Computers & Security 97 (2020) 101956

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

# Assessing blockchain selfish mining in an imperfect network: Honest and selfish miner views

Runkai Yang[a,1], Xiaolin Chang[a,1,*], Jelena Mišić[b,2], Vojislav B. Mišić[b,2]

[a] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing, China
[b] Ryerson University, Toronto, Canada

ARTICLE INFO

*Article history:*
Received 10 June 2020
Accepted 9 July 2020
Available online 16 July 2020

*Keywords:*
Blockchain
Markov model
Natural fork
Selfish mining
Uncle reward

ABSTRACT

Bitcoin and Ethereum, the most famous blockchain-based cryptocurrencies, both use Proof-of-Work protocol to achieve consensus, which is vulnerable to selfish mining. The existing researches on selfish mining analysis focused on Bitcoin, or ignored the blockchain details, or only investigated mining revenue without considering system performance and security.

This paper aims to quantitatively evaluate the influence of selfish mining in an imperfect blockchain network from the perspective of honest miners (system performance and security) and selfish miners (selfish mining revenue ratio). We develop a novel Markov model to capture the behaviors of honest and selfish miners in both Bitcoin and Ethereum. Our model can also capture natural forks (occurring due to block propagation delay) and the varying distance between uncle and nephew blocks in Ethereum. Formulas are derived to calculate mining revenue, system performance and security metrics. The proposed model and metric formulas are validated by (1) comparing our numerical with simulation results, and (2) comparing our numerical results with the existing work results. Numerical analysis is carried out to investigate selfish mining impact over diverse parameters. These quantitative results can help detect whether there are any selfish miners in the system, help design blockchain reward mechanisms and enhance security.



## 1. Introduction

BITCOIN (Nakamoto and Bitcoin, 2008) and Ethereum (Buterin, 2014) are the representatives of blockchain-based cryptocurrency. Bitcoin is the most popular and valuable cryptocurrency, and its market cap dominance is more than 60% in May 2020 (https://coinmarketcap.com/charts/ 2020). Ethereum, known as the representative of blockchain 2.0, is the largest and most active blockchain platform in the world. There are a large number of core protocol developers, fortune 500 companies, mining organizations, and Ether (the native cryptocurrency of Ethereum) holders in Ethereum community (https://ethereum.org/what-is-ethereum/ 2020).

Bitcoin and Ethereum both use Proof-of-Work (PoW) to reach consensus (Chicarino et al., 2020). In a PoW system, people or organizations attempt to generate a new block by solving a math puzzle. When a block is produced, based on the new block, all participants start to solve a new math puzzle. The above process is known as 'mining blocks', and the people or organizations, who take part in the mining, are known as 'miners'. The essence of mining is solving a hash function, so the computing power for mining is called 'hash power'. In both Bitcoin and Ethereum, the first miner who successfully produces a new regular block (the block in the main chain) is entitled to rewards. To increase the chance of generating a block, several miners form a mining pool and then work together to mine blocks and share their revenues when one of the pool members generates a block (Pachal and Ruj, 2019).

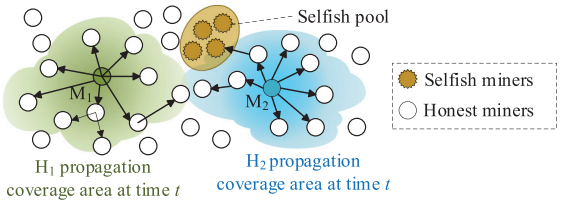
In an imperfect network, it takes time to propagate information from one node to the other nodes. Blockchain miners are all over the world and some miners can be in a bad network connection, leading to large network latency. In a blockchain network, more than one block can be generated and propagated nearly simultaneously. Therefore, the other miners can receive these blocks in a different order, and miners only validate the block they receive firstly. In this case, an inconsistency occurs, and a fork is created. The fork

* Corresponding author.
*E-mail address:* bjtuxiaolin@126.com (X. Chang).

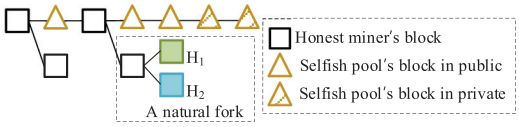
[1] The research of the first two authors was supported by the National Natural Science Foundation of China under Grant U1836105 and the Fundamental Research Funds for the Central Universities (Grant No. 2019YJS034).

[2] The research of Jelena Mišić and Vojislav Mišić was supported in part by the National Science and Engineering Research Council of Canada (NSERC) through Discovery Grants.

https://doi.org/10.1016/j.cose.2020.101956
0167-4048/




a) Our system consists of selfish and honest miners, and honest miners can create natural forks due to block propagation delay.




b) Selfish miners publish blocks strategically, fooling honest miners to mine stale blocks and waste power.

**Fig. 1.** Blockchain forking and selfish mining.

caused by block propagation delay is called a natural fork, which must exist in an imperfect blockchain network. See an example shown in Fig. 1a (Shahsavari et al., 2019). A green miner ($M_1$) and a blue miner ($M_2$) independently produce a new block, denoted as $H_1$ and $H_2$, respectively. Assume that $H_1$ and $H_2$ are produced and propagated almost at the same time. At time $t$, some miners receive $H_1$ firstly and validate it, but others validate $H_2$. Thus, a natural fork with two branches ($H_1$ and $H_2$) is created (depicted in Fig. 1b). The time intervals of generating a block in Bitcoin and Ethereum are about 10 minutes and 13 seconds, respectively (etherchain.org 2020). Therefore, forks are created more frequently in Ethereum, about 6.6% (etherchain.org 2020). Thus, Ethereum adopts a modified version of GHOST protocol (Sompolinsky and Zohar, 2015) which has a more complex reward mechanism. If there is a fork occurrence, Bitcoin and Ethereum both apply the longest rule to choose the main chain (Wang et al., 2019; Grunspan and Pérez-Marco, 2019a; Grunspan and Pérez-Marco, 2019; Bai et al., 2019; Göbel et al., 2016; Feng and Niu, 2019; Kasireddy, 2017).

Blockchain is exposed to various security threats. Blockchain-based cryptocurrencies, such as Bitcoin Gold, ZenCash, Zcash and Litecoin Cash, suffered from double-spending attacks, and millions of US dollars were lost in 2018 (Jang and Lee, 2019). Ethereum Classic, which is a hard fork of the original Ethereum, suffered selfish mining in 2019 (https://www.cryptoglobe.com/latest/2019/01/ethereum-classic-etc-network-s-hashpower-consolidation-is-not-51-attack-developers-claim/ 2020). Eyal and Sirer, (2014) first proposed the selfish mining attack in Bitcoin. Selfish miners get unfair revenue by keeping the new block privately for a while instead of publishing it immediately. For example, several selfish miners form a selfish pool (Fig. 1a), and the pool has produced four blocks but only publishes two blocks (Fig. 1b), which can prevent honest miners from mining blocks on the last block and then waste the power of honest miners. If a blockchain system is attacked by selfish mining, honest miners get less mining revenue than they deserve. Thus, honest miners are willing to participate selfish mining, and the decentralization of blockchain is destroyed.

There existed several studies on blockchain selfish mining analysis. However, these researches only focused on selfish mining in Bitcoin (Eyal and Sirer, 2014) and (Bai et al., 2019; Göbel et al., 2016), or ignored natural forks (Grunspan and Pérez-Marco, 2019; Grunspan and Pérez-Marco, 2019a; Bai et al., 2019; Göbel et al., 2016; Feng and Niu, 2019), or assumed that the distance between uncle and nephew blocks (which is denoted as 'reference distance' in this paper) is a constant, which is 1, in Ethereum (Wang et al., 2019; Grunspan and Pérez-Marco, 2019a HYPERLINK \l "bib29" , 2019b), or only evaluated the revenue for selfish miners without investigating performance and security (Eyal and Sirer, 2014; Wang et al., 2019; Grunspan and Pérez-Marco, 2019a; Grunspan and Pérez-Marco, 2019b; Bai et al., 2019; Göbel et al., 2016; Feng and Niu, 2019). However, natural forks are inevitable and do affect blockchain mining, especially in Ethereum. In addition, the reference distance is varying instead of being always equal to one when blocks are produced. Furthermore, selfish mining not only makes selfish miners get unfair revenue but also disrupts the system performance and security.

In this paper, we consider a system which consists of honest and selfish miners (defined in Sec. III-A). We develop an analytical model to quantitatively study blockchain selfish mining in an imperfect network. Then, we derive formulas to evaluate the impact of selfish mining on two typical blockchain systems (Bitcoin and Ethereum), from the perspective of honest miners (system performance and security) and selfish miners (selfish mining revenue ratio), respectively.

Our major contributions are summarized as follows.

- The system studied in this paper is more realistic than that in the existing works. The system is in an imperfect network, so natural forks can be created by honest miners. In addition, the reference distance can vary with the blocks produced. We develop a Markov model to capture the behaviors of honest and selfish miners in the system. Our model also captures natural forks and varying reference.
- We derive formulas for calculating the revenue of honest and selfish miners in both Bitcoin and Ethereum. Ethereum mining revenue consists of static block reward, uncle reward and nephew reward. Bitcoin mining revenue consists of only static block reward. The formula for calculating each reward is different for honest and selfish miners. Furthermore, based on our model, we compare two uncle block reference strategies in Ethereum, and the formulas for calculating the revenues in different reference strategies are different. These formulas can help honest miners to determine the selfish mining revenue ratio (defined in Sec. III-C) and set the threshold (it is used to limit selfish mining hash power to gain unfair revenue) and design mining reward mechanism under different situations in Bitcoin and Ethereum, respectively.
- We derive formulas for calculating several performance and security metrics, including the stale block ratio, transactions per second, probability of double-spending success, and so on. To the best of our knowledge, we are the first to quantify the impact of selfish mining on blockchain performance and security by modeling the system. These metrics can offer the reference for honest miners to detect whether there exist selfish miners in the blockchain system and to estimate the hash power of the selfish pool. The security analysis can help to enhance blockchain security.

The rest of the paper is organized as follows. Section II presents related works. In Section III, we describe the behaviors of honest miners and selfish pool, respectively. Then, we present the model and metric formulas. Section IV shows the numerical and simulation results. Section V concludes the paper and then talks about future work.

## 2. Related work

Blockchain security has been a concerning problem since the birth of blockchain. There are multiple kinds of attacks threatening blockchain. Rot and Blaicke, (2019) summarized blockchain threats into five types: double-spending, mining threat, wallet threat, network threat, and smart contract threat. Zhang and Lee, (2019) proposed a double-spending attack combining with Sybil attack on

**Table 1**
Comparison of the state of the art for selfish mining analysis models.

| Work | Blockchain system | Method | Distance | Natural fork | Metric |
|---|---|---|---|---|---|
| (Eyal and Sirer, 2014) | Bitcoin | State machine | - | No | Mining revenue |
| (Wang et al., 2019) | Bitcoin and Ethereum | Markov state machine | Constant | Yes | Mining revenue |
| (Grunspan and Pérez-Marco, 2019) | Ethereum | Random walk | Constant | No | Mining revenue |
| (Grunspan and Pérez-Marco, 2019a) | Bitcoin and Ethereum | Random walk | Constant | No | Mining revenue |
| (Bai et al., 2019) | Bitcoin | Markov chain model | - | No | Mining revenue |
| (Göbel et al., 2016) | Bitcoin | Markov chain model | - | No | Mining revenue, orphan block rate |
| (Feng and Niu, 2019) | Ethereum | Markov chain model | Varying | No | Mining revenue |
| Our model | Bitcoin and Ethereum | Markov chain model | Varying | Yes | Mining revenue, performance, security |

Bitcoin, which can increase the probability of attack success with lower hash power. Volety et al., (2019) successfully attacked two Bitcoin wallet software by offline brute force password attempt and can own the Bitcoins. Shurov et al., (2019) conducted a simulation of the Ethereum in an unreliable network and observed that a miner with 26% hash power can control the whole Ethereum network, if the network were down for 10 hours. Feng et al., (2019) presented a system in which adversarial contracts can be synthesized automatically, and adversarial contracts were able to identify and exploit vulnerabilities in a victim smart contract. In this paper, we focus on selfish mining attack which is a kind of mining threats, and we aim to quantitatively study the impact of selfish mining.

