Sucker punch makes you richer

Rethinking incentives in Proof-of-Work-based Blockchains

Runchao Han
Monash University and CSIRO-Data61
runchao.han@monash.edu

Zhimei Sui
Monash University
zhimeisui@gmail.com

Jiangshan Yu[*]
Monash University
jiangshan.yu@monash.edu

Joseph Liu
Monash University
joseph.liu@monash.edu

Shiping Chen
CSIRO-Data61
shiping.chen@data61.csiro.au

Abstract

Honest majority is the key security assumption of Proof-of-Work (PoW) based blockchains like Bitcoin. However, recent 51% attacks render this assumption unrealistic in practice. In this paper, we propose the *"sucker punch attack"*, where an attacker temporarily utilises external mining power to launch 51% attacks on a blockchain, and gains a better revenue than performing honest mining. The *sucker punch attack* indicates that the currently employed incentive mechanisms may incentivise profit-driven miners to turn into evil and break the "honest majority" assumption, rather than incentivising miners to stay honest and keep the system safe.

We develop a Markov Decision Process based model to evaluate the attack, and provide an anslysis on the feasibility and profitability of launching *sucker punch attacks* on mainstream PoW-based blockchains. Our results show that the attacks are feasible and profitable on most of them. In addition, we also leverage our model to investigate the recent 51% attack on Ethereum Classic (Jan. 2019), which is suspected to be an incident of our sucker punch attacks. We provide insights on the attacker strategy and expected revenue, and show that the attacker's strategy is near optimal.

1 Introduction

Proof-of-work (PoW) based consensus was first introduced by Bitcoin [25], to allow distributed nodes agreeing on the same global state of the system. In Bitcoin, all transactions are recorded as a chain of blocks. Anyone can create a block of transactions, and append it into the Bitcoin blockchain as a unique successor of the last block. To create a block, one needs to solve a crypto puzzle which is computationally hard. In particular, one needs to find a random nonce to make the hash value of the block smaller than a target value.

Different miners may create different valid blocks as successors of the same block, which creates a fork in the system. To resolve a fork, miners only accept the longest branch. However, a branch that is currently longer may be over taken by the other branch, and all transactions in the currently longer branch will be deem invalid. This creates an opportunity for the attacker to spend a coin in a branch, and later on creates another longer branch to erase this transaction. This is called "double spending attack". Intuitively, to launch a double spending attack, an attacker should have enough mining power to create a branch that is longer than the current one. This requires the attacker to control a majority of mining power in the network. An attacker with a majority of mining power to double spend is known as "51% attacks".

Key assumption. Currently, the security of all PoW-based blockchains relies on the assumption that the majority of mining power is honest. If this assumption does not hold, then 51% attacks would be possible. It is obvious that the security guarantee is relative rather than absolute, as the difficulty to break the assumption is directly related to the total mining power in the system. So, the more the mining power in the system, the more difficult to control 51% mining power of the entire system, and so the higher the security guarantee will be.

To encourage more miners to join the system to increase the mining power (so the security guarantee of the system), Bitcoin introduced an incentive mechanism. In particular, miners who solved a puzzle in Bitcoin will be rewarded for their contribution. The mining reward includes a pre-defined number of bitcoins and transaction fees of all transactions contained in the block. In this way, miners are encouraged to contribute their mining power to make the system more secure. This incentive concept has been employed by almost all major blockchains.

Fact v.s. Fiction. As mentioned, the honest majority assumption is relative to the mining power in the system. If there exists only one blockchain in the entire

[*]Corresponding author.

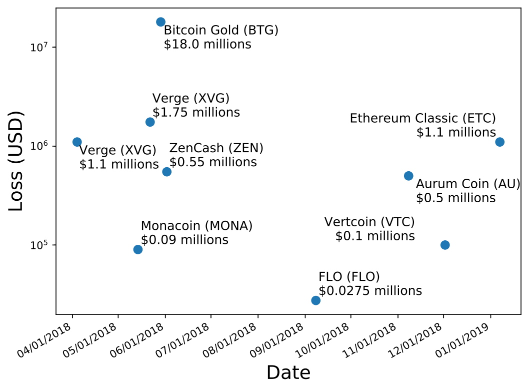


Figure 1: Successful 51% Attacks in 2018-2019 [19]. We omit the 51% attack on Litecoin Cash on June 4, 2018 as the loss is unknown.

world, then all potential miners will contribute their mining power into this blockchain. In this case, controlling 51% mining power could be extremely difficult. However, if there is a competing blockchain of the same mining algorithm, then the total available mining power for this algorithm in the world will be split into two communities, one for each blockchain. This reduces the difficulty of controlling 51% mining power in both of the blockchains. Currently, there are over 2,000 cryptocurrencies[1], and many of them share the same mining algorithm, so that no single blockchain can take full advantage of the proof-of-work mechanism for their security.

In 2016, bribery attacks [5] (a.k.a. cloud mining attack) considered the possibility of renting cloud mining power to gain a majority of mining power in Bitcoin-style blockchains. In fact, since 2018, many successful 51% attacks have been identified, as shown in Figure 1. These attacks result in the loss of more than $23 million US dollars. One main stream suspicion [24, 23] is that the attacks are equipped with cloud mining power from e.g. Nicehash[28].

Miners are rational. In 2005, years before the birth of Bitcoin, BAR (Byzantine, Altruistic, Rational) model [3] was considered in the context of fault tolerance for cooperative services. In this model, a participant can be Byzantine, altruistic, or rational. A Byzantine (a.k.a. malicious) participate would try everything to break the security of the system with any cost; an altruistic (a.k.a. honest) participant will always follow the system specification; and a rational attacker would only depart from the protocol specification and being Byzantine if s/he can gain more profit.

As the nature of the incentive mechanism in blockchains is to encourage miners joining the system with reward, so it is reasonable to assume that most miners are rational, rather than Byzantine or altruistic.

Mining power migration attack. Given the fact that (1) multiple blockchains share the same mining algorithm; (2) total mining power in different blockchains could be very different; and (3) miners are rational, we suspect the possibility of a new attack, which we call mining power migration attack.

In this attack, we consider two blockchains (BC_1 and BC_2) of the same type of mining algorithms, i.e. mining power in BC_1 is compatible with BC_2. A rational miner of BC_1 controls a considerable portion (much less than 50%) of mining power. However, if moving (a part of) the rational miner's mining power to BC_2, then the miner may control a majority of mining power in BC_2.

This may motivate this miner to move a sufficient portion of his mining power from BC_1 to BC_2 to launch 51% attacks, if this would give him some extra revenue.

1.1 Our contributions

New formalised attacks on PoW-based consensus. We propose "Sucker Punch Attacks" (SPA), where an attacker "hits-and-run" the weaker blockchain BC_2 to gain extra profit, through our identified mining power migration attack, or the previously proposed cloud mining attack. To formally study *sucker punch attacks*, we adopt the Markov Decision Process (MDP)-based model from Gervais et al. [15], and extend it to support external mining power. We name the extended model SPA-MDP.

Evaluating sucker punch attacks. We evaluate both ways of launching sucker punch attacks using SPA-MDP. In particular, we analyse the impact of each parameter in SPA-MDP on sucker punch attacks. In addition, to test the feasibility of sucker punch attacks, we apply our model on leading blockchains. For the mining power migration attack, we analyse three pairs of top-ranked blockchains of the same mining algorithm, including 1) Bitcoin (BTC) and BitcoinCash (BCH), 2) Ethereum (ETH) and EthereumClassic (ETC), and 3) Monero(XMR) and ByteCoin (BCN). For the cloud mining attack, we analysed 15 leading blockchains as shown in Section 5.

The result shows that it is not challenging to successfully launch both types of sucker punch attacks. For example, a miner with 12.5% mining power in BTC can gain approximately 6% ($18,946.5 USD) extra profit (than honest mining) by double-spending a transaction of 3000 BCH (equivalent to $378,930) on BitcoinCash. This required mining power is in fact not difficult to achieve – at the time of writing, BTC.com controls 19.2% mining power in Bitcoin[2]

[1]https://coinmarketcap.com/all/views/all/

[2]https://www.blockchain.com/en/pools. Data fetched on

Investigate the 51% attack on Ethereum Classic in 2019 [14] using SPA-MDP. We leverage our SPA-MDP to gain extra insights on the 51% attack incident, including the estimated revenue of the attack, and explanations on the attacker's strategy. We show that the attacker's strategy is near optimal.

1.2 Paper organisation

This paper is structured as follows. Section 2 provides a discussion on the related work. Section 3 presents our system model and formalisation on the two sucker punch attacks. The model is evaluated in Section 4, and the feasibility of the attacks is analysed in Section 5. We investigate the recent 51% attack on EthereumClassic in Section 6, before we conclude in Section 7.

In addition, Appendix A discusses potential remedies of *sucker punch attacks*; Appendix B summarises other related work; Appendix C provides a study on the optimal strategy of a BTC miner to launch *sucker punch attacks* on BCH; and Appendix D presents all experimental data used in this paper.

2 Related work

A growing number of attacks leveraging incentives in the blockchain have recently been identified. The most related work to this paper is the fickle mining [21] and bribery attacks [5]. Due to limited space, other related work can be found in Appendix B.

Fickle mining [21] observed that a miner may gain extra profit by performing honest mining on two blockchains (e.g. BTC and BCH), and proposed a game to model and analyse such behavior. Spiegelman et al. [35] proposed a more generic game to model and analyse a more complex case, where adaptive miners may perform different mining strategies to do honestly mining among multiple blockchains.

Bribery attacks [5] identified several ways for an attacker to bribe other miners to purchase their mining power for a short duration to launch 51% attacks. New ways [22, 36, 16] to bribe miners through higher transaction fees have also been explored.

Our work explores the impact of incentive from a different angle. In particular, we show that if miners are profit-driven, then they may turn to evil to gain extra profit. If this is true, then it indicates that rather than attracting honest mining power to increase the security of the system, the current incentive mechanism may incentivise profit-driven miners to launch attacks. In particular, we consider two classes of sucker punch attacks, i.e., mining power migration attack and cloud mining attack.

19/Aug/2019.

Similar to the fickle mining, our mining power migration attack also considers a miner moving between blockchains. However, in contrast to the existing work where the miner only performs honest mining, we consider the miner may turn to evil and launch 51% attacks when possible. In particular, we consider an existing miner of a blockchain BC_1, such that the miner does not have a majority of mining power in BC_1, but it will control a majority of mining power if moving from BC_1 to a different blockchain BC_2 with a compatible mining algorithm. In this way, the miner may gain more profit by turning malicious and launching 51% attacks on BC_2, then go back to BC_1 to perform honest mining. A more detailed discussion is provided in Section 5. With cloud mining attacks, we consider an attacker renting cloud mining power from the Crypto-Mining Marketplace, which has been suspected to be accountable for the recent 51% incidences.

3 Formalising sucker punch attacks

In this section, we provide a formalisation, called SPA-MDP, on the *sucker punch attacks*. SPA-MDP takes our defined blockchain parameters as input, and outputs an optimal attack strategy with expected revenue of this attack.

