Incentive Analysis of Bitcoin-NG, Revisited

Jianyu Niu*, Ziyu Wang†, Fangyu Gai*, and Chen Feng*
*School of Engineering, University of British Columbia (Okanagan Campus)
†School of Cyber Science and Technology, Beihang University
*{jianyu.niu, fangyu.gai, chen.feng}@ubc.ca, †wangziyu@buaa.edu.cn

***Abstract*—Bitcoin-NG is among the first blockchain protocols to approach the *near-optimal* throughput by decoupling blockchain operation into two planes: leader election and transaction serialization. Its decoupling idea has inspired a new generation of high-performance blockchain protocols. However, the existing incentive analysis of Bitcoin-NG has several limitations. First, the impact of network capacity is ignored. Second, an integrated incentive analysis that jointly considers both key blocks and microblocks is still missing.**

In this paper, we aim to address the two limitations. First, we propose a new incentive analysis that takes the network capacity into account, showing that Bitcoin-NG can achieve *better* incentive compatibility against the microblock mining attack under limited network capacity. Second, we leverage a Markov decision process (MDP) to jointly analyze the incentive of both key blocks and microblocks, showing that Bitcoin-NG is as secure as Bitcoin when the adversary controls less than 35% of the computation power. We hope that our in-depth incentive analysis for Bitcoin-NG can shed some light on the mechanism design and incentive analysis of next-generation blockchain protocols.

I. Introduction

Bitcoin—the largest and most influential cryptocurrency—has sparked many other cryptocurrencies, gaining much attention from both academia and industry [1]. The key innovation behind Bitcoin is the *Nakamoto Consensus* (NC), which is used to realize a distributed ledger known as the blockchain. Due to its unique features of decentralization, security, and privacy, blockchain technology has been envisioned as a fundamental trust infrastructure for supporting future decentralized Internet applications.

However, Bitcoin and other NC-based blockchains have suffered from the low transaction throughput (e.g., 7 TPS[1] in Bitcoin) and the poor network utilization (e.g., less than 2% in Bitcoin [2]). The low throughput of Bitcoin is mostly due to its choice of two system parameters: small block size (originally 1 MB) and long block interval (on average 10 minutes). Although increasing the block size or shortening the block interval can increase the throughput, the resulting severer forks would incredibly reduce the advantage of honest participants against the adversaries, thereby sacrificing its security level [3]–[5]. Indeed, it has been shown in various studies [5]–[7] that redesigning the underlying NC (rather than fine-tuning the system parameters) is essential to improve the throughput.

Bitcoin-NG (Next Generation) [7] is among the first and the most prominent NC-based blockchains to approach the *near-optimal* throughput. Bitcoin-NG creatively leverages two types of blocks: 1) a *key block* that is very similar to a conventional block in Bitcoin except that it doesn't carry any transactions, and 2) a *microblock* that carries a set of outstanding transactions. Any key block is generated through the leader election process (often known as the mining process) in NC, and the corresponding leader will receive a block reward (if the key block ends up in the longest chain). In addition, this leader can issue multiple microblocks and receive the transaction fees until the next key block is generated. Unlike Bitcoin, Bitcoin-NG decouples leader election and transaction serialization. Intuitively, it is this decoupling that enables Bitcoin-NG to approach the near-optimal throughput, since the microblocks can be produced at a rate up to the network capacity. Perhaps for this reason, Bitcoin-NG has been adopted by two major cryptocurrencies: Waves[2] and Aeternity[3].

More importantly, this decoupling idea has inspired a new generation of blockchain protocols including ByzCoin [8], Hybrid consensus [9], Solida [10], Prism [11], and many others. These protocols are able to achieve lower latency and/or higher throughput than Bitcoin-NG. Nevertheless, their reward mechanism design and formal incentive analysis are less well known in the literature. Indeed, even the existing incentive analysis of Bitcoin-NG has several limitations, as we will explain shortly. As a starting point to bridge this research gap, we aim to provide an in-depth incentive analysis for Bitcoin-NG, hoping that it would shed some light on the mechanism design and incentive analysis of aforementioned next-generation blockchain protocols.

The prior works of Bitcoin-NG found that Bitcoin-NG cannot maintain the incentive compatibility of microblocks when an adversary controlling larger than about 29% of the total computation power [7], [12]. In addition, an adversary with the same computation power can gain higher reward proportion in Bitcoin-NG than in Bitcoin by launching key-block and microblock attack [13]. Despite these important findings, previous incentive analysis of Bitcoin-NG has the following limitations. *First*, the impact of network capacity is ignored in the previous analysis [7], [12], [13]. Do the above findings still hold under network capacity constraints? *Second*, an integrated incentive analysis that jointly considers both key blocks and microblocks is still missing. Do the above findings still hold in such a scenario?

[1]TPS is short for transactions per second.

[2]Waves: https://docs.wavesplatform.com/

[3]Aeternity: https://aeternity.com/

To answer the first question, we develop a new probabilistic analysis that takes network capacity into account. In particular, we introduce two random variables: 1) the interval between two consecutive key blocks and 2) the generation rate of microblocks to capture the impact of network capacity with incentive analysis. Then, we apply the Chernoff-type bounding techniques to derive the long-term average revenue of the adversary. We find that by choosing suitable system parameters, Bitcoin-NG can remain incentive compatibility even when the adversary has more than 29% of the computation power.

To address the second question, we leverage an MDP model to jointly analyze the incentive of key blocks and microblocks. Although similar analysis has been conducted by Sapirshtein et al. [14] in the context of Bitcoin[4], the microblock structure introduces additional complexity for the MDP design (e.g., more mining strategies and rewards). To make the MDP tractable, we confine our analysis to a family of joint mining strategies. Our results show that the optimal selfish mining threshold of Bitcoin-NG is just a little lower than Bitcoin when the selfish computation power is greater than 35%.

The main contributions of this paper are summarized as follows:

- We propose a new incentive analysis of Bitcoin-NG considering the network capacity. The results show that Bitcoin-NG can achieve even better incentive compatibility against the microblock mining attack.
- We model the selfish mining of key blocks and microblocks jointly into an MDP. The results show that the threshold of Bitcoin-NG is a little lower than in Bitcoin only when the selfish mining power α is greater than 35%.
- We discuss the instability and possible security attacks caused by high-fee transactions. We also design a pre-proposing mechanism to resist the leader corruption attack. Our solutions can be easily adopted in other Bitcoin-NG variants.

The rest of this paper is organized as follows. Section II provides some necessary backgrounds for our work. Section III introduces the system model and the Bitcoin-NG microblock selfish mining strategies. The mathematical analysis of the Bitcoin-NG incentive is given in Section IV in the more formal model. Section V presents the MDP of selfish mining in Bitcoin-NG. Section VI discusses how to improve the security of Bitcoin-NG. Related work is discussed in Section VII. Finally, Section VIII concludes the paper.

II. Background

A. A Primer on Bitcoin

Bitcoin relies on Nakamoto Consensus (NC) to make a group of distributed and mutually distrusting participants reach agreement on a transparent and immutable ledger, also known as the blockchain. A blockchain is a list of blocks linked by hash values with each block containing some ordered transactions. To make all participants agree on the same blockchain, NC leverages two components: the Proof-of-Work (PoW) mechanism and the longest chain rule (LCR). Each participant in NC (also referred to as a miner) collects valid and unconfirmed transactions from the network, orders, and packs these transactions into a block. In addition, a valid block needs to contain a proof of work, i.e., its owner needs to find a value of the nonce (i.e., a changeable data filed) such that the hash value of the new block has required leading zeros [1]. The length of leading zeros is also known as the mining difficulty, which can be tuned by the system so that new blocks are mined every ten minutes on average.

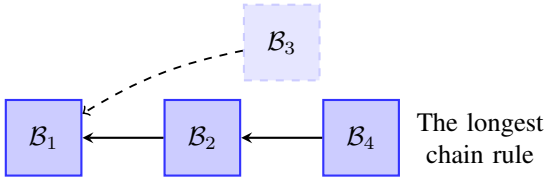

Fig. 1. An illustration of the chain structure in Bitcoin.

Once a new block is produced, it will be broadcast to the entire network. Ideally, the block should be accepted by all participants before the next block is produced. In reality, two new blocks could be mined around the same time, leading to a fork in which two "child" blocks share a common "parent" block. See Fig. 1 for an illustration. To resolve the blockchain forks, an honest miner always accepts the longest chain as the valid one. In Bitcoin, any block that is eventually included in the longest chain will offer its miner a block reward, and the associated transactions also have some fees for the miner as another type of reward. These incentives encourage rational miners to devote their computational resources to the system.

