Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack

Kartik Nayak*†, Srijan Kumar*†, Andrew Miller*° and Elaine Shi‡°
*University of Maryland, College Park,
‡Cornell University, Ithaca,
°Initiative for Cryptocurrency and Contracts (IC3)

***Abstract*—Selfish mining, originally discovered by Eyal et al. [9], is a well-known attack where a selfish miner, under certain conditions, can gain a disproportionate share of reward by deviating from the honest behavior.**

In this paper, we expand the mining strategy space to include novel "stubborn" strategies that, for a large range of parameters, earn the miner more revenue. Consequently, we show that the selfish mining attack is not (in general) optimal.

Further, we show how a miner can further amplify its gain by non-trivially composing mining attacks with network-level eclipse attacks. We show, surprisingly, that given the attacker's best strategy, in some cases victims of an eclipse attack can actually benefit from being eclipsed!

1. Introduction

Decentralized cryptocurrencies like Bitcoin [21] and other altcoins have captured the public's interest, and have been much more successful than any prior incarnations of electronic cash. Many would call the rise of these electronic currencies a technological revolution, and the "wave of the future" [1].

These emerging cryptocurrencies embody a novel blockchain technology, where miners reach consensus about the history of transactions and the status of the ledger. Initial analyses of Bitcoin's security relied on the assumption that a majority of the network, as measured by computational resources, would honestly run the default reference client. It soon became clear that this assumption should be justified by consideration of the incentives of users or attackers to deviate.

Since cryptocurrencies carry monetary value, they naturally become a valuable target of attacks. Intuitively, for a secure-by-design cryptocurrency, an attacker controlling α fraction of the network's computational resource should be able to obtain only α fraction of the mining reward. However, a malicious attacker can employ various types of attacks to gain an unfair share of the mining reward. We refer to such attacks generically as *mining attacks*. Among the most well-known are "selfish-mining"-style attacks that exploit weaknesses in the distributed consensus protocol [9], [8], and network-level attacks that seek to create network partitions between mining powers, referred to as the "eclipse attack" [13].

1.1. Our Contributions

In this paper, we take an in-depth look at the mining attack strategy space, and make several interesting revelations. We investigate strategies that increase the revenue of the attacker.

- *Selfish mining is not optimal for a large parameter space.* We introduce a new family of alternative "stubborn mining" strategies that generalize and outperform the selfish mining attack. For a large fraction of the interesting parameter space, our new strategies significantly increase the attacker's revenue. Depending on the environment parameters, stubborn mining strategies can beat selfish mining by up to 25% (even without leveraging any network-level attacks). Depending on the parameters, and at the price during the time of writing, this can translates to $73K additional gains per day in comparison with the selfish miner.
  In one of our mining strategies, called trail-stubbornness, the attacker keeps mining on her private fork even if the public fork is ahead, thus violating the longest-chain rule. This surprisingly benefits the attacker since if she happens to overtake the public later, she will have wasted more of the public's mining power. We show that in some cases, a trail-stubborn strategy can result in 13% gains in comparison with a non-trail-stubborn counterpart.
- *An attacker's revenue is increased by non-trivial combinations of stubborn mining and network-level attacks.* A stubborn miner can additionally exploit network-level attacks to further increase its gains. In particular, with a successful *eclipse attack*, the attacker isolates a victim from the rest of its peers at the network-level, by controlling its incoming and outgoing connections. We show that a space of non-trivial strategies exist when combining stubborn mining with eclipse attacks. Depending on the parameters, these strategies can sometimes result in 30% gains in comparison with the naïve usage of eclipsed nodes. We also show that surprisingly, in some parameter ranges, the

†*The first two authors contributed equally to this work.*

attacker's best strategy actually *helps* the eclipsed nodes, hence victims may have little incentive to prevent, detect, or react to such attacks.

- *Systematic exploration of strategy space.* We are the first to systematically explore the space of strategies combining selfish-mining-style attacks with network-level eclipse attacks. We show that no single strategy is a "one-size-fits-all" optimal strategy. Instead, an attacker should choose her strategy based on estimated parameters including the amount of computation power it can wield, the fraction of network it can potentially eclipse and the fraction of the remaining network that it can influence.

1.2. Interpreting Our Results

Although we significantly expand the strategy space considered by the selfish mining paper, we do not claim to study the full possible strategy space. Under the strategy space we consider, we show dominant strategies for different regions of the parameter space. However, we do not preclude the possibility of other strategies outside our strategy space that perform better.

