APPENDIX A
DESCRIPTION OF THE MINING STRATEGY

We introduce some commonly used terms and describe the mining strategy as follows. Say, given an interval I_k, the interval's *absolute change* of hashes during a time period of length Δt is commonly defined as follows,

$$\int_{t_{k-1}+\Delta t}^{t_k+\Delta t} h_i(t)dt - \int_{t_{k-1}}^{t_k} h_i(t)dt \tag{1}$$

The *percent change* is the absolute change with respect to the current interval's hashes $\int_{t_{k-1}}^{t_k} h_i(t)dt$. It is commonly defined as follows,

$$\frac{\int_{t_{k-1}+\Delta t}^{t_k+\Delta t} h_i(t)dt - \int_{t_{k-1}}^{t_k} h_i(t)dt}{\int_{t_{k-1}}^{t_k} h_i(t)dt} \tag{2}$$

The *mining strategy* is the percent change with respect to the time period of length Δt, which is used to describe the rate of the percent change. It is commonly defined as follows,

$$\frac{\int_{t_{k-1}+\Delta t}^{t_k+\Delta t} h_i(t)dt - \int_{t_{k-1}}^{t_k} h_i(t)dt}{\Delta t \int_{t_{k-1}}^{t_k} h_i(t)dt} \tag{3}$$

where the mining strategy has two forms: the *continuous-time mining strategy* and the *discrete-time mining strategy*. Besides, the relationship between the continuous-time mining strategy and the discrete-time mining strategy is as follows: We can use the mathematical property of the continuous-time mining strategy to calculate the hash rate function $\int_{t_{k-1}}^{t_k} h_i(t)dt$, from which the discrete-time mining strategy with $\Delta t = |t_{k+1} - t_k|$ is simply available.

APPENDIX B
ESTIMATE DAILY COMPUTING POWER

We introduce the method of estimating computing power. People commonly use the concept of Bitcoin *mining difficulty* to estimate the hash rate of the Bitcoin network.

Let denote the mining difficulty of the Bitcoin network at time t by $\xi(t)$. Bitcoin uses $\xi(t)$ to roughly guarantee that the duration of each block takes 600 seconds on average. Vividly, Bitcoin uses the historical block times to predict future block times. Say, $\xi(t)$ changes every 2,016 blocks time. For each change, its value for the future 2,016 blocks is set, such that the rate for the historical 2,016 blocks equals 600 seconds per block. Hence, the relationship between $\xi(t)$ and the hash rate of the Bitcoin network at time t can be roughly predicted as follows,

$$\sum_{i=1}^{N} h_i(t) \approx \frac{2^{32}\xi(t)}{600 \text{ secs}} \tag{4}$$

where $t \in I_k$ and the length of I_k equals one day. However, mining difficulty has a *property* that after each change, $\xi(t)$ will remain a constant in a period of 2,016 blocks (i.e., around 2,016 $\times$ 600 seconds) until the next change. Such a property leads to a problem in Eq. (4), where it predicts that the hash rate of the Bitcoin network should remain a specific value in the next period of 2,016 $\times$ 600 seconds. Nevertheless, in real-world situations, the hash rate of the Bitcoin network may be different from that specific value.

Therefore, we need to measure the real-world hash-rate related data to adjust the Eq. (4). Based on the Bitcoin mining mechanism, we measure the number of blocks found by mining pools, and have a better estimation on the hash rate of the Bitcoin network at time t as follows,

$$\sum_{i=1}^{N} h_i(t) \approx \frac{\#_{\text{actual bitcoin}}}{\#_{\text{expected bitcoin}}} \times \frac{2^{32}\xi(t)}{600 \text{ secs}} \tag{5}$$

where $t \in I_k$ and the length of I_k equals one day. The symbol $\#_{\text{actual bitcoin}}$ means the actual number of blocks found on the Bitcoin network in the interval. And, the symbol $\#_{\text{expected bitcoin}}$ means the expected number of blocks to be found on the Bitcoin network in this interval (i.e., $\#_{\text{expected bitcoin}} = \left\lfloor \frac{t_k - t_{k-1}}{600 \text{ secs}} \right\rfloor$ blocks), where each block is mathematically expected to take exactly 600 seconds.