Selfish Mining. Although NC is designed to fairly reward miners according to their contributions to the system (i.e., miners' revenue is proportional to their devoted computation power), the studies in [14]–[17] show that a selfish miner can gain more revenue than its fair share by deviating from the protocol. This mining attack is called *selfish mining*. In this attack, a selfish miner can keep its newly discovered blocks secret, mine on top of its blocks, and leverage its lead to create forks on purpose. When some honest miners mine a new block, the selfish miners will publish one secret block to match the current block as a competition, or publish two blocks to override an honest block because honest miners follow LCR. By invalidating honest blocks and lowering the valid computation power, a selfish miner can obtain a higher block reward fraction than it deserves. Sapirshtein et al. [14] converted selfish mining strategies of Bitcoin into an MDP and concluded that the optimal computation power threshold making selfish mining profitable is 23.21%.

B. A Primer on Bitcoin-NG

In Bitcoin, the mining of blocks has two functionalities: 1) electing leaders (i.e., the owners of valid blocks) by NC, and

[4]Due to the similarity, the MDP can be directly used to model the key-block mining in Bitcoin-NG.

2) ordering and verifying transactions. By differentiating block functionalities, Bitcoin-NG decouples the leader election with the transaction serialization. Specifically, Bitcoin-NG uses key blocks mined by proof-of-work to elect a leader at some stable rate (i.e., one key block per 100 seconds). Each leader can produce several microblocks containing outstanding transactions at another rate (i.e., tens of seconds). Clearly, the decoupling method enables Bitcoin-NG to process many microblocks between two consecutive key blocks and significantly increases the Bitcoin-NG transaction throughput. Fig. 2 illustrates the two types of blocks. Note that microblocks do not contain proof of work.

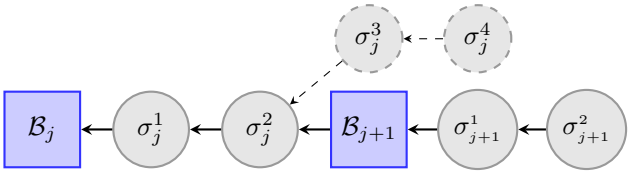

Fig. 2. An illustration of Bitcoin-NG. A square block (circle block) denotes the key block (microblock, respectively). The microblocks are issued by the two key-block miners $\mathcal{B}_j, \mathcal{B}_{j+1}$ with their signatures σ_j, σ_{j+1}, respectively.

Once a key block or microblock is produced, it will be broadcast to the network. Due to network delay, there may exist blockchain forks. In Bitcoin-NG, microblocks carry no weight, not even a secondary index for miners to choose which key block to mine. For instance, if an honest miner observes two chains, one with 3 key blocks and 6 microblocks and the other with 4 key blocks and 3 microblocks, the miner will still choose the chain with 4 key blocks. In other words, an honest miner still follows LCR to choose a "right" key block (i.e., the last key block in the longest chain counting by key blocks), and then mine on the latest microblock produced by the key-block miner. Thus, without microblocks, the Bitcoin-NG mining process of key blocks is the same as the one in Bitcoin.

III. System Model

A. Mining Model

Following the mining models [7], [15], [18], we consider a collection of n miners, denoted by the set $\mathcal{N}$. We assume a subset of miners $\mathcal{S} \subset \mathcal{N}$ are selfish and can deviate from the protocol to maximize their own profit. The other miners in $\mathcal{N} \setminus \mathcal{S}$ are honest and follow the protocol. The i-th miner has m_i fraction of the total hash power. In addition, the selfish miners (honest miners) control α (β, respectively) fraction of total mining power. That is, $\alpha = \sum_{i \in \mathcal{S}} m_i$ and $\beta = \sum_{i \in \mathcal{N} \setminus \mathcal{S}} m_i$. Clearly, $\alpha + \beta = 1$. We assume that all the selfish miners form a mining pool controlled by a single player, which is referred to as the selfish miner.

The mining process of the key block can be modeled as a Poisson process with rate f, as shown in [1], [11], [19] [5]. The key block mining process of the ith miner is also a Poisson process with the rate $m_i f$. Hence, the selfish miner generates a key block at the rate αf and the remained honest miners generate a key block at the rate βf. In addition, the miner of each key block becomes a leader and can issue a series of microblocks containing as many outstanding transactions as possible (up to the maximum microblock size) at a constant rate v until the next key block is mined. Specifically, block (including key block and microblock) mined by an honest miner (or the selfish miner) is referred to as *honest block* (or selfish block), respectively.

B. Network Model

Following the network model of Bitcoin [15], [18], we assume that honest miners are fully connected through the underlying network, and an honest miner spends negligible time to broadcast a key block or microblock in Bitcoin-NG[6]. In addition, we assume that the selfish miner can broadcast its private blocks immediately after it sees a new honest key block.

C. Mining Rewards

In Bitcoin-NG, there are two types of rewards, namely key-block reward and transaction fee. Every miner obtains a key-block reward if it mines a key block by successfully solving a PoW puzzle. In addition, each transaction has a fee (i.e., transaction fee) as a reward to incentive a key-block miner to verify and execute this transaction. This fee also encourages the miner to include the transaction in a microblock. Limited by the microblock size, the number of included transactions in one microblock is constant. So the maximum transaction fee for each microblock is also constant[7].

We use K_b and K_t to denote the key-block reward and the maximum transaction fee included in one microblock. Let $k = K_b/K_t$ denote the ratio of block reward with the transaction fee. This ratio k ranges from $(0, \infty)$. When k approaches 0 (∞), it implies that the transaction fee (key-block reward, respectively) dominates the reward. The different values of k exhibit the various impact of rewards on the Bitcoin-NG system.

D. Mining Strategies

In this section, we introduce the mining strategies for honest and selfish miners. In particular, we focus on mining strategies for microblocks, since strategies for key blocks have been extensively studied [14]–[16]. In Bitcoin-NG, an honest key-block miner includes outstanding transactions in microblocks and publishes these microblocks to win transaction fees. This is called the transaction inclusion rule. In addition, an honest miner should accept as many microblocks issued by the previous key-block miner as possible and mine on the latest

[5]The key block is mined by solving the PoW puzzle, and the value of f can be calculated by the average block interval, which is 10 minutes per block in Bitcoin and 100 seconds per block in Bitcoin-NG.

[6]This assumption is reasonable for key blocks because the inter-arrival time of two consecutive key blocks is often much larger than the block propagation delay. On the other hand, this assumption can be relaxed for microblocks, as we will show later.

[7]The microblock size is set to several KB for high efficiency, and in reality, the majority of the transactions carries small transaction fee.

received microblock, i.e., obeying the longest chain extension rule. By contrast, a selfish miner could break the transaction inclusion and the longest chain extension rules to maximize its profit as explained below:

- **Transaction inclusion attack.** When the selfish miner publishes one or more key blocks, it withholds the last several microblocks created after the last key block. That is to say, the selfish miner continues to mine on top of its microblock chain, while honest miners mine on the last published selfish key block. Fig. 3 shows the case in which the selfish miner withholds all microblocks mined after the key block $\mathcal{B}_j$.

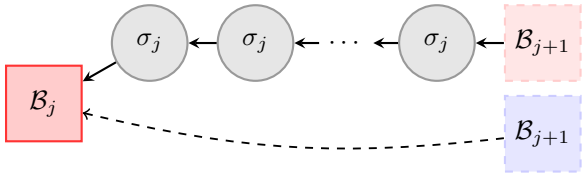

Fig. 3. An example of the transaction inclusion attack. A red (blue) square block is a selfish (honest, respectively) key block.

- **Longest chain extension attack.** When the selfish miner adopts an honest key block, it can reject some (or all) microblocks and mine directly on the last accepted microblock block (or the last key block, respectively). In other words, the selfish miner rejects the transactions in these microblocks issued by the previous honest key-block miner. This attack is illustrated in Fig. 4.

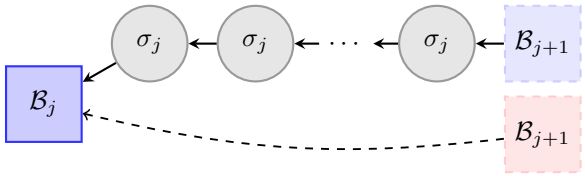

Fig. 4. An example of the longest chain extension attack. A red (blue) square block is a selfish (honest, respectively) key block.

E. Mining Revenue

The selfish miner is incentivized to find an optimal selfish mining strategy to increase its revenue. Specifically, the utility of the selfish miner can be presented by relative revenue [14], [15], i.e., the ratio of the selfish miner's revenue (including key-block reward and transaction fees) to all miners' revenue. In other words, the selfish miner would like to increase its share of key-block reward and transaction fees generated by the system[8], which is given by

$$u = \lim_{t\to\infty} \frac{r_a(t) + t_a(t)}{r_a(t) + r_h(t) + t_a(t) + t_h(t)}, \tag{1}$$

where $r_a(t)$, $r_h(t)$, $t_a(t)$, and $t_h(t)$ are the key-block rewards and transaction fees won by selfish miners and honest miners during time $[0, t]$, respectively.

[8]The number of key block and outstanding transactions to be included into the longest chain is regarded as constant.