Our findings can be interpreted as below. First, we show that the space of viable mining strategies is complicated, and that selfish mining [9] is not optimal in general. Second, we show that the possibility of combining mining attacks with network-level attacks further complicate the space of possible strategies. A notable challenge that we pose is the task of designing a consensus protocol (without pre-established trust or PKI) whose security is *formally* founded under *rationality* assumptions rather than honest majority – the latter was adopted by Garay et al. in their proof of the Bitcoin backbone protocol [10]. The findings of our paper demonstrate that achieving this could be difficult – especially if one also wishes to take into account realistic models of network formation and propagation.

1.3. Related Work

Mining attacks. Garay et al. proved certain security properties of the Bitcoin consensus protocol under highly idealized assumptions about the network propagation, and more importantly, assuming honest majority in hashpower [10]. However, it has been shown that the security of the consensus protocol can be broken when nodes are *rational* rather than honest majority [9].

Bitcoin's reference implementation mandates that, whenever some miner produces a valid block, it distributes this to the rest of the network. Eyal and Sirer [9] show selfish miners can gain an unfair share of the block reward by deviating from the reference client. Specifically, a miner with more than 33% computational power can attain disproportionate gains by maintaining a private blockchain and withholding blocks that have been mined. This forces honest miners to perform wasted computations on a stale public branch. Selfish mining works because honest miners are forced to spend (some of) their computation cycles on blocks that are destined to not be on the public chain.

Various other attacks have been studied. For example, members of a mining pool can launch a block withholding attack against the pool itself [5] – this harms the victim pool and its other members, but in fact increases the revenue of the rest of the network. Several recent works have examined the game theoretic consequences of attacks and cooperation between pools [8], [16], [14].

Network-level attacks. At the network layer, each Bitcoin node (about 7000 in total, today) is connected over TCP to many peers, with a default maximum of 125. The peer-to-peer connections between these nodes can be inferred through various techniques [2], [19].

Heilman et al. [7] demonstrated a network-level eclipse attack where a single node monopolizes all possible connections to a victim and *eclipses* it from the network. This way the eclipsing node can filter the eclipsed node's view of the blockchain. Their paper [7] describes elaborate techniques to achieve eclipse attacks on the Bitcoin network. Although a few proposed counter-measures have been implemented that reduce the feasibility of carrying out an eclipse attack by a single node, multiple nodes can collude and still succeed in eclipsing – in particular, in Section 7, we argue that it may in fact be incentive compatible for selfish players to collude in launching eclipsing attacks. Furthermore, we provide a basis for understanding *how* an attacker could exploit an eclipsed node to profit and analyze the gains that can be achieved.

Knowledge of the Bitcoin network can further help a network-level attacker. For example, Coinscope[19] proposes non-trivial techniques to map out the Bitcoin network topology as well as the hashpower of various nodes. Such knowledge would enable an attacker to make targeted attacks to eclipse mining entities.

Concurrent and independent work. Concurrent to and independent of our work, Sapirshtein et al.[23] also observe that selfish mining is suboptimal. They define a broad strategy space and use a combination of analytic bounds and numeric solvers to compute approximately-optimal strategies from this space. Their strategy space is a generalization of our stubborn mining strategies; however, they do not consider how to compose mining attacks with eclipse attacks.

2. Modeling of Bitcoin Mining

We begin by defining the basic model of Bitcoin mining (cf. [9]) in the presence of an attacker. We will use this basic model throughout this paper; however, later we will extend the model to include eclipse attacks as well.

2.1. Defining Key Parameters

Our mining model has three main parameters:

- **Hashpower of attacker (i.e., Alice)** α: the fraction of the network's total hashpower controlled by the attacker, henceforth referred to as "Alice".

