Let there be two sets of data, $\mathbb{R} = (r_1, r_2, \ldots, r_n)$ and $\mathbb{T} = (t_1, t_2, \ldots, t_n)$, representing n observations of R_t and their corresponding times, respectively. The Pearson correlation coefficient, r, is obtained by using the following calculation:

$$r = \frac{Cov\left(\mathbb{R}, \mathbb{T}\right)}{s_R s_T}, \tag{1}$$

where $Cov\left(\mathbb{R}, \mathbb{T}\right)$ represents the covariance between $\mathbb{R}$ and $\mathbb{T}$, and $s_{\mathbb{N}}$ denotes the standard deviation of variable $\mathbb{N}$. The aforementioned calculation can also be expressed as:

$$r = \frac{\sum_{i=1}^{n}\left(r_i - \bar{\mathbb{R}}\right)\left(t_i - \bar{\mathbb{T}}\right)}{\sqrt{\sum_{i=1}^{n}\left(r_i - \bar{\mathbb{R}}\right)^2}\sqrt{\sum_{i=1}^{n}\left(t_i - \bar{\mathbb{T}}\right)^2}}, \tag{2}$$

where $\bar{\mathbb{X}}$ represents the mean of variable $\mathbb{X}$. Please note that when $|r|$ is close to 1, it indicates a strong linear correlation between variable $\mathbb{X}$ and $\mathbb{T}$.

The coefficient obtained, which is 0.913, indicates a strong positive linear correlation between transaction fee rewards and the time dimension. It is noteworthy that transaction fee rewards are contingent upon time due to the expanding number of pending transactions awaiting inclusion in the network. Consequently, these rewards accumulate over time until a new block is generated, at which point they reset to zero. In contrast, coin-base rewards remain constant over time.

To delineate the "abrupt change points" observed in the evolution of transaction fee rewards, we introduce an auxiliary metric termed whale transaction rewards. A comprehensive analysis is conducted on the 58 identified change points across 6,000 test blocks, as referenced in [30]. The results indicate that the temporal occurrence of whale transactions exhibits a memoryless characteristic, broadly conforming to an exponential distribution. Furthermore, the reward values associated with whale transactions follow an approximate log-normal distribution, with a mean that is approximately an order of magnitude smaller than the coin-base reward. Consequently, we have formulated the concept of whale transaction rewards to encapsulate these "abrupt change points". Specifically, we employ an exponential distribution function to model the temporal frequency of whale transaction occurrences and a log-normal distribution function to characterize the distribution of reward values corresponding to whale transactions.

To delineate the player's expenditures, we introduce the concepts of capital expenses ($capex$) and operating expenses ($opex$). Initially, a player must rent mining machines to engage in the mining activity, which incurs $capex$. Additionally, the operation of mining machines necessitates the consumption of electrical resources, thereby generating $opex$. The $capex$ is contingent upon the duration of the machine leasing, whereas $opex$ is dependent on the commencement time of machine operation. It is important to note that both $opex$ and $capex$ are applicable to both successful and unsuccessful players.

Due to the variations in mining protocols across different blockchain platforms ($\prod$), mining machines that are compatible with $\prod_A$ are not interoperable with $\prod_B$. If a miner, denoted as p_i, who previously acquired (or owns) a mining machine m_i compatible with $\prod_A$, decides to transfer their hash power to $\prod_B$, m_i will become incompatible. Furthermore, when compared to the expense of purchasing m_i, renting it incurs lower expenses and provides greater adaptability. Therefore, without compromising generality, we focus solely on the capital expenditure associated with renting mining machines, rather than purchasing (or owning) them, as referenced in [7].

TABLE II
$opex$ AND $capex$ SETTINGS.

Scenario	c_{op}	c_{cap}
$high_{op}$	0.02	0
$middle_{op}$	0.01	0.01
low_{op}	0	0.02

Once a new block is propagated within the network, all miners proceed to search for the subsequent block, and this iterative process continues indefinitely. The utility of a miner is defined as the difference between their rewards and expenses. Rational miners adopt general selfish mining strategies in an endeavor to maximize their utility, thereby triggering a mining game scenario.

IV. MINING GAME

We now introduce the real-world parameters, block finding time and system property, respectively.

A. Real-World Parameters

Parameters, encompassing rewards, expenses, and network parameters, are influenced by a multitude of factors stemming from diverse sources. The transaction fees are impacted by system users and market dynamics, as referenced in [31]–[35]. The coin-base rewards are contingent upon the minting rate, which is specified by the system protocol. The capital expenditure ($capex$) is influenced by advancements in mining machine efficiency [36], as well as real estate market conditions [37]. The operating expense ($opex$) is primarily driven by the electrical expenses associated with running mining machines [37]–[39], which encompass the actual electricity consumed during the problem-solving process and machine cooling. These parameters are not only challenging to estimate but also exhibit variability across different cryptocurrencies and over time for the same blockchain.

Reward parameters. The prior research [7] examines the temporal evolution of fees in PoW-based blockchains and concludes that a linear approximation of fees is a plausible assumption. Similarly, in our work, we adopt a linear approximation for fees and denote λ_t as the rate of fee accumulation and λ_0 as the fundamental coin-base reward. For the purposes of all conducted experiments, and without compromising generality, we select the following parameter values: the fee accumulation rate is set to $\lambda_t = 1$, the expected block interval is established as $Block_{Interval} = 1$, and the number of mining machines is designated as $k = 256$.

TABLE III
MACHINES START TIMES.

Scenario	p_1	p_2	p_3	p_4
No halt	0	0	0	0
Uniform halt	0.2	0.2	0.2	0.2
Arbitrary halt	0	0.3	0.6	0.9

Expense parameters. Recall that the system under analysis is in a quasi-static equilibrium state, implying no influx or efflux of miners. Consequently, the profits of players should marginally exceed the interest rates and associated risks. To avoid the introduction of extraneous parameters and without compromising generality, we assume that the sum of capital expenditure (c_{cap}) and operating expenditure (c_{op}) is a constant value, akin to the assumption employed in [7]. Notably, the ratios of $opex$ to $capex$ may vary considerably across different cryptocurrencies. Therefore, we employ three distinct ratio settings between $opex$ and $capex$, as detailed in Table II. Specifically, $high_{op}$ and low_{op} represent two extreme scenarios: one where only $opex$ is considered and the other where only $capex$ is considered. In the $middle_{op}$ scenario, both $opex$ and $capex$ are taken into account with an equal ratio. These three expenditure scenarios are applicable to nearly all blockchain systems. It is important to note that different values satisfying the same ratios yield qualitatively consistent results.

Network parameters. We define γ as the intensity of the rushing capability. In the context of fork competition, the expected proportion of honest miners working on the private branch is denoted by γ. For example, if honest miners adhere to a uniform tie-breaking rule, then $\gamma = 1/2$. However, rational miners may initiate network attacks, such as Sybil attacks [40] or eclipse attacks [41], to gain control over block propagation, resulting in a value of γ that approaches 1. Therefore, the parameter γ reflects the communication prowess of rational miners. In the context of this paper, and without compromising generality, we primarily consider $\gamma = 1/2$ in most scenarios. It is worth noting that our model analysis and experimental findings are applicable to all values of γ within the range $0 \leq \gamma \leq 1$.

B. Block Finding Time

To find each player's utility, we first analyze the probability distribution of block finding time, which is a function of each player's start time. We model the block finding time as a random variable B with cumulative distribution function (CDF) and probability density function (pdf) denoted $F_B(t;\bar{s};\mu(\bar{s}))$ and $f_B(t;\bar{s};\mu(\bar{s}))$, respectively. The first step towards analyzing the system is to derive an expression for the distribution based on the player's strategy, namely $f_B(t;\bar{s};\mu(\bar{s}))$ and $F_B(t;\bar{s};\mu(\bar{s}))$. We now formally present Lemma 1 and Lemma 2 to delineate the CDF and pdf of B, respectively.

Lemma 1 (B's cumulative distribution function). *The block finding time as a random variable B with cumulative distribution function (CDF) denoted $F_B(t;\bar{s};\mu(\bar{s}))$ is*

$$F_B(t;\bar{s};\mu(\bar{s})) = 1 - \exp(-\mu(\bar{s}) \cdot \sum_{j \in a(t)} (t - s_j)) \quad . \tag{3}$$

Proof of Lemma 1. We first derive the distribution of a single mining machine, and then extend it to any single machine m_j with a start time s_j. We denote a random variable B_j as the block finding time of machine m_j, and denote $\mu(\bar{s})$ as the rate of a single machine, which is set by the protocol. The value of B_j is drawn from a shifted exponential distribution with a shift s_j as well as a rate $\mu(\bar{s})$. Hence, the pdf of B_j is

$$f_{B_j}(t;s_j;\mu(\bar{s})) = \begin{cases} 0, & t \leq s_j \\ \mu(\bar{s}) \cdot \exp(-\mu(\bar{s})(t - s_j)), & t > s_j \end{cases} \tag{4}$$

and the CDF of B_j is

$$F_{B_j}(t;s_j;\mu(\bar{s})) = \begin{cases} 0, & t \leq s_j \\ 1 - \exp(-\mu(\bar{s})(t - s_j)), & t > s_j \end{cases} . \tag{5}$$

As $F_{B_j}(t;s_j;\mu(\bar{s})) = Pr(B_j \leq t) = Pr(t \geq B_j) = 1 - Pr(t \leq B_j)$, we derive

$$Pr(t \leq B_j) = \begin{cases} 1, & t \leq s_j \\ \exp(-\mu(\bar{s})(t - s_j)), & t > s_j \end{cases} . \tag{6}$$

All active mining machines are competing for the next block, and the first machine to find the next block is the machine with minimal value of B_j. Hence, the time required for the next block is $B = \min_{j \in \{1,2,\ldots,k\}} B_j$. The probability that none of the machines have found a block by time t, $Pr(t \leq B)$, is the product of $Pr(B_j \geq t)$ for all j (as the mining machines are independent of each other). Hence, we have

$$\begin{aligned} Pr(t \leq B) &= \bigcap_{j \in \{1,2,\ldots,k\}} Pr(t \leq B_j) = \prod_{j=1}^{k} Pr(t \leq B_j) \\ &= \exp(-\mu(\bar{s}) \cdot \sum_{j \in a(t)} (t - s_j)) \end{aligned} . \tag{7}$$

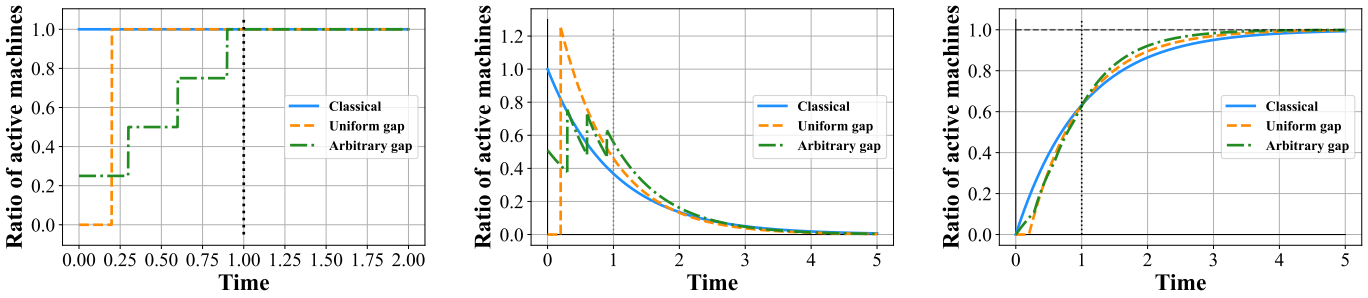
According to $Pr(t \leq B)$, we calculate the CDF of B that is

$$\begin{aligned} F_B(t;\bar{s};\mu(\bar{s})) &= Pr(B \leq t) = Pr(t \geq B) = 1 - Pr(t \leq B) \\ &= 1 - \exp(-\mu(\bar{s}) \cdot \sum_{j \in a(t)} (t - s_j)) \end{aligned} . \tag{8}$$

□

Lemma 2 (B's probability density function). *The block finding time as a random variable B with probability density function (pdf) denoted $f_B(t;\bar{s};\mu(\bar{s}))$ is*

$$f_B(t;\bar{s};\mu(\bar{s})) = \mu(\bar{s}) \cdot |a(t)| \exp(-\mu(\bar{s}) \cdot \sum_{j \in a(t)} (t - s_j)) \quad . \tag{9}$$



(a) Ratio of turned on machines (b) Probability density function (c) Cumulative distribution function

Fig. 2. System properties for different scenarios.