IV. Incentive Analysis for Microblock

In this section, we analyze the incentive of microblock mining. We will present both the prior analysis [7], [12] and our new analysis.

A. Original Analysis

To resist the transaction inclusion attack and the longest chain extension attack, Bitcoin-NG divides the transaction fees included in microblocks between two consecutive key blocks into two parts. The first key-block miner gets the r fraction ($r \in [0, 1]$), while the second one obtains the remaining $1 - r$ fraction. Fig. 5 illustrates the distribution rule. The remaining subsection shows how to decide the value of r to resist the microblock mining attacks.

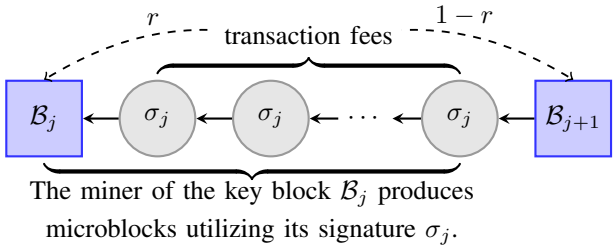

Fig. 5. Bitcoin-NG transaction fees distribution [7]

1) Resisting Transaction Inclusion Attack: When an honest key-block miner includes transactions into a microblock, it will publish the microblock and share the transaction fees with the subsequent key-block miner. However, the selfish key-block miner can create a microblock with these transactions and withhold this microblock to win all the transaction fees. Specifically, the selfish miner tries to mine a new key block on top of this secret microblock, while other honest miners mine on top of an older microblock. If the selfish miner succeeds in mining the subsequent key block, it obtains 100% of the transaction fees. Otherwise, some honest miners mine the next key block and place these transactions in their new microblock. Then, the selfish miner continues to mine on the microblock to win the remaining $1 - r$ transaction fees. To guarantee the average revenue of the selfish miner launching the above attack is smaller than he deserves, the distribution ratio should be

$$\overbrace{\alpha \times 100\%}^{\text{win } 100\%} + \overbrace{(1-\alpha) \times \alpha \times (100\% - r)}^{\text{Lose } 100\%\text{, but mine after txn}} < r,$$

therefore $r > 1 - \frac{1-\alpha}{1+\alpha-\alpha^2}$. This ratio requirement encourages the selfish miner to place a transaction in a public microblock.

Later, Yin et al. [12] found that the above condition can be made even tighter by including an extra term $\alpha(1-r)$ on the right hand side (which considers the possibility that the selfish miner is elected as the next leader):

$$\overbrace{\alpha \times 100\%}^{\text{win } 100\%} + \overbrace{(1-\alpha) \times \alpha \times (100\% - r)}^{\text{Lose } 100\%\text{, but mine after txn}} < r + \alpha(1-r),$$

therefore $r > \frac{\alpha}{1-\alpha}$.

2) Resisting Longest Chain Extension Attack: To increase revenue from some transactions, the selfish miner can ignore these transactions in honest microblock and mine on a previous block. If the selfish miner mines a key block, it can place these transactions in its own microblock. To resist this attack, the selfish miner's revenue in this case must be smaller than the revenue obtained by obeying the longest chain extension rule. Therefore, we have

$$\overbrace{\alpha \times r}^{\text{Mine next key block}} + \overbrace{\alpha^2 \times (100\% - r)}^{\text{Mine the third key Block}} < \overbrace{\alpha(100\% - r)}^{\text{Mine on microblock}},$$

which leads to $r < \frac{1-\alpha}{2-\alpha}$. Taking the upper bound into consideration, the distribution ratio r is $1 - \frac{1-\alpha}{1+\alpha-\alpha^2} < r < \frac{1-\alpha}{2-\alpha}$. When α is less than 25%, we obtain $37\% < r < 43\%$. Hence, $r = 40\%$ is chosen in the Bitcoin-NG [7].

The analysis well fits the selfish mining of high-fee transactions, which are usually too few to consider the network capacity. In other words, the selfish key-block miner can create enough microblocks to contain these transactions despite the microblock size and generation rate. However, for the majority of low-fee transactions, the analysis is not accurate anymore. Let us revisit the example of the longest chain extension attack in Fig. 4. Honest key-block miners can include as many as possible transactions (up to microblock size) into its microblocks. Assume that the selfish miner rejects all the microblocks and the associated transactions, and then tries to include these transactions into its own microblocks after finding a key block. It is easy to see that these transactions will exhaust the capacity of the selfish miner microblocks. The selfish miner would lose the chance to win new transaction fees. In other words, without considering the impact of microblock capacity, the analysis magnifies the selfish miner revenue for the majority of low-fee transactions.

Remark 1. *The analysis does not consider the impact of key-block selfish mining. Specifically, when α is less than the selfish mining threshold (i.e., 23.21% introduced later), the optimal mining strategy of key blocks is honest mining, i.e., miners publish their newly mined key block immediately and mine on others' new key block. Thus, the key-block reward fraction of each miner equals its computation power proportion. Our later analysis in this section also follows the same setting when $\alpha < 23.21\%$. However, once α is above the key-block selfish mining threshold, the selfish miner has the chance to launch the key-block mining attack to make more profit, and the impact of the key-block selfish mining cannot be ignored. We will study the selfish mining of both microblock and key block when $\alpha \in [0, 0.45)$ in Sec. V.*

B. New Analysis

In this section, we follow the mining model in Sec. III-A and consider the revenue for the selfish miner and honest miners during a time interval $[0, t]$. Without loss of generality, we assume there exists a block B_0 that the selfish miner and honest miners both agree to mine on at the starting time. (For example, B can be the genesis block.) The following lemma bounds the number of key blocks mined during the interval $[0, t]$.

Lemma 1. *During the interval $[0, t]$, the number of key blocks mined M has the following Chernoff-type bound: For $0 < \delta < 1$,*

$$\Pr(|M - ft| > \delta ft) < e^{-\Omega(\delta^2 ft)}. \tag{2}$$

Proof: Recall from Sec.III-A that the key block generation process is a Poisson process with rate f. Thus, during the interval $[0, t]$, the expectation of the total number of mined key blocks is ft, and M is a Poisson random variable with rate ft. By Lemma 5, we have $\Pr(M > (1+\delta)ft) < e^{-\Omega(\delta^2 ft)}$ and $\Pr(M < (1-\delta)ft) < e^{-\Omega(\delta^2 ft)}$, which completes the proof. ∎

This lemma shows that as t increases, the number of key blocks M is between $(1+\delta)ft$ and $(1-\delta)ft$ with high probability. Now, let $m \in [(1-\delta)ft, (1+\delta)ft]$ be an integer and let X_i ($i \in [0, m]$) denote an indicator random variable which equals one if the i-th key block is a selfish key block, as described below

$$X_i = \begin{cases} 1, \text{ selfish key block} \\ 0, \text{ honest key block.} \end{cases}$$

Without loss of generality, we assume block B_0 is an honest key block. For other key blocks, the possibility that a key block is a selfish key block equals the selfish miner computation power α.

After mining a key block, its owner can issue a series of microblocks at a constant rate v until the next key block is mined. Let Y_i denote the interval between the i-th key block and $(i+1)$-th key block. Thus, the number of produced microblocks between i-th and $(i+1)$-th key blocks is vY_i. In addition, each microblock contains a fixed number of transactions and a maximum total transaction fee K_t (see Sec. III-C). For simplicity, the microblocks produced between the 0-th and 1-th key block and/or after the m-th key block, and the corresponding transaction fees are ignored. In other words, we only consider the associated rewards with key block B_i ($i \in [1, m-1]$). Based on this model, the remaining subsections will re- the suitable value of r to resist the two microblock attacks.

1) Resisting Transaction Inclusion Attack: Recall the attack from Sec. III-D that the selfish miner hides some of its microblocks generated after a key block but keeps mining on top of the microblock chain. Hence, honest miners directly mine on top of the selfish miner's last published block. Let ρ denote the fraction of the unpublished microblocks among all the selfish microblocks between two consecutive key blocks[9]. Clearly, if $\rho = 1$, there are no microblocks between two consecutive key blocks separately mined by the selfish miner and honest miner. Thus, if any two consecutive key blocks satisfy $(X_i, X_{i+1}) = (1, 0)$, there are $(1-\rho)vY_i$ microblocks

[9]In reality, the selfish miner cannot foresee how many microblocks it can produce until the next key block is mined, but it can set a number for the published microblocks according to its history data.

between them from the view of an honest miner; otherwise, there are vY_i microblocks.