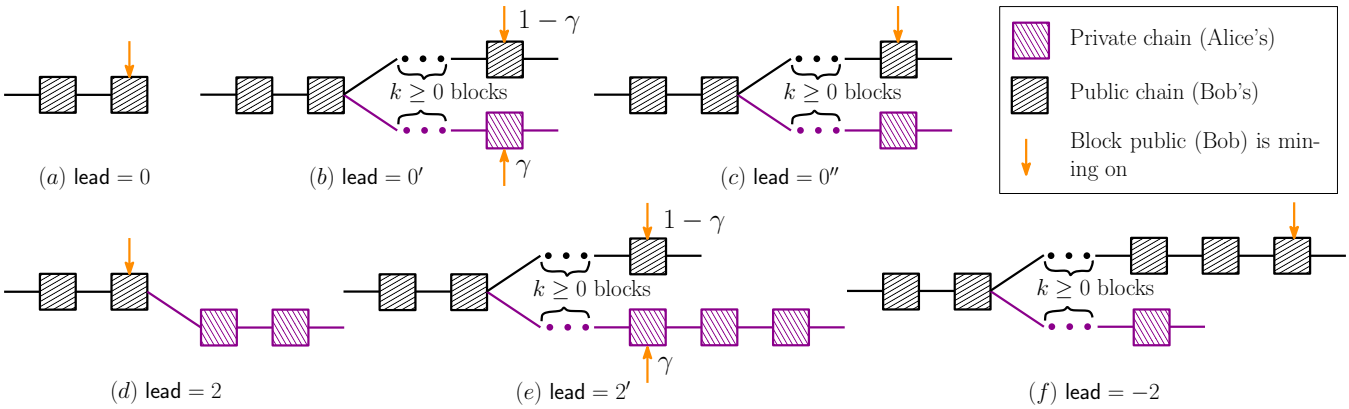


Figure 1: **Representation of Alice and Bob's blockchain at different Markov chain states.** Bob's public chain is represented using black boxes, while Alice's private chain is represented using magenta boxes. The orange arrow indicates the block(s) at which Bob is presently mining. When two orange arrows exist, it means that Bob's mining power is divided across two different forks.

TABLE 1: Table of notations

Bitcoin mining model	
α	Computation power of Alice (stubborn and/or eclipsing miner)
β	Computation power of Bob (public miners, honest)
γ	Fraction of Bob's network that will mine on Alice's block when Alice and Bob have released a block at (approximately) the same time resulting in an equal length fork.
Stubborn mining strategies (Section 3)	
S	Selfish mining strategy
H	Honest strategy
L-stubborn	Lead stubbornness - risks blocks after a high lead
F-stubborn	Equal fork stubbornness - does not reveal next block after an equal length fork
T_j-stubborn	Trail stubbornness - does not merge with public chain even when trailing the public by j blocks
Eclipse mining strategies (Section 5)	
λ	Computation power of Lucy (eclipsed miner, honest)
C	Collude strategy - Alice colludes with Lucy to create a common private chain
D	Destroy strategy - Alice destroys Lucy
DNS	Destroy if no stake - Alice colludes with Lucy only if it has a stake in their common private chain

- **Hashpower of the honest public (i.e., Bob)** β: the fraction of the hashpower of the remaining network, henceforth referred to as "Bob". Note that $\beta + \alpha = 1$.
- **Alice's network influence** γ: fraction of Bob's network (in terms of hashpower) that will mine on Alice's (i.e., the attacker's) block when Alice and Bob have released a block at (approximately) the same time resulting in an equal length fork.

The notations used in the paper are summarized in Table 1. We represent the attacker and the rest of the network only in aggregate; our model does not distinguish between the various separate entities (e.g., mining pools, industrial mining operations) that constitute the network.

2.2. Markov Chain Model of Mining

We model Bitcoin mining using a Markov decision process, where the state captures relevant aspects of the information available to Alice and Bob, and the transitions occur when Alice or Bob find a new block. If a honest miner (i.e., one that runs the default reference client software) finds a block, he publishes the block immediately. In our model, however, Alice may keep a *private chain* (i.e., hidden from Bob) of blocks she has found.

We note that our model generalizes that of the selfish mining work [9] – in particular, as a warm up, we will express selfish mining in our model in Section 2.3.

Defining states. Roughly speaking, each state in the Markov chain encodes the following dimensions of information:

- Alice's *lead*, i.e., how much Alice's private chain is ahead of Bob's, i.e.,

$$\mathsf{lead} := \mathsf{len}(\text{Alice's chain}) - \mathsf{len}(\text{Bob's chain})$$

- Whether there exists a *fork*. As shown in Figure 1, for the same value of lead, there are two cases: 1) no fork, i.e. Bob and Alice are mining on the same chain, either

on top of the same block (Figure 1a) or Alice is ahead of Bob such that the last few blocks are mined by Alice and not known to Bob (Figure 1d) ; 2) with fork, i.e., Alice and Bob are mining on different chains. In other words, the head of the longest chain seen by Bob is not a predecessor of Alice's chain (Figures 1b, 1e).
- Which chain's head Bob is mining on. Bob always mines on the longest chain it sees. When there is a fork, if the revealed portion of Alice's fork is equal length as Bob's fork, then depending on when these blocks are revealed, Bob may be mining on 1) divided between its own fork and Alice's fork – if the two competing blocks are released closely in succession (Figures 1b, 1e); or 2) all mining on its own fork – if Bob's fork was released first (see Figure 1c).