There exists latency in a blockchain network, and forks can occur. There existed diverse researches on the analysis of blockchain forks. Misic et al., (2019) introduced the characteristics of forks and the process after a fork was created in Bitcoin. They further evaluated the network performance and the forking probability in (Mišić et al., 2019a) and (Misic et al., 2019b). Shahsavari et al., (2019) developed a random graph model to analyze the influence of system parameters on the natural fork in Bitcoin. The authors (da Silva et al., 2019) proposed a model to study how communication delay impacted the fork in Ethereum. Wang et al., (2019) studied the vulnerability of the blockchain network, which was triggered by intentional forks. To reduce the probability of forking, Liu et al., (2019) designed a scheme to reduce the block propagation delay. The fork is inevitable in blockchain, so it is necessary to consider natural forks when we research selfish mining. On the other hand, our results show that the more the natural forks, the more easily selfish miners get unfair revenue and disrupt system performance and security. Thus, our work can be complementary to these works to establish a healthy blockchain.

Multiple blockchain mining threats have been researched, and researchers have proposed several attack methods to get more revenue with less hash power. Eyal and Sirer, (2014) first presented the selfish mining in Bitcoin, and they discovered that the threshold which made selfish miners get unfair revenue was 25% of the total hash power. Nayak et al., (2016) modeled Bitcoin mining by using a Markov decision process, and they expanded selfish mining to stubborn mining. Dong et al., (2019) proposed an attack which was a combination of selfish mining attack and block withholding attack. Saad et al., (2019) introduced a kind of blockchain selfish mining attacks and showed the profit margins of the top six cryptocurrencies. In our paper, instead of proposing a new selfish mining strategy, we aim to quantify the influence of selfish mining by using an analytical model. Our model can be extended to analyze these selfish mining strategies.

There are researches about selfish mining analysis in the blockchain. Wang et al., (2019) used a Markov state machine to analyze the selfish mining in a perfect network and in an imperfect network. Different from (Wang et al., 2019), natural forks in our system are caused by two miners creating their blocks nearly simultaneously (Shahsavari et al., 2019; da Silva et al., 2019; Wang et al., 2019; Liu et al., 2019), instead of some miners creating blocks on an old block. Besides, the reference distance is varying in our work, which enables our model to describe a more realistic system. Grunspan and Pérez-Marco, (2019a, 2019b) showed that selfish mining attacks can affect the difficulty adjustment in Bitcoin and Ethereum. However, they assumed that the reference distance in Ethereum is a constant, which is equal to 1. Bai et al., (2019) analyzed the system with two adversaries who attacked Bitcoin by selfish mining through a Markov model. Bai's model only considered the scenario and selfish mining in Bitcoin, and the model was too complex to consider the situation that the selfish miners create a branch with more than four blocks. Based on Eyal and Sirer, (2014), Göbel et al., (2016) evaluated the effect of the probability that honest miners work on the selfish miner's block in Bitcoin. Feng and Niu, (2019) developed a model to evaluate the threshold of making selfish mining profitable in Ethereum, but they ignored natural forks. Ritz and Zugenmaier, (2018) designed a simulation experiment to evaluate the selfish mining in Ethereum. Different from these papers above, our work aims to model Bitcoin and Ethereum which are attacked by selfish mining, and we consider natural forks. Furthermore, to get accurate results, the reference distance in our model is not constant, and its value can be calculated based on our model. Moreover, we evaluate the impact of selfish mining on system performance and security.

Table 1 shows the comparison of the related works. 'Blockchain system' in Table 1 denotes the system to be investigated. 'Method' denotes the corresponding research method. 'Distance' denotes whether the reference distance is a constant or not in the work when the authors analyzed Ethereum. 'Natural fork' denotes whether the work considers natural forks caused by honest miners. 'Metric' indicates what metrics are calculated in the work.

## 3. System description and model

This section first describes the system to be considered and introduces the behaviors of honest miners and the selfish pool in Section III-A. Then we present the model in Section III-B and the formulas in Section III-C.

### 3.1. System description

In the system we consider, there are a large number of miners. Part of them adopt honest mining protocol (namely, they are honest miners). The left miners form a mining pool and follow the selfish mining protocol to get undeserved revenue, which are denoted as selfish miners and selfish pool. We assume that all miners keep mining blocks and try their best to get the maximum revenue.

In Ethereum mining, uncle blocks have to be considered. Uncle block is a kind of stale blocks which are not in the main chain. If a block is the direct child of a regular block, and it is referenced by a regular block, the block is an uncle block. In Ethereum, an uncle block can only be referenced by one regular block, and a regular

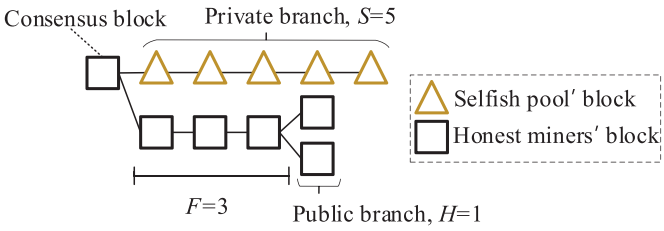



**Fig. 2.** An example to illustrate the definitions of *S, H, F* and consensus block.

block can reference up to two uncle blocks. The miner of the uncle block receives uncle rewards, according to the reference distance.

We refer to the last block which is validated by both honest miners and the selfish pool as a consensus block. Let $S$ be the length of the branch created by the selfish pool, which is called a **private branch.** Let $H$ be the length of the branch after the natural fork or the consensus block, which is called a **public branch**. We define $F$ to denote the natural fork position which is the distance between the consensus block and the last natural fork created by honest miners. $F = 0$ denotes that the natural fork is created on the consensus block, and $F = 0'$ denotes the honest miners do not make a natural fork. We define $\Delta = S - (H + F)$, where $H + F$ is the distance between the consensus block and the last block of honest miners. Fig. 2 is an example to illustrate these definitions.

#### 3.1.1. Honest miner behaviors

Honest miners comply with the mining protocol. They propagate a block as soon as they generate it and mine blocks in the longest branch as they know. In our system, if an honest miner creates two blocks nearly simultaneously, a natural fork is created. If there is a fork, honest miners select the branch they receive firstly and then mine blocks in it. Honest miners can gain revenue which depend on their hash power. In Ethereum, honest miners reference uncle blocks as many as they can (no more than two) to get maximum revenue.

The probability that honest miners create a natural fork with three branches is lower than 0.0001 in Bitcoin (Misic et al., 2019). In Ethereum, only the direct child of the regular block can be referenced as an uncle block, and a regular block can reference up to two blocks. For these reasons, we assume that honest miners can only create a natural fork with two branches and ignore the case that honest miners produce blocks in different branches simultaneously.

#### 3.1.2. Selfish miner behaviors

A group of selfish miners form a mining pool, and they share hash power and mining revenue. The pool is a professional mining organization and has a well-connected network, and it is maintained by administrators. Thus, the pool does not create natural forks, and always mines blocks on the last block. Different from the strategy taken by honest miners, when the selfish pool produces a block, the pool publishes block strategically instead of publishing it at once.

Based on Eyal and Sirer, (2014), the selfish mining strategy is adjusted to the situation that honest miners can create natural forks. The strategy that the selfish pool takes in our paper is given in Algorithm 1. If $\Delta \geq 0$, the selfish miners mine the block in the private branch. If $\Delta < 0$, the selfish miners mine blocks in the public branch. By using selfish mining strategy, the selfish pool can take the unfair lead, and the selfish pool can get more revenue than they deserve.

**Algorithm 1** A selfish mining strategy.

```
1   If the selfish pool produces a block
2     S = S+1
3     if S == 1,S+H == 1
4       mine blocks on the selfish block
5     if S == 2, S+H == 1
6       publish the private branch; the consensus is achieved
7       all miners mine blocks on the consensus block
8     Else
9       keep mining on the new block
10
11  if honest miner produces a block
12    H=H+1
13    if S < H
14      if a fork is generated
15        S, H, F = 0, 1, 0
16        choose a branch and mine on it
17      Else
18        S, H, F = 0, 0, 0'
19        all miners mine blocks on the consensus block
20    else if Δ== 0
21    if a fork is generated
22        S, H, F = 1, 1, 0
23    Else
24        S, H, F = 1, 1, 0'
25    else if Δ== 1
26      publish the private branch, and S, H, F = 0, 0, 0'
27      all miners mine blocks on the consensus block
28    else if Δ> 1
29      if the new block is on the private branch
30      if a fork is generated
31        S, H, F = S-H-F+1, 1, 0
32      Else
33        S, H, F = S-H-F+1, 1, 0'
34    else if the new block is on the public branch
35      if a fork is generated
36        S, H, F = S, 1, H-1
37      Else
38        S, H, F = S, H, 0'
```

### 3.2. System model

The time of generating a block can be approximated by an exponential distribution (Eyal and Sirer, 2014; Bai et al., 2019; Göbel et al., 2016; Feng and Niu, 2019). Without loss of generality, we fix the total hash power of a blockchain system as 1, and let $\alpha$ and $\beta$ be the hash power controlled by the selfish pool and honest miners. $\theta$ denotes the probability that honest miners create a natural fork. We normalize the block generation time as 1 on average. Namely, the average rate of generating a block is set to 1. Thus, we re-define $\alpha$ and $\beta$ as the block generation rate of the selfish pool and honest miners, respectively. $\beta_f$ denotes the rate of creating a natural fork, and $\beta_f = \beta\theta$. As creating a fork means that two blocks are generated, the block generation rate in the case of creating a natural fork is $2\beta_f$. The total block generation rate of honest miners is $\beta$, so the rate of honest miners generating a single block is $\beta_s = \beta - 2\beta_f$. Note that $\alpha + 2\beta_f + \beta_s = 1$. $\beta_e$ denotes the rate that honest miners produce an effective block, and$\beta_e = \beta_s + \beta_f$. Effective blocks are the blocks that can increase the length of the public branch or decrease the difference between the length of the public branch and the private branch. In other words, $\beta_e$ denotes the rate that honest miners increase the length of the public chain.

The state of our model is described by (*S, H, F*). For example, (1, 1, 0)denotes that the length of the private branch and the honest branch are both 1 ($\Delta = 0$), and the honest miners create a natural fork on the consensus block. (4, 1, 1) denotes that there are four blocks in the private branch. The natural fork position is 1, and the length of the public branch after the natural fork is 1. $\Delta = 2$ when the system is in the state (4, 1, 1). As described in Algorithm 1, the selfish pool keeps the last two blocks secret when the system is in

**Table 2**
Definitions of notations.

| Notation | Definition |
|---|---|
| $S$ | The length of the private branch. |
| $H$ | The length of the public branch after the last natural fork or the consensus block. |
| $F$ | The natural fork position. |
| $\alpha$ | The rate of the selfish pool mining a block, used to represent the hash power controlled by the selfish pool. |
| $\beta$ | The rate of honest miners mining a block, used to represent the hash power controlled by honest miners. |
| $\beta_s/\beta_f$ | The rate of honest miners generating a single block/a natural fork. |
| $\beta_e$ | The rate of honest miners effectively increasing the public branch. |
| $\theta$ | The probability of natural fork occurrence. |
| $\gamma_i$ | The probability that honest miners mine a block in the private branch when the branch number of the fork is $i$. |
| $R_{S,S}/R_{S,H}$ | The static block rewards for the selfish pool/honest miners. |
| $R_{U,S,i}/R_{U,H,i}$ | The uncle rewards for the selfish pool/honest miners when the selfish pool adopts strategy $i$. |
| $R_{N,S,i}/R_{N,H,i}$ | The nephew rewards for the selfish pool/honest miners when the selfish pool adopts reference strategy $i$. |
| $R_u(d)/R_n(d)$ | The uncle reward/nephew reward for a block when the reference distance is $d$. |
| $N_{main}$ | The number of regular blocks. |
| $N_{total}$ | The total number of blocks are produced. |
| $N_{uncle}$ | The number of uncle blocks. |
| $R_{stale}/R_{uncle}$ | The stale/uncle block ratio. |
| TPS | Transactions per second (TPS) is the number of transactions executed per second. |
| $p_\alpha/p_\beta$ | The probability that the selfish pool/honest miners produce a block. |
| $p_s/p_f$ | The probability that honest miners create a single block/a fork. |
| $P_{ds}$ | The probability of double-spending success. |

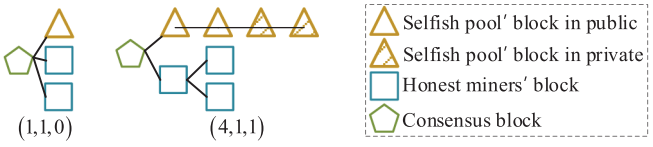



**Fig. 3.** Examples to illustrate the system state.

the state (4, 1, 1). The two cases described by (1, 1, 0) and (4, 1, 1) are illustrated as Fig. 3.