Similarly, the number of hashes of the Bitcoin network in the interval is as follows,

$$\sum_{i=1}^{N} \int_{t_{k-1}}^{t_k} h_i(t)dt \approx \frac{\#_{\text{actual bitcoin}}}{\#_{\text{expected bitcoin}}} \times \int_{t_{k-1}}^{t_k} \frac{2^{32}\xi(t)}{600 \text{ secs}} dt \tag{6}$$

From Eq. (5), the hash rate of mining pool i at time t can be estimated as follows,

$$h_i(t) \approx \frac{\#_{\text{actual pool } i}}{\#_{\text{expected bitcoin}}} \times \frac{2^{32}\xi(t)}{600 \text{ secs}} \tag{7}$$

where $t \in I_k$ and the length of I_k equals one day. The symbol $\#_{\text{actual pool } i}$ means the actual number of blocks found by mining pool i in the interval.

Similarly, the number of hashes of mining pool i in the interval is as follows,

$$\int_{t_{k-1}}^{t_k} h_i(t)dt \approx \frac{\#_{\text{actual pool } i}}{\#_{\text{expected bitcoin}}} \times \int_{t_{k-1}}^{t_k} \frac{2^{32}\xi(t)}{600 \text{ secs}} dt \tag{8}$$

where the definite integral $\int_{t_{k-1}}^{t_k} \xi(t)dt$ is simple since the value of $\xi(t)$ changes once every $20,160$ minutes. And the mining difficulty is a field of Bitcoin block header that is permanently stored in the Bitcoin blockchain. By default, the related parameters about daily computing power are estimated with a reference to Eqs. (5-8).

APPENDIX C
PROOFS

C.1. Proof of Proposition 4.5

Proof: Let the number of hashes of new mining pools in the interval I_k be $H_j(I_k)$, where $j \notin [1, N]$ and $H_j(I_k) = 0$. In the next interval I_{k+1}, it will add $H_j(I_{k+1})$ hashes to the Bitcoin network, where $H_j(I_{k+1}) > 0$ since mining activities are profitable. Also we have $R(I_{k+1}) = R(I_k)$ and $\sum_{i=1}^{N} H_i(I_{k+1}) = \sum_{i=1}^{N} H_i(I_k)$ since $R(I_k)$ and $\sum_{i=1}^{N} H_i(I_k)$ are stable. Therefore, the problem of adding $H_j(I_{k+1})$ hashes to the Bitcoin network for maximizing the expected payoff of new mining pools in the next interval I_{k+1} can be formulated as the optimization problem (9).

$$\begin{aligned} \underset{H_j(I_{k+1})}{\text{maximize}} \quad & \frac{H_j(I_{k+1})R(I_k)}{H_j(I_{k+1}) + \sum_{i=1}^{N} H_i(I_k)} - cH_j(I_{k+1}) \\ \text{subject to} \quad & \frac{R(I_k)}{\sum_{i=1}^{N} H_i(I_k)} > c \end{aligned} \tag{9}$$

It implies that the solutions of either increasing too much computing power or too little, to the Bitcoin network are not optimal. Therefore, we are going to find the optimal one. Specifically, we let the first derivative $\frac{\partial}{\partial H_j(I_{k+1})}\left[\frac{H_j(I_{k+1})R(I_k)}{H_j(I_{k+1})+\sum_{i=1}^{N} H_i(I_k)} - cH_j(I_{k+1})\right] = 0$, and get two saddle points $H_j(I_{k+1}) = -\sqrt{\frac{R(I_k)\sum_{i=1}^{N} H_i(I_k)}{c}} - \sum_{i=1}^{N} H_i(I_k)$, $H_j(I_{k+1}) = \sqrt{\frac{R(I_k)\sum_{i=1}^{N} H_i(I_k)}{c}} - \sum_{i=1}^{N} H_i(I_k)$, respectively. The first saddle point can be removed since it does not satisfy the condition of $H_j(I_{k+1}) > 0$. The second saddle point can be proven to be the optimal one since the second order derivative $\frac{\partial^2}{\partial H_j(I_{k+1})\partial H_j(I_{k+1})}\left[\frac{H_j(I_{k+1})R(I_k)}{H_j(I_{k+1})+\sum_{i=1}^{N} H_i(I_k)} - cH_j(I_{k+1})\right] = -2\sqrt{cR(I_k)\sum_{i=1}^{N} H_i(I_k)} < 0$. ■