Naming states. For clarity, we use the notation

$$\mathsf{lead} = 0, 0', 0'', 2, 2', -2$$

to denote different states in the Markov chain. The numeric value encodes Alice's lead (a negative value indicates trailing behind). The prime (i.e., single apostrophe) notation denotes that there exists a fork, the revealed portion the two forks are equal in length, and that Bob is divided between mining on these two forks. Similarly, states with only numerical value (no apostrophe) denotes that there does not exist a fork. The double-prime (i.e., double apostrophe) notation denotes that there exists a fork, the revealed portion the two forks are equal in length, and that Bob is all mining on its own fork. Figure 1 graphically illustrates all of these cases and may aid understanding.

We note that besides the states depicted in Figure 1, other conceivable states are either impossible by definition of Bob's honest behavior or are undesirable choices for Alice. We therefore elide these.

State transitions. In any state, Alice has α probability of mining the next block, and Bob has β probability of mining the next block – but in some cases (the single-prime states), Bob's mining power can be divided between two forks. As we shall see, the effects of each transition is defined by Alice's strategy.

Mining revenue. Whenever a miner finds a block, she earns a reward of freshly minted Bitcoins. Currently, and for the near future, this reward is 25 BTC per block (i.e., at the time of writing, $\approx$ \$6000USD per block). However, only blocks on the *main chain* count towards their income. For example, whenever there is a fork (two blocks of the equal length, i.e., $lead = 0'$), one of the two blocks will eventually be discarded. Furthermore, every two weeks, Bitcoin automatically adjusts the difficulty of the puzzle so that on average one block is added to the *main chain* every 10 minutes; thus regardless of the rate of discarded blocks, miners on the whole earn mining rewards at a constant rate. We are therefore primarily interested in studying the *steady state* ratio of blocks in the main chain mined by Alice or Bob, as this ultimately reflects the revenue (i.e., in dollars per day) to each party.

2.3. Expressing Honest and Selfish Mining

Alice's mining strategy is defined by two decisions: *i)* when to *reveal* the private chain to the public, and *ii)* when to *accept* the public's chain (i.e., mine off the public's block chain head).

Honest mining: If Alice follows the protocol and runs the reference client, then she *reveals* a block immediately after she mines it. Alice *accepts* the longest block and mines on top. Note that equal forks do not occur[1].

Selfish mining [9]: The selfish mining strategy (illustrated in Figure 2) consists of the following deviations from honest mining:
- When $\mathsf{lead} = 2$ and Bob mines the next block: *Reveal* Alice's entire private chain to Bob (resulting in $\mathsf{lead} = 0$).
- When $\mathsf{lead} = 0'$ and Alice mines the next block: *Reveal* Alice's private chain to Bob (resulting in $\mathsf{lead} = 0$).
- When $\mathsf{lead} = 0$ or $\mathsf{lead} > 0$ and Alice mines the next block: Do *not* reveal Alice's private chain
- Alice always *accepts* the longest chain.

The net effect of selfish mining is to "waste" mining power on blocks that eventually get discarded. Sometimes Alice spends effort to no avail and her private chain falls behind Bob's; on other occasions, Bob wastes work while Alice is already ahead. The surprising result [9] is that for many values of α and γ, this strategy causes Bob to waste more work than Alice. After the network adjusts the puzzle difficulty to compensate for discarded blocks (which do not contribute to the main chain), Alice earns more revenue than if she were mining honestly.

2.4. Revenue Gain

We consider relative gain of Alice in comparison to Bob as a metric to evaluate a strategy. The relative gain of selfish mining (SM) strategy compared to a honest strategy (H) is defined as:

$$\text{relative\_gain}(SM, H) = \frac{\text{gain}_{SM} - \alpha}{\alpha}$$

More generally, given strategies X and Y, the relative gain of X over Y is defined as:

$$\text{relative\_gain}(X, Y) = \frac{\text{gain}_X - \text{gain}_Y}{\alpha}$$

where gain_X is the fraction of blocks earned by Alice under strategy X. Observe that the relative gain metric is normalized with respect to the "fair share α" that represents Alice's gain under a honest strategy.

2.5. Plausible Choices of α and γ

As the attacker's profit will depend on the key parameters α and γ, in this section, we discuss realistic choices

1. Our framework effectively models instantaneous block propagation. In reality, due to network latency, sometimes a fork will occur if two honest miners find a block at approximately the same.