The notations are defined in Table 2. Note that the value of $H$ must be equal to 1 when a natural fork is created, or the state is useless. If the length of the public branch is longer than the private branch, the pool publishes the private branch and mines blocks in the public branch, or the state is useless. Thus, if $S > 0$, $S$ is no less than $H + F$.

We model the behaviors of honest and selfish miners. There are two types of events in our model, and according to the system state, each type of event includes multiple cases. The state transition is shown in Fig. 4. The details of state transition are given in Appendix A. By these transition rules, a transition matrix is constructed. Then, we can get the steady-state probability of every system state, denoted by $\pi_{(S,H,F)}$. The steady-state probabilities of useless states are equal to 0.

### 3.3. Metric formulas

According to our model, we first evaluate the selfish mining revenue ratio of the total mining revenue. Then, we analyze the impact of selfish mining on the system performance. Finally, we analyze the effect of selfish mining on blockchain from the perspective of system security. Let $p_\alpha$ and $p_\beta$ be the probabilities that blocks are mined by the selfish pool and honest miners. $p_s$ and $p_f$ denote the probabilities that honest miners create a single block or create a fork, respectively. Note that $p_\alpha = \alpha$, $p_\beta = \beta$, $p_s = \beta \cdot (1-\theta)$ and $p_f = \beta \cdot \theta$.

#### 3.3.1. Selfish mining revenue

Bitcoin and Ethereum have different mining reward mechanisms. Bitcoin miners can only get one type of rewards when they produce a regular block, namely, static block reward. However, in Ethereum, there are three types of mining rewards, namely, static block reward, uncle reward, and nephew reward. In this paper, we consider two reference strategies of the selfish pool in Ethereum as follows.

- Strategy 1 (**S1**): The selfish pool references as many uncle blocks as it can.
- Strategy 2 (S2): The selfish pool does not reference any blocks.

As the rewards for miners change over some time, without loss of generality, we normalize a static block reward as 1. The formulas of mining rewards are proved in Appendix B. Eq. (1) and Eq. (2) give the formulas of calculating static block rewards for the selfish pool and honest miners, denoted by $R_{S,S}$ and $R_{S,H}$, respectively.

$$R_{S,S} = p_\alpha\left(1-\pi_{(0,0,0')}\right) + p_\alpha\pi_{(0,0,0')}\left(p_\alpha + p_\alpha p_\beta + p_f p_\beta \gamma_3 + p_s p_\beta \gamma_2\right) \quad (1)$$

$$R_{S,H} = p_\beta\left(\pi_{(0,0,0')} + \pi_{(0,1,0)} + \pi_{(1,1,0')} + \pi_{(1,1,0)}\right) + \left(p_\beta p_s(1-\gamma_2) + p_\beta p_f(1-\gamma_3)\right)\pi_{(1,0,0')} \quad (2)$$

By Eq. (1) and Eq. (2), we can evaluate the mining revenue in Bitcoin. To quantitatively analyze the selfish mining in Ethereum, we derive the formulas for calculating uncle rewards and nephew rewards.

Uncle reward is a specific mining reward for miners who find uncle blocks in Ethereum. $R_{U,S}$ and $R_{U,H}$ denote the uncle rewards of the selfish pool and honest miners, respectively. $R_U(d)$ given in Eq. (3) is the uncle reward function, representing the uncle reward for a miner who produces an uncle block which is referenced by a regular block, and the reference distance is $d$.

$$R_U(d) = \begin{cases} (8-d)/8, & 1 \le d \le 6 \\ 0, & others \end{cases} \quad (3)$$

We can calculate the uncle rewards of the selfish pool which applies **S1** by Eq. (4).

$$R_{U,S,1} = R_U(1)\left(p_\beta(1-\gamma_2)\pi_{(1,1,0')} + p_\beta(1-\gamma_3)\pi_{(1,1,0)}\right) \quad (4)$$

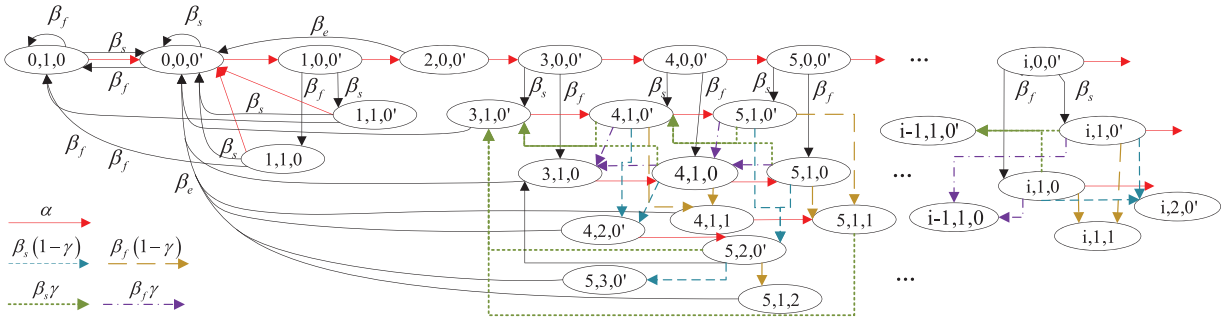



**Fig. 4.** State transition.

Eq. (5) gives the formula for uncle rewards of honest miners, where $k \neq 0'$.

$$\begin{aligned} R_{U,H,1} &= R_U(1)p_f\pi_{(0,0,0')} + R_U(1)\left(p_\alpha + p_\beta\gamma_2\right)\pi_{(1,1,0')} \\ &+ \left(2R_U(1)p_\alpha + R_U(1)p_\beta(1+\gamma_3)\right)\pi_{(1,1,0)} \\ &+ \sum_{i=2}^{\infty}\left(R_U(i)\left(p_s+2p_f\right)\pi_{(i,0,0')}\right) \\ &+ \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\left(R_U(i)\left(p_s+2p_f\right)\gamma_3\pi_{(i+j+k,j,k)}\right) \\ &+ \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\left(R_U(i)\left(p_s+2p_f\right)\gamma_2\pi_{(i+j,j,0')}\right) \end{aligned} \tag{5}$$

Nephew reward is a type of rewards which is used to encourage miners to reference uncle blocks in Ethereum. The function of nephew rewards in the current Ethereum version is given as Eq. (6), where $d$ denotes the reference distance.

$$R_n(d) = \begin{cases} 1/32, & 1 \le d \le 6 \\ 0, & others \end{cases} \tag{6}$$

Eq. (7) is the formula for computing nephew rewards of the selfish pool which applies **S1**. Here, $k \neq 0'$, and $\chi = p_\alpha - p_\alpha p_\beta(p_s(1-\gamma_2) + p_f(1-\gamma_3))$.

$$\begin{aligned} &R_{N,S,1} = R_n(1)\left(\left(p_s p_\alpha + 2p_f p_\alpha\right)\pi_{(1,0,0')} + p_\alpha\pi_{(0,1,0)}\right) \\ &+\sum_{i=2}^{\infty}\left(R_n(i)p_\beta^{i-2}\left(p_s+2p_f\right)\chi\cdot\pi_{(i,0,0')}\right) \\ &+\sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\left(R_n(i)p_\beta^{i-2}\gamma_3\left(p_s+2p_f\right)\chi\cdot\pi_{(i+j+k,j,k)}\right) \\ &+\sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\left(R_n(i)p_\beta^{i-2}\gamma_2\left(p_s+2p_f\right)\chi\cdot\pi_{(i+j,j,0')}\right) \end{aligned} \tag{7}$$

We now show the formula of the nephew rewards for honest miners, when the selfish pool applies **S1** is given in Eq. (8). Here, $k \neq 0'$.

$$\begin{aligned} &R_{N,H,1} = R_n(1)p_\beta\left(2\pi_{(1,1,0)} + \pi_{(0,1,0)} + \pi_{(1,1,0')}\right) \\ &+\sum_{i=2}^{\infty}\left(R_n(i)\left(p_s+2p_f\right)p_\beta^{i-1}\pi_{(i,0,0')}\right) \\ &+\sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0,0'}^{\infty}\left(R_n(i)\left(p_s+2p_f\right)\gamma_3 p_\beta^{i-1}\pi_{(i+j+k,j,k)}\right) \\ &+\sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\left(R_n(i)\left(p_s+2p_f\right)\gamma_2 p_\beta^{i-1}\pi_{(i+j,j,0')}\right) \end{aligned} \tag{8}$$

Uncle rewards for the selfish pool adopting **S2** are the same as the reward that the selfish pool can gain when it uses **S1**, because honest miners still reference blocks as many as possible, including the selfish pool's uncle blocks. Namely, $R_{U,S,2} = R_{U,S,1}$.

The uncle reward for honest miners when the selfish pool adopts **S2** is shown in Eq. (9).

$$R_{U,H,2} = R_{U,H,1}\cdot p_\beta \tag{9}$$

The selfish pool nephew reward is zero when the pool does not reference blocks (**S2**). Namely, $R_{N,S,2} = 0$. The selfish pool reference strategy does not impact the number of honest miners referencing blocks. Thus $R_{N,H,2} = R_{N,H,1}$.

According to Eqs. (1) - (9), we can evaluate the selfish pool total mining revenue $R_{p,\ total}$, and the total mining revenue $R_{total}$ in a block generation time.

$$R_{p,total} = R_{S,S} + R_{U,S} + R_{N,S}$$

$$R_{total} = R_{S,S} + R_{U,S} + R_{N,S} + R_{S,H} + R_{U,H} + R_{N,H}$$

Relative Revenue Ratio (*RRR*) and Absolute Revenue Ratio (*ARR*) are metrics that have been widely used for selfish mining evaluation. *RRR* aims to show the ratio of the revenue for the selfish pool of the total revenue. *ARR* focuses on the selfish mining revenue ratio during the process of creating a regular block. The formulas for calculating *RRR* and *ARR* are given as Eq. (10) and Eq. (11), respectively.

$$RRR = \frac{R_{p,total}}{R_{total}} \tag{10}$$

$$ARR = \frac{R_{p,total}}{R_{S,S} + R_{S,H}} \tag{11}$$

### 3.3.2. The impact of selfish mining on blockchain performance

Based on our model, we evaluate the influence of selfish mining on blockchain performance. Forking is very common in the blockchain system, but selfish mining results in more unnatural forks. On the other hand, selfish mining can cause hash power waste and create a large number of stale blocks, which further disrupts blockchain system performance.

*3.3.2.1. Stale block ratio.* Stale block occurrence means that some hash power is wasted. We divide stale blocks into two categories, natural stale blocks and intended stale blocks. Natural stale blocks are generated due to block propagation delay. Intended stale blocks are made by selfish mining. The total number of blocks generated is denoted as $N_{total}$, and $N_{total}$=$\alpha + \beta$ in a block generation time. $N_{main}$ is the number of blocks in the main chain. As the necessary and sufficient condition of a miner getting a static block reward is generating a block in the main chain, the static block reward is equal to the number of blocks in the main chain. Namely, $N_{main} = R_{S,S} + R_{S,H}$. Eq. (12) describes the formula for the stale block ratio $R_{stale}$.

$$R_{stale} = (N_{total} - N_{main})/N_{total} \tag{12}$$

If there is no selfish miner in the blockchain system, all stale blocks are natural stale blocks, and the stale block ratio is equal to the probability of creating natural forks. Namely, $R_{stale} = \theta$ when $\alpha = 0$.

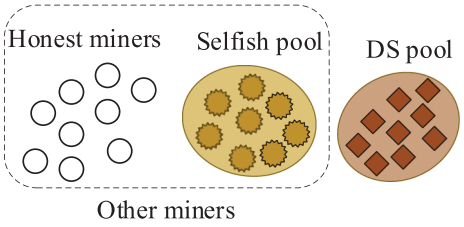



**Fig. 5.** From the perspective of double-spending attacks, the total hash power is divided into two parts, controlled by the DS pool and controlled by the other miners which consist of honest miners and a selfish pool.