Let Z_i denote an indicator random variable equal to one if $\{X_i = 1, X_{i+1} = 0\}$, and equal to zero otherwise. Next, let $Z = \sum_{i=1}^{m-1} Z_i$. The following lemma will aid us to bound the value of Z with high probability:

Lemma 2. *For m consecutive key blocks, the number of block pairs $(X_i, X_{i+1}) = (1, 0)$ has the following Chernoff-type bound: For $0 < \delta < 1$,*

$$\Pr(|Z - \alpha\beta(m-1)| > \delta\alpha\beta(m-1)) < e^{-\Omega\left(\delta^2\alpha\beta m\right)}. \quad (3)$$

Proof: Without loss of generality, we assume the value of m is odd. Let $Z^j = \sum_{k=0}^{(m-3)/2} Z_{j+2k+1}$ where $j \in \{0, 1\}$. Then, $Z = Z^0 + Z^1$. Our key observation is that $\{Z_1, Z_3, \ldots, Z_{m-2}\}$ are independent random variables, because Z_i is a function of (X_i, X_{i+1}). Particularly, $E\left(Z^j\right) = E\left(\sum_{k=0}^{(m-3)/2} Z_{j+2k+1}\right) = \sum_{k=0}^{(m-3)/2} E\left(Z_{j+2k+1}\right)$. Note that $P\{Z_i = 1\} = P\{X_i = 1\}P\{X_{i+1} = 0\} = \alpha\beta$, since X_i and X_{i+1} are independent. Thus, we have $E\left(Z^j\right) = \alpha\beta(m-1)/2$. By Lemma **??**, we have

$$\Pr\left(Z < (1-\delta)\alpha\beta(m-1)\right) < e^{-\Omega\left(\delta^2\alpha\beta(m-1)\right)} = e^{-\Omega\left(\delta^2\alpha\beta m\right)}.$$

Similarly, we can prove that $\Pr\left(Z > (1+\delta)\alpha\beta(m-1)\right) < e^{-\Omega\left(\delta^2\alpha\beta m\right)}$. ∎

This lemma shows as m increase, the number of key pairs $(X_i, X_{i+1}) = (1, 0)$ is between $(1-\delta)\alpha\beta m$ and $(1+\delta)\alpha\beta m$ with high probability. In addition, as the value of m increases, with high probability the total amount of transaction fees is close to

$$\begin{aligned}\sum_{i=1}^{m-1}\left(vY_iR_t - \rho vZ_iY_iR_t\right) &= vR_t\sum_{i=1}^{m-1}(1-\rho Z_i)Y_i \\ &= vR_t(1-\alpha\beta\rho)(m-1)/f.\end{aligned} \quad (4)$$

In other words, due to the transaction inclusion attack, the $vR_t\alpha\beta\rho(m-1)/f$ transaction fees would be lost for all miners. Clearly, the selfish miner loses $rvR_t\alpha\beta\rho(m-1)/f$ transaction fees when an honest key block is found after its last published block. Thus, the transaction fees won by the selfish miner is $vR_t(\alpha - r\alpha\beta\rho)(m-1)/f$.

The Lemma 1 shows as interval t increase, the number of mined key blocks M also increases and is close to ft with high probability. Thus, combined with Lemma 2, it's easy to see as interval t increases, with high probability, the total amount of transaction fees is close to $vR_t(1-\alpha\beta\rho)(ft-1)/f$, while the transaction fees won by the selfish miner is close to $vR_t(\alpha - r\alpha\beta\rho)(ft-1)/f$. Thus, with high probability the selfish miner's relative revenue of transaction fees during $[0, t]$ is close to

$$\begin{aligned}u' &= \max_{0\le\rho\le1}\frac{vR_t(1-\alpha\beta\rho)(ft-1)/f}{vR_t(\alpha - r\alpha\beta\rho)(ft-1)/f} \\ &= r + \max_{0\le\rho\le1}\frac{\alpha - r}{1-\alpha\beta\rho}.\end{aligned} \quad (5)$$

Clearly, if $r \le \alpha$ and $\rho = 1$,

$$u' = r + \frac{\alpha - r}{1-\alpha\beta};$$

otherwise, $\rho = 0$ and $u' = \alpha$. To guarantee the adversary cannot gain more from the transaction inclusion attack, the relative revenues of the selfish miner in the above two cases can not be greater than α. By solving the equations, we have $r > \alpha$.

2) Resisting Longest Chain Extension Attack: Recall the attack from Sec. III-D that the selfish miner can bypass some honest microblocks, and mines directly on an old honest block. Similarly, let ρ denote the rejected microblock fraction. If $\rho = 1$, this means the selfish miner rejects all honest microblocks and mines directly on the last honest key block. Thus, if two consecutive key blocks are $(X_i, X_{i+1}) = (0, 1)$, there are $(1-\rho)vY_i$ honest microblocks accepted by the longest chain. Let K_i denote an indicator random variable equal to one if $\{X_i = 0, X_{i+1} = 1\}$, and equal to zero otherwise. Let $K = \sum_{i=1}^{m-1} K_i$. The following lemma will aid us to bound the expectation of K for m blocks:

Lemma 3. *For the m block sequence, the number of block-pair $(X_i, X_{i+1}) = (0, 1)$ has the following Chernoff-type bound: For $0 < \delta < 1$,*

$$\Pr(|K - \alpha\beta(m-1)| > \delta\alpha\beta(m-1)) < e^{-\Omega\left(\delta^2\alpha\beta m\right)}. \quad (6)$$

Proof: The proof is similar to Lemma 2. We omit it here due to the space constraint. ∎

This lemma 2 shows as m increase, the number of key pairs $(X_i, X_{i+1}) = (0, 1)$ is between $(1-\delta)\alpha\beta m$ and $(1+\delta)\alpha\beta m$ with high probability. In addition, as the value of m increases, with high probability the total amount of transaction fees is close to

$$\begin{aligned}\sum_{i=1}^{m-1}\left(vY_iR_t - K_i\rho vY_iR_t\right) &= vR_t\sum_{i=1}^{m-1}(1-\rho K_i)Y_i \\ &= vR_t(1-\alpha\beta\rho)(m-1)/f.\end{aligned} \quad (7)$$

Due to the longest chain extension attack, the $vR_t\alpha\beta\rho(m-1)/f$ transaction fee would be lost for all miners. Clearly, the selfish miner loses $(1-r)vR_t\alpha\beta\rho(m-1)/f$ transaction fee for rejecting $v\alpha\beta\rho(m-1)/f$ honest microblocks. Similarly, as interval t increases, with high probability, the total amount of transaction fees is close to $vR_t\alpha\beta\rho(ft-1)/f$, while the transaction fees won by the selfish miner is close to $vR_t(\alpha - (1-r)\alpha\beta\rho)(ft-1)/f$. Thus, with high probability the selfish miner's relative revenue of transaction fees during $[0, t]$ is close to

$$\begin{aligned}u' &= \max_{0\le\rho\le1}\frac{vR_t(\alpha - (1-r)\alpha\beta\rho)(ft-1)/f}{vR_t(1-\alpha\beta\rho)(ft-1)/f} \\ &= 1 - r + \max_{0\le\rho\le1}\frac{r-\beta}{1-\alpha\beta\rho}.\end{aligned} \quad (8)$$

Clearly, if $r \ge \beta$, $\rho = 1$ and

$$u' = 1 - r + \frac{\beta - r}{1-\alpha\beta};$$

otherwise, $\rho = 0$ and $u' = \alpha$. To guarantee the adversary cannot gain more from the transaction inclusion attack, the relative revenues of the selfish miner in the above two cases can not be greater than α. By solving the equations, we have $r < \beta$. Combining the two incentive sub-mechanisms of transaction inclusion and longest chain extension, the value of r needs to satisfy that

$$\alpha < r < \beta.$$

Remark 2. *Our analysis assumes that the number of microblocks is vY_i between the ith key block and the (i+1)th key block. It can be extended to the case that the number of microblocks is a function of Y_i. This function can be used to take into account some practical factors, such as the network delay.*

C. Comparison

We now compare our bounds of split ratio r with the results in prior work [7], [12]. The split ratio r in Bitcoin-NG [7] is $1 - \frac{1-\alpha}{1+\alpha-\alpha^2} < r < \frac{1-\alpha}{2-\alpha}$ (see Sec. IV-A). The bound of r is tighten to be $\frac{\alpha}{1-\alpha} < r < \frac{1-\alpha}{2-\alpha}$ under the same model by Yin et al. [12]. Compared with these results, the range of r in our model is pretty wide and quite different as $\alpha < r < \beta$, depicted in Fig. 6. It's easy to see when the α is more than 29%, the intersection of the two conditions [7], [12] is empty. That implies the incentive compatibility of Bitcoin-NG cannot be maintained anymore. By contrast, as α is less than 50%[10], our analysis shows there always exists a valid value for r to maintain the incentive compatibility of Bitcoin-NG.

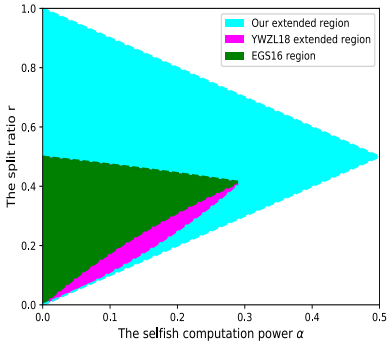

Fig. 6. The comparison of the transaction fee distribution ratio.