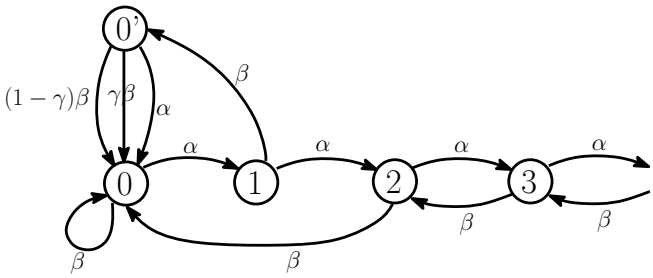


Figure 2: **Selfish Mining [9].**

TABLE 2: Hypothetical attack scenarios and parameters

α	40%	largest mining pool over most of 2014
	41%	the two largest mining pools today
	21%	largest pool today
	17%	if "unknown" mining power is a rogue miner
γ	$0-0.92$	depending on the attacker's influence on the overlay network (see Section 2.5)

of values for these parameters based on a combination of realistic measurements and reasoning of likely attack scenarios.

Plausible values for α. As mentioned earlier, α is the fraction of the attacker's hashpower relative to the entire network. As is well-known, most Bitcoin miners participate in mining pools where they join efforts in solving computational puzzles and share block rewards. Today, the top 9 pools account for 75% of the network. Mining pools might deviate from the honest protocol to maximize their rewards [9]. We can easily measure overall hashpower of the Bitcoin network using the timestamps and proofs-of-work published in the blockchain itself. The distribution of hashpower among mining pools and other entities can be inferred using several heuristics (e.g., mining pools typically sign the blocks they mine). Figure 3 shows an estimate from blockchain.info based on such heuristics. According to this snapshot (May 2015), the largest pool at the time of writing represents 21% of the network's cumulative hashpower. In most of the past year (2014) the largest mining pool was over 40%, and briefly exceeded 51% [17]. The "unknown" hashpower cannot be attributed to any publicly-known entities. It could include individual "solo-miners", private pools, or even deliberately obfuscated hashpower from other pools.

We only consider values of α under 50% in this paper – when the attacker controls over 50% of the hashpower, the fundamental security properties of the cryptocurrency are broken anyway (e.g., the attacker can guarantee she earns *all* of the mining reward, and can revert or delay transactions).

Table 2 summarizes what we consider realistic values for parameter α. For example, *we will consider* $\alpha = 40\%$ *to be a typical value* – this could correspond to either the compromise of a large mining pool (i.e., such as GHash.IO during most of 2014), or else to a coalition or simultaneous compromise of several top mining pools (such as AntPool and F2Pool) today. (We discuss implications of various scenarios in Section 7).

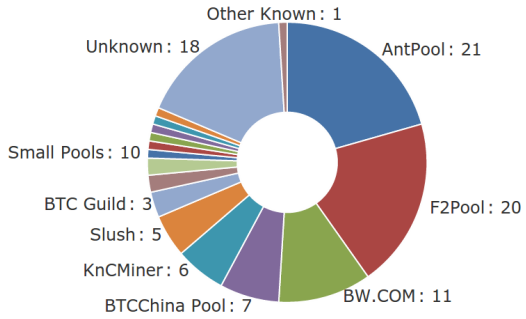


Figure 3: Known mining entities and their hashpower (from `https://blockchain.info/pools`, accessed May 16 2015).

Plausible values for γ. Suppose that at some point, the Markov chain is in state lead $= 2'$ or lead $= k'$ for any $k > 1$ (see Figure 1e). At this point, Bob finds a block and announces it. Alice can now immediately announce a next block on her private chain to create a race with Bob's new block – and the Markov chain state transitions to lead $= (k-1)'$. The fraction of Bob that will accept on Alice's fork depends on to what extent Alice can successfully race the arrival of Bob's new block.

Let A denote Alice, and let i and j denote two public miners (either mining pool or solo miner) that are part of Bob. When i is the miner that found the new block, whether j will accept it depends on the effective latency of the two overlay paths $i \rightsquigarrow A \rightsquigarrow j$ and $i \rightsquigarrow j$.

There are numerous possible avenues Alice can take to increase the likelihood of j accepting her own block instead of the one found by i. First, Alice can perform attacks targeting the overlay network's topology and place corrupted nodes on paths between (i, j) pairs. These attacks can be exacerbated by known weaknesses of Bitcoin's networking stack. For example, each Bitcoin node by default accepts only 117 incoming connections, and an attacker can easily consume these slots by establishing multiple connections to every peer. Second, Alice can attempt to reduce the end-to-end latency of the overlay paths she controls. As is known, the Bitcoin protocol takes three rounds of interaction to actually deliver a block. The first two rounds are optional, but serve to reduce unnecessary network traffic; an attacker can trivially skip these, and achieve better latency accordingly. Finally, known vulnerabilities such as the "Inv-Block" attack [12], [19] effectively allows the attacker to sway j to only request i's block through the overlay path $i \rightsquigarrow A \rightsquigarrow j$, as long as the attacker can relay a small metadata message to j more quickly than the $i \rightsquigarrow j$ overlay path.