Proof of Lemma 2. By calculating the derivative of the CDF of B with respect to time t, the pdf of B can be obtained as

$$f_B(t;\bar{s};\mu(\bar{s})) = \frac{\partial F_B(t;\bar{s};\mu(\bar{s}))}{\partial t} = \mu(\bar{s}) \cdot |a(t)| \exp(-\mu(\bar{s}) \cdot \sum_{j \in a(t)} (t - s_j)). \tag{10}$$

□

As expected, when $|a(t)| = 0$ and then $\sum_{j\in a(t)}(t-s_j) = 0$, we have $F_B(t;\bar{s};\mu(\bar{s})) = 0$ as well as $f_B(t;\bar{s};\mu(\bar{s})) = 0$. We can verify whether $f_B(t;\bar{s};\mu(\bar{s}))$ is a valid pdf by checking that $\int_0^{+\infty} f_B(t;\bar{s};\mu(\bar{s}))dt = 1$ holds.

C. System Property

We further propose three different scenarios for selecting the start time and then analyze the block finding time probability distribution of the system. The following table lists the values used in each scenario, and we derive the ratio active on machines, the CDF, as well as pdf of B (denoted a random variable of the block generation time) with respect to time. Meanwhile, recall that the expected block time interval is $Block_{Interval}$, thus the expected total fees of finding a new block are $Block_{Interval} \cdot \lambda_t$. We denote $RTF = \frac{Block_{Interval} \cdot \lambda_t}{\lambda_0}$ as the expected ratio of expected cumulative transaction fee reward to expected coin-base rewards. Specifically, the larger RTF leads to accumulated fees becoming the main factor stimulating players to participate in the mining game. In this section, without loss of generality, we consider $Blcok_{Interval} = 1$. Each scenario has four equal-size players, each controlling 64 out of the total $k = 256$ machines in the system. The detailed description is shown in Table III.

In the no halt scenario, all players set the start time to zero. In Figure 2a, we see a constant ratio of one as all players' start time $a(t) = 0$. In Figure 2b and 2c, we notice that $F_B(t;\bar{s};\mu(\bar{s})) > 0$ and $f_B(t;\bar{s};\mu(\bar{s})) > 0$ for all t, which is expected, as all machines are active throughout the entire scenario.

In the uniform halt scenario, all players set the start time to 0.2. In Figure 2a, we see the ratio is zero while $t < 0.2$ and to one while $t \geq 0.2$, as all the start time is set to 0.2. In Figure 2b and 2c, we notice that $f_B(t;\bar{s};\mu(\bar{s})) = 0$ and $F_B(t;\bar{s};\mu(\bar{s})) = 0$ while $t < 0.2$, as no machines are active during this scenario. When $t \geq 0.2$, all machines are turn on, thus $F_B(t;\bar{s};\mu(\bar{s})) > 0$ and $f_B(t;\bar{s};\mu(\bar{s})) > 0$. In this case, the block finding time of the single machine follows the exponential distribution, and thus the block finding time of the entire system is also drawn from the shifted exponential distribution with the shift 0.2 and a rate parameter $\frac{5}{4} \cdot \frac{1}{Block_{Interval}}$. Note that the expected block time interval is still $Blcok_{Interval}$.

In the arbitrary scenario, each player sets the start time to a different value. Specifically, p_1 sets the start time to zero, p_2 sets to 0.3, p_3 sets to 0.6 and p_4 sets to 0.9. In Figure 2a, we notice that the ratio increases over time, and these sudden changes in ladder values occur when machines are turned on. Note that for all scenarios, we have $\lim_{t\to+\infty} f_B(t;\bar{s};\mu(\bar{s})) > 0$, $F_B(t=0;\bar{s};\mu(\bar{s})) = 0$ and $\lim_{t\to+\infty} F_B(t;\bar{s};\mu(\bar{s})) = 1$.

V. GENERIC SELFISH MINING

We now formally introduce the generic selfish mining and honest mining strategies, stationary distribution, reward analysis and utility analysis, respectively.

A. Attack Description

Overview. We are now poised to elucidate the mining strategies employed by honest and generic selfish participants within the blockchain system. Honest participants adhere strictly to the protocols stipulated by the blockchain system. They individually observe the prevalent state of the blockchain and, based on the longest valid chain rule, select a main chain for mining purposes. Upon discovering a new block, honest participants promptly disseminate and broadcast this information to all members of the network. In the context of Ethereum 1.x, honest miners aim to include as many references to uncle blocks as feasible within the target block, in order to maximize nephew rewards. This behavior underscores the profit-driven nature of participants, who remain honest except in the case of attackers.

In stark contrast, generic selfish miners employ a strategy of retaining newly discovered blocks and releasing them strategically to maximize their earnings. The core principle underpinning generic selfish mining is to augment the proportion of base rewards and procure as numerous uncle and nephew rewards as feasible. More specifically, generic selfish

miners accumulate their newly discovered blocks, thereby establishing a private branch. When honest miners discover and disseminate a new block, generic selfish miners subsequently unveil their retained blocks, which are of equivalent length, thereby giving rise to blockchain forks. Subsequently, generic selfish miners persist in mining on their private branch, while honest miners continue mining on an arbitrary branch, until the private branch is devoid of retained blocks.

Analogous to the general blockchain protocol, there exist two categories of participants: honest players who adhere to the protocol and generic selfish players who adopt the generic selfish mining strategy. We define $L_S(T)$ as the length of the private branch perceived by generic selfish players and $L_H(T)$ as the length of the public branch perceived by honest players. Consequently, the tuple $(L_S(T), L_H(T))$ represents the system state at time T. Let S and H denote the sets of generic selfish players and honest players, respectively. Furthermore, M_S and M_H represent the total machine sets controlled by generic selfish players and honest players, respectively. Therefore, we have $M_S = \bigcup_{i \in S} m_i$ and $M_H = \bigcup_{i \in H} m_i$, where $M_S \cap M_H = \emptyset$ and $M_S \cup M_H = \Omega$. For simplicity and without compromising generality, we assume that M_S is controlled by a single generic selfish player.

Generic selfish mining. Using Ethereum 1.x as an illustrative example, we employ the generation of new blocks as the triggering mechanism for state transitions. When a generic selfish player discovers a new block, they designate all unreferenced blocks as uncle blocks, thereby incrementing the length of the private branch by one. The following three scenarios are considered: **(1)** In the initial phase of generic selfish mining, where the length of the private branch is one $((L_S(T), L_H(T)) = (1, 0))$, the generic selfish player retains the block and continues mining on their private branch. **(2)** When the length of the private branch exceeds the public branch by one and the length of the private branch is at least two $(L_S(T) - L_H(T) = 2$ and $L_S(T) \geq 2)$, the generic selfish player promptly releases all retained blocks to claim the corresponding coin-base rewards. **(3)** If the length of the private branch is at least two blocks longer than the public branch $(L_S(T) - L_H(T) \geq 2)$, the generic selfish player retains the block and continues mining on their private branch.

Honest mining. When an honest player discovers a new block, they designate all unreferenced blocks as uncle blocks, thereby increasing the length of the public branch by one. Note that there are no nephew blocks and uncle blocks in a Satoshi-like blockchain system. We can extend this model to any Satoshi-like blockchain system by simply setting the corresponding nephew and uncle rewards to zero. The subsequent scenarios are outlined as follows: **(1)** If the length of the public branch surpasses the private branch $(L_S(T) - L_H(T) < 0)$, the generic selfish player adopts the public branch and proceeds with mining on it. **(2)** In the event that the lengths of the private and public branches are equivalent $(L_S(T) - L_H(T) = 0)$, the generic selfish player promptly releases all retained blocks and attempts to persuade the majority to mine on the private branch. **(3)** When the length of the private branch exceeds the public branch by one $(L_S(T) - L_H(T) = 1)$, the generic selfish player immediately disseminates all retained blocks to claim corresponding coin-base rewards, as the risk associated with the fork competition becomes too unpredictable for the generic selfish player. **(4)** If the length of the private branch is at least two blocks longer than the public branch $(L_S(T) - L_H(T) \geq 2)$, the generic selfish player releases only one retained block, given their substantial advantage. Notably, if a new block is generated on the prefix of the private branch, the system state transitions to $(L_S(T), L_H(T)) = (L_S(T) - L_H(T) + 1, 1)$, indicating the emergence of a new fork.

A comprehensive analysis of the state transition is presented in Section V. It is worth noting that in this research, the generic selfish player within our model possesses a more extensive strategy space compared to the traditional selfish mining paradigm. This expansion includes decisions regarding when to activate the mining machinery, as both $capex$ and $opex$ cannot be overlooked.

B. Stationary Distribution

Recall that the tuple $(L_S(T), L_H(T))$ captures the system state at time T. The state space comprises the states $(0, 0), (1, 0), (1, 1)$ and (i, j), where $i - j \geq 2, j \geq 0$. Under our proposed generic selfish mining strategy and network model, it is straightforward to confirm that the process $(L_S(T), L_H(T))$ ultimately evolves into a Markov process. Furthermore, we can rigorously verify that the process $(L_S(T), L_h(T))$ is positive recurrent, implying that it possesses a unique stationary distribution.

To calculate the stationary distribution of the process $(L_S(T), L_H(T))$, we need to derive the transition rate of the state evolution. In previous works, the transition rate of the state evolution is calculable, while in our model, we cannot directly provide an explicit expression for the transition rate of state evolution, as it is related to the mining strategy of each player and the block finding time of the blockchain system. Here, we denote $\pi_{(i,j)}$ as the stationary distribution of the Markov process $(L_S(T), L_H(T))$. Let $f_H^{\pi_{(i,j)}}$ denote the rate that the next new block is generated by the honest players H, and $f_S^{\pi_{(i,j)}}$ denote the rate that the next new block is generated by the generic selfish players S, thus we have $f_H^{\pi_{(i,j)}} + f_S^{\pi_{(i,j)}} = 1$. We also denote $p_{(a,b)\Rightarrow(c,d)}$ as the rate that the state (a, b) transitions to the state (c, d). The generic selfish mining's transition rate of the state evolution is provided below.