*3.3.2.2. Uncle block ratio.* In Ethereum, uncle block ratio is an important metric. The uncle reward is the product of the number of uncle blocks and the uncle reward function $R_U(d)$ in each case. Thus, if $R_U(d) = 1$, the number of uncle blocks is equal to the total uncle rewards for all miners. The uncle block ratio is given in Eq. (13).

$$R_{uncle} = (R_{U,S,i} + R_{U,H,i})/N_{total} \tag{13}$$

*3.3.2.3. Transaction per second (TPS).* When a block is produced, a number of transactions are recorded in the blockchain. TPS is an important performance metric in cryptocurrency. In Bitcoin, only the transactions in regular blocks can be recorded in the blockchain. However, in Ethereum, if there is no conflicting transaction in the uncle block, the transactions in both regular blocks and uncle blocks are recorded in the blockchain. $T_{block}$ denotes the block generation time, and $N_t$ denotes the number of transactions in a block. TPS is given in Eq. (14).

$$TPS = \begin{cases} (1 - R_{stale}) \cdot N_t / T_{block}, & \text{Bitcoin} \\ (1 - R_{stale} + R_{uncle}) \cdot N_t / T_{block} & \text{Ethereum} \end{cases} \tag{14}$$

*3.3.3. The impact of selfish mining on blockchain security*

Double-spending is one of the most important challenges in cryptocurrency. It existed before Bitcoin was born. This paper evaluates the impact of selfish mining on the ability of blockchain resisting double-spending. We assume that part of miners forms a malicious pool (a DS pool) to launch a double-spending attack when the system suffers from selfish mining attack. In this scenario, the system consists of three kinds of miners, honest miners, selfish miners and double-spending miners. Namely, there are two malicious pools in the system, a selfish pool and a DS pool. These two pools do not know the existence of each other. From the perspective of double-spending attacks, the total hash power is divided into two parts, controlled by the DS pool and controlled by the other miners. The DS pool is a mining pool aiming to launch a double-spending attack, and the other miners consist of a selfish pool and honest miners. The system in this scenario is shown in Fig. 5.

Let $p$ and $q$ denote the probability that a block is mined by the other miners and the DS pool, respectively. The DS pool creates a branch and increases it secretly, and it is immune to selfish mining attack. Thus, the probability that the DS pool's branch increases one block is $q$. However, the block mined by the other miners may not increase the branch length because of the natural fork and selfish mining. Eq. (15) describes the probabilities that the DS pool generates a block, and the other miners increase their branches, denoted as $q'$ and $p'$, respectively.

$$p' = \frac{p(1 - R_{stale})}{p(1 - R_{stale}) + q}, \quad q' = \frac{q}{p(1 - R_{stale}) + q}, \tag{15}$$

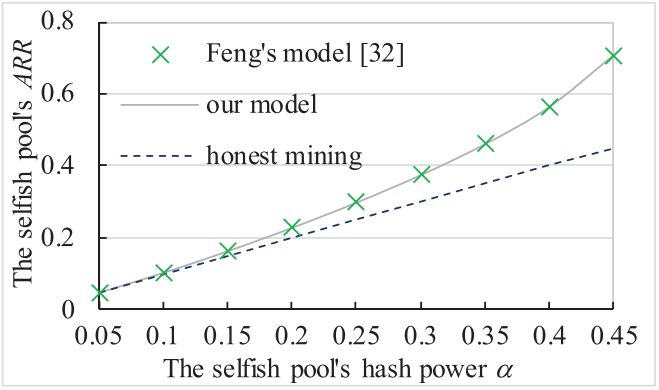



**Fig. 6.** Verify that Feng model (Feng and Niu, 2019) is a special case of our model.

Eq. (16) describes the formula for calculating the probability of double-spending success ($P_{ds}$) in (Rosenfeld, 2014).

$$P_{ds} = \begin{cases} 1 - \sum_{m=0}^{n-1} \binom{m+n-1}{m} \left(p'^n q'^m - p'^m q'^n\right), & if\ q' < p' \\ 1, & if\ q' \geq p' \end{cases} \tag{16}$$

## 4. Analysis results

This section shows the analysis results of selfish mining in Bitcoin and Ethereum based on our model. We first verify our model and metric formulas by comparing other research works and our simulation experiments, which are shown in Figs. 6–8. Then we use numerical analysis to evaluate mining revenue for the selfish pool, the system performance and security, which are given in Figs. 9–16. In our result figures, '-S1' and '-S2' suffixes denote that the selfish pool adopts **S1** and **S2** (which are described in Sec. III-C), respectively. '-sim' suffix denotes simulation results. 'The selfish pool's hash power' denotes that the selfish pool hash power ratio of the total hash power. Both simulation and numerical analysis are conducted by using Maple 18 (https://www.maplesoft.com/ 2020).

### 4.1. Verification of model and metric formulas

This section presents experiment results to ensure the correctness and accuracy of our model and metric formulas. Our model and formulas are verified from threefold: ① We reproduce the existing model of Feng (Feng and Niu, 2019), which is a special case of our model. ② We compare the results of our model and an existing simulation (Ritz and Zugenmaier, 2018). ③ We verify our model by comparing a simulation.

We design a simulation to verify our model in different parameters. There are two types of miners who are selfish miners and honest miners in the simulation. The honest miners adopt the honest mining strategy, and they may create natural forks because of the block propagation delay. The selfish miners are organized in a selfish pool with a perfect network connection. Moreover, the pool can adopt two reference strategies, respectively, as described in Sec. III. In each simulation experiment, 100,000 blocks are generated in the main chain, and the experiments are run ten times. The results are averages of 10 times of simulation experiments.

The model of Feng and Niu, (2019) ignored the natural fork in Ethereum. In other words, if the probability of natural fork occurrence of our model is set to 0, namely $\theta = 0$, we can get the same results as in Feng's model. Fig. 6 shows the results of Feng's model and the results of our model in no natural fork scenario. As shown in Fig. 6, our model's results fall in line with the results of Feng's model. According to this experiment, we can verify our model. We further get more evaluation results by changing the probability of natural fork occurrence ($\theta$) in our model.

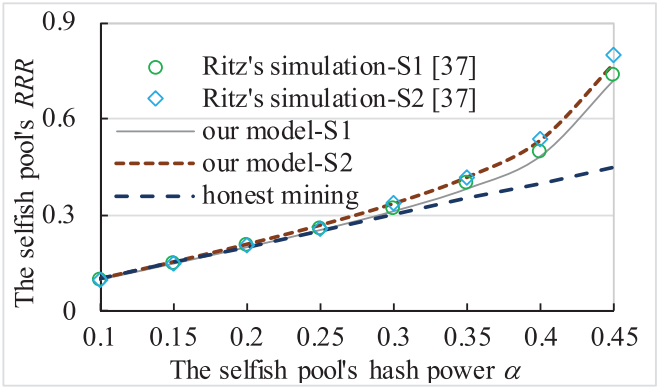



**Fig. 7.** Verify our model by comparing Ritz's simulation (Ritz and Zugenmaier, 2018).

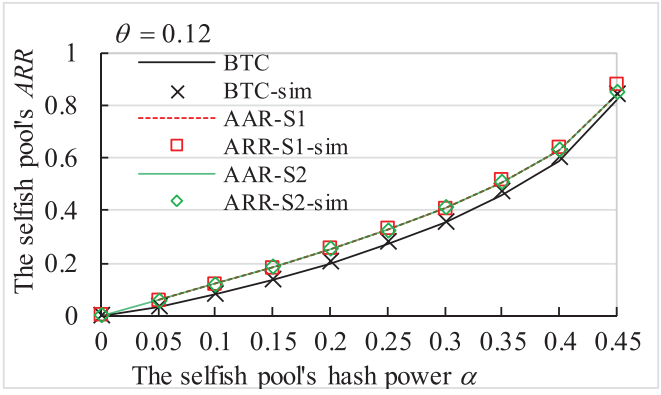



**Fig. 8.** Verify our model by comparing our simulation.

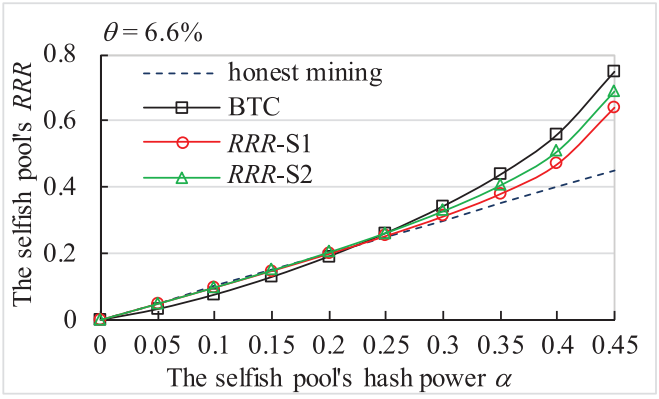



**Fig. 9.** The selfish pool's *RRR* over the selfish pool's hash power.

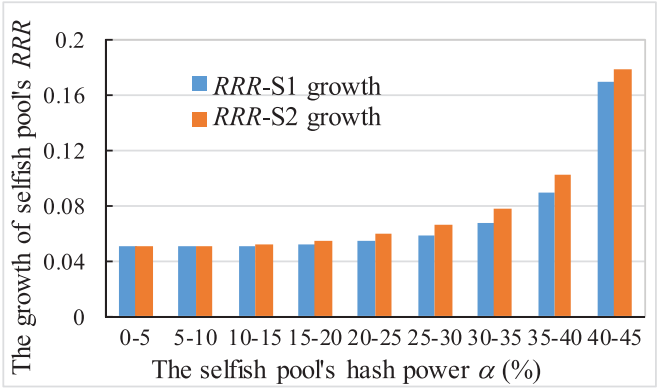



**Fig. 10.** The growth of *RRR* over the selfish pool's hash power.

Ritz and Zugenmaier, (2018) made a Monte Carlo simulation to evaluate the selfish mining in Ethereum. We set our parameters to be the same as the parameters in Ritz's work, and we compare our results and Ritz's by observing their result chart. Fig. 7 shows *RRR* of the selfish pool adopting two strategies, respectively, under the condition of $\theta = 0.12$. As illustrated in Fig. 7, the results in two strategies by our model are both close to the results of Ritz's simulation experiments. These experiments are evidence for the correctness and accuracy of our model. Different from Ritz's work, we propose a stochastic model to get a more accurate analysis.

We do a series of experiments to compare the results of the model and the Monte Carlo simulation, and their results are very close. We calculate *ARR* of the selfish pool on the condition that $\theta = 0.12$ as an example to prove our model, which is shown in Fig. 8.

### 4.2. Mining revenue analysis

Based on our model, we evaluate the revenue ratio that the selfish pool can get. In our experiments, we set the default value as follows. $\gamma_i$=1/*i*, $\theta$= 0.5% in Bitcoin, $\theta$= 6.6% in Ethereum, $N_t$ = 2137, $T_{block}$ = 600s in Bitcoin, $N_t$ = 142, $T_{block}$ = 13s in Ethereum, and $\alpha$ changes from 0 to 0.5 (etherchain.org 2020).

First, we observe the *RRR* of the selfish pool in Ethereum. For a clear comparison of mining revenue between Bitcoin and Ethereum, the probability of natural fork occurrence of Bitcoin is set at 6.6% too. Fig. 9 shows the *RRR* for selfish pool over hash power in Ethereum and Bitcoin. It is obvious that in both Bitcoin and Ethereum systems, the selfish pool can get more revenue than what they deserve when the pool's hash power is large enough. The threshold of the selfish pool gaining unfair revenue is the intersection of honest mining revenue and selfish mining revenue. As shown in Fig. 9, the threshold that makes the selfish pool profitable in Ethereum is lower than that in Bitcoin because of uncle and nephew rewards. The threshold of Ethereum is 0.1596 (**S2**), and the threshold of Bitcoin is 0.22, when the probability of natural fork occurrence is 6.6%.

The *RRR* for the selfish pool is different in Ethereum and Bitcoin. *RRR* for the selfish pool in Ethereum is more than that in Bitcoin when the hash power of the pool changes from 0 to 0.25. However, the pool can get less *RRR* in Ethereum than that in Bitcoin when the hash power is more than 0.3. This is because uncle rewards can compensate the loser in the mining competition. It is obvious that the GHOST protocol has an important influence on selfish mining in Ethereum.

The reference strategy also affects the revenue for the pool in Ethereum. For a selfish pool with 20% hash power of the total power, what reference strategy it adopts has little influence on the *RRR* of the pool. However, if the hash power controlled by the pool is larger than 20%, **S2** can take more *RRR* than **S1** for the selfish pool, and the *RRR* difference between the two strategies is directly proportional to the hash power controlled by the pool. This is because the pool references honest miners' uncle blocks by adopting **S1**, leading to honest miners profitable.