The key ideas of sucker punch attacks (SPA) are simple (but quite effective and practical as shown in Section 5). It leverages the observation that the increasing number of proof-of-work based blockchains deployed in the world may reduce the total amount of available mining power in each blockchain. This reduces the difficulty for an attacker to launch 51% attacks on a chosen blockchain. In particular, sucker punch attacks make use of external mining power to launch 51% attacks on a PoW-based blockchain, where the external mining power comes from either other blockchains or cloud mining services.

3.1 System and threat model

Sucker punch attacks consider profit-driven miners, and potentially multiple blockchains such that their mining power is compatible with each other's mining algorithm. For simplicity, our model only considers two blockchains with the same mining algorithm. We call the blockchain with more total mining power a *stronger blockchain*, and the other one a *weaker blockchain*. As the attack (on the weaker blockchain) will be completed within a relatively short time period, our model assumes that during the attack, the difficulty parameter, total amount of honest mining power, and block mining reward do not change.

For *mining power migration attacks*, we assume that the rational miner already has a considerable portion (much less than 50%) of mining power on the *stronger blockchain*. For *cloud mining attacks*, we assume that the rentable cloud mining power from cloud mining services such as NiceHash is compatible with the victim blockchain. In addition, we assume that the adversary has enough money to rent sufficient cloud mining power according to the mining strategy. Later in Section 5, we will analyse the soundness of such an environment.

3.2 Notations

Table 1 provides a summary of notations. Let BC_1 and BC_2 be the stronger blockchain and the weaker blockchain, respectively. Let pr be the price of renting a unit of mining power (e.g. hash per second) for a time unit. Let D_1 and D_2 be the difficulties, R_1 and R_2 be the mining rewards of BC_1 and BC_2, respectively. We have $D_1 > D_2$ and $R_1 > R_2$. We define the fractions of the difficulties and mining rewards as: $d = \frac{D_1}{D_2}$, $r = \frac{R_1}{R_2}$.

For *mining power migration attacks*, we define mining-related parameters for two blockchains. Let $H_{h,1}$, $H_{h,2}$ be the honest mining power, and $H_{a,1}$, $H_{a,2}$ be the adversary's mining power of BC_1 and BC_2, respectively. Let $H_1 = H_{h,1} + H_{a,1}$ and $H_2 = H_{h,2} + H_{a,2}$ be the total mining powers on BC_1 and BC_2, respectively. Note that $H_1 > H_2$ according to the security model. Let $h_1 = \frac{H_a}{H_{h,1}}$ and $h_2 = \frac{H_a}{H_{h,2}}$ be the fractions of the adversary's mining power towards the honest mining power on BC_1 and BC_2. Let $\beta = \frac{H_{a,2}}{H_a}$ be the fraction of migrated mining power by the adversary.

For *cloud mining attacks*, since the mining power is not coming from another blockchain, we only consider the target blockchain BC_2. Let $H_{h,2}$ be the honest mining power, and H_a be the rentable mining power from the cloud mining service. Let $h_2 = \frac{H_a}{H_{h,2}}$ be the fraction of rented mining power out of rentable mining power. Let $\beta = \frac{H_{a,2}}{H_a}$ be the fraction of rented mining power towards the rentable mining power. Let pr be the price of renting mining power (in $\$/h/s$).

Let $\gamma \in [0,1]$ be the propagation parameter of the adversary, i.e. the connectivity of the adversary within the network. When the adversary and the honest miners release blocks simultaneously, γ equals to the fraction of the network that agrees on the adversary's block. Let N_c be the required number of blocks for the blockchain network to confirm a transaction.

3.3 The SPA-MDP model

For a profit-driven miner, the willingness of launching an attack is largely influenced by its expected net revenue. To quantitatively analyse the optimal attack strategy of *sucker punch attacks* (SPA) and its expected revenue, we develope a Markov Decision Process (MDP), called SPA-MDP. It describes the attacks as a series of actions performed by an adversary. At any time, the adversary lies in a state, and can perform an action, which transits his state to another state by a certain probability. For each state transition, the adversary may get some reward or penalty.

Formally, our SPA-MDP model is a four-element tuple $M := (S, A, P, R)$ where:

- S is the state space containing all possible states of an adversary.
- A is the action space containing all possible actions performed by an adversary.
- P is the stochastic transition matrix presenting the probabilities of all state transitions.
- R is the reward matrix presenting the rewards of all state transitions.

We detail each element of M below, and present an overview on the state transitions and reward matrices of SPA-MDP in Table 2.

Table 1: Notations of parameters in SPA-MDP

Notations	definition
D_1	The difficulty of BC_1
D_2	The difficulty of BC_2
d	The fraction of BC_1's difficulty towards BC_2's difficulty ($d = \frac{D_1}{D_2}$)
$H_{h,1}, H_{a,1}$	The honest and adversary's mining power on BC_1, respectively
$H_{h,2}, H_{a,2}$	The honest and adversary's mining power on BC_2, respectively
H_a, H_h	The total honest and adversary's mining power, respectively ($H_a = H_{a,1} + H_{a,2}$, $H_h = H_{h,1} + H_{h,2}$)
h_1	The fraction of the adversary's mining power towards the honest mining power on BC_1 ($h_1 = \frac{H_a}{H_{h,1}}$)
h_2	The fraction of the adversary's mining power towards the honest mining power on BC_2 ($h_2 = \frac{H_a}{H_{h,2}}$)
R_1	The mining reward of a block of BC_1
R_2	The mining reward of a block of BC_2
r	The fraction of BC_1's mining reward of a block towards BC_2's ($r = \frac{R_1}{R_2}$)
v_{tx}	The amount of the attacking transactions
γ	The propagation parameter of the adversary
pr	Renting price of a mining algorithm
β	The fraction of migrated mining power by the adversary
N_c	The number of blocks required to confirm a transaction

Table 2: State transitions and reward matrices of SPA-MDP. Notations are summarised in Table 1.

State × Action	Resulting State	Probability	Reward				Condition
			$\mathbb{R}'_{migration}$	$\mathbb{R}'_{cloudmining}$	$\mathbb{R}_{mine}$	$\mathbb{R}_{tx}$	
(l_h, l_a, β, $fork$), ADOPT	(0, 0, β, ir)	1	0	0	0	0	$l_h > l_a \geq N_c$
(l_h, l_a, β, $fork$), OVERRIDE	(0, 0, β, ir)	1	0	0	$l_a R_2$	v_{tx}	$l_a > l_h \geq N_c$
(l_h, l_a, β, $fork$), WAIT	(l_h, l_a +1, β, p)	$\frac{\beta h_2}{\beta h_2+1}$	$\frac{-\beta h_2 R_1}{d(1+\beta h_2)}$	$\frac{-\beta h_2 D_2 pr}{1+\beta h_2}$	0	0	$l_h < N_c$
	(l_h+1, l_a, β, p)	$\frac{1}{\beta h_2+1}$	$\frac{-\beta h_2 R_1}{d(1+\beta h_2)}$	$\frac{-\beta h_2 D_2 pr}{1+\beta h_2}$	0	0	$l_h < N_c$
(l_h, l_a, β, $fork$), WAIT_INC	(l_h, l_a+1, β +0.2, p)	$\frac{(\beta+0.2)h_2}{(\beta+0.2)h_2+1}$	$\frac{-(\beta+0.2)h_2 R_1}{d(1+(\beta+0.2)h_2)}$	$\frac{-(\beta+0.2)h_2 D_2 pr}{1+(\beta+0.2)h_2}$	0	0	$l_h < N_c$
	(l_h+1, l_a, β +0.2, p)	$\frac{1}{(\beta+0.2)h_2+1}$	$\frac{-(\beta+0.2)h_2 R_1}{d(1+(\beta,+0.2)h_2)}$	$\frac{-(\beta+0.2)h_2 D_2 pr}{1+(\beta+0.2)h_2}$	0	0	$l_h < N_c$
(l_h, l_a, β, $fork$), WAIT_DEC	(l_h, l_a+1, β -0.2, p)	$\frac{(\beta-0.2)h_2}{(\beta-0.2)h_2+1}$	$\frac{-(\beta-0.2)h_2 R_1}{d(1+(\beta-0.2)h_2)}$	$\frac{-(\beta-0.2)h_2 D_2 pr}{1+(\beta-0.2)h_2}$	0	0	$l_h < N_c$
	(l_h+1, l_a, β -0.2, p)	$\frac{1}{(\beta-0.2)h_2+1}$	$\frac{-(\beta-0.2)h_2 R_1}{d(1+(\beta-0.2)h_2)}$	$\frac{-(\beta-0.2)h_2 D_2 pr}{1+(\beta-0.2)h_2}$	0	0	$l_h < N_c$
(l_h, l_a, β, $fork$), MATCH	(l_h, l_a +1, β , ir)	$\frac{\beta h_2+\gamma}{\beta h_2+1}$	$\frac{-\beta h_2 R_1}{d(1+\beta h_2)}$	$\frac{-\beta h_2 D_2 pr}{1+\beta h_2}$	$\frac{(l_a+1)R_2 \beta h_2}{\beta h_2+\gamma}$	v_{tx}	$l_h = l_a \geq N_c$
	(l_h +1, l_a, β , r)	$\frac{1-\gamma}{\beta h_2+1}$	$\frac{-\beta h_2 R_1}{d(1+\beta h_2)}$	$\frac{-\beta h_2 D_2 pr}{1+\beta h_2}$	0	0	$l_h = l_a \geq N_c$
(l_h, l_a, β, $fork$), MATCH_INC	(l_h, l_a +1, β +0.2, ir)	$\frac{(\beta+0.2)h_2+\gamma}{(\beta+0.2)h_2+1}$	$\frac{-(\beta+0.2)h_2 R_1}{d(1+(\beta+0.2)h_2)}$	$\frac{-(\beta+0.2)h_2 D_2 pr}{1+(\beta+0.2)h_2}$	$\frac{(l_a+1)R_2(\beta+0.2)h_2}{(\beta+0.2)h_2+\gamma}$	v_{tx}	$l_h = l_a \geq N_c$
	($l_h + 1$, l_a, β +0.2 , r)	$\frac{1-\gamma}{(\beta+0.2)h_2+1}$	$\frac{-(\beta+0.2)h_2 R_1}{d(1+(\beta+0.2)h_2)}$	$\frac{-(\beta+0.2)h_2 D_2 pr}{1+(\beta+0.2)h_2}$	0	0	$l_h = l_a \geq N_c$
(l_h, l_a, β, $fork$), MATCH_DEC	(l_h, l_a +1, β -0.2, ir)	$\frac{(\beta-0.2)h_2+\gamma}{(\beta-0.2)h_2+1}$	$\frac{-(\beta-0.2)h_2 R_1}{d(1+(\beta-0.2)h_2)}$	$\frac{-(\beta-0.2)h_2 D_2 pr}{1+(\beta-0.2)h_2}$	$\frac{(l_a+1)R_2(\beta-0.2)h_2}{(\beta-0.2)h_2+\gamma}$	v_{tx}	$l_h = l_a \geq N_c$
	(l_h +1, l_a, β -0.2 , r)	$\frac{1-\gamma}{(\beta-0.2)h_2+1}$	$\frac{-(\beta-0.2)h_2 R_1}{d(1+(\beta-0.2)h_2)}$	$\frac{-(\beta-0.2)h_2 D_2 pr}{1+(\beta-0.2)h_2}$	0	0	$l_h = l_a \geq N_c$

3.3.1 The State Space S

The state space S is defined as a tuple $S := (l_h, l_a, \beta, fork)$, where l_h and l_a are the length of the honest blockchain and the adversary's fork, respectively; β is the ratio of mining power at BC_2 allocated by the adversary; and $fork$ represents the state of the adversary's fork.