(1) $p_{(0,0)\Rightarrow(0,0)} = f_H^{\pi_{(0,0)}}$.

This transition occurs when the honest player finds the new block and then broadcasts it to other players in the network. The generic selfish player receives it and continues to mine on the public chain. The rate is $f_H^{\pi_{(0,0)}}$.

(2) $p_{(0,0)\Rightarrow(1,0)} = f_S^{\pi_{(0,0)}}$.

This transition occurs when the generic selfish player finds the new block and then withholds it. In this case, the private branch forms with length set to one. The rate is $f_S^{\pi_{(0,0)}}$.

(3) $p_{(1,0)\Rightarrow(2,0)} = f_S^{\pi_{(1,0)}}$.

This transition occurs when the generic selfish player finds the new block and then withholds it on the private branch. In

this case, the length of the private branch increases by one. The rate is $f_S^{\pi_{(1,0)}}$.

(4) $p_{(1,0)\Rightarrow(1,1)} = f_H^{\pi_{(1,0)}}$.

This transition occurs when the honest player finds the new block. Then the generic selfish player immediately releases the withholding block, as the length of public branch is equal to the private branch, bringing about the fork competition. The rate is $f_H^{\pi_{(1,0)}}$.

(5) $p_{(1,1)\Rightarrow(0,0)} = f_S^{\pi_{(1,1)}} + f_H^{\pi_{(1,1)}}(1-\gamma) + f_H^{\pi_{(1,1)}}\gamma$.

This transition occurs when the following three cases occur: 1) the generic selfish player finds the new block and then releases the private branch (as $L_S(T), L_H(T) = (2,1)$). The private branch wins in the fork competition with length set to zero. The rate is $f_S^{\pi_{(1,1)}}$; 2) the honest player finds the new block and publishes it on the public branch. The public branch wins in the fork competition with length set to zero. The rate is $f_H^{\pi_{(1,1)}}(1-\gamma)$; 3) the honest player finds the new block and publishes it on the private branch. The private branch wins in the fork competition with length set to zero. The rate is $f_H^{\pi_{(1,1)}}\gamma$.

(6) $p_{(i,j)\Rightarrow(i+1,j)} = f_S^{\pi_{(i,j)}}$ for $i \geq 2$ and $j \geq 0$.

This transition occurs when the generic selfish player finds the new block and then withholds it on the private branch. In this case, the length of the private branch increases by one. The rate is $f_S^{\pi_{(i,j)}}$.

(7) $p_{(i,j)\Rightarrow(i-j,1)} = f_H^{\pi_{(i,j)}}\gamma$ for $i - j \geq 3$ and $j \geq 1$.

This transition occurs when the honest player finds the new block and publishes it on the prefix of the private branch. The generic selfish player immediately releases a withholding block to form a new fork. In this case, the length of the private branch decreases by j, and the length of the public branch is set to one. The rate is $f_H^{\pi_{(i,j)}}\gamma$.

(8) $p_{(i,j)\Rightarrow(0,0)} = f_H^{\pi_{(i,j)}}$ for $i - j = 2$ and $j \geq 1$.

This transition occurs when the honest player finds the new block. The generic selfish player immediately releases the private branch (as $L_S(T) = L_H(T) + 1$). In this case, the length of the private branch and the public branch is set to zero. The rate is $f_H^{\pi_{(i,j)}}$.

(9) $p_{(2,0)\Rightarrow(0,0)} = f_H^{\pi_{(2,0)}}$.

This transition occurs when the honest player finds the new block. The generic selfish player immediately releases the private branch $(L_S(T), L_H(T) = (2,1))$. In this case, the length of the private branch and the public branch is set to zero. The rate is $f_H^{\pi_{(2,0)}}$.

(10) $p_{(i,0)\Rightarrow(i,1)} = f_H^{\pi_{(i,0)}}$ for $i \geq 3$.

This transition occurs when the honest player finds the new block and then the generic selfish player immediately releases a private block. In this case, the public branch forms with length set to one. The rate is $f_H^{\pi_{(i,0)}}$.

(11) $p_{(i,j)\Rightarrow(i,j+1)} = f_H^{\pi_{(i,j)}}(1-\gamma)$ for $i - j \geq 3$ and $j \geq 1$.

This transition occurs when the honest player finds the new block and publishes it on the public branch. The generic selfish player immediately releases a private block and the length of public branch increases by one. The rate is $f_H^{\pi_{(i,j)}}(1-\gamma)$.

According to the above transition rate, the set $\{\pi_{(i,j)}\}$ satisfies the following global equilibrium equation:

$$\begin{cases} \pi_{(0,0)} \cdot f_S^{\pi_{(0,0)}} = \pi_{(1,1)} + \sum_{j=0}^{+\infty} \pi_{(2+j,j)} \cdot f_H^{\pi_{(2+j,j)}} \\ \pi_{(1,1)} = \pi_{(1,0)} \cdot f_H^{\pi_{(1,0)}} \\ \pi_{(3,1)} = \pi_{(3,0)} \cdot f_H^{\pi_{(3,0)}} + \sum_{j=1}^{+\infty} \pi_{(3+j,j)} \cdot f_H^{\pi_{(3+j,j)}} \cdot \gamma \\ i \geq 1 : \pi_{(i,0)} = \pi_{(i-1,0)} \cdot f_S^{\pi_{(i-1,0)}} \\ i \geq 4 : \pi_{(i,1)} = \pi_{(i,0)} \cdot f_H^{\pi_{(1,0)}} + \pi_{(i-1,1)} \cdot f_S^{\pi_{(i-1,1)}} \\ + \sum_{j=1}^{+\infty} \pi_{(i+j,j)} \cdot f_H^{\pi_{(i+j,j)}} \cdot \gamma \\ i \geq 4 : \pi_{(i,i-2)} = \pi_{(i,i-3)} \cdot f_H^{\pi_{(i,i-1)}} \cdot (1-\gamma) \\ i \geq 5, j \geq 2 : \pi_{(i,j)} = \pi_{(i-1,j)} \cdot f_S^{\pi_{(i-1,j)}} \\ + \pi_{(i,j-1)} \cdot f_H^{\pi_{(i,j-1)}} \cdot (1-\gamma) \end{cases} \tag{11}$$

Please note that we can only obtain numerical solutions for each $\pi(i,j)$ through Monte Carlo simulation, due to the potential variability in player strategies across different states. Unless otherwise specified, the subsequent experiments utilize Monte Carlo simulations to calculate the steady-state probabilities of each state $\pi(i,j)$.

C. Reward Analysis

In this section, we proceed to perform a reward analysis under the assumption that each player adheres to the honest protocol. For brevity, we designate c_{cap} as $capex$ and c_{op} as $opex$. Let $utility_i$ represent the expected utility of player p_i, which is computed as the difference between the player's total expected rewards (denoted as $reward_i$) and total expected expenses (denoted as $expense_i$).). Recall that pending fees escalate over time until a new block is mined. To quantify fee accumulation over time, we employ historical Bitcoin data and apply a plausible approximation within our generalized model. We formulate the total block reward as a linear function, where the slope corresponds to the expected fee accumulation rate (denoted as λ_t), and the intercept comprises the sum of the expected coin-base reward and the whale transaction reward (denoted as $\lambda_0 + R_w$), with R_w denoting the random variable representing the whale transaction reward.

We now formally present Property 1 and Property 2 to delineate the pdf for the occurrence time of the whale transaction and its reward value, respectively.

Property 1 (**Whale transaction occurrence time's pdf**). *The pdf for the occurrence time of whale transactions is given by:*

$$g(t,\lambda) = \lambda e^{-\lambda t} \tag{12}$$

where $\lambda > 0$ is the rate parameter of the distribution, t represents the time interval, and the expected value $E(T) = \frac{1}{\lambda}$.

In the version of Ethereum 1.x, we have $E(T) = \frac{1}{\lambda} \approx 0.1$, which implies that $\lambda \approx 10$.

We represent the random event of an occurrence of a whale transaction with the symbol set $A \in \{0,1\}$. Here, the element '0' signifies the absence of a whale transaction, whereas '1' indicates the occurrence of such a transaction.

Property 2 (**Whale transaction reward value's pdf**). *The pdf for the value of whale transaction rewards is given by:*

$$R_w(x,\mu,\sigma) \begin{cases} \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}, & A = 1 \\ 0, & A = 0 \end{cases} \tag{13}$$

where $x > 0$, μ, σ represent the expected value and standard deviation of the underlying normal distribution, respectively.

We now show in Theorem 1 the expected total profits of each player under honest game.

Theorem 1 (**Player's expected total profits under honest game**). *The expected total profits of p_i under honest game are (denoted as $profit_i$)*

$$\begin{aligned} profit_i = & \frac{a_i(t_1)}{\cup_{j=1}^{n} a_j(t_1)} \cdot (\lambda_0 + R_w + \lambda_t \cdot t_1) \\ & - c_{cap} \cdot t_1 \cdot |m_i| - c_{op} \cdot \sum_{s \in a_i(t_1)} (t_1 - s) \end{aligned}. \tag{14}$$

Proof of Theorem 1. We consider that all players follow the honest protocol. We assume the value of B is t_1 and $a_i(t_1)$ denotes the set of machines that the generic selfish player p_i has turned on by time t_1. Hence, the probability of p_i finding the new block is $\frac{a_i(t_1)}{\cup_{j=1}^{n} a_j(t_1)}$ at time t_1. Then we can obtain the expected total rewards of p_i are (denoted as $reward_i$)

$$reward_i = \frac{a_i(t_1)}{\cup_{j=1}^{n} a_j(t_1)} \cdot (\lambda_0 + R_w + \lambda_t \cdot t_1). \tag{15}$$

The expected total expenses of p_i are (denoted as $expense_i$)

$$expense_i = c_{cap} \cdot t_1 \cdot |m_i| + c_{op} \cdot \sum_{s \in a_i(t_1)} (t_1 - s). \tag{16}$$

Hence, the expected total profits of p_i are (denoted as $profit_i$)

$$\begin{aligned} profit_i &= reward_i - expense_i \\ &= \frac{a_i(t_1)}{\cup_{j=1}^{n} a_j(t_1)} \cdot (\lambda_0 + R_w + \lambda_t \cdot t_1) \\ &\quad - c_{cap} \cdot t_1 \cdot |m_i| - c_{op} \cdot \sum_{s \in a_i(t_1)} (t_1 - s) \end{aligned}. \tag{17}$$

□

Note that fees accumulate over time until the subsequent block is found. Once the subsequent block is found, fees become zero, as all the pending transactions will be included to that block. Hence, the cumulative fees can be regarded as a pseudo loop function.