With the increasing hash power of the selfish pool, the growth of *RRR* does not grow linearly. Fig. 10 shows the growth of *RRR* under the increase of the selfish pool's hash power is 5%. If the pool adopts **S1**, the growth of *RRR* is almost unchanged when the pool's hash power is less than 25%, and the growth speeds up when the pool's hash power is more than 30%. If the pool adopts **S2**, the *RRR* growth begins to increase significantly when the pool's hash power is around 20%. This is because the pool with large hash power

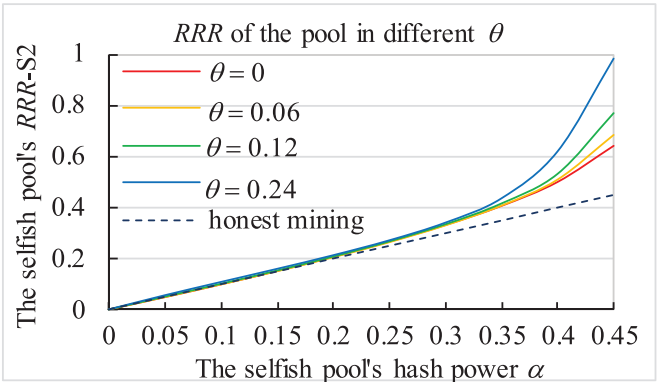



**Fig. 11.** *RRR* of the selfish pool which adopts **S2** in different natural fork probability.

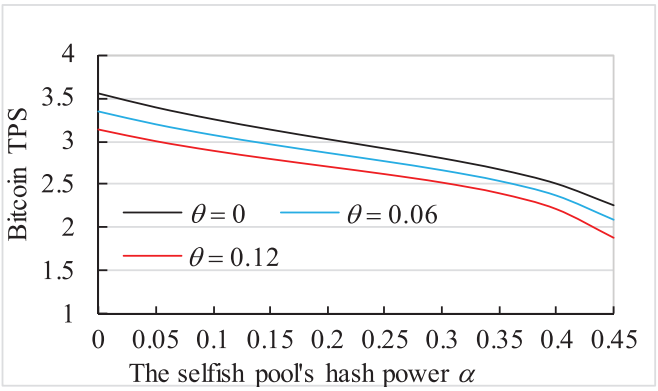



**Fig. 13.** Bitcoin TPS over the probability of natural fork occurrence.

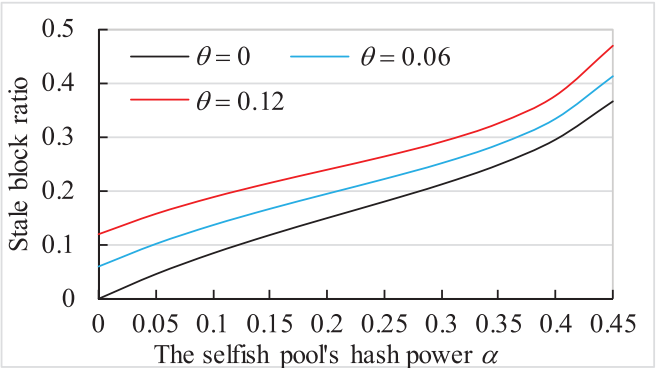



**Fig. 12.** Stale block ratio over the probability of natural fork occurrence.

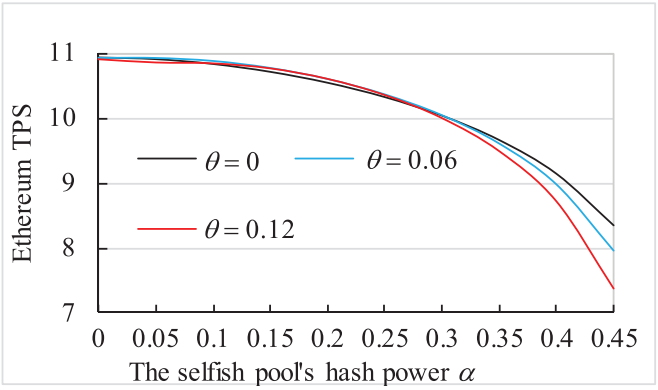



**Fig. 14.** Ethereum TPS over the probability of natural fork occurrence.

is easier to create a private branch and make a clear advantage, which causes the pool to have a larger probability of getting revenue.

Fig. 11 shows the relationship between the probability of natural fork occurrence ($\theta$) and the selfish mining revenue. Honest miners hash power is wasted by creating natural forks, and more forks mean that less computing power is used to create a new block to increase the public branch length. As illustrated in Fig. 11, with $\theta$ increases, the selfish pool is easier to get unfair revenue, and the pool can get more revenue when the hash power of the pool is fixed.

### 4.3. Blockchain system performance analysis

Selfish mining causes a number of stale blocks larger than the system without selfish miners (Fig. 12). When there is no selfish miner in the system, the stale blocks are caused by the natural fork, namely honest miners make several blocks stale. The number of forks of the system increases with the selfish pools' hash power growth. Namely, the more selfish hash power, the more blocks are stale. According to our quantitative results, people can detect if there are any selfish miners in the blockchain system.

Then we focus on blockchain TPS (Fig. 13), which can evaluate the performance of the blockchain system. With the increasing hash power of the selfish pool, TPS decreases. In Bitcoin, TPS decreases approximately linearly (Fig. 13). When the hash power of the pool reaches 45%, TPS is nearly 60% of the system without selfish miners. However, as the transactions in the uncle blocks can be included in the blockchain, natural forks and small selfish pool affect little on TPS of Ethereum (Fig. 14). With the increasing hash power of the pool, TPS of Ethereum decreases more quickly. Fig. 13 and Fig. 14 suggest that the selfish mining is detrimental to blockchain performance. Besides, Ethereum is more resilient than Bitcoin from the perspective of blockchain TPS.

### 4.4. Blockchain system security analysis

We analyze the scenario that the system suffers from two kinds of attacks (selfish mining and double-spending). Double-spending can pose a huge threat to a blockchain system. To prevent double-spending attack, transactions in blockchain usually have to be confirmed by several blocks. For example, Bitcoin transactions have to be confirmed by six blocks (Nakamoto and Bitcoin, 2008), and most Ethereum wallet applications require 12 blocks to confirm a transaction (Bez et al., 2019). We quantify the impact of selfish mining on system resilience of the probability of a successful double-spending attack. Fig. 15 and Fig. 16 show the probability that the DS pool attacks Bitcoin and Ethereum by launching double-spending attack. Meanwhile, the system suffers selfish mining.

As illustrated in Fig. 15, more selfish hash power and more DS hash power can both lead to a larger probability of double-spending success. This is because the more hash power controlled by the DS pool means a higher rate of the DS pool generating a secret block, and the more selfish hash power means the lower rate of generating a block in the main public chain. Because of the higher natural fork probability and some other reasons, the number of recommended confirmation blocks in Ethereum is larger than that in Bitcoin, which aims to resist double-spending attacks. As can be seen in Fig. 16, the probability of double-spending success in Ethereum has a similar tendency with that in Bitcoin. However, attackers are easier to launch a double-spending attack successfully in Ethereum when the hash power of the selfish pool is fixed. When the hash power of the DS pool is fixed, both two

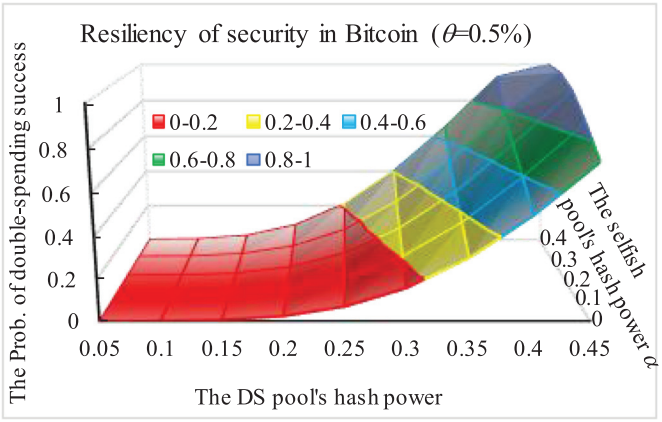



**Fig. 15.** The probability of double-spending success in Bitcoin system, where the natural fork probability is 0.5%, and the number of confirmation blocks is 6.

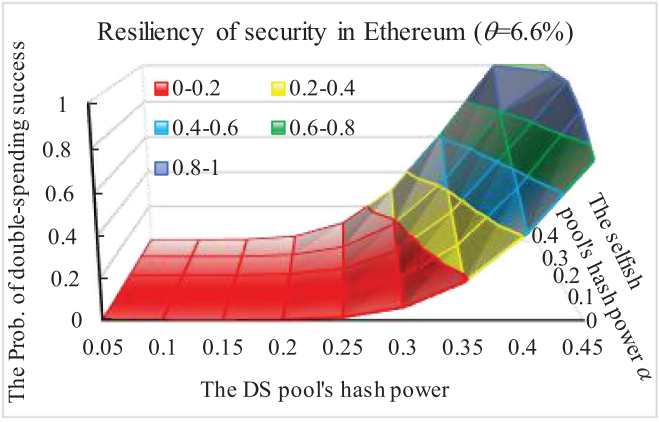



**Fig. 16.** The probability of double-spending success in Ethereum system, where the natural fork probability is 6.6%, and the number of confirmation blocks is 12.

figures show that the probability of double-spending success increases with the selfish pool's hash power increases. Namely, a blockchain system with more selfish miners is less resilient. Moreover, Bitcoin is more resilient than Ethereum, from the perspective of the probability of double-spending success.

## 5. Conclusion and future work

In this paper, we propose a continuous-time Markov chain model to analyze selfish mining in the top two largest cryptocurrencies by market capitalization, namely, Bitcoin and Ethereum. Our model enables us to capture the behaviors of the system with selfish miners. Differently from the existing models, our model considers the natural forks made by honest miners, and it has three-dimensional states which can describe the accurate reference distance in Ethereum. As illustrated in the experiment results, the hash power threshold that makes selfish miners gain unfair revenue in Ethereum is lower than that in Bitcoin, indicating that Ethereum is more vulnerable to Bitcoin in selfish mining attack. On the other hand, the selfish miners get less undeserved revenue in Ethereum than in Bitcoin, when they exceed the threshold. Selfish mining not only causes honest miners' losses but also impacts on the performance of blockchain system, such as TPS. Furthermore, we carry out a quantitative analysis of the impact of selfish mining on blockchain security when the system suffers from selfish mining and double-spending attacks. Finally, our results demonstrate that selfish mining lowers the resiliency of blockchain systems on resisting double-spending attacks.

Note that this paper assumes that there is only a two-way natural fork, and natural forks are created in the same branch. However, honest miners may produce blocks in different branches simultaneously. One future work is to extend our model to quantitatively investigate this situation. Besides, this paper assumes that Bitcoin and Ethereum only suffer from a classical selfish mining attack (Eyal and Sirer, 2014). Actually, based on selfish mining, more complex attacks can be developed, such as stubborn mining (Nayak et al., 2016) and selfholding attack (Dong et al., 2019). We will extend our modeling approach to analyze selfish mining under these complex attacks.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A. The system state transition rules

This section presents the state transition rules of Fig. 4 in the Sec. III-B according to two types of triggering events. The definitions of the system state ($S$, $H$, $F$) and the variables used in the following are given in Sec. III.

**Event 1**: The selfish pool creates a new block.

Case 1: $(0,0,0') \xrightarrow{\alpha} (1,0,0')$ All miners mine the block on the top of one consensus block, and the selfish pool generates a new block. The selfish pool keeps mining on the new block, and honest miners mine blocks on the consensus block.

**Case 2**: There exist several branches with equal length, which start from one block. The selfish pool produces a new block, and $\Delta = 1$. According to the mining strategy, the selfish pool publishes its new block to make the block in the main chain. After receiving the new block, all miners achieve a consensus and mine blocks on the selfish pool's block.

Subcase 1: $(0,1,0) \xrightarrow{\alpha} (0,0,0')$ Two branches with equal length are both mined by honest miners, and the selfish pool generates a block on one of these branches.

Subcase 2: $(1,1,0') \xrightarrow{\alpha} (0,0,0')$ There are two branches, which are created by honest miners and the selfish pool, respectively. The selfish pool publishes the new block when it generates a block.

Subcase 3: $(1,1,0) \xrightarrow{\alpha} (0,0,0')$ There is a fork with three branches, and the selfish pool produces a block in the private branch.

**Case 3**: $(S,H,F) \xrightarrow{\alpha} (S+1,H,F) S > 1$.The public branch is null, or $\Delta > 1$. In this case, the selfish pool has a clear advantage, and it does not publish the new block and keeps mining.