For simplicity, we choose β from $(0.0, 0.2, 0.4, 0.6, 0.8, 1.0)$. The state $fork$ of the adversary's fork has three values, defined as follows:

- relevant ($fork = r$) means the adversary's fork is published but the honest blockchain is confirmed by the network. This indicates that the attack is unsuccessful at present. (Note that the adversary can keep trying and may succeed in the future.)
- irrelevant ($fork = ir$) means the adversary's fork is published and confirmed in network. This indicates a successful attack.
- private ($fork = p$) means the adversary's fork is private and only the adversary is mining on it. This indicates that an attack is in process.

3.3.2 The Action Space A

An adversary can perform the following actions:

- **ADOPT** The adversary accepts the honest blockchain and discards his fork, which means the adversary aborts his attack.
- **OVERRIDE** The adversary publishes his fork (which is longer than the honest one). Consequently, the honest blockchain is overridden, and the payment transaction from the adversary is successfully reverted.
- **MATCH** The adversary publishes his fork with the same length as the honest blockchain. This action has three variants *MATCH*, *MATCH_INC*, and *MATCH_DEC*, where *_INC* and *_DEC* represent the increase and decrease of malicious mining power ratio β, respectively.
- **WAIT** The adversary keeps mining on his fork. This action can be performed in two scenarios. One is $l_h < N_c$, indicating that the merchant is still waiting for the payment confirmation. Another one is that after the adversary publishes his blockchain by *MATCH*, $l_a \leq l_h$ still holds. In addition, there are three types in this action *WAIT*, *WAIT_INC*, *WAIT_DEC*, which represent the adversary's mining power adjustment, similar to that of **MATCH**.

3.3.3 The State Transition Matrix P

The State Transition Matrix P (Table 2) describes all state transition possibilities. It is defined as a 3-dimensional matrix $S \times A \times S : Pr(s, a \Rightarrow s')$, where the participant in the state s does the action a to transit his state to s' with the probability $Pr(s, a \Rightarrow s')$. Here $s, s' \in S$ and $a \in A$. In the double-spending context, the state transition is invoked by a new block (during **WAIT**), a blockchain fork selection (by **OVERRIDE** or **MATCH**) or quitting the attack (by **ADOPT**).

In MDP, an action can trigger a state s to transit to another state s' with some probability. Note that the resulting state s' can have multiple possibilities $s'_1, s'_2, \cdots, s'_n$, and $\sum_{i=1}^{n} Pr(s, a \Rightarrow s'_i) = 1$.

Not using eclipsed mining power ($A \in \{$ WAIT_INC, WAIT_DEC $\}$) When $A \in \{$ **WAIT_INC**, **WAIT_DEC** $\}$, the adversary is mining his fork alone. The next state update is triggered by a newly created block either by the honest network or by the adversary, with a probability directly associated to their mining power on BC_2, namely $H_{a,2} = \beta H_a$ of the adversary and $H_{h,2}$ of the honest network.

Therefore, the probability that the adversary gets the next block ($l_a \rightarrow l_a + 1$) is

$$\begin{aligned} P(l_a \rightarrow l_a+1) &= \frac{H_{a,2}}{H_{a,2}+H_{h,2}} \\ &= \frac{\beta H_a}{\beta H_a + H_{h,2}} = \frac{\beta h_2}{\beta h_2 + 1} \end{aligned} \tag{1}$$

And vice versa for $l_h \rightarrow l_h + 1$.

$$P(l_h \rightarrow l_h+1) = 1 - P(l_a \rightarrow l_a+1) = \frac{1}{\beta h_2 + 1} \tag{2}$$

Using eclipsed mining power ($A \in \{$ MATCH_INC, MATCH_DEC $\}$) Besides the mining power owned by the adversary, the eclipsed mining power of $\gamma H_{h,1}$ mines on the adversary's blockchain after a **MATCH** attempt. Therefore, the possibility of $l_a \rightarrow l_a + 1$ becomes

$$\begin{aligned} P(l_a \rightarrow l_a+1) &= \frac{H_{a,2}+H_{eclipsed}}{H_{a,2}+H_{h,2}} \\ &= \frac{\beta H_a + \gamma H_{a,2}}{\beta H_a + H_{h,2}} = \frac{\beta h_2 + \gamma}{\beta h_2 + 1} \end{aligned} \tag{3}$$

And vice versa for $l_h \rightarrow l_h + 1$.

$$P(l_h \rightarrow l_h+1) = 1 - P(l_a \rightarrow l_a+1) = \frac{1-\gamma}{\beta h_2 + 1} \tag{4}$$

3.3.4 The Reward Matrix R

The Reward Matrix R is defined as $S \times A \times S : Re(s, a \Rightarrow s')$, where the participant in the state s transits to a new state s' with the reward $Re(s, a \Rightarrow s')$. Here $s, s' \in S$ and $a \in A$. The reward from a double-spending attack contains two parts: the block reward (including transaction fees) of the published longer blockchain and the double-spending transaction. To calculate the net reward, we also need to consider the cost of launching an attack. Here, we define the reward $Re(s, a \Rightarrow s')$ as a tuple ($\mathbb{R}'$, $\mathbb{R}_{mine}$, $\mathbb{R}_{tx}$) to fit into the MDP model, where $\mathbb{R}'$ represents the cost of launching a double-spending attack, $\mathbb{R}_{mine}$ represents the reward from the mined blocks including transaction fees, and $\mathbb{R}_{tx}$ represents the reward from the double-spent transaction.

With a state transition $S \times A \rightarrow S'$, the adversary will get some reward, which can be of a positive or negative value. The state transition matrix R is a 3-dimension matrix $(S \times A \times S')$, where S, A and S' are the same as P, but the value of this matrix is the reward of the corresponding state transition.

The reward of an *sucker punch attack* consists of the reward of mining $\mathbb{R}_{mine}$ and the reward from the double-spent transaction $\mathbb{R}_{tx}$. Besides, the reward should minus the attack cost $\mathbb{R}'$. We define $\mathbb{R}_{mine}$, $\mathbb{R}_{tx}$ and $\mathbb{R}'$ as follows.

Attack cost $\mathbb{R}'$ To calculate net revenue, we also need to consider a "negative reward", which is the cost of launching attacks. We use $\mathbb{R}'_{migration}$ and $\mathbb{R}'_{cloudmining}$ to denote the cost of launching the *mining power migration attack* and the *cloud mining attack*, respectively. Compared to honest mining on BC_1, the cost $\mathbb{R}'_{migration}$ of mining power migration is mainly from the loss of block rewards from BC_1 due to the migrated mining power. Consequently, the cost can be computed as hashrate $\cdot$ time $\cdot$ difficulty $\cdot$ R_1, which is the estimated mined block multiplies the block reward of BC_1. For **ADOPT** and **OVERRIDE** actions, the time of finishing a state transition is negligible. Meanwhile, for **WAIT**-style and **MATCH**-style actions, the state transition is triggered by mining a new block, so the time it takes is depending on the difficulty of mining a block. Therefore, the $\mathbb{R}'_{migration}$ under **WAIT**-style and **MATCH**-style actions can be computed as follows:

$$\begin{aligned} \mathbb{R}'_{migration}(l_a \rightarrow l_a+1) &= \mathbb{R}'_{migration}(l_h \rightarrow l_h+1) \\ &= \text{hashrate} \cdot R_1 \cdot \text{time} \cdot \frac{1}{\text{difficulty}} \\ &= -\beta H_a \cdot R_1 \cdot \frac{D_2}{H_{h,2} + \beta H_a} \cdot \frac{1}{D_1} \\ &= \frac{-\beta h_2 R_1}{d(1+\beta h_2)} \end{aligned} \tag{5}$$

When launching *cloud mining attacks*, the cost is from renting the cloud mining power. The cloud mining power is priced as $\$/(h/s)/s$, which means the price of renting a mining power unit for a time unit. We denote the cloud mining power price as pr. Similar with the *mining power migration attack*, only **WAIT**-style and **MATCH**-style actions take a non-negligible time period. Therefore, either the cost of **ADOPT** or **OVERRIDE** is 0, and the cost of **WAIT**-style and **MATCH**-style actions is computed as

$$\begin{aligned} R_{BC_1}(l_a \to l_a+1) &= R_{BC_1}(l_h \to l_h+1) \\ &= \text{hashrate} \cdot \text{price} \cdot \text{time} \\ &= -\beta H_a \cdot Pr \cdot \frac{D_2}{H_{h,2}+\beta H_a} \\ &= \frac{-\beta h_2 D_2 Pr}{1+\beta h_2} \end{aligned} \tag{6}$$

Mining reward $\mathbb{R}_{mine}$ The adversary can get the block reward $\mathbb{R}_{mine}$ on BC_2 only when his fork is published and accepted by the honest network. Therefore, only **OVERRIDE** and the winning scenarios of **MATCH**-style actions have a positive reward, while $\mathbb{R}_{mine} = 0$ under other scenarios.

When performing **OVERRIDE**, the adversary's blockchain of length l_a is directly accepted, so $\mathbb{R}_{mine} = l_a R_2$. When performing **MATCH**-style actions, the adversary needs to get the next block so that his blockchain overrides the honest one. Therefore, $\mathbb{R}_{mine} = (l_a+1)R_2$ holds for the winning scenarios.

Reward from the double-spent transaction $\mathbb{R}_{tx}$ Similar with $\mathbb{R}_{mine}$, the adversary gets the double-spent money only when it successfully overrides the honest blockchain. Therefore, $\mathbb{R}_{tx}$ equals to the transaction amount v_{tx} for **OVERRIDE** and the winning scenarios of **MATCH**-style actions, while $\mathbb{R}_{tx} = 0$ for other scenarios.

4 Evaluating Sucker Punch Attacks

In this section, we present our implemented SPA-MDP model, and evaluate the impact of each parameter on the two *sucker punch attacks* using SPA-MDP. The evaluation reveals most important aspects on the profitability of *sucker punch attacks*. By combining this evaluation and public blockchain data, the attackers can find most feasible targets to attack, and the defenders can be aware of the potential threats in advance.

4.1 Experimental methodology

We implement our SPA-MDP model using Python 2.7 and the *pymdptoolbox* library [7]. All experiments run on a MacBook Pro with a 2.2 GHz Intel Core i7 Processor, a 16 GB DDR4 RAM and 256 SSD storage disk.

The SPA-MDP model is infinite due to the unbounded l_a and l_h. In order to implement SPA-MDP, we convert it into the finite one by giving an upper bound *limit* for l_a and l_h. We apply the *ValueIteration* algorithm [33] with a discount value of 0.9 and an epsilon value of 0.1 in our MDP-based model.