Bitcoin is shown to resist more than 29% selfish computation power in prior works [7], [15], [20]. Specifically, under the optimal network assumptions (i.e., zero block latency network), Bitcoin is secure against almost $1/3$ selfish mining power [7], [15]. In addition, the work [20] shows Bitcoin is secure under 32% selfish computation power by optimizing the protocol. For these cases, the analyses [7], [12] show that Bitcoin-NG is less resilient than Bitcoin because there is no valid value for r. By contrast, our analysis shows that security can be guaranteed by adopting a suitable value of r.

[10]This is because once α is above the $1/2$, the selfish miner can launch the well-known 51% attack and gains more.

The transaction model and analysis [7], [12] well fit the high-fee transactions, which account for a minority of all transactions. Thus, miners compete to include them in microblocks and do not need to worry about the microblocks' capacity limit. Once included, there are no more high-fee transactions for the consecutive leaders. However, in reality, the majority of transactions have small transaction fees, and the volume of outstanding transactions is usually larger, such as Bitcoin and Ethereum [21]. Thus, our model is more realistic in such a scenario where outstanding transactions are more than what the network can process. More importantly, the high transaction fees usually cause instability of the whole system (see Sec. VI-B). For safety, a blockchain system should limit the maximum value of the fee that a transaction can include.

V. Incentive Analysis for both Microblock and Key Block

In this section, we first model the selfish mining strategies of both key blocks and microblocks with a Markov decision process (MDP) and then use the MDP to compute the optimal revenues of the selfish miner with different computation power. The results enable us to have a more complete picture of the security of Bitcoin-NG since our previous analysis focuses on $\alpha < 23.21\%$.

A. Designing a Markov Decision Process (MDP)

The MDP of the selfish mining process in Bitcoin-NG is based on the previous work [14] in which Sapirshtein et al. transformed the selfish mining strategies in Bitcoin into an MDP. Unlike Bitcoin, the microblock architecture in Bitcoin-NG introduces two new attacks, i.e., the transaction inclusion attack and the longest chain extension attack (see Sec. III-D). Thus, merging these two microblock attacks into the selfish mining of key blocks is a key step in our new analysis.

However, it is non-trivial to realize this. On one hand, the additional microblocks introduce extra complexity for the MDP design (e.g., more mining strategies and rewards); on the other hand, the number of microblocks between two consecutive key blocks varies, which relies not only on the key block interval, but also on the selfish miners' mining strategies. So, we introduce two simplifications here based on our analysis in Sec. IV. First, the key block interval is set to a constant $1/f$, and the number of microblocks produced between two consecutive key blocks is v/f. Second, only the binary choices (i.e., publishing or hiding all selfish microblocks in the transaction inclusion attack, and accepting or rejecting all honest microblocks in the longest chain extension attack) are considered.

An MDP can be presented by a 4-tuple $M := (S, A, P, R)$, where S is the state space, A is the action space, P is the stochastic state transition matrix, and R is the reward matrix. Specifically, S contains all possible states in the selfish mining process; A includes the available actions (e.g., publishing or hiding blocks by the selfish miner) at each state; P contains the transition probabilities from the current state to the next state according to the taken action; R records how much the

selfish miner obtains when there are some state transitions. Table I illustrates the MDP of selfish mining in Bitcoin-NG. Below we will discuss each component of the 4-tuple:

Actions. The selfish miner has eight actions in our model.

- **Adopt and include.** The selfish miner accepts all honest key blocks and the corresponding honest microblocks. In other words, the selfish miner will mine its key block on the last honest block and abandon its own private chain. This action is referred to as adopt.
- **Adopt and exclude.** The selfish miner accepts all honest key blocks and microblocks except for microblocks produced after the last honest key block. Specifically, the selfish miner directly mines on top of the last honest key block, which is referred to as adoptE.
- **Override and publish.** The selfish miner publishes all its key blocks and corresponding microblocks whenever its private chain is longer than the honest one. The chain length is counted by the key block. This action is denoted as override.
- **Override and hide.** The selfish miner publishes all its key blocks and the microblocks except for these mined after the last selfish key block whenever its private chain is longer than the honest one. This action is denoted as overrideH.
- **Match and publish.** When an honest miner finds one new key block, the selfish miner publishes its key block of the same height and the microblocks built after this key block. This action is available when the selfish miner has one block in advance and is referred to as match.
- **Match and hide.** When an honest miner generates a new key block, the selfish miner publishes its key block of the same height while hides the microblocks built after this key block. This action is also available when the selfish miner has one block in advance. This action is denoted as matchH.
- **Wait.** In this action, the selfish miner does not publish any new key blocks and microblocks, while keeps mining on its private chain until a new key block and corresponding microblocks are found.
- **Revert.** The selfish miner reverts its previous actions. Specifically, the selfish miner can publish its hidden microblocks when there is no honest key block mined after its block; the selfish miner can include the honest microblocks (decided to excluded in the previous decision) or excludes the honest microblocks (decided to included in the previous decision) once there is no selfish key block mined on honest block.

State space. The state space S is also composed by 4-tuple $(l_a, l_h, \mathsf{fork}, \mathsf{lastMicroBlock})$.

- $\boldsymbol{l_a}$ accounts for the length of the chain mined by the selfish miner after the last common ancestor key block. Note that the length is counted by the selfish key blocks in this branch.
- $\boldsymbol{l_h}$ is the length of the public chain which can be viewed by all honest miners.
- **fork**. The field fork obtains three possible values, dubbed noTie, tie and tie$'$. Specifically, tie means the selfish miner publishes l_h selfish key block and the corresponding microblocks; tie$'$ presents the selfish miner publishes l_h selfish key block and the corresponding microblocks except for these after the last selfish key block; noTie signifies that there are not two public branches with the equivalent length.
- **lastMicroBlock**. This field also includes four possible values, dubbed H_{in}, H_{ex}, S_p, and S_h. Specifically, H_{in} (H_{ex}) represents the common ancestor is an honest key block, and the corresponding microblocks are accepted (rejected, respectively) by the selfish miner. While S_p (or S_h) which stands for the common ancestor is a selfish key block, and the corresponding microblocks mined are published (or hidden) by the selfish miner.

State Transition and Reward. We use a 4-tuple (R_h, T_h, R_a, T_a) to indicate the rewards won by the selfish and honest miners in the state transitions. Specifically, R_h (R_a) is the key block rewards for honest miners (the selfish miner, respectively), while T_h (T_a) is the transaction fee for honest miners (the selfish miner, respectively). In addition, instead of recording the number of rewards, each field only records the number of key block reward or transaction fees (the total transaction fee in v/f microblocks as one unit) won by miners. More importantly, the transaction fees included in the microblocks after the common ancestor key block are not assigned to miners until the next ancestor key block is decided. This is because these transaction fees are affected by some future actions of the selfish miner (see Sec. IV).

In adopt or adoptE actions, the selfish miner accepts l_h honest key blocks and the microblocks mined before these key blocks. Honest miners obtain $l_h K_b$ key block rewards and $(l_h - 1)v/fK_t$ transaction fees. In override or overrideH actions, the selfish miner publishes $l_h + 1$ selfish key blocks. Honest miners accept these key blocks and the microblocks produced before the key blocks. Thus, the selfish miner obtains $(l_h + 1)R_b$ key block rewards and $l_h v/f R_t$ transaction fees. In the match actions, the next state depends on whether the next key block is created by the selfish miner (w.p. α), by some honest miners working on the honest branch (w.p. $(1-\gamma)(1-\alpha)$), or by the left honest miners mining on the selfish branch (w.p. $\gamma(1-\alpha)$). In the latter case, the selfish miner effectively overrides the honest miners' branch. It can obtain $l_h K_b$ key block reward and $(l_h - 1)v/fK_t$ transaction fees. Note that the value of γ is decided by the adopted fork solution (e.g., $\gamma = 0.5$ in the uniform tie-break policy).

Once the common ancestor key block changed, the transaction fees in the microblocks produced after the previous ancestor key block will be assigned. There are two cases.

- The previous common ancestor key block is mined by an honest miner. This case can be further divided into two subcases: 1) the next key block is mined by honest miners, and honest miners get v/fk_t transaction fees;

TABLE I
STATE TRANSITION AND REWARD MATRICES FOR THE OPTIMAL SELFISH MINING.