**Event 2**: The honest miners generate a new block or create a new fork.

**Case 1**: There is a natural fork with two branches, and the length of each branch is one.

Subcase 1: $(0,1,0) \xrightarrow{\beta_f} (0,1,0)$Honest miners create a new fork in one of two branches. Both honest and selfish miners mine blocks on one of the two branches.

Subcase 2: $(0,1,0) \xrightarrow{\beta_s} (0,0,0')$Honest miners produce a new block in one of the equal length branches, and all the miners mine blocks on the new block since it is the consensus block.

**Case 2**: All miners mine on a consensus block. There are two subcases as follows, depending on whether a fork is created.

Subcase 1: $(0,0,0') \xrightarrow{\beta_f} (0,1,0)$.Honest miners create a fork, and all miners select a branch to mine new blocks.

Subcase 2: $(0,0,0') \xrightarrow{\beta_s} (0,0,0')$.Honest miners create a block, and all miners start to mine blocks on this new block since a consensus is achieved.

**Case 3**: The length of the private branch is one. There are two subcases if honest miners produce blocks. After blocks are produced by honest miners, the selfish pool publishes its private

block, and the selfish pool keeps mining in its private branch. The honest miners mine in one of these branches.

Subcase 1: $(1,0,0') \xrightarrow{\beta_s} (1,1,0')$. Honest miner produce a single new block.

Subcase 2: $(1,0,0')\beta_f(1,1,0)$Honest miners create a natural fork.

**Case 4**: $(2,0,0') \xrightarrow{\beta_e} (0,0,0')$. The system is in the state (2, 0, 0′), and $\Delta = 2$. Honest miners generate a block or a fork. The selfish pool publishes the private branch. According to the longest-chain rule, the private branch is the main chain.

**Case 5**: There is a fork with two or three equal length branches. New blocks are created by honest miners.

Subcase 1: $(1,1,0') \xrightarrow{\beta_s} (0,0,0')$.There are two branches, honest miners produce a single block, and all miners achieve a consensus.

Subcase 2: $(1,1,0) \xrightarrow{\beta_s} (0,0,0')$.There are three branches. Honest miners produce a block, and the blockchain reaches consensus.

Subcase 3: $(1,1,0') \xrightarrow{\beta_f} (0,1,0)$.There is a fork with two branches, and honest miners create a fork on one of these branches.

Subcase 4: $(1,1,0) \xrightarrow{\beta_f} (0,1,0)$.There are three branches. Honest miners produce two blocks and create a fork on the top of a branch.

**Case 6:** $(S,H,F) \xrightarrow{\beta_e} (0,0,0')$,$S = H + F + 2$.The selfish pool publishes the whole private branch when $\Delta = 1$ after honest miners produce blocks.

Case 7 and case 8 describe the cases that the selfish pool has a clear advantage, namely, $\Delta \geq 2$.

**Case 7**: Honest miners mine blocks on the consensus block, the selfish pool publishes the first unpublished block when honest miners produce a block or create a fork.

Subcase 1: $(S,0,0') \xrightarrow{\beta_s} (S,1,0')$, $S \geq 3$.Honest miners generate a block.

Subcase 2: $(S,0,0') \xrightarrow{\beta_f} (S,1,0)$, $S \geq 3$.A natural fork is created by honest miners on the consensus block.

**Case 8**: $H + F > 0$ and $\Delta \geq 2$. According to whether honest miners create a fork or honest miners mine blocks in the private branch, there are four subcases as follows. In this case, the probability that honest miners produce blocks in the private branch depends on the number of branches that honest miners can recognize. In this case, $\gamma$ denotes the probability that honest miners produce blocks in the private branch. If $F = 0'$, $\gamma = \gamma_2$. If $F \neq 0'$, $\gamma = \gamma_3$.

Subcase 1: $(S,H,F) \xrightarrow{\beta_f\gamma} (S-(H+F),1,0)$.When honest miners create a fork in the private branch, the selfish pool publishes the first unpublished block, and a fork with three branches is created.

Subcase 2: $(S,H,F) \xrightarrow{\beta_s\gamma} (S-(H+F),1,0')$.Honest miners produce a block in the private branch. The selfish pool publishes a block, and a fork is created in the private branch.

Subcase 3: $(S,H,F) \xrightarrow{\beta_s(1-\gamma)} (S,H+1,0')$. If a new block is generated by honest miners in the public branch, the selfish pool publishes a block.

Subcase 4: $(S,H,F) \xrightarrow{\beta_f(1-\gamma)} (S,1,H+F)$.If honest miners create a natural fork, the selfish pool publishes the first unpublished block in the private branch. Then honest miners can recognize three equal length branches.

## Appendix B. Proof of the formulas for mining rewards

Miners in Ethereum can get three kinds of rewards, which are static block reward, uncle reward and nephew reward. Miners in Bitcoin can only gain static block reward. We consider two reference strategies of the selfish pool in Ethereum as follows.

Strategy 1 (**S1**): The selfish pool references as many uncle blocks as it can.

Strategy 2 (**S2**): The selfish pool does not reference any blocks.

In Appendix B, we prove the formulas for calculating mining rewards for the selfish pool and honest miners. The formulas are given in Sec. III-C. The notations and their definitions are given in Sec. III. $R$ denotes the rewards for miners. $\pi_{(S,H,F)}$ denotes the steady-state probability of the state $(S,H,F)$. $r_{(S,H,F)}$ denotes the rewards for miners when the system is in the state $(S,H,F)$, and the function for calculating $r_{(S,H,F)}$ are different in different kinds of rewards. $\Gamma$ denotes the set of the system states in our model. The rewards for miners can be calculated by $R = \sum_{(S,H,F)\in\Gamma}(r_{(S,H,F)} \cdot \pi_{(S,H,F)})$. Let $p^r_{(S,H,F)}$ denote the expect number of blocks which can gain the corresponding rewards when the system is in the state $(S,H,F)$. For example, when we calculate the uncle reward for miners, $p^r_{(S,H,F)}$ denotes the expect number of uncle blocks in the state $(S,H,F)$.

As described in the Sec. III-C, we normalize a static block reward as 1. Therefore, when we calculate the static block rewards for miners, $r_{(S,H,F)} = p^r_{(S,H,F)}$.

**Proposition 1.** Eq. (B.1) calculates the static block rewards for the selfish pool

$$R_{S,S} = p_\alpha\left(1-\pi_{(0,0,0')}\right) + p_\alpha\pi_{(0,0,0')}\left(p_\alpha + p_\alpha p_\beta + p_f p_\beta \gamma_3 + p_s p_\beta \gamma_2\right). \tag{B.1}$$

**Proof.** $R_{S,S} = \sum_{(S,H,F)\in\Gamma}(p^r_{(S,H,F)} \cdot \pi_{(S,H,F)})$, according to the $R_{S,S}$ definition in the Sec. III-C. We discuss how to calculate $p^r_{(S,H,F)}$ in the two cases.

Case 1: The system is not in the state (0, 0, 0′). If $\Delta \geq 1$, the private branch must be the main chain, and the selfish pool must gain a static block reward according to the selfish mining algorithm which is proposed in Sec. III-A. If the system is in the state (0, 1, 0), the pool publishes its new block when it produces the block, and the selfish pool gets a static block reward, because the new block is a regular block (the block in the main chain). Thus, if the selfish pool generates a block, the pool can get a static block reward, unless the system is in the state (0, 0, 0′). In this case, the probability that the selfish pool gains static block rewards is $p_\alpha$. Namely, $p^r_{(S,H,F)} = p_\alpha$ on the condition that $(S,H,F) \neq (0,0,0')$.

Case 2: The system is in the state (0, 0, 0′), namely, all miners mine blocks on a consensus block. The selfish pool can get rewards in some cases when the selfish pool generates a block. Namely, the system state transits to (1, 0, 0′). It cannot be sure whether the selfish pool can get the static block rewards. We need to consider the following subcases to determine the probability that the selfish pool gets static block reward.

Case 2.1: The next block is mined by the selfish pool. That is, the state of the system is (2, 0, 0′), and $\Delta = 2$, so the selfish pool can get the reward. The probability of case 2.1 occurring is $p_\alpha$.

Case 2.2: A block or a fork is created by honest miners on the consensus block. A fork with two or three equal branches is generated. At this time, the system is in the state (1, 1, 0′) or (1, 1, 0), and the probabilities of these cases happening are $p_s$and $p_f$, respectively. If the next block is created in the private branch, the private branch becomes the main chain, and the pool gets a reward. According to the owner of the next block, there are two cases as follows.

Case 2.2.1: A new block is mined by the selfish pool in the private branch with the probability $p_\alpha$, and it is published immediately. In this case, the selfish pool gets a static block reward. The state transition is $(1,1,F) \rightarrow (0,0,0')$. The probability of case 2.2.1 happening is $p_\alpha$.

Case 2.2.2: Honest miners produce a block or create a fork with two or more branched on the private branch, and the private branch is longer than the public branch. Namely, the private branch becomes the main branch, and the selfish pool gains a static block reward. Namely, the system state transits from (1, 1, 0′) or (1, 1, 0) to (0, 0, 0′) or (0, 1, 0) with probabilities $p_\beta\gamma_2$ and $p_\beta\gamma_3$, respectively.

Thus, $p^r_{(0,0,0')} = p_\alpha(p_\alpha + p_\alpha p_\beta + p_f p_\beta\gamma_3 + p_s p_\beta\gamma_2)$.

Based on above analysis, Eq. (B.1) is proved. ■

**Proposition 2.** Static block rewards for honest miners are

$$R_{S,H} = p_\beta\left(\pi_{(0,0,0')} + \pi_{(0,1,0)} + \pi_{(1,1,0')} + \pi_{(1,1,0)}\right) + \left(p_\beta p_s(1-\gamma_2) + p_\beta p_f(1-\gamma_3)\right)\pi_{(1,0,0')}. \quad \text{(B.2)}$$

**Proof.** According to the definition of $R_{S,H}$ in the Sec. III-C, $R_{S,H} = \sum_{(S,H,F)\in\Gamma}(p^r_{(S,H,F)} \cdot \pi_{(S,H,F)})$. There are three cases that $p^r_{(S,H,F)} \neq 0$, which are given as follows.

Case 1: The system is in the state (0, 0, 0′)or (0, 1, 0)). Namely, all miners mine blocks on the consensus block(s). If honest miners create a block or a fork, honest miners gain the static block reward. Thus, $p^r_{(0,0,0')} = p_\beta$ and $p^r_{(0,1,0)} = p_\beta$.

Case 2: The system is in the state (1, 1, 0′) or (1, 1, 0). Namely, honest miners and the selfish pool have equal length branches, and there is one block in each branch. Honest miners get the static block reward when honest miners produce a block or create a fork. In this case, $p^r_{(1,1,0')} = p_\beta$ and $p^r_{(1,1,0)} = p_\beta$.

Case 3: The selfish pool has produced a block, and the pool keeps it in secret. Namely, the system state is (1, 0, 0′). If honest miners produce two blocks on the consensus block continuously and create a public branch which is longer than the private branch, and honest miners can get a static block reward because the public branch is the main chain. Depending on whether a fork is created on the consensus block, there are two cases as follows.

Case 3.1: Honest miners produce a single block on the consensus block, and honest miners generate blocks in the public branch. This happens with probability $p_\beta p_s(1-\gamma_2)$.

Case 3.2: A natural fork is created by honest miners on the consensus block. The selfish pool publishes the unpublished block, and there is a fork with three branches. Then, honest miners generate blocks in the public branch. This happens with probability $p_\beta p_f(1-\gamma_3)$.

Thus, $p^r_{(1,0,0')} = p_\beta p_s(1-\gamma_2) + p_\beta p_f(1-\gamma_3)$.