We classify parameters in SPA-MDP into five aspects, namely 1) mining status, 2) incentive, 3) adversary network, 4) the vigilance of the merchant, and 5) mining power price. The mining status includes the mining difficulty (D_1 and D_2) and the ratio of adversary's mining power (h_1 and h_2). The incentive includes mining reward (R_1 and R_2) and the adversary's transaction amount v_{tx}. The adversary's network includes the propagation parameter γ of the adversary. The vigilance of the merchant includes the number N_c of required block confirmations. The mining market includes pr.

4.2 Evaluation results

We evaluate the impact of each aspect on the relative revenue of an adversary launching both types of *sucker punch attacks*. Table 5 in Appendix D summarises the parameter values used for the evaluation. For aspects with multiple parameters (the mining status and the incentive), we correlate them by using a 3D surface. Since both attacks have common parameters D_2, h_2, R_2, v_{tx}, γ and N_c, we evaluate them on *mining power migration attack* only to avoid the repetition.

4.2.1 Impact of mining status

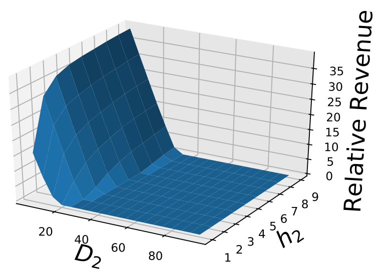
Figure 2a shows the impact of mining-related parameters on the adversary's relative revenue. We observe that the relative revenue increases monotonically with D_2 decreasing and h_2 increasing.

When D_2 decreases, mining on BC_2 will be easier, so that migrating to BC_2 will be more profitable. This encourages both types of our attacks on BC_2. When h_2 increases, launching both types of attacks on BC_2 will be more possible to succeed, so the net revenue of attacks will increase. This also encourages 51% attacks on BC_2. Therefore, both decreasing D_2 and increasing h_2 incentivise 51% attacks on BC_2.

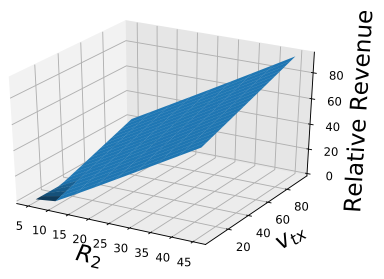
4.2.2 Impact of incentive-related parameters

Figure 2b shows the impact of incentive-related parameters on the relative revenue. We observe that increasing R_2 and v_{tx} leads the adversary to profit more.

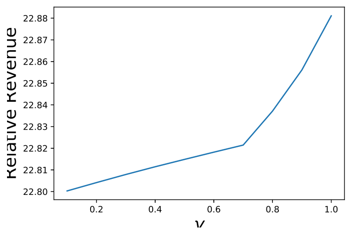
When R_2 increases, mining BC_2 will be more profitable, and 51% attacks on BC_2 will also be more profitable. This encourages both types of 51% attacks on BC_2. The 51% attack generates v_{tx} out of thin air, so v_{tx} is the direct revenue of the 51% attack, and increasing v_{tx} directly increases the relative revenue. Therefore, both increasing R_2 and v_{tx} incentivise 51% attacks on BC_2.



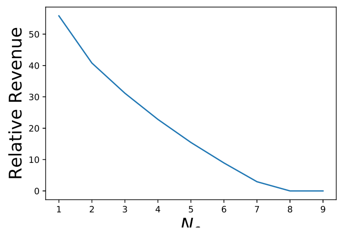
(a) Impact of h_2 and D_2.



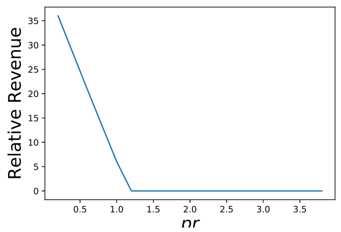
(b) Impact of v_{tx} and R_2.



(c) Impact of γ.



(d) Impact of N_c.



(e) Impact of pr.

Figure 2: Impacts of parameters on the relative revenue of *sucker punch attacks*. Table 5 in Appendix D outlines all experimental data.

4.2.3 Impact of adversary network

Figure 2c shows the impact of γ on the relative revenue. In particular, we can see that the relative reward increases slightly with γ increasing. Interestingly, when the attacker's propagation parameter $\gamma = 0.7$, the curve slope increases.

According to our model, γ counts only when the adversary launches the *MATCH* action. When $h_2 \geq 1$, the adversary can always launch the 51% attack, regardless of the reward. Therefore, the *MATCH* action is an infrequent choice compared to *OVERRIDE*, so the influence of γ is negligible in our case.

The slope change is suspected to be when $\beta H_a + \gamma H_{h,2} \geq (1-\gamma)H_{h,2}$. At that point, the allocated mining power from the adversary plus his eclipsed honest mining power outperforms the un-eclipsed honest power. Consequently, the adversary is confident to override the small blockchain by *MATCH* action.

4.2.4 Impact of the vigilance of the merchant

Figure 2d shows the impact of N_c on the relative revenue. We observe the relative revenue decreases monotonically with N_c increasing, and finally reaches 0.

More block confirmations require the adversary to keep mining secretly for a longer time. This leads to greater cost for launching the 51% attack through both types of attacks, and discourages 51% attacks on BC_2.

4.2.5 Impact of mining power price

The impact of the mining power price pr is shown in Figure 2e. We observe that the relative revenue decreases sharply with pr increasing, and finally reaches 0.

When the price of renting mining power is low, the related blockchains are vulnerable to the cloud mining attack as the attack cost is also low. Increasing pr leads to greater cost of launching 51% attack through renting cloud mining power, which will discourage this kind of 51% attacks on BC_2.

4.3 Analysis

We observed several insights from our evaluation. First, an attacker's profit is mainly affected by the parameters that are out of the attacker's control. That is, to maximise attack revenue, an attacker should choose its target carefully. In particular, even though to some extent an attacker might have a chance to control the network propagation parameter γ, it has little impact on the revenue according to Figure 2c. All other parameters are out of the attacker's control. Thus, once choosing the targeted blockchains, the adversary has little control on the feasibility and profitability of the attack.

Second, the adversaries and the defenders should be aware of the public parameters D_2, h_2, v_{tx} and R_2. According to Figure 2a and Figure 2b, all of these parameters have a significant impact on the adversary's relative revenue. Although either the adversaries or the defenders can hardly control these parameters, monitoring them in real time is possible. By monitoring these parameters, an adversary can identify a target with more expected profit, and an defender can be aware of potential attacks then react to this situation (e.g. by following the recommendations in Appendix A.1.2).

Last, defenders (e.g. the cryptocurrency exchanges and the merchants) can reduce the feasibility and profitability of *sucker punch attacks* by increasing the number N_c of blocks for confirming transactions. Within these parameters, the defenders can only control N_c, and increasing N_c can greatly reduce the adversary's relative revenue according to Figure 2d.

5 Sucker Punch Attacks in the wild

This section evaluates the security of mainstream PoW-based blockchains against sucker punch attacks. We evaluate the *mining power migration attack* on 3 pairs of top-ranked blockchains with the same mining algorithm, namely 1) Bitcoin (BTC) and BitcoinCash (BCH) with Sha256d, 2) Ethereum (ETH) and EthereumClassic (ETC) with Ethash, and 3) Monero(XMR) and ByteCoin (BCN) with CryptoNight. Our evaluation shows that, the *mining power migration attack* is quite easy and profitable on BTC/BCH and ETH/ETC, but it is not as effective on XMR/BCN. As an example, we also provide an optimal strategy derived from our model for a BTC miner to launch *mining power migration attacks* on BCH, together with explanations and observed insights, in Appendix C.

For the *cloud mining attack*, we evaluated the security of 15 leading PoW-based blockchains (chosen from the top 100 blockchains by their market caps [8]). Our evaluation shows that the *cloud mining attacks* are feasible and profitable on all selected blockchains except Komodo (KMD), where the attack is feasible but not profitable.

5.1 Mining power migration attacks

We evaluate the profitability and feasibility of the mining power migration attack on 3 pairs of top-ranked cryptocurrencies with the same mining algorithm: BTC/BCH, ETH/ETC, and XMR/BCN. Table 6 in Appendix D summarises the blockchain data we use as the input of our model. By permuting the adversary mining power H_a and the transaction value v_{tx}, our experiments reveal their relationship with the relative revenue.

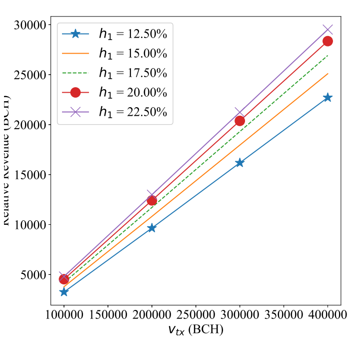
As shown in Figure 3, it is surprisingly easy and profitable for a miner of BTC (or ETH) to launch a 51% attack on BCH (resp. ETC), while it is difficult and unprofitable for a XMR miner to attack BCN. For example, as shown in Figure 3:

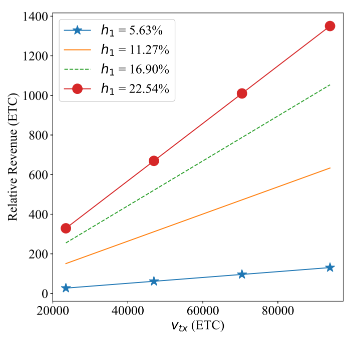
- With approximately 12.5% mining power of BTC ($5000E + 15h/s$), an adversary can gain 6% (150 BCH, or \$18,946.5 USD) extra profit (than honest mining) by double-spending a transaction of 3000 BCH (equivalent to \$378,930).
- With approximately 11.27% mining power of ETH ($16E + 12h/s$), the adversary can gain 1.33% (600 ETC, or \$2,556 USD) extra profit by double-spending a transaction of 90000 ETC (equivalent to \$383,400).
- With approximately 43.05% mining power of XMR ($4E + 8h/s$), the adversary can gain 0.67% (1,000,000 BCN or \$619 USD) extra profit by double-spending a transaction of 600,000,000 BCN (equivalent to \$247,600).

The required mining power is not difficult to achieve. According to Table 9 in Appendix D [3], the top three mining pools in ETH are Ethermine (27.7%), Sparkpool (22.2%), f2pool2 (12.5%); and the top three mining pools in BTC are BTC.com (23.0%), AntPool (16.4%), and F2Pool (11.6%).

However, for XMR, a miner cannot profit much from the *mining power migration attack*. This is because the total available mining power in Monero is only about 2.8 times of the mining power in the BCN, although their market caps differ greatly. In comparison, the total available mining power in BTC is about 27.8 times of the total mining power in BCH; and the total available mining power in ETH is about 16.4 times of the total mining power in ETC.

Kwon et al. [20] also observed that BTC and BCH share the same mining algorithm, but their analysis and results differ from ours. First, our analysis is based on MDP, while their analysis is based on game theory. Second, their work still assumes the "honest majority", but we show that the adversary can launch 51% attacks on BCH and profit with less than 12.5% BTC mining power. Third, our *sucker punch attacks* are a type of attacks, while their proposed *fickle mining* is only a mining strategy. Last, our main result is that PoW's current incentive mechanism breaks the security guarantee of PoW consensus due to the *sucker punch attacks*, while their main result is that PoW's current incentive mechanism weakens the security guarantee of PoW due to the *fickle mining*.