State $\times$ Action	State	Probability	Reward	Condition
$(l_a, l_h, \cdot, S_h)$, adopt	$(1, 0, \mathsf{noTie}, H_{\mathsf{in}})$	α	$(l_h, l_h, 0, 0)$	–
$(l_a, l_h, \cdot, S_p)$, adopt	$(0, 1, \mathsf{noTie}, H_{\mathsf{in}})$	$1-\alpha$	$(l_h, l_h - 1 + (1-r), 0, r)$	
$(l_a, l_h, \cdot, \{H_{\mathsf{in}}, H_{\mathsf{ex}}\})$, adopt			$(l_h, l_h - 1, 0, 0)$	
$(l_a, l_h, \cdot, S_h)$, adoptE	$(1, 0, \mathsf{noTie}, H_{\mathsf{ex}})$	α	$(l_h, l_h, 0, 0)$	–
$(l_a, l_h, \cdot, S_p)$, adoptE	$(0, 1, \mathsf{noTie}, H_{\mathsf{ex}})$	$1-\alpha$	$(l_h, l_h - 1 + (1-r), 0, r)$	
$(l_a, l_h, \cdot, \{H_{\mathsf{in}}, H_{\mathsf{ex}}\})$, adoptE			$(l_h, l_h - 1, 0, 0)$	
$(l_a, l_h, \cdot, H_{\mathsf{ex}})$, override	$(l_a - l_h, 0, \mathsf{noTie}, S_p)$	α	$(0, 0, l_h + 1, l_h + 1)$	$l_a > l_h$
$(l_a, l_h, \cdot, H_{\mathsf{in}})$, override	$(l_a - l_h - 1, 1, \mathsf{noTie}, S_p)$	$1-\alpha$	$(0, r, l_h + 1, l_h + (1-r))$	
$(l_a, l_h, \cdot, \{S_p, S_h\})$, override			$(0, 0, l_h + 1, l_h)$	
$(l_a, l_h, \cdot, H_{\mathsf{ex}})$, overrideH	$(l_a - l_h, 0, \mathsf{noTie}, S_h)$	α	$(0, 0, l_h + 1, l_h + 1)$	$l_a > l_h$
$(l_a, l_h, \cdot, H_{\mathsf{in}})$, overrideH	$(l_a - l_h - 1, 1, \mathsf{noTie}, S_h)$	$1-\alpha$	$(0, r, l_h + 1, l_h + (1-r))$	
$(l_a, l_h, \cdot, \{S_p, S_h\})$, overrideH			$(0, 0, l_h + 1, l_h)$	
$(l_a, l_h, \mathsf{noTie}, \cdot)$, wait	$(l_a + 1, l_h, \mathsf{noTie}, *)$	α	$(0, 0, 0, 0)$	–
	$(l_a, l_h + 1, \mathsf{noTie}, *)$	$1-\alpha$		
$(l_a, l_h, noTie, H_{\mathsf{in}})$, match	$(l_a + 1, l_h, \mathsf{tie}, H_{\mathsf{in}})$	α	$(0, 0, 0, 0)$	$la \geq l_h$
$(l_a, l_h, \mathsf{tie}, H_{\mathsf{in}})$, wait	$(l_a - l_h, 1, \mathsf{noTie}, S_p)$	$\gamma(1-\alpha)$	$(0, r, l_h, l_h - 1 + (1-r))$	
	$(l_a, l_h + 1, \mathsf{noTie}, H_{\mathsf{in}})$	$(1-\gamma)(1-\alpha)$	$(0, 0, 0, 0)$	
$(l_a, l_h, noTie, H_{\mathsf{ex}})$, match	$(l_a + 1, l_h, \mathsf{tie}, H_{\mathsf{ex}})$	α	$(0, 0, 0, 0)$	$la \geq l_h$
$(l_a, l_h, \mathsf{tie}, H_{\mathsf{ex}})$, wait	$(l_a - l_h, 1, \mathsf{noTie}, S_p)$	$\gamma(1-\alpha)$	$(0, 0, l_h, l_h - 1)$	
	$(l_a, l_h + 1, \mathsf{noTie}, H_{\mathsf{ex}})$	$(1-\gamma)(1-\alpha)$	$(0, 0, 0, 0)$	
$(l_a, l_h, \mathsf{noTie}, \{S_p, S_h\})$, match	$(l_a + 1, l_h, \mathsf{tie}, *)$	α	$(0, 0, 0, 0)$	$la \geq l_h$
$(l_a, l_h, \mathsf{tie}, \{S_p, S_h\})$, wait	$(l_a - l_h, 1, \mathsf{noTie}, S_p)$	$\gamma(1-\alpha)$	$(0, 0, l_h, l_h)$	
	$(l_a, l_h + 1, \mathsf{noTie}, *)$	$(1-\gamma)(1-\alpha)$	$(0, 0, 0, 0)$	
$(l_a, l_h, \mathsf{noTie}, H_{\mathsf{in}})$, matchH	$(l_a + 1, l_h, tie', H_{\mathsf{in}})$	α	$(0, 0, 0, 0)$	$la \geq l_h$
$(l_a, l_h, tie', H_{\mathsf{in}})$, wait	$(l_a - l_h, 1, \mathsf{noTie}, S_h)$	$\gamma(1-\alpha)$	$(0, r, l_h, l_h - 1 + (1-r))$	
	$(l_a, l_h + 1, \mathsf{noTie}, H_{\mathsf{in}})$	$(1-\gamma)(1-\alpha)$	$(0, 0, 0, 0)$	
$(l_a, l_h, noTie, H_{\mathsf{ex}})$, matchH	$(l_a + 1, l_h, \mathsf{tie}', H_{\mathsf{ex}})$	α	$(0, 0, 0, 0)$	$l_a \geq l_h$
$(l_a, l_h, \mathsf{tie}', H_{\mathsf{ex}})$, wait	$(l_a - l_h, 1, \mathsf{noTie}, S_h)$	$\gamma(1-\alpha)$	$(0, 0, l_h, l_h - 1)$	
	$(l_a, l_h + 1, \mathsf{noTie}, H_{\mathsf{ex}})$	$(1-\gamma)(1-\alpha)$	$(0, 0, 0, 0)$	
$(l_a, l_h, noTie, \{S_p, S_h\})$, matchH	$(l_a + 1, l_h, \mathsf{tie}', *)$	α	$(0, 0, 0, 0)$	$l_a \geq l_h$
$(l_a, l_h, \mathsf{tie}', \{S_p, S_h\})$, wait	$(l_a - l_h, 1, \mathsf{noTie}, S_h)$	$\gamma(1-\alpha)$	$(0, 0, l_h, l_h)$	
	$(l_a, l_h + 1, \mathsf{noTie}, *)$	$(1-\gamma)(1-\alpha)$	$(0, 0, 0, 0)$	
$(l_a, l_h, \mathsf{tie}', \cdot)$, revert	$(l_a, l_h, \mathsf{tie}, *)$	1	$(0, 0, 0, 0)$	–
$(l_a, l_h, \cdot, S_h)$, revert	$(l_a, l_h, *, S_p)$	1	$(0, 0, 0, 0)$	$l_h = 0$
$(l_a, l_h, \cdot, H_{\mathsf{ex}})$, revert	$(l_a, l_h, *, H_{\mathsf{in}})$	1	$(0, 0, 0, 0)$	$l_a = 0$

$*$ denotes the state element remains the same in the state transition.

2) the next key block is mined by the selfish miner and **lastMicroBlock** $= H_{\mathsf{in}}$, honest miners get rv/fk_t transaction fees and the selfish miner gets $(1-r)v/fk_t$ transaction fees.

- The previous common ancestor key block is mined by the selfish miner. This case can be further divided into two subcases: 1) the next key block is mined by the selfish miner, and the selfish miner gets v/fk_t transaction fees; 2) the next key block is mined by some honest miners and **lastMicroBlock** $= S_{\mathsf{p}}$, the selfish miner gets rv/fk_t transaction fees and honest miners get $(1-r)v/fk_t$ transaction fees.

B. Evaluation Results

We use the MDP toolbox developed in MATLAB [22] to obtain the selfish miner's relative revenue, denoted in the equation (1). Note that in the following evaluations, Bitcoin-NG adopts the uniform tie-breaking policy ($\gamma = 0.5$).

The selfish mining threshold. Fig. 7 shows the relative revenues of the selfish miner when $r = 0.4$ (used in Bitcoin-NG [7]) and $\alpha \in [0, 0.45]$. We consider three reward settings: $k \to 0$, $k = v/f$, and $k \to \infty$. Specifically, in the first setting, the transaction fees dominate the miners' revenue; in second setting, the transaction fees included in v/f microblocks between two consecutive key blocks have the same weight with one key block reward; in the third setting, the key block rewards dominate miners' revenue. Note that the key block reward dominated case has a similar reward distribution as Bitcoin, i.e., the microblock architecture does not impact the system.

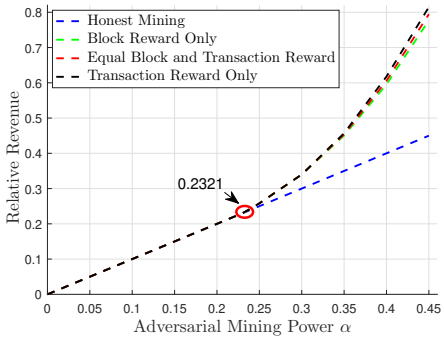

Fig. 7. The selfish miner's relative revenue.