In summary, Eq. (B.2) is proved. ■

$R_U(d)$ given in Eq. (B.3) is the uncle reward function, representing the uncle reward for a miner who produces an uncle block in reference distance (the distance between uncle and nephew blocks) $d$. When we calculate uncle rewards for miners, $r_{(S,H,F)}$ can be calculated by $R_U(d)$ and $p^r_{(S,H,F)}$ in the state $(S, H, F)$. $r_{(S,H,F)} = p^r_{(S,H,F)} \cdot R_U(d)$ and $d$ can be calculated in each state.

$$R_U(d) = \begin{cases} (8-d)/8, & 1 \le d \le 6 \\ 0, & others \end{cases} \quad \text{(B.3)}$$

**Proposition 3.** Uncle rewards for the selfish pool when the pool adopts **S1** is given as Eq. (B.4).

$$R_{U,S,1} = R_U(1) \cdot \left(p_\beta(1-\gamma_2)\pi_{(1,1,0')} + p_\beta(1-\gamma_3)\pi_{(1,1,0)}\right) \quad \text{(B.4)}$$

**Proof.** According to the definition of $R_{U,S,1}$ in the Sec. III-C, $R_{U,S,1} = \sum_{(S,H,F)\in\Gamma}(r_{(S,H,F)} \cdot \pi_{(S,H,F)})$, where $r_{(S,H,F)} = p^r_{(S,H,F)} \cdot R_U(d)$. As honest miners always publish their blocks as soon as they produce them, the selfish pool always mines blocks on the last block. The pool's blocks cannot be stale unless there is a fork with a private branch and (a) public branch(s). Namely, only the system is in the state (1, 1, 0′) or (0, 1, 0), $p^r_{(S,H,F)} \neq 0$. We discuss two cases to determine $p^r_{(S,H,F)}$ and $d$.

Case 1: The system is in the state (1, 1, 0′). There are two branches, and one of the branches is the private branch. The selfish pool can get an uncle reward if honest miners produce a block or create a fork in the public branch. Thus, $p^r_{(1,1,0')} = p_\beta(1-\gamma_2)$, and $d = 1$ .

Case 2: The system is in the state (1, 1, 0). There are three branches, and one of the branches is a private branch. If honest miners produce blocks in the public branch, the pool's block becomes an uncle block, with probability $p_\beta(1-\gamma_3)$. Thus, $p^r_{(1,1,0)} = p_\beta(1-\gamma_3)$, and $d = 1$.

In summary, uncle rewards for the selfish pool are $R_{U,S,1} = R_U(1) \cdot (p_\beta(1-\gamma_2)\pi_{1,1,0'} + p_\beta(1-\gamma_3)\pi_{1,1,0})$, when the pool adopts **S1**. ■

**Proposition 4.** Uncle rewards for honest miners when the pool adopts **S1** are given as Eq. (B.5), where $k \neq 0'$.

$$\begin{aligned} R_{U,H,1} = {} & R_U(1)p_f\pi_{(0,0,0')} + R_U(1)\left(p_\alpha + p_\beta\gamma_2\right)\pi_{(1,1,0')} \\ & + \left(2R_U(1)p_\alpha + R_U(1)p_\beta(1+\gamma_3)\right)\pi_{(1,1,0)} \\ & + \sum_{i=2}^{\infty}\left(R_U(i)\left(p_s + 2p_f\right)\pi_{(i,0,0')}\right) \\ & + \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\left(R_U(i)\left(p_s + 2p_f\right)\gamma_3\pi_{(i+j+k,j,k)}\right) \\ & + \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\left(R_U(i)\left(p_s + 2p_f\right)\gamma_2\pi_{(i+j,j,0')}\right) \end{aligned} \quad \text{(B.5)}$$

**Proof.** According to the definition of $R_{U,H,1}$ in the Sec. III-C, $R_{U,H,1} = \sum_{(S,H,F)\in\Gamma}(r_{(S,H,F)} \cdot \pi_{(S,H,F)})$, where $r_{(S,H,F)} = p^r_{(S,H,F)} \cdot R_U(d)$. There are several cases that a block mined by honest miners becomes an uncle block. To calculate $p^r_{(S,H,F)}$ and $d$, four cases that $p^r_{(S,H,F)} \neq 0$ are discussed.

Case 1: The system is in the state (0, 0,0′). If honest miners create a fork on the consensus block, one of the two blocks must be the uncle block, and honest miners must get one uncle block. In case 1, $p^r_{(0,0,0')} = p_f$, and $d = 1$.

Case 2: A fork has several equal length branches, and a block is produced in the private branch. In this case, the public branch is abandoned, and the owner of the first block in the public branch can get uncle rewards.

Case 2.1: The system is in the state (1, 1,0′). The fork has two branches, and the length of the two branches are both one. If the selfish pool or honest miners produce blocks in the private branch, honest miners gain an uncle reward, because the block in the public branch is referenced with reference distance 1.

Thus, $p^r_{(1,1,0')} = p_\alpha + p_\beta\gamma_2$, and $d = 1$.

Case 2.2: The system is in the state (1, 1, 0). If the fork has three equal length branches, there are two cases that the blocks mined by honest miners are referenced by a new block.

If the next block is generated by the selfish pool in the private branch with probability $p_\alpha$, honest miners can get an uncle reward, and the reference distance is one.

If the next block is created by honest miners in the private branch with probability $p_\beta\gamma_3$, honest miners can get two uncle rewards, and the reference distance is one.

If the next block is created by honest miners in the public branch with probability $p_\beta(1-\gamma_3)$, honest miners can get an uncle reward, and the reference distance is one.

In summary, $p^r_{(1,1,0)} = (2p_\alpha + p_\beta(1+\gamma_3))$, and $d = 1$.

Case 3: The system is in the state $(i, 0, 0'), i \ge 2$. Namely, there is more than one block in the private branch, and the length of the public branch is zero. In this case, the public branch must be stale, and the owners of uncle blocks receive uncle rewards. Although honest miners can reference the uncle block with a shorter distance, the uncle block reference distance is the length of the private branch, because only the regular block can reference the uncle block. Depending on whether a natural fork is created, honest miners can gain one or two uncle rewards in this case.

Thus, $p^r_{(i,0,0')} = p_s + 2p_f$, and $d = i$, where $i \geq 2$.

Case 4: The system is in the state $(i+H+F, H, F)$, and $i \geq 3$, $H \geq 1$. There is a fork with several branches, and the private branch is longer than the public branch at least two blocks. According to the number of branches, there exist the following subcases.

Case 4.1: The system is in the state $(i+H+F, H, F)$, and $i \geq 3$, $H \geq 1$ and $F \neq 0'$. Honest miners can see three equal branches. A block or a fork is created by honest miners in the private branch, and the previous public branch has been abandoned. The uncle rewards are determined by the reference distance between the new fork and the latest block of the private branch. The probability that honest miners mine blocks in the private branch is $\gamma_3$. In this case, $p^r_{(i+H+F,H,F)} = p_s\gamma_3 + 2p_f\gamma_3$ where $i \geq 3$, $H \geq 1$ and $F \neq 0'$.

Case 4.2: The system is in the state $(i+H, H, 0')$, and $i \geq 3$, $H \geq 1$. Different from case 4.1 the probability that honest miners produce a block or create a fork in the private branch is $\gamma_2$. If honest miners create a block in the private branch, honest miners gain an uncle block. If honest miners create a natural fork, two uncle rewards are rewarded to honest miners. Thus, $p^r_{(i+H+F,H,F)} = p_s\gamma_3 + 2p_f\gamma_3 p^r_{(i+H,H,0')} = p_s\gamma_2 + 2p_f\gamma_2$ and $d = i$, where $i \geq 3$ and $H \geq 1$.

In summary, Eq. (B.5) are proved. ■

The function of a nephew reward in Ethereum is given as Eq. (B.6). When we calculate nephew rewards for miners, $r_{(S,\,H,\,F)}$ can be calculated by $R_n(d)$ and $p^r_{(S,H,F)}$ in the state $(S,\,H,\,F)$. $r_{(S,H,F)} = p^r_{(S,H,F)} \cdot R_U(d)$ and $d$ can be calculated in each state.

$$R_n(d) = \begin{cases} 1/32, & 1 \leq d \leq 6 \\ 0, & others \end{cases} \tag{B.6}$$

**Proposition 5**. Nephew rewards for selfish miners when the selfish pool adopts **S1** are given as Eq. (B.7), where $k \neq 0'$ and $\chi = p_\alpha - p_\alpha p_\beta(p_s(1-\gamma_2) + p_f(1-\gamma_3))$.

$$\begin{aligned} R_{N,S,1} = {} & R_n(1)\left(\left(p_s p_\alpha + 2p_f p_\alpha\right)\pi_{(1,0,0')} + p_\alpha \pi_{(0,1,0)}\right) \\ & + \sum_{i=2}^{\infty}\left(R_n(i)p_\beta^{i-2}\left(p_s+2p_f\right)\chi \cdot \pi_{(i,0,0')}\right) \\ & + \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\left(R_n(i)p_\beta^{i-2}\gamma_3\left(p_s+2p_f\right)\chi \cdot \pi_{(i+j+k,j,k)}\right) \\ & + \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\left(R_n(i)p_\beta^{i-2}\gamma_2\left(p_s+2p_f\right)\chi \cdot \pi_{(i+j,j,0')}\right) \end{aligned} \tag{B.7}$$

**Proof.** According to the definition of $R_{N,\,S,\,1}$ in the Sec. III-C, $R_{N,S,1} = \sum_{(S,H,F)\in\Gamma}(r_{(S,H,F)} \cdot \pi_{(S,H,F)})$, where $r_{(S,H,F)} = p^r_{(S,H,F)} \cdot R_U(d)$. There are three cases that $p^r_{(S,H,F)} \neq 0$, and we discuss the three cases to compute $p^r_{(S,H,F)}$ and $d$.

Case 1: The system is in the state (1, 0,0′). The length of the private branch is one. If honest miners produce a block or create a fork, the lengths of all branches are equal. Then the selfish pool produces a block in the private branch, and the pool can get nephew rewards. Depending on the number of uncle blocks, the pool can get one or two nephew rewards. Thus, $p^r_{(1,1,0')} = p_s p_\alpha + 2p_f p_\alpha$ and $d = 1$.

Case 2: The system is in the state (0, 1, 0). If the selfish pool produces a block on one of the two blocks, the pool can gain a nephew reward. $p^r_{(0,1,0)} = p_\alpha$ and $d = 1$.

Case 3: The system is in the state $(i+H+F, H, F)$, $i \geq 2$, $F \neq 0'$. The private branch is longer than the public branch, so the private branch must be the main branch. If a block is mined in the private branch or on the consensus block, the block can be referenced as an uncle block, and the owner of the regular block can get nephew rewards. After honest miners producing $(i-2)$ blocks, the probability that the selfish pool produces a regular block is$p_\alpha - p_\alpha p_\beta(p_s(1-\gamma_2) + p_f(1-\gamma_3))$. Depending on whether a natural fork is created, the pool can reference one or two uncle blocks and get one or two nephew rewards. According to the number of branches, there are two subcases.

Case 3.1: The system is in the state $(i,\,0,\,0')$ where $i \geq 2$. The length of the public branch is zero. If honest miners produce a block or create a fork on the consensus block, the selfish pool gains nephew rewards when the pool produces regular blocks. In this case, $d = i$ and

$$\begin{aligned} p^r_{(i,0,0')} = {} & p_\beta^{i-2}\left(p_s+2p_f\right) \\ & \cdot\left(p_\alpha - p_\alpha p_\beta\left(p_s(1-\gamma_2) + p_f(1-\gamma_3)\right)\right), \end{aligned}$$

where $i \geq 2$.

Case 3.2: The system is in the state $(i+H+F, H, F)$, and $i \geq 3$, $H \geq 1$ and $F \neq 0'$. In this case, honest miners can see three equal branches, and the probability of honest miners producing blocks in the private branch is $\gamma_3$. Thus, $d = i$ and

$$\begin{aligned} p^r_{(i+H+F,H,F)} = {} & p_\beta^{i-2}\gamma_3\left(p_s+2p_f\right) \\ & \cdot\left(p_\alpha - p_\alpha p_\beta\left(p_s(1-\gamma_2) + p_f(1-\gamma_3)\right)\right), \end{aligned}$$

where $i \geq 3$, $H \geq 1$ and $F \neq 0'$.

Case 3.3: The system is in the state $(i+H, H, 0')$, and $i \geq 3$, $H \geq 1$. In this case, honest miners produce blocks in one of the two branches, and the probability of honest miners producing blocks in the private branch is $\gamma_2$. Thus, $d = i$and

$$\begin{aligned} p^r_{(i+H,H,0')} = {} & p_\beta^{i-2}\gamma_2\left(p_s+2p_f\right) \\ & \cdot\left(p_\alpha - p_\alpha p_\beta\left(p_s(1-\gamma_2) + p_f(1-\gamma_3)\right)\right), \end{aligned}$$

where $i \geq 3$, $H \geq 1$.

Let $\chi = p_\alpha - p_\alpha p_\beta(p_s(1-\gamma_2) + p_f(1-\gamma_3))$.