(a) BTC and BCH


(b) ETH and ETC

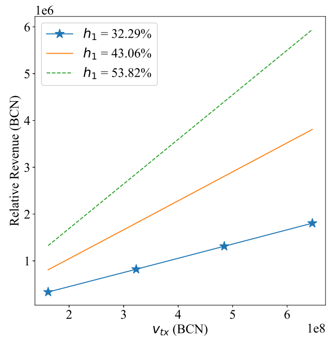

(c) XMR and BCN

Figure 3: Mining power migration attacks on three different pairs of blockchains. We use $\gamma = 0.3$ for this group of experiments. Table 6 in Appendix D outlines experimental data of this analysis.

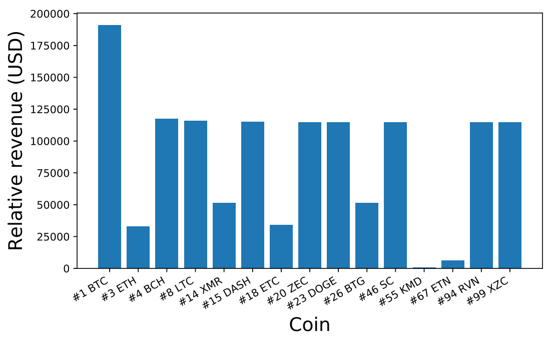


Figure 4: Cloud mining attacks on selected 15 PoW blockchains. We use the transaction amount v_{tx} \$500,000, $h_2 = 2$ and $\gamma = 0.3$. We use the value of N_c recommended by cryptocurrency communities. Table 7 in Appendix D outlines experimental data of this analysis.

5.2 Cloud mining attacks

We evaluate the security of 15 leading PoW-based blockchains against the *cloud mining attack*. We select these 15 blockchains from Top 100 blockchains by their market cap (at 19 February 2019, the time we fetched experimental data). Among these 100 blockchains, 22 blockchains adopt PoW consensus. Among these 22 blockchains, DigiByte (DGB) and Verge (XVG) use multiple mining algorithms simultaneously so cannot fit into our model; Bytom (BTM) uses the Tensority mining algorithm which NiceHash does not support; and NiceHash does not support ByteCoin (BCN), Electroneum(ETN), WaltonChain (WTC), and Aion (AION), so has no renting price data for these blockchains. Therefore, only 15 blockchains are left, and we evaluate these 15 blockchains.

Table 7 in Appendix D summarises our experimental data. We set $v_{tx} = \$500,000$ (i.e. the double-spending transaction amount is \$500,000), and $h_2 = 2$ (i.e. the rentable mining power is twice of the honest mining power). We choose the value of N_c according to the recommended values from cryptocurrency community.

Figure 4 summarises our evaluation results. It shows that, unfortunately, all selected blockchains are vulnerable towards the *cloud mining attack*. For example:

- the attacker needs approximately \$2,000 to launch *cloud mining attack* on ETC for an hour, and the relative revenue will be \$33899 if successful;
- the attacker needs approximately \$2,600 to launch the attack on BCH for an hour, and the relative revenue will be \$117198 if successful;
- the attack needs approximately \$730 to launch the attack on ETN for one hour, and the relative revenue will be \$6222 if successful.

The only exception within these 15 blockchains is Komodo (KMD): the attacker cannot profit much by launching *cloud mining attacks* on KMD. To investigate the reason of this, we additionally evaluate the security of KMD against both *mining power migration attacks* and *cloud mining attacks*. More specifically, we evaluate the impact of the adversary's mining power (h_1 and h_2) and the transaction value (v_{tx}) on the attacker's profit (in Figure 5). The evaluation result shows that, although feasible, both attacks on KMD will not give much extra profit - the attacker can only gain $1\% \sim 2\%$ more revenue compared to honest mining. In addition, while both attacks are not very profitable, the revenue of launching *cloud*

[3]https://www.etherchain.org/charts/topMiners. Data collected on 19/Feb/2019.

mining attack is still better than the revenue of launching *mining power migration attack*. For the profitability, the reason is that KMD requires 30 blocks to confirm a transaction (i.e., $N_c = 30$) [1], which is a much higher requirement than other blockchains. As shown in Section 4, increasing N_c can reduce the profit of sucker punch attacks. Later in Appendix A, we will also demonstrate in detail how effective it is to prevent sucker punch attacks by adjusting different parameters, including N_c.

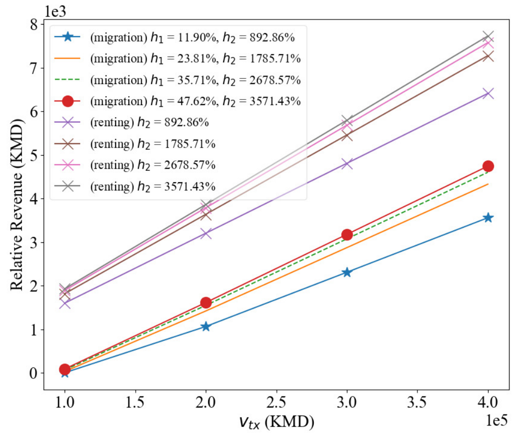


Figure 5: Profitability of mining power migration attacks and cloud mining attacks on Komodo (KMD). We choose $\gamma = 0.3$, and N_c according to the recommended value by cryptocurrency communities. Table 7 in Appendix D outlines experimental data of this analysis. For mining power migration attacks, we use Zcash (ZEC) as BC_1.

6 Investigating the 51% attack on ETC at Jan. 2019

Ethereum Classic (ETC) is a PoW-based blockchain using the same mining algorithm as Ethereum (ETH). As shown in Section 5, both ETH and ETC are vulnerable to the cloud mining attack, and ETC is also vulnerable to the *mining power migration attack* (from ETH or other GPU-friendly PoW-based blockchains). On 7 Jan. 2019, a 51% attack happened to ETC, where the attacker double-spent transactions of more than 1.1 million USD on a cryptocurrency exchange (i.e., Gate.io [14]). Though the actual source is still a mystery, NiceHash, a cloud mining service, is highly suspected as the mining power source [24, 23].

In this section, we investigate this cloud mining attack on ETC by using SPA-MDP. First, we reveal the attacker's strategy from his behaviours during the attack, and conclude that the attacker's strategy in this incident is the best practise of launching *cloud mining attacks*.

Second, we estimate the attacker's relative revenue using SPA-MDP (assuming he was using NiceHash to launch *cloud mining attacks*). Our estimated relative revenue is approximate to $100,000, the amount that the attacker returned to Gate.io [2]. This indicates that the attacker's revenue is near optimal, and further implies that the attacker may launch the *cloud mining attacks* in a fine-grained way as in Appendix C. Last, we compare the attacker's revenue between the *cloud mining attack* and the *mining power migration attack*. Our comparison shows that, when the rentable mining power is sufficient, *cloud mining attack* is much more profitable than *mining power migration attack*.

6.1 The attack details

Table 3: All double-spent transactions during the 51% attack on ETC [27]. In this attack, 12 transactions were double-spent from two accounts.

Trans. ID (in short)	From	To	Amount (ETC)	Height	waiting time (#block)
0x1b47a700c0	0x3ccc8f7415	0xbbe1685921	600	7249357	-
0xbba16320ec	0x3ccc8f7415	0x2c9a81a120	4000	7254430	**5073**
0xb5e0748666	0x3ccc8f7415	0x882f944ece	5000	7254646	216
0xee31dffb66	0x3ccc8f7415	0x882f944ece	9000	7255055	409
0xfe2da37fd9	0x3ccc8f7415	0x2c9a81a120	9000	7255212	157
0xa901fcf953	0x3ccc8f7415	0x2c9a81a120	15700	7255487	275
0xb9a30cee4f	0x3ccc8f7415	0x882f944ece	15700	7255554	67
0x9ae83e6fc4	0x3ccc8f7415	0x882f944ece	24500	7255669	115
0xaab50615e3	0x3ccc8f7415	0x53dffbb307	5000	7256012	343
0xd592258715	0x07ebd5b216	0xc4bcfee708	26000	7261492	**5480**
0x9a0e8275fc	0x07ebd5b216	0xc4bcfee708	52800	7261610	118
0x4db8884278	0x07ebd5b216	0xc4bcfee708	52200	7261684	74
					Total: 219500 ETC

At the beginning of 2019, a 51% attack on ETC resulted in the loss of more than 1.1 million dollars. The attack lasted for 4 hours, from 0:40am UTC, Jan. 7th, 2019 to 4:20am UTC, Jan. 7th, 2019, approximately. During the attack, the attacker repetitively launched a coin withdrawal transaction on the Gate.io cryptocurrency exchange [13], then launched double-spending attacks [14]. Among these attempts, 12 transactions were successfully double-spent (listed in Table 3). Interestingly, several days after the attack, the attacker returned ETC equivalent to $100,000 to Gate.io [2].

The source of the mining power for this attack remains uncertain due to the anonymity of miners. However, the NiceHash cloud mining platform [28] is highly suspected: One day before the attack, an anonymous person rented all available Ethash (the mining algorithm used by ETH and ETC) mining power from NiceHash [24, 23].

6.2 Analysing the attacking strategy

According to Table 3, the attacker continuously increased the value of new transactions throughout the attack (except the last double spending of the first account). It is suspected that this behavior belongs to the strategies used by the attacker to maximise and stabilise his revenue, with the following reasons.

First, launching multiple small double-spending attempts can stabilise the expected revenue. The double-spending attack may fail in a limited time period, even if the adversary controls more than 50% of the computing power. Compared to a one-off attempt, the revenue will be more stable if dividing a transaction to multiple smaller transactions.

Second, this strategy may be used for avoiding the risk management system of the cryptocurrency exchanges. Most cryptocurrency exchanges run their own risk management system to combat the misbehaviors, like the fraudulent payments and the abnormal login attempts. A huge coin withdrawal transaction is highly possible to trigger the risk management system, while multiple small transactions would be overlooked. Meanwhile, a big transaction may lead to longer confirmation time, and a longer attack period is easier to be detected. Therefore, defeating the risk control system is naturally a part of the attacker's strategy. According to the Gate.io report [14], the risk management system ignored transactions from the attacker, as the attack was decently prepared - they registered and real-name authenticated the account on Gate.io more than 3 months before the attack. Slowly increasing the transaction value is also highly suspected as an approach for reverse-engineering the threshold of invoking the risk management system.

In addition, we investigate the waiting time between each two attacks (quantified by using the number of blocks). The waiting time varies mostly from 67 blocks to 409 blocks. Interestingly, there are two much bigger gaps of more than 5000 blocks before the transactions 0xbba16320ec and 0xd592258715. The first gap is after the first attack, and the second gap is before the attacker changed to another account to send double-spending transactions. The first gap may be because the attacker was cautious when first launching the double-spending attack. The attacker launched a double-spending transaction of only 600 ETC coins, which is much smaller than his following transactions. After the first attack, the attacker waited for a long time to confirm that the attack is successful, then he started to increase the transaction value. The second gap may be because the attacker ran out of money in his first account 0x3ccc8f7415, and managed to change to another account 0x07ebd5b216. The last transaction 0xd592258715 sent by 0x3ccc8f7415is is right before the second gap. It's value is 5000 ETC coins, which is much smaller than its previous transaction of 24500 ETC coins. After the transaction 0xd592258715, the attacker changed to his another account 0x07ebd5b216, which caused the time gap of 5480 blocks.