D. Utility Analysis

In this section, we proceed to undertake a rigorous utility analysis for each state transition. Recall that each state transition is accompanied by a new block generated. Immediately upon the creation of this new block, the determination of its associated rewards is infeasible, as the ultimate "disposition" of the block hinges on the subsequent evolution of the system. Consequently, we will estimate the expected rewards of the newly generated block, employing a virtual function for this purpose. In contrast, the previous analysis tracks published blocks related to the state transition whose "disposition" has been determined, rather than the new block, thus it can derive the exact rewards. In the rest of this section, we use the time T as the block finding time (i.e., $B = T$). First, we calculate the basic rewards of selfish and honest players, respectively (denoted as $E(reward_S^b \mid B = T)$ and $E(reward_H^b \mid B = T)$). These calculations are detailed in Appendices A and B.

Recall that $a_S(t)$ denotes the set of machines that the generic selfish player p_S has turned on by time t, and $a_H(t)$ denotes the set of machines that the honest player p_H has turned on by time t. M_S denotes the total machine set controlled by the generic selfish players, and M_H denotes the total machine set controlled by the honest players. Next, we calculate the expected total expenses of the selfish and honest players, respectively (denoted as $E(expense_P \mid B = T)$, where $P \in \{P_H, P_S\}$). We obtain the following results:

$$\begin{aligned} E(expense_P \mid B = T) = & c_{cap} \cdot |M_P| \cdot T \\ & + c_{op} \cdot \sum_{s \in a_P(T)} (T - s) \end{aligned} \tag{18}$$

Finally, we calculate the expected total utility of the selfish and honest players, respectively (denoted as $E(utility_P \mid B = T)$, where $P \in \{P_H, P_S\}$). We obtain the following results:

$$\begin{aligned} E(utility_P \mid B = T) = & E(reward_P \mid B = T) \\ & - E(expense_P \mid B = T) \end{aligned} \tag{19}$$

We now show in Theorem 2 the optimal strategy of the generic selfish player and its existence.

Theorem 2 (**Adversary's optimal strategy $\hat{s}$**). *There exists an optimal strategy $\hat{s}$ for generic selfish players, which fulfills the following conditions*

$$\begin{aligned} & \arg\min_{\hat{s}} -E(utility_{P_S} \mid B = T), \\ & s.t. \;\; f_1(\hat{s}) = -\hat{s} \leq 0; \\ & \quad\;\; f_2(\hat{s}) = \hat{s} - 1 \leq 0. \end{aligned} \tag{20}$$

Proof of Theorem 2. The optimal strategy $\hat{s}$ of the adversary corresponds to the time s when the expected total utility $E(utility_P \mid B = T)$ of the adversary attains its maximum value. Thus, we obtain the following equation:

$$\begin{aligned} & \arg\max_{\hat{s}} E(utility_{P_S} \mid B = T), \\ & s.t. \;\; 0 \leq \hat{s} \leq 1. \end{aligned} \tag{21}$$

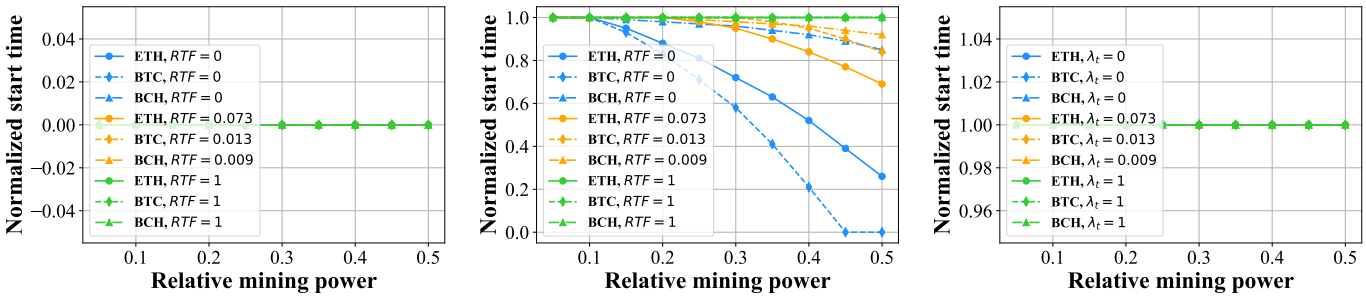
We further use Lagrange multipliers to solve it. Specifically, we rewrite Equation 21 as follows:

$$\begin{aligned} & \arg\min_{\hat{s}} -E(utility_{P_S} \mid B = T), \\ & s.t. \;\; f_1(\hat{s}) = -\hat{s} \leq 0; \\ & \quad\;\; f_2(\hat{s}) = \hat{s} - 1 \leq 0. \end{aligned} \tag{22}$$

As expected, the objective function $-E(utility_P \mid B = T)$ is a convex function when $\hat{s} \in [0, 1]$ (the Hessian matrix is positive definite), hence we can find the optimal $\hat{s}$ by solving the Karush-Kuhn-Tucker (KKT) conditions [42]. □

In the version of Ethereum 1.x, the function $R_u(*)$ is given below:

$$R_u(d) = \begin{cases} (8 - d)/8, & 1 \leq d \leq 6 \\ 0, & otherwise \end{cases} \tag{23}$$



(a) low_{op} (b) $middle_{op}$ (c) $high_{op}$

Fig. 3. Start times at equilibria of equal-size players with different strategies.

where d is the distance between the uncle block and the nephew block. Note that our analysis applies to an arbitrary function of $R_u(*)$.

In the version of Ethereum 1.x, the function $R_n(*)$ is given below:

$$R_n(n) = n \times \frac{1}{32} \times \lambda_0 \tag{24}$$

where n is the number of the uncle block referenced by the nephew block, and λ_0 is the basic rewards. Note that our analysis applies to an arbitrary function of $R_n(*)$.

VI. IMPLEMENTATION AND EVALUATION

In this section, we first introduce a modified PoW blockchain system simulator and Equilibrium serch tool to assist us in finding mining equilibria, and then use them to quantitatively evaluate the game in multiple scenarios.

A. Implementation

In order to determine the cumulative utility of each participant over an extended period and to ascertain the optimal strategy within the context of a Nash equilibrium framework, we devise and implement a modified PoW blockchain system simulator, which is integrated with an Equilibrium search tool.

Modified PoW blockchain system simulator. To substantiate our theoretical analysis, we have implemented a modified PoW blockchain system simulator. This simulator operates as a block-driven model in continuous time space and comprises a set of players who oversee machines. Each participant maintains a private copy of the blockchain locally and engages in competition with others for accounting rights. We employ exponential distribution random events to mimic the intervals between blocks. Each player selects their own machine activation time and adjusts the system's exponential distribution rate parameters to maintain a constant average block time interval. The active machine continues mining until a new block is discovered by any machine within the network.

Throughout this process, fees accumulate continually, and each player is responsible for incurring $capex$ and $opex$. Upon discovering a new block, an honest player promptly announces it. In contrast, a generic selfish player adopts the mining strategy outlined in Section V. It is crucial to note that the generation of a new block not only terminates the current mining epoch but also initiates the subsequent one.

Equilibrium search tool. When a player alters his strategy, the expected utility of the other players will undergo a corresponding impact. Our objective is to identify the equilibrium point, at which no player can further enhance his expected utility by modifying his strategy. Given that the utility of a miner is determined by both the system state and the strategies employed by other players, it is not feasible to express it explicitly through symbolic methodologies. Therefore, we represent it as an implicit function and employ numerical analysis tools to ascertain the equilibrium of the system.

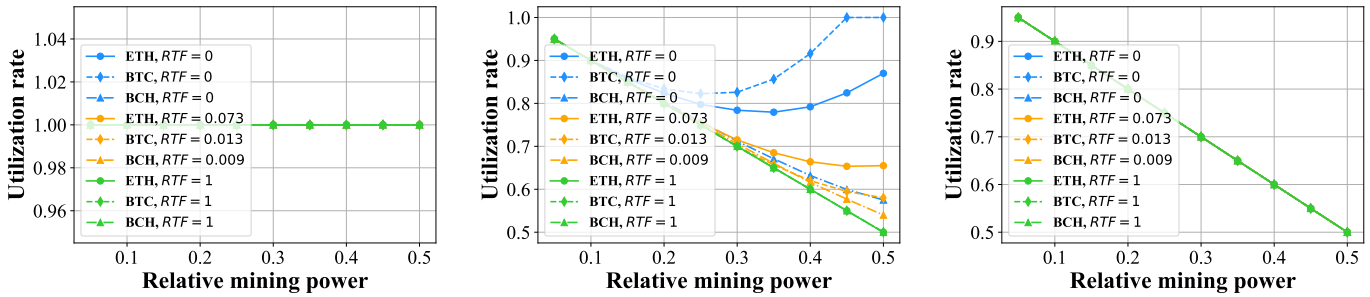
We implement an equilibrium search tool that works as follows. The tool receives reward and expense parameters, as well as a composite tuple list representing all players' strategies as an input. The form of each composite tuple list is $\left[p_i, m_i, \left[s_1, s_2, \ldots, s_{|m_i|}\right]\right]$, where p_i is the index of the players, $m_{\{i\}}$ is the machine set that p_i controls, and $[s_1, s_2, \ldots, s_j]$ is p_i's strategy (i.e., the normalized start time of $|m_i|$ machines that p_i controls). Note that we have $m_{\{i\}} \in \{m_1, m_2, \ldots, m_k\}$.

Iteratively, the equilibrium search tool randomly selects reward and expense parameters, as well as a composite tuple $\left[p_i, m_i, \left[s_1, s_2, \ldots, s_{|m_i|}\right]\right]$ as an input, and searches what output of the strategy will result in the maximum utility for p_i. This process continues to iterate until no player can increase his utility by changing his own strategy, which means the system reaches the equilibrium.

We emphasize that all equilibria found through such iterative process are only ϵ-Nash equilibria, as they are limited by the accuracy of numerical calculations. To address the dilemma, we repeat the search process with different random start time as well as optimization sequences. In all experiments conducted, the randomness introduced has no effect on the equilibrium of the system. It strengthens our theoretical analysis of the system equilibrium.

B. Evaluation

We now undertake an empirical validation of our theoretical analysis by utilizing the equilibrium search tool in conjunction with the execution of the modified PoW blockchain system simulator. We delve deeper into the parameter configurations



(a) low_{op} (b) $middle_{op}$ (c) $high_{op}$

Fig. 4. Utilization rate of mining power at equilibria.

that are pertinent to real-world scenarios, aiming to corroborate our theoretical findings and evaluate generic selfish mining within the framework of three distinct PoW blockchain platforms: Bitcoin (BTC), Ethereum 1.x (ETH), and Bitcoin Cash (BCH). To represent transaction fee rewards, we employ RTF values of 1, 0, and a decimal between 0 and 1 to signify significant, non-existent, and real-world scenarios, respectively. Specifically, the real-world RTFs are set at 0.073, 0.013, and 0.009 for ETH, BTC, and BCH, respectively. Furthermore, we calculate the maximum Pearson correlation coefficients, which are found to be 0.913, 0.958, and 0.942 for ETH, BTC, and BCH, respectively.