In summary, Eq. (B.7) is proved. ■

**Proposition 6.** Nephew rewards for honest miners when the pool adopts **S1** are given as Eq. (B.8), where $j,\, k \neq 0,\, 0'$.

$$\begin{aligned} R_{N,H,1} = {} & R_n(1) \cdot p_\beta\left(2\pi_{(1,1,0)} + \pi_{(0,1,0)} + \pi_{(1,1,0')}\right) \\ & + \sum_{i=2}^{\infty}\left(R_n(i)\left(p_s+2p_f\right)p_\beta^{i-1}\pi_{(i,0,0')}\right) \\ & + \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0,0'}^{\infty}\left(R_n(i)\left(p_s+2p_f\right)\gamma_3 p_\beta^{i-1}\pi_{(i+j+k,j,k)}\right) \\ & + \sum_{i=3}^{\infty}\sum_{j=1}^{\infty}\left(R_n(i)\left(p_s+2p_f\right)\gamma_2 p_\beta^{i-1}\cdot\pi_{(i+j,j,0')}\right) \end{aligned} \tag{B.8}$$

**Proof.** According to the definition of $R_{N,\,H,\,1}$ in the Sec. III-C, $R_{N,H,1} = \sum_{(S,H,F)\in\Gamma}(r_{(S,H,F)} \cdot \pi_{(S,H,F)})$ and $r_{(S,H,F)} = p^r_{(S,H,F)} \cdot R_U(d)$. We discuss two cases, which $p^r_{(S,H,F)} \neq 0$, to calculate $p^r_{(S,H,F)}$ and $d$.

Case 1: There are several branches, and the lengths of the branches are equal. Then a block is mined by honest miners, and honest miners get nephew rewards. There are three subcases, as follows.

Case 1.1: The system is in the state (1, 1, 0). There is a fork with one private branch and two public branches. If honest miners produce blocks, honest miners can get two nephew rewards. Thus, $p^r_{(1,1,0)} = 2p_\beta$, and $d = 1$.

Case 1.2: The system is in the state (1, 1, 0′). There is a fork with one private branch and one public branch. If honest miners produce blocks on one of the two branches, honest miners can get a nephew rewards. Thus, $p^r_{(1,1,0')} = p_\beta$, and $d = 1$.

Case 1.3: The system is in the state (0, 1, 0). There is a fork with two public branches. honest miners can get a nephew reward when honest miners produce blocks. Thus, $p^r_{(0,1,0)} = p_\beta$ and $d = 1$.

Case 2: The system is in the state $(i+H+F, H, F)$, and $i \geq 2$. Honest miners can get a nephew reward after they create $i$ blocks. There are three subcases as follows.

Case 2.1: The system is in the state $(i,\,0,\,0')$, $i \geq 2$. The length of the public branch is zero, and honest miners can produce a block

or create a fork on the consensus block. Honest miners can get nephew rewards when honest miners produce regular blocks. In this case, $p^r_{(i,0,0')} = (p_s + 2p_f)p_\beta^{i-1}$ and $d = i$, where $i \geq 2$.

Case 2.2: The system is in the state $(i + H + F, H, F)$, and $i \geq 3$, $H \geq 1$ and $F \neq 0'$. Honest miners can see three branches, so honest miners produce in the private branch is $\gamma_3$. Honest miners can get nephew rewards when honest miners produce regular blocks. Thus, $p^r_{(i+H+F,H,F)} = (p_s + 2p_f)\gamma_3 p_\beta^{i-1}$, and $d = i$, where $i \geq 3$, $H \geq 1$ and $F \neq 0'$.

Case 2.3: The system is in the state $(i + H, H, 0')$, and $i \geq 3$, $H > 1$. In this case, honest miners can see two branches, so honest miners produce in the private branch is $\gamma_2$. Honest miners can get nephew rewards when honest miners produce regular blocks. Thus, $p^r_{(i+H,H,0')} = (p_s + 2p_f)\gamma_2 p_\beta^{i-1}$ and $d = i$, where $i \geq 3$, $H \geq 1$ and $F \neq 0'$.

In summary, Eq. (B.8) is proved. ■

**Proposition 7.** Uncle rewards for honest miners when the pool adopts **S2** are shown as Eq. (B.9).

$$R_{U,H,2} = R_{U,H,1} \cdot p_\beta \tag{B.9}$$

**Proof**: If the selfish pool adopts S1, both selfish pool and honest miners reference honest miner's blocks. As the selfish pool adopts **S2**, only honest miners reference uncle blocks. The probability of honest miners generating blocks is $p_\beta$. Thus, $R_{U,H,2} = R_{U,H,1} \cdot p_\beta$. ■

## References


Bai, Qianlan, Zhou, Xinyan, Wang, Xing, Xu, Yuedong, Wang, Xin, Kong, Qingsheng, 2019. A deep dive into blockchain selfish mining. ICC 1–6.

Bez, Mirko, Fornari, Giacomo, Vardanega, Tullio, 2019. The scalability challenge of Ethereum: an initial quantitative analysis. SOSE.

Buterin, Vitalik, 2014. A next-generation smart contract and decentralized application platform.. White Pap. 3.37.

Chicarino, Vanessa R.L., Albuquerque, Célio, Jesus, Emanuel Ferreira, Rocha, Antônio A.de A., 2020. On the detection of selfish mining and stalker attacks in blockchain networks. Ann. des Télécommun. 75 (3-4), 143–152.

da Silva, Francisco J.C., Damsgaard, Sebastian B., Sorensen, Mikki A.M., et al., 2019. Analysis of Blockchain forking on an ethereum network. Eur. Wirel..

Dong, Xuewen, Wu, Feng, Faree, Anter, Guo, Deke, Shen, Yulong, Ma, Jianfeng, 2019. Selfholding: a combined attack model using selfish mining with block withholding attack. Comput. Secur. 87.

etherchain.org. Accessed: May, 2020.

Eyal, Ittay, Sirer, Emin Gün, 2014. Majority is not enough: Bitcoin mining is vulnerable. Financ. Cryptogr. 436–454.

Feng, Chen, Niu, Jianyu, 2019. Selfish mining in Ethereum. ICDCS 1306–1316.

Feng, Yu, Torlak, Emina, Bodík, Rastislav, 2019. Precise attack synthesis for smart contracts. CoRR abs/1902 06067.

Göbel, Johannes, Keeler, Holger Paul, Krzesinski, Anthony E, Taylor, Peter G., 2016. Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. Perform. Eval. 104, 23–41.

Grunspan, Cyril, Pérez-Marco, Ricardo, 2019a. Selfish mining in Ethereum. CoRR abs/1904 13330.

Grunspan, Cyril, Pérez-Marco, Ricardo, 2019b. Selfish mining and Dyck words in Bitcoin and Ethereum networks. CoRR abs/1904 07675.

https://coinmarketcap.com/charts/. Accessed: May, 2020.

https://ethereum.org/what-is-ethereum/. Accessed: Jan.30, 2020.

https://www.cryptoglobe.com/latest/2019/01/ethereum-classic-etc-network-s-hashpower-consolidation-is-not-51-attack-developers-claim/. Accessed: May, 2020.

https://www.maplesoft.com/. Accessed: May, 2020.

Jang, Jehyuk, Lee, Heung-No, 2019. Profitable double-spending attacks. CoRR abs/1903 01711.

Preethi Kasireddy: How does Ethereum work, anyway? published on Medium (https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369) (2017).

Liu, Bing, Qin, Yang, Chu, Xiaowen, 2019. Reducing forks in the Blockchain via probabilistic verification. In: ICDE Workshops, pp. 13–18.

Mišić, Jelena, Mišić, Vojislav B., Chang, Xiaolin, Motlagh, Saeideh Gholamrezazadeh, Ali, Zulfiker M., 2019a. Modeling of Bitcoin's blockchain delivery network. IEEE Trans. Netw. Sci. Eng. DOI: 10.1109/TNSE.2019.2928716.

Misic, Vojislav B., Misic, Jelena V., Chang, Xiaolin, 2019. On forks and fork characteristics in a Bitcoin-like distribution network. Blockchain 212–219.

Misic, Jelena, Misic, Vojislav B., Chang, Xiaolin, 2019b. On ledger inconsistency time in Bitcoin's blockchain delivery network. GLOBECOM 1–6.

Satoshi Nakamoto and A Bitcoin: a peer-to-peer electronic cash system. Bitcoin.–URL: https://bitcoin.org/bitcoin.pdf (2008).

Nayak, Kartik, Kumar, Srijan, Miller, Andrew, Shi, Elaine, 2016. Stubborn mining: generalizing selfish mining and combining with an eclipse attack. EuroS&P 305–320.

Pachal, Soumen, Ruj, Sushmita, 2019. Rational mining of Bitcoin. COMSNETS 1–8.

Ritz, Fabian, Zugenmaier, Alf, 2018. The impact of uncle rewards on selfish mining in Ethereum. In: EuroS&P Workshops, pp. 50–57.

Rosenfeld, Meni, 2014. Analysis of hashrate-based double spending. CoRR abs/1402 2009.

Rot, Artur, Blaicke, Bartosz, 2019. Blockchain's future role in cybersecurity. Analysis of defensive and offensive potential leveraging blockchain-based platforms. ACIT 447–451.

Saad, Muhammad, Njilla, Laurent, Charles, A, Kamhoua, Aziz Mohaisen, 2019. Countering selfish mining in Blockchains. ICNC 360–364.

Shahsavari, Yahya, Zhang, Kaiwen, Talhi, Chamseddine, 2019. A theoretical model for fork analysis in the Bitcoin network. Blockchain 237–244.

Shurov, Artem, Malevanniy, Daniil, Iakushkin, Oleg, Korkhov, Vladimir, 2019. Blockchain network threats: the case of PoW and ethereum. ICCSA 2, 606–617.

Sompolinsky, Yonatan, Zohar, Aviv, 2015. Secure high-rate transaction processing in Bitcoin. Financ. Cryptogr. 507–527.

Volety, Tejaswi, Saini, Shalabh, McGhin, Thomas, Liu, Charles Zhechao, Choo, Kim-Kwang Raymond, 2019. Cracking Bitcoin wallets: I want what you have in the wallets. Future Gener. Comput. Syst. 91, 136–143.

Wang, Shengling, Wang, Chenyu, Hu, Qin, 2019. Corking by forking: vulnerability analysis of Blockchain. INFOCOM 829–837.

Wang, Ziyu, Liu, Jianwei, Wu, Qianhong, Zhang, Yanting, Yu, Hui, Zhou, Ziyu, 2019. An analytic evaluation for the impact of uncle blocks by selfish and stubborn mining in an imperfect Ethereum network. Comput. Secur. 87.

Zhang, Shijie, Lee, Jong-Hyouk, 2019. Double-spending with a sybil attack in the bitcoin decentralized network. IEEE Trans. Ind. Inform. 15 (10), 5715–5722.



**Runkai Yang** received his B.Eng. in Computer Science and Technology from Beijing Information Science and Technology University, China, in 2016. He received his M.Eng. in Computer Technology from Beijing Jiaotong University, China, in 2018. He is pursuing a Ph.D. degree in Cyberspace Secure at Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, China. His research interests include virtualization technology, cloud computing, and blockchain security.

**Xiaolin Chang** is Professor at School of Computer and Information Technology, Beijing Jiaotong University. She has published over 80 papers in journals and conferences. Her current research interests include Cloud-edge computing, cybersecurity, secure and dependable in machine learning. She is a member of IEEE.

**Jelena Mišić** is Professor of Computer Science at Ryerson University in Toronto, Ontario, Canada. She has published over 120 papers in archival journals and close to 200 papers at international conferences in the areas of wireless networks, in particular wireless personal area network and wireless sensor network protocols, performance evaluation, and security. She serves on editorial boards of IEEE Transactions on Vehicular Technology, Computer Networks, Ad hoc Networks, Security and Communication Networks, Ad Hoc & Sensor Wireless Networks, Int. Journal of Sensor Networks, and Int. Journal of Telemedicine and Applications. She is a Fellow of IEEE and Member of ACM.

**Vojislav B. Mišić** is Professor of Computer Science at Ryerson University in Toronto, Ontario, Canada. He received his PhD in Computer Science from University of Belgrade, Serbia, in 1993. His research interests include performance evaluation of wireless networks and systems and software engineering. He has authored or co-authored six books, 20 book chapters, and over 280 papers in archival journals and at prestigious international conferences. He serves on the editorial boards of IEEE transactions on Cloud Computing, Ad hoc Networks, Peer-to-Peer Networks and Applications, and International Journal of Parallel, Emergent and Distributed Systems. He is a Senior Member of IEEE and member of ACM.