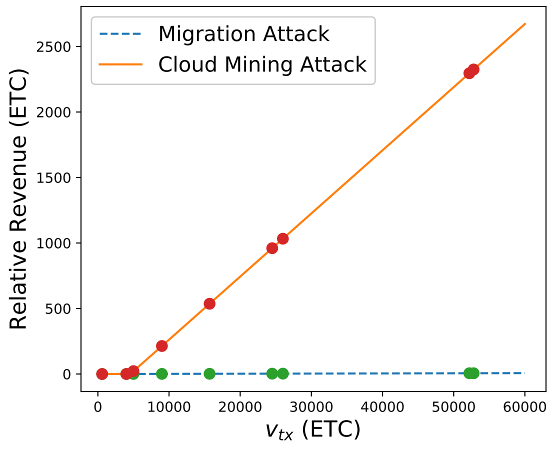


Figure 6: Simulated 51% attack on ETC. The blue line denotes the relative reward of *cloud mining attacks*. The orange line denotes the relative reward of *mining power migration attacks* for making comparisons. We also marked different transaction amounts in the attack using dots.

6.3 Estimating the attacker's revenue

We use our model to estimate the revenue of the attacker. To analyse this attack, we collect attack-related data during the time period of the attack (on 07/01/2019). Table 8 in Appendix D summarises the experimental data.

With Gate.io, the required number N_c of block confirmation is 12, which is also recommended by of ETH community and ETC community [32]. The price of ETC on that day was \$5.32, and the price of BTC was \$4061.47. The mining difficulty of ETC was 131.80E+12, and the ratio h_2 of attacker's mining power over the honest mining network was about 1.16, i.e., the attacker approximately controls 53.7% mining power during the attack. The reward of successfully mining a block is 4 ETC coins, and the price of renting Nicehash mining power on that day is 3.8290 BTC/TH/day. As there is no data on the attacker's connectivity w.r.t. propagating his blocks, and the impact of γ is relatively small (as previously discussed), we assume that $\gamma = 0.3$.

We permute and mark the transaction values used by the attacker. We also plot the same curve in the *mining power migration attack* to compare the profitability of two mining power sources. The result is shown in Figure 6.

The result shows that when the transaction value is over 5000 ETC, double-spending is more profitable than by honest mining. Having a transaction (or a set of transactions) of value over 5000 ETC (approximately 26,000 USD at the time of attack) should not be difficult for an attacker, so the incentive of launching double-spending attacks is very strong.

Moreover, our results give the estimated net revenue of the attacker: \$84773.40. It is approximate to \$100,000 - the value that attacker returned to Gate.io after the attack [2]. Summing all relative revenues of successful transactions, the total relative revenue of the attacker derived from our model is approximately 9000 ETC coins. Recall that the attacker controlled $p = 53.7\%$ of ETC mining power. the probability P of a successful 51% attack is one minus the possibility of failing to attack. The failing scenario is that the adversary mines $n < N_c$ blocks when the honest network has mined N_c blocks, where N_c is the number of confirmation blocks. Mining can be modeled as a binomial distribution model $B(n+N_c, p)$, where $n+N_c$ blocks will be mined and the adversary mines the next block with the probability p. Therefore, given $N_c = 12$, P is calculated as:

$$\begin{aligned} P &= 1 - \sum_{i=0}^{12} C_{i+12-1}^{i} p^i (1-p)^{12} \\ &= 56.48\% \end{aligned} \tag{7}$$

Our model produces the expected net revenue of a single attack, regardless whether it is successful or not. In our case, only successful attacks were observed, but the failed attacks also contribute to the theoretical expected net revenue. The successful attacks contribute to the expected net revenue of 9000 ETC coins, and their possibilities of success are 56.48%. Accordingly, the failed attacks contribute to the expected net revenue of $\frac{9000}{56.48\%}(1 - 56.48\%) = 6934.85$ ETC coins. Therefore, the expected total net revenue is $9000 + 6934.85 = 15934.85$ ETC coins, which is equivalent to \$84773.40 at the time of attack.

This results implies that, the attacker may adopt the optimal strategy for launching *cloud mining attacks*. \$84773.40 - our estimated revenue of the attacker - is slightly less than \$100,000 - the amount of money the attacker returned to Gate.io. The attacker is impossible to return money more than he earned from the attack, so his revenue should be more than \$100,000. If so, the attacker's revenue should be optimal. To achieve the optimal revenue, the attacker should launch *cloud mining attacks* using the optimal strategy, which is fine-grained as shown in Table 4 in Appendix C.

Compared to the *mining power migration attack*, the *cloud mining attack* is much more profitable. This means that for the ETH/ETC pair, renting mining power to attack ETC is much cheaper than migrating mining power from ETH. The reason may be the GPU friendliness of the ETH/ETC mining algorithm. ETH and ETC use Ethash [31] as the mining algorithm of PoW. Ethash is a memory-hard function, making it GPU-friendly while ASIC-resistant [30]. As GPU is not dedicated hardware, its mining power can be migrated to any blockchains. Therefore, renting mining power for ETH/ETH is much cheaper compared to renting mining power with dedicated hardware such as ASICs.

7 Conclusion

Honest majority is the most important assumption of PoW-based blockchains. However, this assumption does not always hold in practice. With this observation, we present the *sucker punch attacks* that utilise external mining power to temporarily dominate the PoW mining and launch 51% attacks. We design the SPA-MDP model to evaluate *sucker punch attacks* which can estimate the cost and the revenue of such attacks. We showed that it is feasible and profitable to launch *sucker punch attacks*, and rational miners have incentive to launch such attacks, as they can profit more from that. We also analyse the recent 51% attacks on ETC in January 2019 using our model, which successfully estimates the attacker's revenue and describes the attacker's strategy of attacking.

The *sucker punch attacks* imply that, the current incentive mechanism in PoW fails to incentivise miners to protect the "honest majority", but incentivises miners to launch *sucker punch attacks*. Thus, the incentive mechanism actually breaks the "honest majority".

References

[1] . Security: Delayed proof of work (dpow), 2018.

[2] . Gate.io Got Back 100k USD Value Of ETC From The ETC 51% Attacker, 2019.

[3] Aiyer, A. S., Alvisi, L., Clement, A., Dahlin, M., Martin, J., and Porth, C. BAR fault tolerance for cooperative services. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, SOSP 2005, Brighton, UK, October 23-26, 2005* (2005), pp. 45–58.

[4] Arnosti, N., and Weinberg, S. M. Bitcoin: A natural oligopoly. *arXiv preprint arXiv:1811.08572* (2018).

[5] Bonneau, J. Why Buy When You Can Rent? - Bribery Attacks on Bitcoin-Style Consensus. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers* (2016), pp. 19–26.

[6] Carlsten, M., Kalodner, H. A., Weinberg, S. M., and Narayanan, A. On the Instability of Bitcoin Without the Block Reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016* (2016), pp. 154–167.

[7] Chadès, I., Chapron, G., Cros, M.-J., Garcia, F., and Sabbadin, R. MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography 37*, 9 (2014), 916–920.

[8] coinmarketcap.com. Top 100 Cryptocurrencies by Market Capitalization, 2019.

[9] Eyal, I. The Miner's Dilemma. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015* (2015), pp. 89–103.

[10] Eyal, I., and Sirer, E. G. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers* (2014), pp. 436–454.

[11] Garay, J. A., Kiayias, A., and Leonardos, N. The Bitcoin Backbone Protocol: Analysis and Applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II* (2015), pp. 281–310.

[12] Garay, J. A., Kiayias, A., and Leonardos, N. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I* (2017), pp. 291–323.

[13] gate.io. Gate.io - the gate of blockchain assets exchange, 2019.

[14] gate.io. Gate.io research: Confirmed the etc 51% attack and attacker's accounts - gate.io news, 2019.

[15] Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., and Capkun, S. On the Security and Performance of Proof of Work Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016* (2016), pp. 3–16.

[16] Judmayer, A., Stifter, N., Zamyatin, A., Tsabary, I., Eyal, I., Gazi, P., Meiklejohn, S., and Weippl, E. Pay-to-win: Incentive attacks on proof-of-work cryptocurrencies. Cryptology ePrint Archive, Report 2019/775, 2019.

[17] Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference* (2017), Springer, pp. 357–388.

[18] Kiffer, L., Rajaraman, R., and Shelat, A. A Better Method to Analyze Blockchain Consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018* (2018), pp. 729–744.

[19] komodoplatform.com. The Anatomy Of A 51% Attack And How You Can Prevent One, 2019.

[20] Kwon, Y., Kim, D., Son, Y., Vasserman, E. Y., and Kim, Y. Be selfish and avoid dilemmas: Fork after withholding (FAW) attacks on bitcoin. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017* (2017), pp. 195–209.

[21] Kwon, Y., Kim, H., Shin, J., and Kim, Y. Bitcoin vs. Bitcoin Cash: Coexistence or Downfall of Bitcoin Cash? *CoRR abs/1902.11064* (2019).

[22] Liao, K., and Katz, J. Incentivizing blockchain forks via whale transactions. In *International Conference on Financial Cryptography and Data Security* (2017), Springer, pp. 264–279.

[23] Messamore, W. Nicehash to smaller cryptocurrency miners : If you can't beat 51% attackers who lease our hash power, join them, 2019.

[24] Morris, D. Z. The Ethereum Classic 51% attack is the height of crypto-irony, 2019.

[25] Nakamoto, S., et al. Bitcoin: A peer-to-peer electronic cash system.

[26] Nayak, K., Kumar, S., Miller, A., and Shi, E. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016* (2016), pp. 305–320.

[27] Nesbitt, M. Deep Chain Reorganization Detected on Ethereum Classic (ETC), 2019.

[28] nicehash. Nicehash - Largest Crypto-Mining Marketplace, 2019.

[29] Pass, R., Seeman, L., and Shelat, A. Analysis of the Blockchain Protocol in Asynchronous Networks. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II* (2017), pp. 643–673.

[30] Ray, J. Ethash Design Rationale, 2018.

[31] Ray, J. Dagger Hashimoto, 2019.

[32] reddit. How many confirms is considered 'safe' in Ethereum?, 2019.

[33] Ross, S. M. *Introduction to stochastic dynamic programming*. Academic press, 2014.

[34] Sapirshtein, A., Sompolinsky, Y., and Zohar, A. Optimal Selfish Mining Strategies in Bitcoin. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers* (2016), pp. 515–532.

[35] Spiegelman, A., Keidar, I., and Tennenholtz, M. Game of coins. *arXiv preprint arXiv:1805.08979* (2018).

[36] Winzer, F., Herd, B., and Faust, S. Temporary censorship attacks in the presence of rational miners. Cryptology ePrint Archive, Report 2019/748, 2019.