All our experiments were conducted on a server running Windows 11 and PyCharm 2021 based on Python version 3.9. The server is equipped with two Core i7 2.5GHz 14-core processors and 32GB of RAM. We set up 256 full nodes on the frameworks of three distinct PoW blockchain platforms: BTC, ETH, and BCH, and independently ran five rounds of experiments on each, with each round consisting of 10^8 blocks from five independent runs. The mean duration of each experimental round for the BTC blockchain framework amounted to 19 hours and 46 minutes, whereas for the ETH framework, it was 23 hours and 46 minutes, and for the BCH framework, it was 18 hours and 21 minutes. Unless stated otherwise, each data point depicted in our figures corresponds to the average of 10^8 blocks obtained from five independent experimental runs.

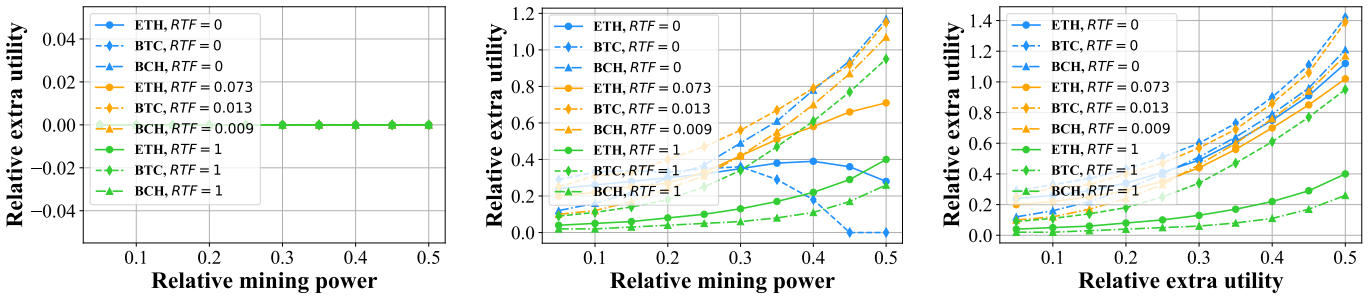
1) Normalized Start Time: We consider a simplified scenario in BTC, ETH, and BCH, where the system comprises two players: an honest player with a constant start time of zero and a generic selfish player with an arbitrarily determined start time. This assumption is plausible, as in real-world blockchain systems such as Bitcoin and Ethereum 1.x, honest players operate from a private perspective and are typically unaware of the presence of generic selfish players. Utilizing the equilibrium search tool, we obtain the normalized start time at equilibrium as the relative mining power of the generic selfish players increases. We then depict the relationship between these variables in Figure 3.

For low_{op} setting, as illustrated in Figure 3c, the optimal normalized start time for generic selfish players at equilibrium consistently equals zero, irrespective of the value of RTF. This outcome is anticipated, as opting for an earlier start time would not incur any additional expenses. By initiating their mining operations promptly, generic selfish players can maximize their probability of generating blocks, thereby achieving a sufficient increase in utility.

For $middle_{op}$ setting, as depicted in Figure 3b, generic selfish players weigh both rewards and expenses in their decision-making process. When RTF is smaller, generic selfish players tend to opt for an earlier start time to enhance their probability of generating blocks. Conversely, for larger RTF values, generic selfish players prefer a later start time at equilibrium, as the basic block rewards are insufficient to incentivize them to initiate mining operations immediately. In such scenarios, they opt to forego a portion of the basic rewards rather than incur high expenses. Therefore, a smaller RTF renders the rewards substantial, influencing generic selfish players to adopt an earlier start strategy, resulting in a more pronounced increase in rewards compared to expenses. For larger generic selfish players, the basic rewards constitute the primary factor in adjusting their mining strategies. Consequently, they prefer an earlier start time, despite the associated higher expenses. In contrast, smaller generic selfish players prioritize minimizing expenses to enhance their utility. Given that an earlier start time has a limited impact on their probability of generating blocks, it fails to offset the losses incurred due to increased expenses.

Another noteworthy finding is that selfish pools of medium size tend to select an optimal start time. The underlying observation is that, in the standard incentive model (which disregards expenses), the minimum threshold for generic selfish players necessitates possessing a normalized computing power of 0.25. As computing power escalates, generic selfish players gain a disproportionately larger advantage. However, in our model, the incorporation of expenses deters generic selfish players from recklessly opting for a start time of zero, as they do in the standard model. A hasty decision to initiate mining immediately may not yield as beneficial outcomes as selecting an optimal start time. Consequently, the presence of expenses promotes rational behavior among generic selfish players.

For $high_{op}$ setting, as illustrated in Figure 3a, all generic selfish players opt for the latest possible start time due to the inadequacy of the basic reward in compensating for the high $opex$. In such a scenario, participants effectively withdraw from the mining process, leading to a significant re-



(a) low_{op} (b) $middle_{op}$ (c) $high_{op}$

Fig. 5. Relative extra utility at equilibria.

duction in computational resources. Consequently, this decrement in computational power undermines the security of the blockchain system.

2) Utilization Rate of Mining Power: Analogous to the configurations described in Section VI, the system comprises two participants: an honest player with a fixed start time of zero and a generic selfish player with an arbitrarily chosen start time. To determine the normalized start time at equilibrium as the relative mining power of the generic selfish player increases, we utilize the equilibrium search tool. Subsequently, we illustrate the resultant relationship in Figure 4.

For low_{op} setting, as depicted in Figure 4c, the utilization rate of mining power at equilibrium consistently equals 1, irrespective of the value of RTF. This scenario is ideal, as the security of the system is intricately tied to this utilization rate. Specifically, as the available computational resources diminish, the system's capacity to withstand attacks diminishes accordingly. Furthermore, as the utilization rate of mining power decreases, the threshold for malicious players to launch 51% attacks also lowers, thereby facilitating such attacks.

For $high_{op}$ and $middle_{op}$ settings, as illustrated in Figures 4a and 4b, generic selfish players strategically adjust their optimal start times, leading to a decline in the utilization rate of mining power. Another intriguing finding is that generic selfish players of medium size contribute to the lowest utilization rate of mining power when RTF is approximately 0 in ETH and BTC. Additionally, as RTF decreases, the minimum utilization of mining power shifts towards scenarios involving large-sized generic selfish players in $middle_{op}$ setting. This is anticipated, as the utilization rate of mining power is determined by the product of the mining power of generic selfish players and the start times of their machines. Neither large-sized generic selfish players opting for earlier start times nor small-sized players choosing later start times significantly reduce the system's overall mining power utilization. In the most extreme scenario under $middle_{op}$ setting with $RTF = 1$, the utilization of mining power drops to approximately 0.5.

3) Relative Extra Utility: We consider the system consisting of two players, an honest player with a constant start time of zero and a generic selfish player with a start time arbitrarily. In this experiment, we investigate the relative extra utility (REU) of the optimal start time for generic selfish players at equilibrium compared to the start time of zero. Before the specific analysis, we first define $REU = \left|\frac{utility_t - utility_0}{utility_0}\right|$, where $utility_0$ is the utility of generic selfish players when the start time is zero, and $utility_t$ is the optimal utility of generic selfish players at equilibrium.

We first calculate the utility of generic selfish players when the start time is zero, and then use the equilibrium search tool to obtain the utility at equilibrium. We obtain the REU for generic selfish players with different relative mining power, as shown in Figure 5.

Observing in Section VI under $high_{op}$ setting, we notice that generic selfish players with large-size can obtain the greater REU at equilibria. Also, generic selfish players prefer the system under lower RTF, as they can gain disproportionate REU by dynamically adjusting to the optimal mining strategy. For low_{op} setting, as shown in Figure 5c, the optimal start time for generic selfish players is zero, and thus their REU is also zero. And for $middle_{op}$ setting, as shown in Figure 5b, in most cases, generic selfish players are more profitable by choosing an appropriate start time, compared with the start time of zero. Another interesting result is that large-sized generic selfish players cannot obtain disproportionate REU when $RTF = 0$, as their optimum start time gradually approaches zero in ETH and BTC. We emphasize that there are multiple factors contributing to an increase in REU, including low RTF and $high_{op}$. For mining pools under $high_{op}$ or $middle_{op}$ settings, they are supposed to strategically adjust mining strategies to reduce unnecessary losses and further improve utilities. We note that in the most extreme case under $middle_{op}$ and $RTF = 0$ settings, the REU of generic selfish players with the normalized relative mining power of 0.4 rises by a factor of 0.4, compared with the start time of zero in ETH.

VII. DISCUSSION

To address the challenges posed by the halt game in PoW blockchain systems, several specific measures can be implemented. These include adjusting the incentive mechanism to align with desired behavioral outcomes, augmenting machine investment to enhance the system's capacity and resilience, and integrating alternative consensus mechanisms that may offer improved security and efficiency.

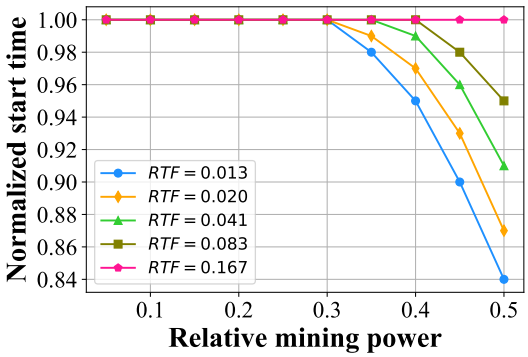


Fig. 6. Normalized start time of generic selfish players at equilibria under different RTFs.

A. Adjust Incentive Mechanism

The results of our incentive model demonstrate that the composition of basic rewards, $capex$, and $opex$ plays a pivotal role in ensuring the security of PoW blockchain systems. Achieving an appropriate balance among these components is crucial and can be facilitated through the implementation of a dynamic allowance scheme or a redesigned fee model. Furthermore, we evaluate the normalized start time of each player at equilibrium under different RTFs within the Bitcoin network, as illustrated in Figure 6. Our empirical findings indicate that an RTF of 0.167 serves as the minimum threshold for preventing mining halts in our incentive model. Based on the principle of periodic halving of rewards in the Bitcoin protocol, we anticipate that Bitcoin may fall below this threshold within the next two to three decades. To mitigate this issue, we propose that novel cryptocurrencies should establish a high ratio of basic coin-base rewards to expected transaction fees. During the design phase of these novel cryptocurrencies, it is imperative to consider this threshold to preemptively guard against the occurrence of halt games.

B. Increase Machine Investment

Our prior analysis has revealed that excessively high $capex$ and $opex$ of mining machines have rendered mining economically unviable, thereby diminishing the attractiveness of PoW blockchain systems. To address these challenges, machine researchers and developers have a strong incentive to develop more cost-effective hardware, specifically machines that exhibit lower $capex$ and $opex$ per second. Rational users will only participate in blockchain mining when the mining rewards fully compensate for the expenses incurred from renting and operating the mining machines. This not only fosters the continued advancement of PoW blockchain systems but also enhances their security. Therefore, we propose that the priority should be to invest more resources in mining equipment to reduce associated expenses.