C.2. Proof of Proposition 4.6

Proof: Let the number of hashes of an existing mining pool i in the interval I_k be $H_i(I_k)$, where $i \in [1, N]$, $H_i(I_k) > 0$. Mining pool i will increase/decrease $H_{i'}(I_{k+1})$ hashes to/from the Bitcoin network in the next interval I_{k+1}. Also, $R(I_{k+1}) = R(I_k)$, $\sum_{i=1}^{N} H_i(I_{k+1}) = \sum_{i=1}^{N} H_i(I_k)$ since $R(I_k)$ and $\sum_{i=1}^{N} H_i(I_k)$ are stable. Moreover, there are no new mining pools. Therefore, the problem of increasing/decreasing $H_{i'}(I_{k+1})$ hashes to/from the Bitcoin network for maximizing the expected payoff of mining pool i in the next interval can be formulated as the optimization problem (10).

$$\begin{aligned} \underset{H_{i'}(I_{k+1})}{\text{maximize}} \quad & [H_{i'}(I_{k+1}) + H_i(I_k)] \times \\ & \left[\frac{R(I_k)}{H_{i'}(I_{k+1}) + \sum_{i=1}^{N} H_i(I_k)} - c\right] \\ \text{subject to} \quad & \frac{R(I_k)}{\sum_{i=1}^{N} H_i(I_k)} > c \end{aligned} \tag{10}$$

We are going to find the optimal solution. And similar to the previous optimization problem (9), the optimal solution is $\sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k)\right)}{c}} - \sum_{i=1}^{N} H_i(I_k)$. It means that for an existing mining pool i whose hashes satisfy $H_i(I_k) < \left(1 - \frac{c\sum_{i=1}^{N} H_i(I_k)}{R(I_k)}\right)\sum_{i=1}^{N} H_i(I_k)$, the optimal strategy is to add $\sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k)\right)}{c}} - \sum_{i=1}^{N} H_i(I_k)$ hashes to the Bitcoin network in the next interval. Otherwise, the optimal solution is to decrease $\sum_{i=1}^{N} H_i(I_k) - \sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k)\right)}{c}}$ hashes from the Bitcoin network in the next interval. ■

C.3. Proof of Proposition 4.8

Proof: Let the number of hashes of an existing mining pool i in the interval I_k be $H_i(I_k)$, where $i \in [1, N]$, $H_i(I_k) > 0$. Mining pool i will decrease $H_{i'}(I_{k+1})$ hashes from the Bitcoin network in the next interval I_{k+1} where $H_{i'}(I_{k+1}) > 0$ since mining activities are non-profitable. Also, $R(I_{k+1}) = R(I_k)$ and $\sum_{i=1}^{N} H_i(I_{k+1}) = \sum_{i=1}^{N} H_i(I_k)$ since $R(I_k)$, $\sum_{i=1}^{N} H_i(I_k)$ are stable. Therefore, the problem of decreasing $H_{i'}(I_{k+1})$ hashes from the Bitcoin network for maximizing the expected payoff of mining pool i in the next interval can be formulated as the optimization problem (11).

$$\begin{aligned} \underset{H_{i'}(I_{k+1})}{\text{maximize}} \quad & [-H_{i'}(I_{k+1}) + H_i(I_k)] \times \\ & \left[\frac{R(I_k)}{-H_{i'}(I_{k+1}) + \sum_{i=1}^{N} H_i(I_k)} - c\right] \\ \text{subject to} \quad & \frac{R(t)}{\sum_{i=1}^{N} H_i(I_k)} \leq c \end{aligned} \tag{11}$$

We are going to find the optimal solution. There are two cases.