[37] Yu, J., Kozhaya, D., Decouchant, J., and Verissimo, P. Repucoin: Your reputation is your power. *IEEE Transactions on Computers* (2019).

[38] Zhang, R., and Preneel, B. Lay Down the Common Metrics: Evaluating Proof-of-Work Consensus Protocols' Security. In *Proceedings of the 40th IEEE Symposium on Security and Privacy* (2019), S&P, IEEE.

A Discussions on attack prevention

This section discusses short term and long term solutions to detect and prevent *sucker punch attacks*. We make use of the 51% attack incident on ETC (see Section 6) as an example, and demonstrate how to make use of SPA-MDP to gain insights that helps to defend aginst such cloud mining attacks in Section A.1.2.

A.1 Quick remedies

We first discuss several quick remedies for cryptocurrency exchanges to reduce the damage of 51% attacks. It consists of detecting potential attack attempts, and reacting upon detection through conventional risk management techniques.

A.1.1 Detecting 51% attacks

For the sucker punch attacks, the attacker needs to move a considerable amount of mining power from somewhere, such as the other blockchain or a cloud mining service.

This gives us an opportunity to detect the anomaly state where a “large” portion of mining power suddenly disappears from a source. For example, a potential victim can monitor the available compatible mining power of other blockchains or cloud mining services. If there is a sudden change on the amount of total availabe mining power, then this might indicate a potential sucker punch attack. The threshold of “large” is blockchain specific according to the risk management rules. For example, a blockchain which cares less on such attacks can set the threshold to 100% of its current total mining power. That is, once the disappearance of this amount of mining power in other sources is detected, then an alarm of a potential attack is raised. However, this will not detect an attacker who gains 90% mining power from one source, and 10% from another sources. A more cautious blockchain may set a tighter threshold, e.g. 5%, however, this may cause false positive alarms.

There are two limitations of this method. First, it may introduce false positive detections, and it is hard to identify which blockchain will be the victim upon detection. Second, it is expensive to monitor all possible stronger blockchains and cloud mining services in real-time.

A.1.2 Reactions upon 51% attacks

Upon detection, a potential victim can react to manage potential risks. Several reactions can be taken to reduce the potential damage from the sucker punch attacks. The first reaction is to increase the number N_c of block confirmations. As shown in Figure 2d, in our experiment setting (Table 5), with the increase of required number of confirmations, the related revenue decreases. In addition, decreasing the maximum amount of cash out in a single transaction also helps. As shown in Figure 2b, the higher the transaction value is, the more relative revenue an attacker can gain. Thus, decreasing the maximum value of a transaction for cash out would discourage a rational miner to launch sucker punch attacks.

Our model can be used to provide recommendations on the above mentioned parameters. For example, Figure 7 shows the impact of the value v_{tx} of transactions and the number N_c of confirmations on the 51% on ETC. This analysis is produced by using our SPA-MDP model, and all other parameters are set up according to the attack happened. This shows that if the value of transactions was limited to 9,000 ETC (approximately $38340.0) per transaction, then the attacker will not get any net revenue. On the other side, if the exchange wants to allow a maximum of 52,800 ETC (approximately $224928.0), then our model recommends that the N_c should be increased to 18 to eliminate the incentive of a rational miner to launch such attacks.

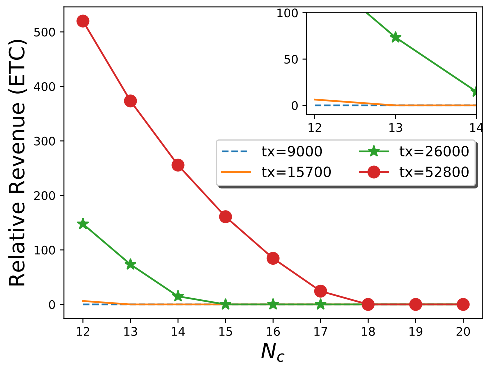


Figure 7: Impacts of v_{tx} and N_c on the ETC attack.

In addition, decreasing the maximum frequency of cash out would also limit the potential damage from an attacker, as it reduces the daily withdraw limit.

Last, if a potential attack is considered very likely, then the potential victim can halt the cash out temporarily, to increase the cost of the attack.

A.2 Long term solutions

Though easy to deploy, aforementioned quick remedies are not sufficient. First, they sacrifice the usability of blockchains. Second, all of them only minimise the effect of the potential attacks, rather than eliminating them.

Improving the PoW protocol from the protocol-level is also a promising approach to defend against our attacks. There are limited works aiming at minimising the effects of powerful miners being malicious. For example, RepuCoin [37] aims at mitigating the 51% attacks in PoW protocols by introducing the “physics-based reputation”. In RepuCoin, the weight of each miner is decided by the reputation rather than the mining power. The reputation of a miner depends on the mining power, but also takes the past contribution of miners into consideration. In this way, a sucker punch attacker cannot gain a high-enough reputation within a short time period, and the “sucker punch”-style attacks become much harder to launch.

B Other related work

B.1 Formalisations of PoW-based consensus

There are several attempts on formalising security properties of PoW-based consensus. Garay et al. [11]

formalised Bitcoin's consensus under the synchronous network setting, and extracted two refined properties, namely *common prefix* and *chain quality*. Pass et al. [29] extended this formalisation to a bounded asynchronous network setting, and Garay et al. [12] further considered the dynamic difficulty adjustment of PoW. Kiayias et al. [17] further defined another property on the liveness called the *chain growth*.

B.2 Evaluation frameworks on PoW-based consensus security

Gervais et al. [15] first proposed an evaluation framework for quantitatively analyse PoW-based consensus security using MDP, with a focus on the resistance of selfish mining and double-spending. Zhang et al. [38] generalised this framework and proposed a cross-protocol evaluation framework with three new properties measuring the resistance of several attacks on PoW-based consensus, namely the *incentive compatibility* (i.e. the relative revenue lower bound of honest miners under selfish mining attacks), the *subversion gain* (i.e. the profit upper bound of an adversary performing double spending), and the *censorship susceptibility* (i.e. the profit loss of honest miners under censorship retaliation attacks).

B.3 Evaluation of attacks on PoW-based consensus

Most research evaluating attacks on PoW-based consensus use the MDP-based models [34, 15, 18, 38] or the game-theoretic models [10, 9, 6, 26, 20, 21, 16, 4]. Our research adopts the MDP-based model to evaluate our proposed *sucker punch attacks*, which we call SPA-MDP. SPA-MDP is similar with models in [34, 15] in the notations and the processes, but in addition supports the presence of external mining power. Model the external mining power is complex, because the number of parameters will be doubled. We reduce the excessive parameters in a way that simplifies the implementation and the simulation of SPA-MDP without losing any correctness of the model.

C Optimal strategy for BTC/BCH

In this section, we show the optimal strategy of launching mining power migration attacks from BTC to BCH. We use the same experimental setting (with one pair of hashrate and transaction amount), as in Section 5. More specifically, we assume the adversary with $\alpha = 0.3$ uses the Sha256d mining power of hashrate $5000E+15$ (i.e. $h_1 = 0.125$ and $h_2 = 3.462$), and a transaction of \$300,000 to launch *mining power migration attacks*. We use the default value 6 for N_c. We apply the *ValueIteration* algorithm [33] with a discount value of 0.9 and an epsilon value of 0.1 for SPA-MDP.

Table 4 outlines the optimal strategy with notations. Each subtable describes the optimal strategy with a fixed β. For each table, the x-axis is l_h, while the y-axis is l_a. For each cell in a table, the three letters denote the optimal actions when $fork = r, ir, p$, respectively. More specifically, A, O, W, M denote **ADOPT**, **OVERRIDE**, **WAIT** and **MATCH**, respectively; **W** and **M** denote **WAIT_INC** and **MATCH_INC**, respectively; and w and m denote **WAIT_DEC** and **MATCH_DEC**, respectively. Among all cells, there are 7 different values labelled by different colours, namely: AAA in light blue; **W**AA in deep blue; **MMW** in green; OAO in yellow; **W**A**W** in red; WAW in brown; and wAw in purple (which only appears when $\beta \geq 0.2$).

We divide each matrix into four parts according to N_c (here $N_c = 6$), namely the upper left, upper right, lower left, and lower right. The upper left part represents the situation such that after the last common block in the blockchain, both honest branch and the attacker's branch do not contain enough number (N_c) of blocks as required for confirmation; whereas in the lower right part, both branches contain enough number (N_c) of blocks. In the upper right part, the honest branch contains enough number (N_c) of blocks as required for confirmation, but not the attacker's branch, whereas the lower left represents the opposite scenario.

The upper left part (when $l_a < N_c \wedge l_h < N_c$). In this scenario, the adversary's optimal action is mostly **WAIT_INC** i.e. increasing his mining power on BCH for mining more blocks. Note that in this scenario, $fork$ can only be p (i.e. the adversary's branch is still private and unpublished), and the first two letters for each cell (represent the action when $fork = r \vee fork = ir$, respectively) are unreachable states. When $l_a < N_c \wedge l_h < N_c$, the merchant does not confirm the transaction, so the adversary cannot publish his branch to double-spend. The adversary needs at least N_c blocks to revert the honest blockchain, as the merchant will accept the transaction only when $l_h \geq N_c$. Therefore, at this stage, the adversary should make $l_a \geq N_c$ as fast as possible, which can be achieved by allocating more mining power on BCH. When $l_a = 5 \wedge l_h = 0 \wedge \beta = 0.8$, the optimal action is **WAIT**. The reason is that the adversary has already gained significant advantage (5 blocks longer than the honest blockchain), and he has already secured the attack with his existing mining power with a high probability. When $\beta = 1.0$, the optimal action is **WAIT** except for $l_a = 5 \wedge l_h = 0$, where the optimal strategy is **WAIT_DEC**. In this scenario, the adversary has no more mining power for BCH, so cannot do **WAIT_INC**. When $l_a = 5 \wedge l_h = 0$, the adversary can even move some min-

Table 4: Optimal strategy for a BTC miner to launch *mining power migration attacks* on BCH, where w denotes **WAIT_DEC**, W denotes **WAIT**, **W** denotes **WAIT_INC**, m denotes **MATCH_DEC**, M denotes **MATCH**, **M** denotes **MATCH_INC**, O denotes **OVERRIDE**, and A denotes **ABORT**.