Despite these known weaknesses, the Bitcoin network also has several defenses that may mitigate such attacks. Coinscope [19] provides evidence that the public Bitcoin network is not uniformly random and the topology is skewed by well-connected "super-nodes" running customized software (e.g., operated by exchanges or other services). Even a single benign super-node can significantly improve network

propagation [6]. Furthermore, large miners and mining pools are believed to peer directly (e.g., a Bitcoin core developer operates a dedicated "*miner backbone*" [4] that is immune to Inv-Block attacks) in order to defend themselves against network attackers.

As a simple model, we consider a graph where public miners i and j have an edge $i \twoheadrightarrow j$ if $i \rightsquigarrow A \rightsquigarrow j$ is a higher latency path than $i \rightsquigarrow j$, i.e., if miner i finds a block, then miner j will accept that block, even if the attacker releases a tying block. An effective "miner backbone" means that the large mining pools form a clique. Mining pools can be expected to have a self-edge — even an eclipse attacker cannot prevent a pool from building on its *own* recently-mined block. On the other hand, a collection of many vulnerable individuals may be modeled as a single vertex without a self-edge. We can compute the effective value of γ from such a graph as:

$$\gamma = 1 - \sum_{i \twoheadrightarrow j} \frac{\beta_i \beta_j}{\beta^2}$$

where β_i is the hashpower of i (expressed as a fraction of the total network). More specifically, the sub-term $\frac{\beta_i}{\beta}$ denotes the conditional probability that i finds the next block given that Bob finds a block. [2]

We now use the known distribution of mining hashpower to pose plausible attack scenarios. For example, suppose the attacker is the largest mining pool ($\alpha = 21\%$,$\beta = 79\%$), another $\beta_0 = 27\%$ (rounding down) comprises the small pools and other unknown or uncategorized miners, and the remaining 52% is distributed among six other large pools as shown in Figure 3. For a worst-case attack, we may model β_0 as disconnected and vulnerable nodes (i.e., no self-edge for β_i), and allow the attacker to defeat the miner backbone and win every race (i.e., no edges between different pools), then the effective γ value would be 0.92. Alternatively, we may suppose the "miner backbone" is effective, and that the other miners are well-connected (i.e., there is a self-edge for β_0), but the attacker can win races between the large pools and the other miners: this would result in a γ value of 0.45. Lower values of γ are likely attainable at lower cost to the attacker.

We will therefore consider the entire range $0 \leq \gamma \leq 0.92$ *to be plausible values.*

3. Stubborn Mining

In this section, we introduce a new class of Bitcoin mining strategies, called *stubborn mining*, that strictly generalizes (and improves upon) the prior known *selfish mining* strategies [9].

Intuition. In a nutshell, all known deviant mining strategies work by selectively withholding blocks mined by the attacker, causing the rest of the network to *waste its hashpower* on redundant blocks. The selfish mining strategy, in particular, withholds blocks when it is "in the lead" (i.e., when it has created a private chain longer than that of the honest network), but cooperates with the honest network when it falls behind. The key insight behind our new *stubborn mining* strategies is that *the attacker should not give up so easily!* Instead, the attacker can often increase profits by mining on its private chain more often, even under circumstances where a selfish-mining attacker would acquiesce to the public chain.

As mentioned earlier in Section 2, abstractly, a mining strategy defines a state machine, where each state represents lead of Alice's private block chain over Bob's public block chain and whether there exists a fork. There are two types of state transitions in this state machine:

1) State transitions that take place when Bob finds a new block. In this case, Bob always announces the new block immediately. At this point, Alice has the freedom of performing the following actions: 1) if Alice's private chain is leading, Alice can decide whether and how many blocks to reveal from its private chain to Bob; and 2) if Alice's private chain is sufficiently behind Bob's chain, Alice may choose to abandon mining on its private chain and accept Bob's chain. Alice's decision in this case will define how the Markov chain will transition when Bob mines a next block.
2) State transitions that take place when Alice finds a new block. In the latter case, i.e., when Alice mines a next block, we assume that Alice may simply continue to mine on her own private chain. Whether Alice immediately reveals her new block to Bob depends on the strategy and the current state.

Roadmap. Depending on Alice's actions when Bob mines a new block, we will introduce the following three stubborn strategies.

1) Lead stubborn, a.k.a., "L-stubborn" (Section 3.1)
2) Equal Fork stubborn, a.k.a., "F-stubborn" (Section 3.2)
3) Trail stubborn, a.k.a., "T_j-stubborn" (Section 3.2)

These three strategies are not mutually exclusive but can be combined to form hybrid strategies (Section 3.3).