Case 1. If an existing mining pool i whose hashes satisfy $\frac{R(I_k)}{-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k)} > c$, then the optimization problem (11) can be represented the same form as the optimization problem (9). Therefore, after applying the proposition 4.5, we get the optimal solution $H_{i'}(I_{k+1}) = \sum_{i=1}^{N} H_i(I_k) - \sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k)\right)}{c}}$ where $H_{i'}(I_{k+1}) > 0$.

Case 2. If an existing mining pool i whose hashes satisfy $\frac{R(I_k)}{-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k)} \leq c$, then intuitively the optimal solution is to decrease all hashes $H_{i'}(I_{k+1})$ where $H_{i'}(I_{k+1}) = H_i(I_k)$ from the Bitcoin network in the next interval. ■

C.4. Proof of Theorem 4.9

Proof: Given a stable mining revenue R. The number of hashes of the Bitcoin network is $\sum_{i=1}^{N} H_i(I_k)$ hashes in interval I_k. We are going to prove that $\lim_{n\to\infty} \sum_{i=1}^{N} H_i(I_{k+n})$ will converge to a Nash equilibrium point among mining pools. There are two cases.

Case 1. If mining activities are profitable, then we apply the proposition 4.5, and know that extra hashes $\sqrt{\frac{R\sum_{i=1}^{N} H_i(I_k)}{c}} - \sum_{i=1}^{N} H_i(I_k)$ will be added to the Bitcoin network by new mining pools. It means that the number of hashes of the Bitcoin network in the next interval will be $\sum_{i=1}^{N} H_i(I_{k+1}) = \sum_{i=1}^{N} H_i(I_k) +$

$\left(\sqrt{\frac{R\sum_{i=1}^{N} H_i(I_k)}{c}} - \sum_{i=1}^{N} H_i(I_k)\right) = \sqrt{\frac{R\sum_{i=1}^{N} H_i(I_k)}{c}}$ hashes. So on and so forth, after n intervals, we have $\lim_{n\to\infty} \sum_{i=1}^{N} H_i(I_{k+n}) = \frac{R}{c}$.

Case 2. If mining activities are non-profitable, then we repeatedly apply the proposition 4.8, and know that the optimal solution for a major mining pool i to maximize its expected payoff is to decrease $\sum_{i=1}^{N} H_i(I_k) - \sqrt{\frac{(-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k))R}{c}} > 0$ hashes from the Bitcoin network in the next interval, and finally go to *Case 1* of this proof. ■

APPENDIX D
CASE STUDY

D.1. Mining Revenue Increases Dramatically

We will analyze the best-response strategy if the mining revenue increases dramatically. We make the following assumptions:

Assumption 1.1: Two existing mining pools X, Y, and one new mining pool Z with corresponding hashes of $H_x(I_k)$, $H_y(I_k)$, $H_z(I_k)$ in the interval I_k, where,

$$H_x(I_k) > 0; H_y(I_k) > 0; H_z(I_k) = 0 \tag{12}$$

$$\sum_{i=1}^{N} H_i(I_k) = H_x(I_k) + H_y(I_k) + H_z(I_k) \tag{13}$$

Assumption 1.2: The mining activities are profitable and satisfy the following conditions,

$$\begin{aligned} H_x(I_k) &< \left(1 - \frac{c\sum_{i=1}^{N} H_i(I_k)}{R(I_k)}\right) \sum_{i=1}^{N} H_i(I_k) \\ H_y(I_k) &< \left(1 - \frac{c\sum_{i=1}^{N} H_i(I_k)}{R(I_k)}\right) \sum_{i=1}^{N} H_i(I_k) \end{aligned} \tag{14}$$

Assumption 1.3: The mining revenue is highly profitable, where,

$$R(I_{k+1}) = R(I_k) \tag{15}$$

Case 1 $\{Increase, Increase, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_x(I_{k+1}) \tag{16}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_y(I_{k+1}) \tag{17}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{18}$$