C. Integrate Consensus Mechanism

The integration of PoW protocol with Proof of Delay (PoD) can significantly reduce the time required for users to mine, thereby lowering the associated expenses. In contrast to PoW, the computational process of PoD does not necessitate substantial computational resources. Consequently, miners have the flexibility to alternate between nearly costless PoD and costly PoW. The initial concept of PoD bears similarities to Verifiable Delay Functions (VDFs) [43]–[45], but it distinguishes itself by mandating a uniform delay period for all miners, a characteristic that VDFs typically cannot guarantee. This uniformity in delay is crucial for the security of the protocol. Although achieving a perfect time-proof architecture remains purely theoretical at present, with reliable hardware configurations, we can approximate the realization of an ideal PoD solution [46]. Recent research findings indicate that Sprints [47] has successfully decreased its carbon footprint by an order of magnitude while maintaining the same level of security assurance as PoW blockchains. Therefore, combining PoW with other non-resource-intensive consensus algorithms, such as PoD, presents an effective and promising approach to mitigating the emergence of the halt game.

VIII. Related Work

Prior research on blockchain incentives primarily focused on models where coin-base rewards are the primary subsidy. The original Bitcoin white paper [1] offers a basic expense analysis and incentive rationale. Kroll et al. [48] view Bitcoin as a consensus game with negligible expense impact. Eyal et al. [4] propose the "selfish mining" strategy, which enhances attackers' rewards by withholding blocks and breaking the longest chain rule, suggesting that honest protocols cannot be Nash equilibria. Their Markov chain analysis provides a revenue formula and a power threshold for miners, but does not preclude sub-threshold miners from profiting through deviations. Eyal's work suggests Bitcoin's security threshold is at most 0.25 under certain assumptions. Sapirshtein et al. [5] and Nayak et al. [6] refine selfish mining strategies, with Nayak et al. using simulations to show that the original strategy is suboptimal and varies by parameters. Sapirshtein et al. model Bitcoin as a nonlinear MDP, approximating the security threshold at 0.2321. Bar-Zur et al. [49] and Davidson et al. [50] explore selfish mining in the context of difficulty adjustment, finding other cryptocurrencies more vulnerable. Bar-Zur et al. [51] use Deep RL to analyze miner motivations and security thresholds, revealing a negative correlation with fee variability. Eyal et al. [23] and Kwon et al. [26] study pool infiltration and combined attacks, showing reduced earnings in mutual attacks. In their work, infiltration machines strategically alternate between infiltration and selfish mining attacks. All of these works consider a model where subsidies are the main incentive for mining, while the expenses can be ignored. In our work, we consider both subsidies (including coin-base rewards, transaction fee rewards and whale transaction rewards) and expenses, where miners' expenses vary based on their mining strategy.

Other studies have adapted selfish mining to Ethereum 1.x, assessing its profitability to determine Ethereum's security threshold. Ritz et al. [52] employ Monte Carlo simulations to assess the revenue from a generalized selfish mining strategy. Niu and Feng [53] similarly extend selfish mining to Ethereum

1.x and use Markov chain theory to compute its revenues. Grunspan et al. [54] analyze two selfish mining variants using combinatorial theory. All the above works assume that the subsidy comes from block rewards, uncle rewards, and nephew rewards, while ignoring the impact of expenses. In our work, we use a different model to consider capital expenses and operating expenses.

Babaiof et al. [55] examine incentives for transaction propagation in blockchain networks, proposing and analyzing various reward schemes. Our work focuses on a system with traditional rewards tied to block rewards and transaction fees, without specific incentives for transaction dissemination. Other studies explore broader network attack models [56], additional network and blockchain factors [21], and external attacker motivations [57]–[59] that involve bribing miners, which is outside the scope of our research.

Möser et al. [60] study the history of transaction fees in Bitcoin and find that miners typically follow protocol rules over profit optimization, a trend they predict will continue only when transaction fees are a minor incentive. Our work focuses on systems where fees are significant and can lead to unexpected participant behavior. Carlsten et al. [61] examine a model without block rewards, where miners are rewarded by incoming fees, and show how miners may be motivated to disrupt the main chain, impacting safety and vitality. They also suggest a conjecture about mining halts, where miners turn off their machines to save expenses, leading to suboptimal mining power utilization. Tsabary et al. [7] study blockchain security in real-world settings, considering both fees and rewards. They analyze the system as a "gap game" and suggest that miners should form alliances for profit maximization, choosing different gap sizes to optimize utility, even with similar operating expenses. Our work finds that the system encourages large miner coalitions, reducing decentralization. Biais et al. [36] discuss the investment in mining equipment for proof-of-work cryptocurrencies, noting the need for excessive equipment purchases to remain competitive. In our work, we consider fixed mining equipment as part of players' expenses.

Pass et al. [62] introduce the FruitChain protocol, which is ϵ-Nash incentive compatible and maintains the consistency and liveness of Nakamoto's protocol with an honest majority. It reduces mining reward variance, potentially eliminating the need for mining pools. Sleepy consensus [63], Hybrid consensus [64], and Solida [65] are novel consensus protocols that assume altruistic participants, ignoring incentives. Algorand [66] and Ouroboros Praos [67] are proof of stake protocols, with the latter addressing block withholding attacks through a new reward mechanism. Bitcoin-NG [68] is a scalable protocol using proof-of-work to select leaders for micro-blocks, distributing rewards sequentially and assuming minimal expenses and fees. Lavi et al. [69] propose new bidding schemes for Bitcoin's fee market, focusing on genuine miner bids. Our work centers on Bitcoin's current and similar incentive schemes, rather than these alternative protocols.

IX. CONCLUSION

Previous studies on static incentive models have laid a solid foundation in the field of PoW blockchain research. By taking into account additional elements such as mining expenditure, transaction fee and whale transaction rewards, our research proposes a novel generic selfish mining strategy and then devises a comprehensive dynamic incentive model for arbitrary PoW blockchains. Within the framework of this proposed incentive model, we reveal the halt game and conduct an in-depth exploration of the factors that lead to mining halts. This phenomenon not only facilitates the emergence of generic selfish mining but also gives rise to centralized incentives. The halt game thereby undermines the robustness of PoW blockchains, which holds significant implications for both current and future blockchain designs. Finally, we propose several specific countermeasures to alleviate this challenge.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[2] V. Buterin, "A next-generation smart contract and decentralized application platform," 2014.

[3] coinmarketcap.com, "Cryptocurrency market capitalizations," (2023), https://coinmarketcap.com/.

[4] I. Eyal and E. G. Sirer, "Majority is not enough: bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, p. 95–102, jun 2018. [Online]. Available: https://doi.org/10.1145/3212998

[5] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *Financial Cryptography and Data Security*, J. Grossklags and B. Preneel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 515–532.

[6] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 305–320.

[7] I. Tsabary and I. Eyal, "The gap game," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 713–728. [Online]. Available: https://doi.org/10.1145/3243734.3243737

[8] Digiconomist.net, "Bitcoin energy consumption index," (2023), https://digiconomist.net/bitcoin-energy-consumption.

[9] Digiconomist.net, "Ethereum energy consumption index," (2023), https://digiconomist.net/Ethereum-energy-consumption.

[10] "Markov model," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Markov_model

[11] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," pp. 139–147, 1993.

[12] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*, ser. CMS '99. NLD: Kluwer, B.V., 1999, p. 258–272.

[13] "Bitcoin 1.x mining process," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Bitcoin_protocol#Mining

[14] "Bitcoin difficulty adjustment," 2024. [Online]. Available: https://en.bitcoin.it/wiki/Difficulty

[15] "Ethereum 1.x difficulty adjustment," 2024. [Online]. Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.md

[16] Blockchain.info, "Transaction fees," (2023), https://www.blockchain.com/explorer/charts/transaction-fees.

[17] Etherscan.io, "Ether supply growth," (2023), https://etherscan.io/chart/ethersupply.

[18] Etherscan.io, "Ether transaction fees," (2023), https://etherscan.io/chart/transactionfee.

[19] "Fork," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Fork_(blockchain)

[20] "Pay-to-script-hash," 2024. [Online]. Available: https://en.bitcoin.it/wiki/Pay_to_script_hash

[21] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 3–16. [Online]. Available: https://doi.org/10.1145/2976749.2978341

[22] L. Bahack, "Theoretical bitcoin attacks with less than half of the computational power (draft)," *CoRR*, vol. abs/1312.7013, 2013. [Online]. Available: http://arxiv.org/abs/1312.7013

[23] I. Eyal, "The miner's dilemma," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 89–103.

[24] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, "On power splitting games in distributed computation: The case of bitcoin pooled mining," in *2015 IEEE 28th Computer Security Foundations Symposium*, 2015, pp. 397–411.

[25] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in bitcoin digital currency," *CoRR*, vol. abs/1402.1718, 2014. [Online]. Available: http://arxiv.org/abs/1402.1718

[26] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 195–209. [Online]. Available: https://doi.org/10.1145/3133956.3134019

[27] S. Gao, Z. Li, Z. Peng, and B. Xiao, "Power adjusting and bribery racing: Novel mining attacks in the bitcoin system," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 833–850. [Online]. Available: https://doi.org/10.1145/3319535.3354203

[28] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 281–310.

[29] G. Huberman, J. D. Leshno, and C. Moallemi, "Monopoly without a Monopolist: An Economic Analysis of the Bitcoin Payment System," *The Review of Economic Studies*, vol. 88, no. 6, pp. 3011–3040, 03 2021. [Online]. Available: https://doi.org/10.1093/restud/rdab014

[30] Blockchain.info, "Test blocks," https://github.com/VanceYan/SelfishMining/tree/master/ExperimentDatas, 2024. [Online]. Available: https://github.com/VanceYan/SelfishMining/tree/master/ExperimentDatas

[31] W. Binns, "How do I calculate my transaction fee," (2023), https://support.earn.com/digital-currency/bitcoin-transactions-and-fees/how-do-i-calculate-my-transaction-fee.

[32] Blockchain.info, "Bitcoin market capitalization," (2023), http://blockchain.info/charts/market-cap.

[33] Earn.com, "Predicting bitcoin fees for transactions," (2023), https://bitcoinfees.earn.com/.

[34] S. Khatwani, "Ethereum: Ether, ether gas, gas limit, gas price and fees," (2023), https://coinsutra.com/ethereum-gas-limit-gas-price-fees/.

[35] M. Möser and R. Böhme, "Trends, tips, tolls: A longitudinal study of bitcoin transaction fees," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 19–33.

[36] B. Biais, C. Bisière, M. Bouvard, and C. Casamatta, "The Blockchain Folk Theorem," *The Review of Financial Studies*, vol. 32, no. 5, pp. 1662–1715, 04 2019. [Online]. Available: https://doi.org/10.1093/rfs/hhy095

[37] Digiconomist.net, "A deep dive in a real-world bitcoin mine," (2023), https://digiconomist.net/deep-dive-real-world-bitcoin-mine.

[38] S. Malkin, "Cheapest places mining bitcoin," (2023), https://cryptocurrencynews.com/daily-news/cryptocurrency-mining/cheapest-places-mining-bitcoin/.

[39] R. Browne, "The cheapest and most expensive countries to mine bitcoin," (2023), https://www.cnbc.com/2018/02/15/the-cheapest-and-most-expensive-countries-to-mine-bitcoin.html.