3.1. Warm Up: "Lead-Stubborn" Mining

We begin by describing a simple strategy, called *lead-stubborn*, that outperforms selfish mining. We will describe lead-stubborn mining by comparing it with selfish mining [9]. Figure 4 shows the state machines of selfish mining and lead-stubborn mining simultaneously.

As we can see, when $\mathsf{lead} = 2$ and if Bob finds the next block and closes the gap by 1, the selfish miner would immediately reveal her private chain to guarantee that the network chooses her private chain over Bob's. Therefore, the state transitions to $\mathsf{lead} = 0$. In lead-stubborn mining, instead of revealing her entire private chain, Alice reveals the next block on her private chain only, to match the length

2. Here we make a simplifying assumption that if $i \twoheadrightarrow j$, then the attacker always loses the race between public miners i to j. In reality, overlay paths have variance in propagation time and bandwidth, therefore the existence of an edge is not binary, but can be fractional. However, our basic model suffices for estimating a plausible range of γ.

TABLE 3: Possible actions of Alice (stubborn miner) upon state transition, and expressing strategies in terms of these actions.

State Transition	Action	Selfish Mining	Stubborn Strategy
lead $= 2$ and Alice finds a block	Reveal private chain to Bob?	Reveal	Do not reveal in L-stubborn
lead $= 0'$ (with a fork) and Alice finds a block	Reveal private chain to Bob?	Reveal	Do not reveal in F-stubborn
Behind by x, $x \geq 0$ and Bob finds a block	Accept Bob's block?	Accept	Do not accept till $x = j$ in T_j-stubborn

of the public chain. In this case, γ fraction of Bob will mine on Alice's private chain, and $1-\gamma$ fraction mines on Bob's fork; and the state transitions to lead $= 1'$. This has pros and cons for the attacker Alice: if the $(1-\gamma)$ fraction of Bob succeeds in advancing Bob's chain, Alice may risk losing her private chain. However, if Alice or the γ fraction of Bob advances Alice's fork, then Alice has successfully diverted a part of Bob, $(1-\gamma)$ fraction, to do useless work.

Moreover, when lead $= k$ for some $k > 2$, and if Bob finds the next block, the selfish miner [9] does not reveal her private chain, and thus the state machine transitions to lead $= k-1$. However, the lead-stubborn miner would again, immediately reveal one more block on her private fork, and the state thus transitions to lead $= (k-1)'$.

More generally, we could consider a more sophisticated lead-stubborn miner whose action (including whether to reveal her private chain and how many blocks to reveal) depends on the value of lead, i.e., state-dependent actions. Our simulation results suggest that even without considering state-dependent actions, a simple-minded lead-stubborn-miner already achieves up to 13.94% higher gains than selfish miner under typical parameterizations of α and γ (see Section 2.5).

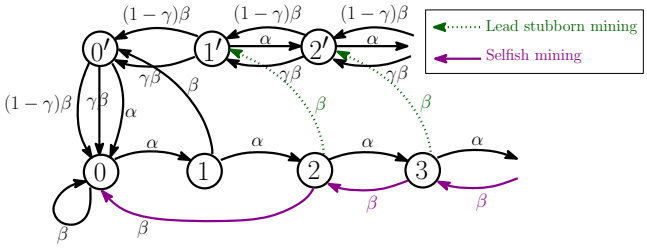


Figure 4: **Lead-stubborn mining.** Black + magenta transitions define selfish mining. Black + green transitions define lead-stubborn mining. Markov chain states are defined in Section 2.2 and Figure 1.

3.2. Other Stubborn Mining Strategies

We now explore other possible strategies besides lead-stubborn mining. All these stubborn-mining strategies are in Table 3 where we highlight the difference from the selfish miner.

1) **Equal Fork stubborn (F-stubborn)**: When Alice wins a race in state lead $= 0'$, the selfish-miner [9] would hurry to reveal her new block to the public, transitioning to lead $= 0$. By contrast, the F-stubborn miner would conceal her new block and continue mining on it privately, thus proceeding to state lead $= 1$. In Figure 5, this event corresponds to the dotted blue edge (labeled with α) leaving state $0'$ into state 1.