Case 2 $\{Increase, Unchange, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_{k+1})} - cH_x(I_{k+1}) \tag{19}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_{k+1})} - cH_y(I_k) \tag{20}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{21}$$

Case 3 $\{Unchange, Increase, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_x(I_k) \tag{22}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_y(I_{k+1}) \tag{23}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{24}$$

Case 4 $\{Unchange, Unchange, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_{k+1})} - cH_x(I_k) \tag{25}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_{k+1})} - cH_y(I_k) \tag{26}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_k) + H_y(I_k) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{27}$$

Case 5 $\{Increase, Increase, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_k)} - cH_x(I_{k+1}) \tag{28}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_k)} - cH_y(I_{k+1}) \tag{29}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_k)} - cH_z(I_k) \tag{30}$$

Case 6 $\{Increase, Unchange, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_k)} - cH_x(I_{k+1}) \tag{31}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_k)} - cH_y(I_k) \tag{32}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_k)} - cH_z(I_k) \tag{33}$$

Case 7 $\{Unchange, Increase, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_k)} - cH_x(I_k) \tag{34}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_k)} - cH_y(I_{k+1}) \tag{35}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_k)} - cH_z(I_k) \tag{36}$$

Case 8 $\{Unchange, Unchange, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_k)} - cH_x(I_k) \tag{37}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_k)} - cH_y(I_k) \tag{38}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_k)} - cH_z(I_k) \tag{39}$$

D.2. Mining Revenue Decreases Dramatically

We will analyze the best-response strategy if the mining revenue decreases dramatically. We make the following assumptions:

Assumption 2.1: Two existing mining pools X, Y, and one new mining pool Z with corresponding hashes of $H_x(I_k)$, $H_y(I_k)$, $H_z(I_k)$ in the interval I_k, where,

$$H_x(I_k) > 0; H_y(I_k) > 0; H_z(I_k) = 0 \tag{40}$$

$$\sum_{i=1}^{N} H_i(I_k) = H_x(I_k) + H_y(I_k) + H_z(I_k) \tag{41}$$

Assumption 2.2: The mining activities are profitable and satisfy the following conditions,

$$\begin{aligned} H_x(I_k) &\geq \left(1 - \frac{c\sum_{i=1}^{N} H_i(I_k)}{R(I_k)}\right) \sum_{i=1}^{N} H_i(I_k) \\ H_y(I_k) &\geq \left(1 - \frac{c\sum_{i=1}^{N} H_i(I_k)}{R(I_k)}\right) \sum_{i=1}^{N} H_i(I_k) \end{aligned} \tag{42}$$

Assumption 2.3: The mining revenue is slightly profitable, where,

$$R(I_{k+1}) = R(I_k) \tag{43}$$

Case 1 $\{Decrease, Decrease, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_x(I_{k+1}) \tag{44}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_y(I_{k+1}) \tag{45}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{46}$$

Case 2 $\{Decrease, Unchange, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_{k+1})} - cH_x(I_{k+1}) \tag{47}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_{k+1})} - cH_y(I_k) \tag{48}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{49}$$

Case 3 $\{Unchange, Decrease, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_x(I_k) \tag{50}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_y(I_{k+1}) \tag{51}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{52}$$

Case 4 $\{Unchange, Unchange, Increase\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_{k+1})} - cH_x(I_k) \tag{53}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_{k+1})} - cH_y(I_k) \tag{54}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_{k+1})}{H_x(I_k) + H_y(I_k) + H_z(I_{k+1})} - cH_z(I_{k+1}) \tag{55}$$

Case 5 $\{Decrease, Decrease, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_k)} - cH_x(I_{k+1}) \tag{56}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_k)} - cH_y(I_{k+1}) \tag{57}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_{k+1}) + H_y(I_{k+1}) + H_z(I_k)} - cH_z(I_k) \tag{58}$$

Case 6 $\{Decrease, Unchange, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_{k+1})}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_k)} - cH_x(I_{k+1}) \tag{59}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_k)} - cH_y(I_k) \tag{60}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_{k+1}) + H_y(I_k) + H_z(I_k)} - cH_z(I_k) \tag{61}$$