The selfish mining thresholds in these three settings are

all 23.21%, which illustrates that when adopting the suitable r, the microblock architecture does not affect the system security compared with Bitcoin. In addition, the selfish miner's revenues in the three settings are still the same even when $\alpha > 29\%$, which also supports that Bitcoin-NG is as resilient as Bitcoin.

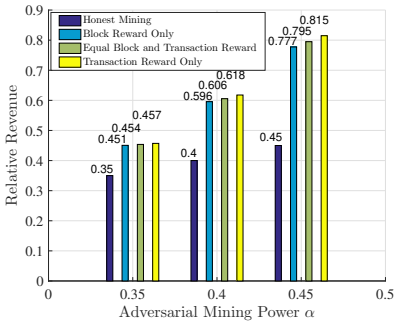

Fig. 8. The relative revenue when $\alpha \in \{0.35, 0.4, 0.45\}$.

When $\alpha > 35\%$, the differences between the selfish miner's revenues in the three settings and the honest revenue are clearly exhibited in Fig. 8. It's easy to see that the selfish miner can obtain the highest revenue in the transaction fee dominated case. This implies that the microblock architecture can slightly increase the selfish miner's revenue, but the increase is much smaller than the results [13]. We delay explaining the reason.

The selfish revenue with different split ratio. Fig. 9 shows the selfish miner's relative revenue with different values of r and different values of α. In Fig. 9, we only consider the case of $k \to 0$ (the transaction fee dominated case). We can see when r ranges from 0.2321 to 0.7679, the selfish revenue is the lowest.

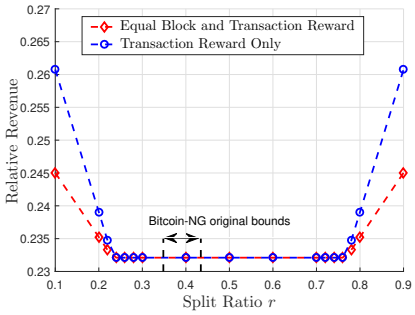

Fig. 9. The selfish miner's relative revenue with different computation power α and split ratio r.

C. Comparison

First, we validate that when adopting the suitable r, the selfish mining threshold in Bitcoin-NG is the same as the threshold in Bitcoin [14]. This shows Bitcoin-NG and Bitcoin have the same security level. Our result first supports the security claim of Bitcoin-NG [7] by using MDP.

Second, we find out that when $\alpha > 35\%$, the selfish miner in Bitcoin-NG can gain more revenue than in Bitcoin. But the increase is not so significant as shown in the previous result [13]. This is because Wang et al. [13] treated the selfish mining of key block and microblocks as two independent issues and take the outputs of key-block mining attack (i.e., the fraction of selfish key blocks) as input into microblock mining analysis. That method will amplify the selfish miner's revenue by serializing the key block and microblocks mining. By contrast, our MDP integrates the selfish mining of key blocks and microblocks together, which makes it more accurate and convincing.

VI. Discussion

A. Leader Corrupted Attack

In Bitcoin-NG, when a miner generates and publishes a key block, it will become a leader and can produce a series of microblocks. However, once publishing its key block, this new leader is likely to be corrupted (e.g., by bribery), and then it can manipulate the transactions in the following microblocks. To address this issue, we design a pre-proposing mechanism by using the Merkle tree as shown in Fig. 10. First, each miner pre-proposes a set of microblocks, and each microblock contains several outstanding transactions. Next, the miner constructs the hash of these microblocks into a Merkle tree and includes the root in its key block header. Finally, the miner starts solving the PoW puzzle of this key block.

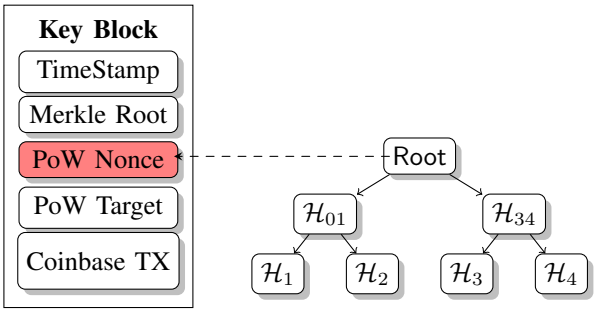

Fig. 10. The new key block structure.

When a key block is mined, the key block header together with the hash values of the pre-proposed microblocks will be published by the leader. Later on, these microblocks will be published whose hash values, as well as the pre-determined order (i.e., their positions in the Merkle tree) should match the Merkle root. Other miners can easily verify these microblocks. Once they find out a mismatch, they can reject these microblocks. Clearly, this mechanism prevents a key block leader manipulates the transactions in its issued microblocks after bribery. In some sense, this mechanism is another way of decoupling leader election and transaction serialization, which we believe is the spirit of Bitcoin-NG.

B. Transaction Fee Limits

A Bitcoin or Bitcoin-NG client can create and broadcast a transaction with an arbitrary transaction fee, and the transaction fee is rewarded to the miner who put the transaction in a block accepted by the network. Thus, it is easy to see that rational miners would like to maximize their own payoff and pick the transaction with high fees into a block. In other words, the higher the transaction fee is, the more miners try to include it into the latest mined block. Thus, some low-fee transactions may wait for days or months to be confirmed. In addition, the high-fee transactions (also known as "whale" transactions) may lead to many security issues. First, the whale transactions can incentivize miners to join attacks, such as a double-spending attack [23]. Second, combined with the smart contract, these transactions can incentivize miners to mine on the attacker's desired chain branch [24]. Third, these transactions can increase the instability of the whole system, such as selfish mining [25]. Thus, we suggest a blockchain system should limit the maximum value of the transaction fee.

VII. Related Work

In this section, we review the NC-based scaling protocols and the incentive analysis of Bitcoin and Bitcoin-NG.

A. Scaling Protocols

There are two main approaches to redesign NC-based blockchains to improve the transaction throughput and/or reduce the transaction latency. The first is Bitcoin-NG and its variants, which decouple the leader election with the transaction serialization. The second is replacing the chain structure with more generalized graphs, e.g., directed acyclic graph (DAG) or parallel chain.

Bitcoin-NG and its variants. By decoupling leader election with transaction serialization, Bitcoin-NG can successfully approach the near-optimal throughput. Due to its simplicity and significant prompt in throughput, this decoupling idea has inspired a lot of blockchain protocols [8]–[10], [26]–[28]. Kokoris-Kogias et al. [8] proposed Byzcoin, which replaces the single key-block owner with a committees to order transactions and leverages the classical Byzantine Fault Tolerance (BFT) consensus to realize agreement between the committee members. Furthermore, Pass and Shi [9] formally modeled this family of blockchain protocols as hybrid consensus. Later, Abraham et al. [10] proposed Solida, a decentralized blockchain protocol based on reconfigurable Byzantine consensus augmented by proof-of-work. It is easy to see that NC can guarantee decentralization by electing the committee, while the classical BFT consensus can provide an instant finality and high efficiency. Some multi-chain designs [26]–[28] make a forward step to extend one single consensus committee to multiple committees.

Non-chain Style Block Structure. Due to the inherent limitation (i.e., the throughput-security trade-off) of NC, many protocols tries to replace the chain structure with directed acyclic graph (DAG) (e.g., Ghost [29], Phantom [30], Spectre [31], and Conflux [32]) or parallel chain (e.g., Prism [11], OHIE [33], and Parallel Chain [34]). However, their reward mechanism design and formal incentive analysis are less well known in the literature partially due to their complicated design.

B. Incentive Analysis

In this subsection, we introduce the prior works of incentive analysis in Bitcoin and Bitcoin-NG.

Bitcoin Incentive Analysis. Eyal and Sirer [15] show that the Bitcoin mining protocol is not incentive competitive. They also introduce a deviant strategy named selfish mining, which wastes honest power and decreases the security threshold to 0.25. Nayak et al. [16] conclude the selfish mining strategy and extend it to the stubborn mining strategies, which also combines an Eclipse attack. Moreover, Sapirshtein et al. [14] and Gervais et al. [17] try to figure out the optimal Bitcoin selfish mining threshold utilizing the MDP tool. Carlsten et al. [25] focus more on the deviating strategy in the transaction-fee regime where the block reward dwindles to a negligible amount. Their undercutting attack works even an attacker only accounts for small computation power and a poor network connection. After confirming the postulate of Carlsten et al. [25], Tsabary and Eyal [35] additionally study the Bitcoin gap game between block reward and transaction fee.

Bitcoin-NG Incentive Analysis. As a variant of NC, Bitcoin-NG also relies on incentive to provide security. Yin et al. [12] extended the transaction fee distribution ratio after considering another situation what the original paper omits [7]. Wang at al. [13] considered advanced selfish mining strategies, i.e., stubborn mining strategies, when an attacker may manipulate the micro-block chains between two honest parties. Wang at al. [36] propose a Bitcoin-NG advance attack in their formalized game-theoretical model. However, these prior works have several limitations in the incentive analysis (see Sec I).