[40] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[41] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's Peer-to-Peer network," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 129–144. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman

[42] "Karush-kuhn-tucker conditions," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Karush-Kuhn-Tucker_conditions#

[43] B. Wesolowski, "Efficient verifiable delay functions," in *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*. Springer, 2019, pp. 379–407.

[44] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass, "Continuous verifiable delay functions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 125–154.

[45] K. Pietrzak, "Simple verifiable delay functions," in *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019.

[46] C. Baum, B. M. David, E. Pagnin, and A. Takahashi, "Cascade:(time-based) cryptography from space communications delay," in *International Conference on Security and Cryptography for Networks*. Springer, 2024, pp. 252–274.

[47] M. Mirkin, L. Zhou, I. Eyal, and F. Zhang, "Sprints: Intermittent blockchain {PoW} mining," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 6273–6289.

[48] J. A. Kroll, I. C. Davey, and E. W. Felten, "The economics of bitcoin mining or, bitcoin in the presence of adversaries," *proceedings of weis*, 2013.

[49] R. B. Zur, I. Eyal, and A. Tamar, "Efficient mdp analysis for selfish-mining in blockchains," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, ser. AFT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 113–131. [Online]. Available: https://doi.org/10.1145/3419614.3423264

[50] M. Davidson and T. Diamond, "On the profitability of selfish mining against multiple difficulty adjustment algorithms," Cryptology ePrint Archive, Paper 2020/094, 2020, https://eprint.iacr.org/2020/094. [Online]. Available: https://eprint.iacr.org/2020/094

[51] R. Bar-Zur, A. Abu-Hanna, I. Eyal, and A. Tamar, "Werlman: to tackle whale (transactions), go deep (rl)," in *Proceedings of the 15th ACM International Conference on Systems and Storage*, ser. SYSTOR '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 148. [Online]. Available: https://doi.org/10.1145/3534056.3535005

[52] F. Ritz and A. Zugenmaier, "The impact of uncle rewards on selfish mining in ethereum," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2018, pp. 50–57.

[53] C. Feng and J. Niu, "Selfish mining in ethereum," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1306–1316.

[54] C. Grunspan and R. Perez-Marco, "Selfish mining in ethereum," in *Mathematical Research for Blockchain Economy*, P. Pardalos, I. Kotsireas, Y. Guo, and W. Knottenbelt, Eds. Cham: Springer International Publishing, 2020, pp. 65–90.

[55] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On bitcoin and red balloons," *SIGecom Exch.*, vol. 10, no. 3, p. 5–9, dec 2011. [Online]. Available: https://doi.org/10.1145/2325702.2325704

[56] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 692–705. [Online]. Available: https://doi.org/10.1145/2810103.2813655

[57] M. Mirkin, Y. Ji, J. Pang, A. Klages-Mundt, I. Eyal, and A. Juels, "Bdos: Blockchain denial-of-service," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 601–619. [Online]. Available: https://doi.org/10.1145/3372297.3417247

[58] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *Financial Cryptography and Data Security*, A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, and M. Sala, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, pp. 3–18.

[59] A. Judmayer, N. Stifter, A. Zamyatin, I. Tsabary, I. Eyal, P. Gaži, S. Meiklejohn, and E. Weippl, "Pay to win: Cheap, cross-chain bribing attacks on pow cryptocurrencies," in *Financial Cryptography and Data Security. FC 2021 International Workshops*, M. Bernhard, A. Bracciali, L. Gudgeon, T. Haines, A. Klages-Mundt, S. Matsuo, D. Perez, M. Sala, and S. Werner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 533–549.

[60] M. Möser and R. Böhme, "Trends, tips, tolls: A longitudinal study of bitcoin transaction fees," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 19–33.

[61] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 154–167. [Online]. Available: https://doi.org/10.1145/2976749.2978408

[62] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, ser. PODC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 315–324. [Online]. Available: https://doi.org/10.1145/3087801.3087809

[63] R. Pass and E. Shi, "The sleepy model of consensus," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 380–409.

[64] R. Pass and E. Shi, "Hybrid Consensus: Efficient Consensus in the Permissionless Model," in *31st International Symposium on Distributed Computing (DISC 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), A. Richa, Ed., vol. 91. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, pp. 39:1–39:16. [Online]. Available: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2017.39

[65] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solida: A Blockchain Protocol Based on Reconfigurable Byzantine Consensus," in *21st International Conference on Principles of Distributed Systems (OPODIS 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), J. Aspnes, A. Bessani, P. Felber, and J. a. Leitão, Eds., vol. 95. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, pp. 25:1–25:19. [Online]. Available: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.OPODIS.2017.25

[66] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 51–68. [Online]. Available: https://doi.org/10.1145/3132747.3132757

[67] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology – CRYPTO 2017*, J. Katz and H. Shacham, Eds. Cham: Springer International Publishing, 2017, pp. 357–388.

[68] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: a scalable blockchain protocol," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'16. USA: USENIX Association, 2016, p. 45–59.

[69] R. Lavi, O. Sattath, and A. Zohar, "Redesigning bitcoin's fee market," *ACM Trans. Econ. Comput.*, vol. 10, no. 1, may 2022. [Online]. Available: https://doi.org/10.1145/3530799

APPENDIX A
CALCULATION OF GENERIC SELFISH PLAYERS' REWARDS

We now formally analyze the rewards of generic selfish players. The Markov process is shown in Figure 7. We refer to a new block associated with a transition as a *target* block.

A. The coin-base reward of generic selfish players

(1) Case 1: $(0,0) \xrightarrow{f_S^{\pi(0,0)}} (1,0)$.

For state $(0,0)$, when a generic selfish player finds the target block, he withholds it, and thus the length of the private branch becomes one. Therefore, the rate is $\pi_{(0,0)} f_S^{\pi(0,0)}$.

Subcase 1-1: $(0,0) \xrightarrow{f_S^{\pi(0,0)}} (1,0) \xrightarrow{f_S^{\pi(1,0)}} (2,0)$.

For state $(1,0)$, when a generic selfish player finds the next block, he withholds it, and thus the length of the private branch is increased by one. In this case, the length of private branch minus the length of public branch is equal to two and thus, the target block will be on the longest valid chain and become a regular block definitely. Then, the generic selfish player obtains a corresponding coin-base reward. Therefore, the rate is $\pi_{(0,0)} f_S^{\pi(0,0)} f_S^{\pi(1,0)}$.

Subcase 1-2: $(0,0) \xrightarrow{f_S^{\pi(0,0)}} (1,0) \xrightarrow{f_H^{\pi(1,0)}} (1,1)$.

For state $(1,0)$, when an honest player finds the next block, he publishes it, and then the generic selfish player releases a withholding block immediately, which brings about forks. In this case, the coin-base reward of generic selfish players is determined by subsequent states. The rate is $\pi_{(0,0)} f_S^{\pi(0,0)} f_H^{\pi(1,0)}$.

Subcase 1-2-1: $(0,0) \xrightarrow{f_S^{\pi(0,0)}} (1,0) \xrightarrow{f_H^{\pi(1,0)}} (1,1) \xrightarrow{f_S^{\pi(1,1)}} (0,0)$.

For state $(1,1)$, when a generic selfish player finds the next block, he publishes it on the private branch, and thus wins in the fork competition. In this case, the target block becomes a regular block, and then the generic selfish player obtains a corresponding coin-base reward. The rate is $\pi_{(0,0)} f_S^{\pi(0,0)} f_H^{\pi(1,0)} f_S^{\pi(1,1)}$.

Subcase 1-2-2: $(0,0) \xrightarrow{f_S^{\pi(0,0)}} (1,0) \xrightarrow{f_H^{\pi(1,0)}} (1,1) \xrightarrow{f_H^{\pi(1,1)}\gamma} (0,0)$.

For state $(1,1)$, when an honest player finds the next block and publishes it on the private branch, the private branch wins in the fork competition. In this case, the target block becomes a regular block, and then the generic selfish player obtains a corresponding coin-base reward. The rate is $\pi_{(0,0)} f_S^{\pi(0,0)} f_H^{\pi(1,0)} f_H^{\pi(1,1)} \gamma$.

(2) Case 2: $(1,0) \xrightarrow{f_S^{\pi(1,0)}} (2,0)$.

For state $(1,0)$, when a generic selfish player finds the target block, he withholds it, and thus the length of the private branch becomes two. In this case, the length of private branch minus the length of public branch is equal to two and thus, the target block will be on the longest valid chain and become a regular block definitely. Then, the generic selfish player obtains a corresponding coin-base reward. Therefore, the rate is $\pi_{(1,0)} f_S^{\pi(1,0)}$.

(3) Case 3: $(1,1) \xrightarrow{f_S^{\pi(1,1)}} (0,0)$.

For state $(1,1)$, when a generic selfish player finds the target block, he publishes it on the private branch, and thus wins in the fork competition. In this case, the target block becomes a regular block, and then the generic selfish player obtains a corresponding coin-base reward. The rate is $\pi_{(1,1)} f_S^{\pi(1,1)}$.

(4) Case 4: $(i+j,j) \xrightarrow{f_S^{\pi(i+j,j)}} (i+j+1,j)$ with $i \geq 2, j \geq 0$.

For states $(i+j,j)$ with $i \geq 2, j \geq 0$, when a generic selfish player finds the target block, he withholds it, and thus the length of the private branch is increased by one. In this case, the length of the private branch minus the length of public branch is equal to $i+1 > 2$ and thus, the target block will be on the longest valid chain and become a regular block definitely. Then, the generic selfish player obtains a corresponding coin-base reward. The rate is $\sum_{i=2}^{+\infty} \sum_{j=0}^{+\infty} \pi_{(i+j,j)} f_S^{\pi(i+j,j)}$.

B. The uncle reward of generic selfish players

(1) Case 1: $(0,0) \xrightarrow{f_S^{\pi(0,0)}} (1,0) \xrightarrow{f_H^{\pi(1,0)}} (1,1) \xrightarrow{f_H^{\pi(1,1)}(1-\gamma)} (0,0)$.

For state $(0,0)$, when a generic selfish player finds the target block, he withholds it. We note that only if the target block is not on the longest valid chain will it become an uncle block. Then when an honest player finds the next block and publishes it, the generic selfish player releases a withholding block immediately, which gives rise to the fork competition.