Case 7 $\{Unchange, Decrease, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_k)} - cH_x(I_k) \tag{62}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_{k+1})}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_k)} - cH_y(I_{k+1}) \tag{63}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_k) + H_y(I_{k+1}) + H_z(I_k)} - cH_z(I_k) \tag{64}$$

Case 8 $\{Unchange, Unchange, Unchange\}$. The expected payoff of mining pool X, Y, Z in the next interval I_{k+1} will be,

$$\mathbb{E}\left[f_x(I_{k+1})\right] = \frac{R(I_{k+1})H_x(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_k)} - cH_x(I_k) \tag{65}$$

$$\mathbb{E}\left[f_y(I_{k+1})\right] = \frac{R(I_{k+1})H_y(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_k)} - cH_y(I_k) \tag{66}$$

$$\mathbb{E}\left[f_z(I_{k+1})\right] = \frac{R(I_{k+1})H_z(I_k)}{H_x(I_k) + H_y(I_k) + H_z(I_k)} - cH_z(I_k) \tag{67}$$

APPENDIX E
TRANSACTION SIZE AFTER VS. BEFORE SEGWIT

E.1. Description of the Transaction Size

Segregated Witness (SegWit) is a soft fork of the Bitcoin protocol that separates the witness data (i.e., digital signatures) from the total transaction data and allows more base transaction data (i.e., without the witness data) to be stored in a single block. SegWit introduces a new unit named *weight unit* (WU) to measure the size of transaction data and block data, and transitions to a new block limit of 4 million WU. The existing 1 MB block limit still holds for these non-SegWit nodes. The relationship between *weight unit* and *byte* is as follows,

$$weight\,unit = base\,size\,in\,byte \times 3 + total\,size\,in\,byte \tag{68}$$

where the base size does not include the witness data while the total size includes the witness data. The equation (68) holds for both transaction and block. That is, a transaction's weight unit equals three times of the base transaction size in byte plus the total transaction size in byte. And a block's weight unit equals three times of the base block size in byte plus the total block size in byte. In practice, both weight unit and total size are directly available via querying the Bitcoin full node. For example, we can query the weight unit and the total size of a transaction using the '*bitcoin-cli getrawtransaction true*' command. Similarly, we can query the weight unit and the total size of a block using the '*bitcoin-cli getblock*' command.

Bitcoin also uses *virtual size* in *vbyte* to measure the size of transaction and block, where the relationship between *vbyte* and *weight unit* is as follows,

$$\begin{aligned} virtual\,size &= weight\,unit/4 \\ &= base\,size\,in\,byte + witness\,data\,in\,byte/4 \end{aligned} \tag{69}$$

where the equation (69) also holds for both transaction and block. It implies that the new block limit of 4 million WU measured in virtual size is 1 million vbytes. In practice, we can query the virtual size of a transaction using the '*bitcoin-cli getrawtransaction true*' command.

E.2. Comparison of the Transaction Size

We will use both the virtual size and the total size to study the transaction size after versus before SegWit activation. Before SegWit activation, there is no SegWit transaction; hence, all transactions are non-SegWit. After SegWit activation, transactions can be grouped into non-SegWit transactions and SegWit transactions.

TABLE I compares the transaction size in byte after versus before SegWit activation on a SegWit-compatible Bitcoin full node from Feb 26, 2016, to Nov 25, 2020. First, we can see that the average transaction size in byte (i.e., overall) of top mining pools after SegWit activation is larger than its size (i.e., overall) before SegWit activation since the transaction from Bitcoin end-user is getting larger in byte. For example, the average transaction size of AntPool after SegWit is 534.19 bytes, which is larger than its size before SegWit (i.e., 509.66 bytes).