VIII. Conclusion

In this paper, we have proposed a new incentive analysis of Bitcoin-NG considering the network capacity. Our model enables us to evaluate the impact of key-block generation interval and microblock generation rate, which is missing in the previous analysis. In particular, we have shown that Bitcoin-NG can achieve even better incentive compatibility against the microblock mining attack in our model. In addition, we have modeled the selfish mining of key blocks and microblocks jointly into an MDP and shown the threshold of Bitcoin-NG is a little lower than in Bitcoin only when the selfish mining power α is greater than 35%. We hope that our in-depth incentive analysis for Bitcoin-NG can shed some light on the mechanism design and incentive analysis of next-generation blockchain protocols.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Working Paper*, 2008.

[2] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *IEEE P2P 2013 Proceedings*, Sep. 2013, pp. 1–10.

[3] J. A. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *EUROCRYPT 2015*, 2015, pp. 281–310.

[4] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *EUROCRYPT 2017*, pp. 643–673.

[5] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in Bitcoin," in *FC 2015*, pp. 507–527.

[6] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *FC 2016*, pp. 106–125.

[7] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *NSDI 2016*, pp. 45–59.

[8] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing Bitcoin security and performance with strong consistency via collective signing," in *USENIX Security 2016*, pp. 279–296.

[9] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *DISC 2017*, vol. 91, pp. 39:1–39:16.

[10] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solida: A blockchain protocol based on reconfigurable Byzantine consensus," in *IACR Cryptology ePrint Archive*, 2016.

[11] V. K. Bagaria, S. Kannan, D. Tse, G. C. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *CCS 2019*, pp. 585–602.

[12] J. Yin, C. Wang, Z. Zhang, and J. Liu, "Revisiting the incentive mechanism of Bitcoin-NG," in *ACISP 2018*, pp. 706–719.

[13] Z. Wang, J. Liu, Z. Zhang, Y. Zhang, J. Yin, H. Yu, and W. Liu, "A combined micro-block chain truncation attack on Bitcoin-NG," in *ACISP 2019*, pp. 322–339.

[14] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *FC 2016*, pp. 515–532.

[15] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, Jun. 2018.

[16] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *EuroS&P 2016*, pp. 305–320.

[17] 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 *CCS 2016*, pp. 3–16.

[18] J. Niu and C. Feng, "Selfish mining in Ethereum," in *ICDCS 2019*, pp. 1306–1316.

[19] N. Papadis, S. C. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *INFOCOM 2018*, pp. 2546–2554.

[20] E. Heilman, "One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner," in *FC 2014*, pp. 161–162.

[21] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger Byzantium version," *Ethereum project yellow paper*, pp. 1–32, 2018. [Online]. Available: https://github.com/ethereum/yellowpaper

[22] I. Chadès, G. Chapron, M.-J. Cros, F. Garcia, and R. Sabbadin, "Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems," *Ecography*, vol. 37, no. 9, pp. 916–920, 2014.

[23] K. Liao and J. Katz, "Incentivizing blockchain forks via whale transactions," in *FC 2017*, pp. 264–279.

[24] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *FC 2019*, pp. 3–18.

[25] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of Bitcoin without the block reward," in *CCS 2016*, pp. 154–167.

[26] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *CCS 2016*, pp. 17–30.

[27] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *S&P 2018*, pp. 583–598.

[28] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *CCS 2018*, pp. 931–948.

[29] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in Bitcoin," in *FC 2015*, pp. 507–527.

[30] Y. Sompolinsky and Z. Aviv, "Phantom: A scalable blockdag protocol." *IACR Cryptology ePrint Archive*, vol. 2018, p. 104, 2018.

[31] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol," *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159.

[32] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018.

[33] H. Yu, I. Nikolic, R. Hou, and P. Saxena, "Ohie: Blockchain scaling made simple," in *Proceedings of the 41th IEEE Symposium on Security and Privacy*, ser. S&P. San Francisco, CA, USA: IEEE, 2020.

[34] M. Fitzi, P. Gazi, A. Kiayias, and A. Russell, "Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition," *Cryptology ePrint Archive, Report 2018/1119*, 2018.

[35] I. Tsabary and I. Eyal, "The gap game," in *CCS 2018*, pp. 713–728.

[36] T. Wang, X. Bai, H. Wang, S. C. Liew, and S. Zhang, "Game-theoretic analysis of mining strategy in Bitcoin-NG blockchain protocol," *CoRR*, vol. abs/1911.00900, 2019.

[37] J. Niu, C. Feng, H. Dau, Y.-C. Huang, and J. Zhu, "Analysis of nakamoto consensus, revisited," *arXiv preprint arXiv:1910.08510*, 2019.

[38] E. Upfal and M. Mitzenmacher, *Probability and computing: randomized algorithms and probabilistic analysis*. Cambridge university press, 2005, vol. 160.

APPENDIX

Lemma 4 (Chernoff bound for a sum of dependent random variables [37])**.** *Let T be a positive integer. Let $X^{(j)} = \sum_{i=0}^{n-1} X_{j+iT}$ be the sum of n independent indicator random variables and $\mu_j = E\left(X^{(j)}\right)$ for $j \in \{1, \ldots, T\}$. Let $X = X^{(1)} + \cdots + X^{(T)}$. Let $\mu = \min_j\{\mu_j\}$. Then, for $0 < \delta < 1$, $\Pr\left(X \leq (1-\delta)\mu T\right) \leq e^{-\delta^2\mu/2}$.*

Proof: Let $\bar{X} = \frac{X}{T} = \frac{1}{T}\sum_{j=1}^{T} X^{(j)}$. Then, for any $t < 0$, we have

$$\Pr\left(X \leq (1-\delta)\mu T\right) = \Pr\left(\bar{X} \leq (1-\delta)\mu\right) \leq \frac{E(e^{t\bar{X}})}{e^{t(1-\delta)\mu}}. \tag{9}$$

Note that $\exp(\cdot)$ is a convex function, we use Jensen's inequality to obtain $E(e^{t\bar{X}}) \leq \frac{1}{T}\sum_{j=1}^{T} E\left(e^{tX^{(j)}}\right)$. Hence,

$$\Pr\left(X \leq (1-\delta)\mu T\right) \leq \frac{1}{T}\sum_{j=1}^{T} \frac{E\left(e^{tX^{(j)}}\right)}{e^{t(1-\delta)\mu}} \leq \frac{1}{T}\sum_{j=1}^{T} \frac{E\left(e^{tX^{(j)}}\right)}{e^{t(1-\delta)\mu_j}}, \tag{10}$$

where the last inequality comes from the fact that $\mu_j \geq \mu$ for all j. Setting $t = \ln(1-\delta) < 0$ and following the footsteps in the proof of Theorem 4.5 in [38], we have $\frac{E\left(e^{tX^{(j)}}\right)}{e^{t(1-\delta)\mu_j}} \leq e^{-\delta^2\mu_j/2}$ for all j. Finally, we note that $e^{-\delta^2\mu_j/2} \leq e^{-\delta^2\mu/2}$ for all j and this completes the proof. ∎

Lemma 5 (Poisson tail)**.** *Let X be a Poisson random variable with rate μ (which is also its expectation). For $0 < \delta < 1$, $\Pr(X > (1+\delta)\mu) < e^{-\Omega(\delta^2\mu)}$ and $\Pr(X > (1-\delta)\mu) < e^{-\Omega(\delta^2\mu)}$.*

Proof: Recall that the moment generating function of a Poisson random variable is $E[e^{tX}] = e^{(e^t-1)\mu}$. By Markov Inequality, we have

$$\begin{aligned}\Pr(X > (1+\delta)\mu) &= \Pr(e^{tX} > e^{t(1+\delta)\mu}) \\ &< \frac{E[e^{tX}]}{e^{t(1+\delta)\mu}} = \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}}.\end{aligned} \tag{11}$$

Setting $t = \ln(1+\delta)$, the right-hand side becomes $e^{[\delta-(1+\delta)\ln(1+\delta)]\mu}$. It is not hard to show (using derivatives) that $(1+\delta)\ln(1+\delta) \geq \delta + \delta^2/3$, which completes the proof of the upper tail bound. For the other side,

$$\begin{aligned} \Pr(X < (1-\delta)\mu) &= \Pr(e^{-tX} > e^{-t(1-\delta)\mu}) \\ &< \frac{E[e^{-tX}]}{e^{-t(1+\delta)\mu}} = \frac{e^{(e^{-t}-1)\mu}}{e^{-t(1+\delta)\mu}}. \end{aligned} \tag{12}$$

Setting $t = -\ln(1-\delta)$, the right-hand side becomes $e^{[-\delta+(1-\delta)\ln(1-\delta)]\mu}$. It is not hard to show that $(1-\delta)\ln(1-\delta) \geq -\delta + \delta^2/2$, which completes the proof. ■