(a) $\beta = 0$

	0	1	2	3	4	5	6	7	8	9
0	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAA**	AAA	AAA	AAA
1	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAA**	AAA	AAA
2	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAA**	AAA
3	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA
4	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
5	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
6	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
7	WAW	WAW	WAW	WAW	WAW	WAW	OAO	**MMW**	**WAW**	**WAW**
8	WAW	WAW	WAW	WAW	WAW	WAW	OAO	OAO	**MMW**	**WAW**
9	WAW	WAW	WAW	WAW	WAW	WAW	OAO	OAO	OAO	**MMW**

(b) $\beta = 0.2$

	0	1	2	3	4	5	6	7	8	9
0	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA	AAA
1	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA
2	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA
3	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
4	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
5	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
6	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
7	wAw	wAw	wAw	wAw	wAw	wAw	OAO	**MMW**	**WAW**	**WAW**
8	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	**MMW**	**WAW**
9	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	OAO	**MMW**

(c) $\beta = 0.4$

	0	1	2	3	4	5	6	7	8	9
0	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA	AAA
1	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA
2	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA
3	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
4	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
5	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
6	WAW	WAW	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
7	wAw	wAw	wAw	wAw	wAw	wAw	OAO	**MMW**	**WAW**	**WAW**
8	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	**MMW**	**WAW**
9	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	OAO	**MMW**

(d) $\beta = 0.6$

	0	1	2	3	4	5	6	7	8	9
0	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA	AAA
1	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA
2	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA
3	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
4	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
5	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
6	wAw	wAw	WAW	WAW	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
7	wAw	wAw	wAw	wAw	wAw	wAw	OAO	**MMW**	**WAW**	**WAW**
8	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	**MMW**	**WAW**
9	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	OAO	**MMW**

(e) $\beta = 0.8$

	0	1	2	3	4	5	6	7	8	9
0	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA	AAA
1	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA	AAA
2	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	AAA
3	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
4	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
5	WAW	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
6	wAw	wAw	wAw	wAw	WAW	**WAW**	**WAW**	**WAW**	**WAW**	**WAW**
7	wAw	wAw	wAw	wAw	wAw	wAw	OAO	**MMW**	**WAW**	**WAW**
8	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	**MMW**	**WAW**
9	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	OAO	**MMW**

(f) $\beta = 1.0$

	0	1	2	3	4	5	6	7	8	9
0	WAW	WAW	WAW	WAW	WAW	WAW	WAW	AAA	AAA	AAA
1	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	AAA	AAA
2	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	AAA
3	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW
4	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW
5	wAw	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW	WAW
6	wAw	wAw	wAw	wAw	wAw	WAW	WAW	WAW	WAW	WAW
7	wAw	wAw	wAw	wAw	wAw	wAw	OAO	MMW	WAW	WAW
8	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	MMW	WAW
9	wAw	wAw	wAw	wAw	wAw	wAw	OAO	OAO	OAO	MMW

ing power on BCH back to BTC, so that he gains more reward from honestly mining BTC while securing the attack on BCH.

The upper right part (when $l_a < N_c \wedge l_h \geq N_c$). In this scenario, the merchant has confirmed the transaction (as $l_h \geq N_c$), but the adversary's branch falls behind the honest blockchain. The adversary's optimal action is **Abort** with $l_h - l_a \geq 7$ (the light blue upper right corner), and mostly **WAIT_INC** (**WAIT** when $\beta = 1.0$) with $l_h - l_a < 7$. When $l_h - l_a \geq 7$, the adversary's branch significantly falls behind the honest blockchain, so he should give up to reduce the damage. When $l_h - l_a \leq 5$, the adversary's branch does not fall behind too much, so he still has a chance to catch up by increasing its minging power (i.e., **WAIT_INC**). When $\beta = 0.0 \wedge l_a - l_h = 6 \wedge l_a \neq 9$ (the dark blue area), the adversary's optimal action is **WAIT_INC** with $fork = r$ (i.e. the adversary's branch is published but the honest blockchain is confirmed), but is **ABORT** with $fork = p$ (i.e. the adversary's branch is unpublished). When the adversary publishes his branch, some miners with γ honest mining power choose to mine on this branch. In this way, the adversary obtains extra mining power from other miners, so becomes more confident on the attack.

The lower left part (when $l_a \geq N_c \wedge l_h < N_c$). In this scenario, the merchant has not confirmed the transaction (as $l_h \leq N_c$). If $l_a > N_c$, the adversary has secured the attack: he can just wait for the merchant to confirm the transaction (when the honest blockchain reaches N_c), then publish his branch to revert the blockchain. If $l_a = N_c$, the adversary only needs to mine one more block to secure the attack. When β becomes bigger, the adversary is more intended to do **WAIT_DEC** (the purple area) compared to **WAIT** (the brown area) and **WAIT_INC** (the red area). Similar with the upper right part, with bigger β, the adversary has a good chance to make the attack successful, so he can use less mining power to attack BCH while using more mining power to honestly mine BTC.

The lower right part (when $l_a \geq N_c \wedge l_h \geq N_c$). In this scenario, the merchant has confirmed the transaction (as $l_h \geq N_c$). When $l_a > l_h$, the adversary can revert the honest blockchain and double-spend his money directly by **OVERRIDE** (i.e. publishing his branch). When $l_a < l_h$, the adversary's branch slightly falls behind the honest blockchain, so he can try to catch up by **WAIT_INC** (except when $\beta = 1.0$). When $l_a = l_h$, if $fork = r$ (i.e. the adversary has published his branch), the adversary's optimal action is **MATCH_INC** (except when $\beta = 1.0$). Meanwhile, if $fork = p$ (i.e. the adversary has not published his branch), the adversary's optimal action is **WAIT_INC** (except when $\beta = 1.0$). This is because when $l_a = l_h \wedge fork = r$ (i.e. the adversary's branch is published and its length is the same as the honest blockchain), the adversary lost control on his branch: he can only do **MATCH**-style actions but cannot do **WAIT**-style actions. Thus, the adversary can maximise the probability of success only by allocating more mining power to BCH. If $l_a = l_h \wedge fork = p$ (i.e. the adversary's branch is private and its length is the same as the honest blockchain), the adversary can keep waiting and increase the mining power to secure the attack.

D Experimental data

As mentioned in the main body, we present our collected data in the tables (Table 5 – Table9). We omit the detailed explaination on each table, as they are already referenced in the according sections when they are mentioned.

We fetched the blockchain data from Coinmarketcap [8] on 19 February 2019, and prices of renting mining power from NiceHash [28] on 07 April 2019. For analysing the recent 51% attack on ETC, we fetched the blockchain data from coinmarketcap [8], and the price of renting Ethash mining power from NiceHash [28], both at the time of the attack (Jan. 7th, 2019).

Note that in table 9, the "portion" represents the ratio of a stronger blockchain over a weaker blockchain, where the stronger blockchain is the first row of each mining algorithm, and all other rows of the same mining algorithm are weaker chains. For a stronger chain, the "Top Miners" represents the percentage of mining power that the top mining pools control in the stronger chain. For weaker chains, the "Top Miners" show the ratio of mining power of a top miner over the total mining power of the weaker chain. For example, the top 1 mining pool in ETH controls 27.7% mining power, and this amount of mining power about 4.563 times of the total mining power in the entire ETC network.

Table 5: Values of parameters for evaluating the SPA-MDP model.

	Notation	Default	Permuted
Mining Status	D_1	100	N/A
	D_2	10	np.arange(5, 100, 5)
	h_1	0.1	N/A
	h_2	2.0	np.arange(1, 10, 1)
Incentive-Related Parameters	R_1	50	N/A
	R_2	5	np.arange(5, 50, 5)
	v_{tx}	100	np.arange(5, 100, 5)
Adversary Network	γ	0.3	np.arange(0.1, 1.0, 0.1)
the Vigilance of the Merchant	N_c	4	np.arange(1, 10, 1)
Mining Power Price	pr	2	np.arange(0.2, 4, 0.2)

Table 6: Data of BTC/BCH, ETH/ETC and XMR/BCN for experiments.

(a) BTC and BCH

	BTC	BCH
Difficulty	6071846049920.0	199070336984
Price (USD)	3585.99	126.31
Algorithm	Sha256d	Sha256d
Hashrate(h/s)	39997.52E+15	1444.26E+15
Coins per Block	12.5	12.5

(b) ETH and ETC

	ETH	ETC
Difficulty	1.91E+15	122025268093982
Price (USD)	118.53	4.26
Algorithm	Ethash	Ethash
Hashrate (h/s)	142.00E+12	8.62E+12
Coins per Block	2	4

(c) XMR and BCN

	XMR	BCN
Difficulty	113361254717.0	40879087965
Price (USD)	43.64	0.000619
Algorithm	CryptoNight	CryptoNight
Hashrate (h/s)	9.29E+08	3.35E+08
Coins per Block	3.075	987.26

Table 7: Data of 15 selected PoW blockchains and NiceHash prices.

	Rank	Rent($/h/s)	Coin Price($)	Hashrate
Bitcoin	1	2E-18	3585.99	4E+19
Ethereum	3	1.36E-13	118.53	142E+14
BitcoinCash	4	2E-18	126.31	1.44E+18
Litecoin	8	3.34E-14	30.84	2.77E+14
Monero	14	9.13E-11	43.64	9.29E+8
Dash	15	3.53E-16	71.79	2.32E+15
EthereumClassic	18	1.36E-13	4.26	8.62E+12
Zcash	20	1.38E-08	54.77	3.36E+9
Dogecoin	23	3.34E-14	0.002132	3.76E+14
BitcoinGold	26	1.38E-08	11.93	3170000
Siacoin	46	3.74E-17	0.002389	1.88E+15
Komodo	55	1.38E-08	0.640292	4.48E+7
Electroneum	67	9.13E-11	0.006184	4.4E+9
Ravencoin	94	3.36E-13	0.011905	5.9E+12
Zcoin	99	2.79E-12	4.83	9.69E+10

Table 8: Details of relevant blockchains and mining power prices at the time of attack (Jan. 7th, 2019).

ETC Price	\$5.32
BTC Price	\$4061.47
Difficulty	131.80E+12
h_2	1.16
Coins per Block	4
Nicehash Price	3.8290 BTC/TH/day

Table 9: Summary of blockchains sharing the same mining algorithm.

Type	Mining Algorithm	Coin	Rank	Hashrate (h/s)	Portion	Top Miners		
						#1	#2	#3
ASIC-resistant	Ethash	Ethereum (ETH)	3	1.42E+14	N/A	27.7%	22.2%	12.5%
		EthereumClassic (ETC)	18	8.62E+12	1647.4%	456.3%	365.7%	205.9%
	CryptoNight	Monero (XMR)	14	9.29E+08	N/A	37%	26%	12%
		ByteCoin (BCN)	39	3.35E+08	277.3%	102.6%	72.1%	33.3%
	Equihash	Zcash (ZEC)	20	3.36E+09	N/A	33.4%	19.2%	17.8%
		BitcoinGold (BTG)	26	3.17E+06	111111.1%	37111.1%	21333.3%	19777.8%
		Komodo (KMD)	55	4.48E+07	7518.8%	2511.3%	1443.6%	1338.3%
		Aion (AION)	84	7.22E+05	1000000.0%	334000.0%	192000.0%	178000.0%
ASIC-friendly	Sha256d	Bitcoin (BTC)	1	4.00E+19	N/A	23%	16.4%	11.6%
		BitcoinCash (BCH)	4	1.44E+18	2777.8%	638.8%	455.6%	322.2%
	Scrypt	Dogecoin (DOGE)	23	3.76E+14	N/A	18.0%	16.0%	10.0%
		Litecoin (LTC)	8	2.77E+14	135.7%	24.4%	21.7%	13.6%
	X11	Dash (DASH)	15	2.32E+15	N/A	13.0%	11.0%	11.0%
		WaltonChain (WTC)	73	1.14E+15	203.5%	26.5%	22.4%	22.4%