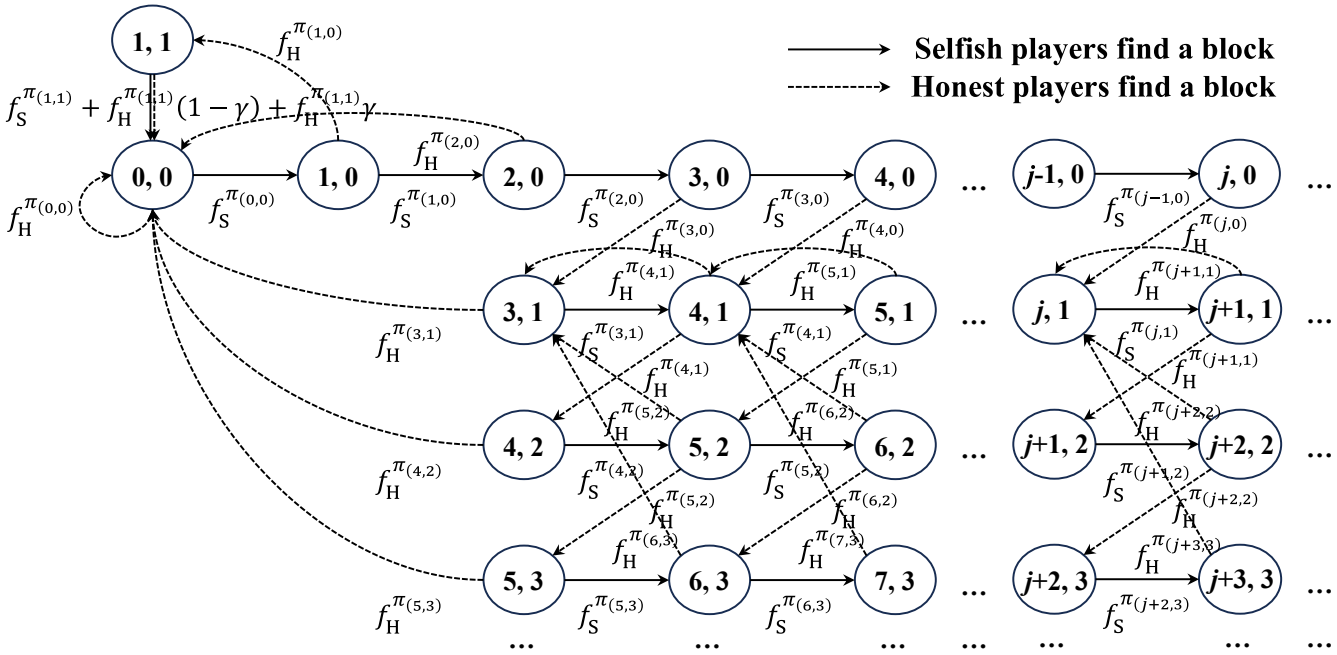


Fig. 7. The Markov process of generic selfish mining.

In this case, once an honest player finds the subsequent block, publishing it on the public branch and referencing the target block as an uncle block, he obtains a corresponding uncle reward. The rate is $\pi_{(0,0)} f_S^{\pi_{(0,0)}} f_H^{\pi_{(1,0)}} f_H^{\pi_{(1,1)}} (1-\gamma)$.

C. The nephew reward of generic selfish players

(1) Case 1: $(1,1) \xrightarrow{f_S^{\pi_{(1,1)}}} (0,0)$.

For state $(1,1)$, when a generic selfish player finds the target block, he publishes it on the private branch and references the honest block that is not on the longest valid chain as an uncle block. In this case, the private branch wins in the fork competition and then the generic selfish player obtains a corresponding nephew reward. The rate is $\pi_{(1,1)} f_S^{\pi_{(1,1)}}$.

(2) Case 2: $(3,1) \xrightarrow{f_H^{\pi_{(3,1)}} \gamma} (0,0) \xrightarrow{f_S^{\pi_{(0,0)}} - f_S^{\pi_{(0,0)}} f_H^{\pi_{(1,0)}} f_H^{\pi_{(1,1)}} (1-\gamma)} (0,0)$.

For state $(3,1)$, when a generic selfish player finds the target block, he withholds it and references honest blocks that may be published on the prefix of the private branch. Yet only if the target block is on the longest valid chain definitely, does the generic selfish player obtain corresponding nephew rewards.

(3) Case 3: $(i+j,j) \xrightarrow{\pi_{(i+j,j)} f_H^{\pi_{(i+j,j)}} \gamma (\prod_{n=3}^{i} f_H^{\pi_{(i+3-n,1)}} \gamma)} (0,0) \xrightarrow{f_S^{\pi_{(0,0)}} - f_S^{\pi_{(0,0)}} f_H^{\pi_{(1,0)}} f_H^{\pi_{(1,1)}} (1-\gamma)} (0,0)$ with $i \geq 3, j \geq 1$.

For state $(i+j,j)$ with $i \geq 3, j \geq 1$, when a generic selfish player finds the target block, he withholds it and references honest blocks that may be published on the prefix of the private branch. Yet only if the target block is on the longest valid chain definitely, does the generic selfish player obtain corresponding nephew rewards.

APPENDIX B
CALCULATION OF HONEST PLAYERS' REWARDS

We now formally analyze the rewards of honest players. We also refer to a new block associated with a transition as a *target* block.

A. The coin-base reward of honest players

(1) Case 1: $(0,0) \xrightarrow{f_H^{\pi_{(0,0)}}} (0,0)$.

For state $(0,0)$, when an honest player finds the target block, he publishes it directly. In this case, the target block becomes a regular block and then, the honest player obtains a corresponding coin-base reward. Therefore, the rate is $\pi_{(0,0)} f_H^{\pi_{(0,0)}}$.

(2) Case 2: $(1,0) \xrightarrow{f_H^{\pi_{(1,0)}}} (1,1) \xrightarrow{f_H^{\pi_{(1,1)}} (1-\gamma)} (0,0)$.

For state $(1,0)$, when an honest player finds the target block and publishes it directly, the generic selfish player releases a withholding block immediately, which brings about forks. Once an honest player finds the subsequent block, he publishes it on public branch. In this case, the target block will be on the longest valid chain and become a regular block definitely. Thus, the honest player obtains a corresponding coin-base reward. The rate is $\pi_{(1,0)} f_H^{\pi_{(1,0)}} f_H^{\pi_{(1,1)}} (1-\gamma)$.

(3) Case 3: $(1,1) \xrightarrow{f_H^{\pi_{(1,1)}}} (0,0)$.

For state $(1,1)$, when an honest player finds the target block, it will be on the longest valid chain and become a regular block definitely regardless of whether he publishes it on the private or public branch. Thus, the honest player obtains a corresponding coin-base reward. The rate is $\pi_{(1,1)} f_H^{\pi_{(1,1)}}$.

B. The uncle reward of honest players

(1) Case 1: $(1,0) \xrightarrow{f_H^{\pi_{(1,0)}}} (1,1)$.

For state $(1,0)$, when an honest player finds the target block and publishes it directly, the generic selfish player releases a withholding block immediately, which gives rise to forks. We notice that only if the target block is not on the longest valid chain will it become an uncle block. In this case, the uncle reward of honest players is determined by subsequent states.

Subcase 1-1: $(1,0) \overset{f_H^{\pi(1,0)}}{\rightarrow} (1,1) \overset{f_H^{\pi(1,1)}\gamma}{\rightarrow} (0,0)$.

For state $(1,1)$, when an honest player finds the subsequent block and publishes it on the private branch, it references the target block as an uncle block. In this case, the private branch wins in the fork competition and thus, the honest player obtains a corresponding uncle reward. The rate is $\pi_{(1,0)} f_H^{\pi(1,0)} f_H^{\pi(1,1)} \gamma$.

Subcase 1-2: $(1,0) \overset{f_H^{\pi(1,0)}}{\rightarrow} (1,1) \overset{f_S^{\pi(1,1)}}{\rightarrow} (0,0)$.

For state $(1,1)$, when a generic selfish player finds the subsequent block, he publishes it and references the target block as an uncle block. In this case, the private branch wins in the fork competition and thus, the honest player obtains a corresponding uncle reward. The rate is $\pi_{(1,0)} f_H^{\pi(1,0)} f_S^{\pi(1,1)}$.

(2) Case 2: $(i,0) \overset{f_H^{\pi(i,0)}}{\rightarrow} (i,1)$ with $i \geq 2$.

For state $(i,0)$ with $i \geq 2$, when an honest player finds the target block and publishes it directly, the generic selfish player releases a withholding block immediately, bringing about forks. In this case, the private branch will be on the longest valid chain such that the target block becomes an uncle block definitely regardless of whether the subsequent block is generated by honest or generic selfish players. Thus, the honest player obtains a corresponding uncle reward. The rate is $\sum_{i=2}^{+\infty} \pi_{(i,0)} f_H^{\pi(i,0)}$.

(3) Case 3: $(i+j,j) \overset{f_H^{\pi(i+j,j)}\gamma}{\rightarrow} (i,1)$ with i $\geq 2, j \geq 1$.

For state $(i+j,j)$ with $i \geq 2, j \geq 1$, when an honest player finds the target block and publishes it on the prefix of the private branch, the generic selfish player releases a selfish block immediately, giving rise to forks. In this case, the private branch will be on the longest valid chain such that the target block becomes an uncle block definitely regardless of whether the subsequent block is generated by honest or generic selfish players. Thus, the honest player obtains a corresponding uncle reward. The rate is $\sum_{i=2}^{+\infty} \sum_{j=1}^{+\infty} \pi_{(i+j,j)} f_H^{\pi(i+j,j)} \gamma$.

C. The nephew reward of honest players

(1) Case 1: $(1,1) \overset{f_H^{\pi(1,1)}}{\rightarrow} (0,0)$.

For state $(1,1)$, when an honest player finds the target block, regardless of whether he chooses to publish it on the private or public branch, the target block will reference the block on another branch as an uncle block to strive for a corresponding nephew reward. Thus, the rate is $\pi_{(1,1)} f_H^{\pi(1,1)}$.

(2) Case 2: $(3,1) \overset{f_H^{\pi(3,1)}\gamma}{\rightarrow} (0,0) \overset{f_H^{\pi(0,0)}+f_S^{\pi(0,0)} f_H^{\pi(1,0)} f_H^{\pi(1,1)}(1-\gamma)}{\rightarrow} (0,0)$.

For state $(3,1)$, when an honest player finds the target block, he publishes it and references honest blocks that may be published on the prefix of the private branch to strive for nephew rewards. Yet only when the target block is on the longest valid chain, does he obtain corresponding nephew reward. The rate is $\pi_{(3,1)} f_H^{\pi(3,1)} \gamma \left(f_H^{\pi(0,0)} + f_S^{\pi(0,0)} f_H^{\pi(1,0)} f_H^{\pi(1,1)} (1-\gamma) \right)$.

(3) Case 3: $(i+j,j) \overset{\pi_{(i+j,j)} f_H^{\pi(i+j,j)} \gamma (\prod_{n=3}^{i} f_H^{\pi(i+3-n,1)} \gamma)}{\rightarrow} (0,0) \overset{f_H^{\pi(0,0)}+f_S^{\pi(0,0)} f_H^{\pi(1,0)} f_H^{\pi(1,1)}(1-\gamma)}{\rightarrow} (0,0)$ with $i \geq 3, j \geq 1$.

For state $(i+j,j)$ with $i \geq 3, j \geq 1$, when an honest player finds the target block, he publishes it and references honest blocks that may be published on the prefix of the private branch to strive for nephew rewards. Yet only when the target block is on the longest valid chain, does he obtain corresponding nephew reward. The rate is $\sum_{i=3}^{+\infty} \sum_{j=1}^{+\infty} \frac{\pi_{(i+j,j)} f_H^{\pi(i+j,j)} \gamma \left(\prod_{n=3}^{i} f_H^{\pi(i+3-n,1)} \gamma \right)}{\left(f_H^{\pi(0,0)} + f_S^{\pi(0,0)} f_H^{\pi(1,0)} f_H^{\pi(1,1)} (1-\gamma) \right)}$.