TABLE I
TRANSACTION SIZE IN BYTE AFTER VS. BEFORE SEGWIT ACTIVATION ON A SEGWIT-COMPATIBLE BITCOIN FULL NODE FROM FEB 26, 2016 TO NOV 25, 2020

Mining Pool		Before SegWit (Overall)	After SegWit		
			Non-SegWit	SegWit	Overall
AntPool	Min	127	157	190	157
	Max	99,877	99,946	224,432	224,432
	Mean	**509.66**	477.04	642.24	**534.19**
	Median	226	226	283	248
	Std	2095.71	2059.78	3000.18	2427.97
F2Pool	Min	69	104	96	96
	Max	188,477	240,181	224,850	240,181
	Mean	**509.79**	463.18	637.98	**535.77**
	Median	226	226	283	249
	Std	2254.78	2202.60	3100.71	2614.74
BTC.com	Min	108	85	190	85
	Max	99,887	99,925	224,447	224,447
	Mean	**506.94**	479.98	634.51	**536.55**
	Median	226	226	283	249
	Std	2057.84	2043.16	3086.50	2477.78
ViaBTC	Min	97	85	189	85
	Max	99,905	99,999	188,537	188,537
	Mean	**514.57**	486.20	659.04	**546.17**
	Median	226	226	283	249
	Std	2094.55	2118.90	3255.24	2572.04

TABLE II
TRANSACTION SIZE IN VBYTE AFTER VS. BEFORE SEGWIT ACTIVATION ON A SEGWIT-COMPATIBLE BITCOIN FULL NODE FROM FEB 26, 2016 TO NOV 25, 2020

Mining Pool		Before SegWit (Overall)	After SegWit		
			Non-SegWit	SegWit	Overall
AntPool	Min	127	157	109	109
	Max	99,877	99,946	99,826	99,946
	Mean	**509.66**	477.04	395.39	**448.80**
	Median	226	226	198	225
	Std	2095.71	2059.78	1788.12	1970.42
F2Pool	Min	69	104	86	86
	Max	188,477	240,181	99,704	240,181
	Mean	**509.79**	463.18	391.54	**433.43**
	Median	226	226	200	225
	Std	2254.78	2202.60	1775.51	2036.45
BTC.com	Min	108	85	100	85
	Max	99,887	99,925	99,581	99,925
	Mean	**506.94**	479.98	391.67	**447.65**
	Median	226	226	190	225
	Std	2057.84	2043.16	1844.19	1973.10
ViaBTC	Min	97	85	100	85
	Max	99,905	99,999	99,919	99,999
	Mean	**514.57**	486.20	405.84	**458.32**
	Median	226	226	198	225
	Std	2094.55	2118.90	1921.27	2052.84

Second, we can see that the average size in byte of the SegWit transaction is larger than the non-SegWit transaction after SegWit activation. For example, the average size of the SegWit transaction of AntPool is 642.24 bytes, which is larger than the average size of the non-SegWit transaction after SegWit activation (i.e., 477.04 bytes). It implies that larger transaction (in byte) prefers SegWit than non-SegWit.

TABLE II compares the transaction size in vbyte after versus before SegWit activation. First, we can see that the average transaction size in vbyte (i.e., overall) of top mining pools after SegWit activation is smaller than its size before SegWit activation, because the witness data in vbyte is only counted as one quarter of its size in byte (see Eq. (69)). For example, the average transaction size in vbyte of AntPool after SegWit is 448.80 vbytes, which is smaller than its size before SegWit (i.e., 509.66 vbytes). Second, we can see that the average size in vbyte of the SegWit transaction is

smaller than the non-SegWit transaction after SegWit activation. For example, the average size of the SegWit transaction of AntPool is 395.39 vbytes, which is smaller than the average size of the non-SegWit transaction after SegWit activation (i.e., 477.04 vbytes).

In summary, first, we find that the average transaction size in byte after SegWit activation is larger than its size before SegWit activation, since transaction from Bitcoin end-user is getting larger in byte; however, the average transaction size in vbyte after SegWit activation is smaller than its size before SegWit activation, because the witness data in vbyte is only counted as one quarter of its size in byte (see Eq. (69)). Second, we find that larger transaction in byte prefers SegWit than non-SegWit since the virtual size of witness data is counted smaller in vbyte under the new block limit of 1 million vbytes (see Eq. (69)).