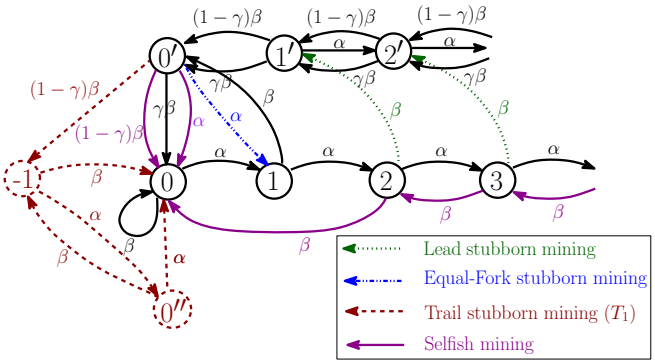


Figure 5: **Lead, Equal-Fork, and Trail Stubborn mining strategies.** Black + magenta transitions denote selfish mining [9]. Black + green transitions denote lead-stubborn mining. Black + blue transitions denote equal-fork stubborn mining. Black + brown transitions denote T_1-stubborn mining. Markov chain states are defined in Section 2.2 and Figure 1.

2) **Trail stubborn (T_j-stubborn)**: When Alice's private chain falls behind the public chain, she may decide to continue mining on it anyway, in the hope of catching up. In this case, the Markov model includes negative states (e.g., lead $= -j$ indicates that Alice's private chain lags behind the public by j blocks).
We consider a family of "trail stubborn" strategies parameterized by a threshold j, such that if Alice is T_j-stubborn then she accepts the public blockchain once her private chain falls $j+1$ behind. Thus a T_j-stubborn strategy is characterized by a β edge from lead $= -j$ to lead $= 0$. Finally, when Alice catches up from lead $= -1$, the state machine transitions to lead $= 0''$: in this special double-primed state, all of Bob is mining on the public fork since Alice's private chain just caught up from behind (*c.f.* in the single-primed states, Bob's mining power is divided between the two forks). The T_1 strategy is illustrated in Figure 5 by the brown dashed states and edges.

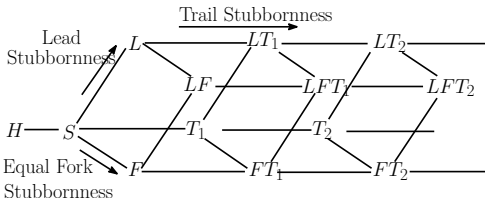


Figure 6: **Partial ordering of (hybrid) stubborn strategies.** All edges' directions are along one of the three arrows. A path from a strategy X to a strategy X' to its right indicates that X' is "more stubborn" than X i.e. X' spends more time mining on a private chain than X.

3.3. Hybrid Stubborn Mining Strategies

Altogether, these three insights (L, F, and T_j stubborn) represent heuristic components of mining strategies that can be applied independently or jointly. We represent these choices pictorially in Figure 6 as dimensions on a directed graph.This chart suggests a partial ordering among the strategies. Indeed, if there is a path from strategy X to X' in the directed graph, then X' is "more stubborn" than X in the sense that X' spends more time mining on a private chain than X. Looking ahead, our simulation results will show that this intuitive notion of stubbornness is a useful heuristic for identifying the best strategy depending on the network parameters.

Further possible extensions. These are not the only strategies that are possible. In fact, there are an infinite number of possible strategies. Such strategies can be developed by combining different strategies. One way to do so is based on conditions. A miner may change its stubbornness conditionally based on certain factors, for example depending on the length of its private blockchain. As an instance, a miner may behave according to the T_2-*stubborn* strategy if its private blockchain has j blocks for $j \geq 3$, and T_0-*stubborn* otherwise. If the miner's private blockchain contains at least three blocks in total, then it will continue to mine on it until it falls behind the public blockchain by 2. Otherwise, if the private blockchain contains fewer than three blocks, the miner accepts the public blockchain as soon as it falls behind.

Allowing strategies to depend on the length of the private blockchain adds another dimension to the potential strategy space. There may be other strategies based on different conditions as well. In this paper, we limit our attention to strategies that depend only on lead, and leave possible generalizations for future work.

4. Stubborn Mining Results

4.1. Methodology

In this section we use numeric simulations to evaluate the profitability of our stubborn mining strategies. We simulate the gains of each of the parties when Alice employs these strategies, while Bob mines honestly. For each choice of α, γ, and each strategy, we simulate 100 sample paths of the state machine, where each sample path contains 10^5 iterations. For each sample path, it is easy to calculate the respective gains of Alice and Bob. Therefore, from the 100 sample paths, we can estimate both the mean of Alice's and Bob's gains, as well as the confidence interval for this estimate.

4.2. Which strategies earn the most revenue?

Result 1. There is no one-size-fits-all dominant strategy for all parameters.

In particular, Figure 7b shows the dominant strategy under different parameters. Region R1 denotes the parameter space when honest mining wins. Region R2 denotes the parameter space when the selfish-miner wins. All other colors denote our new stubborn strategies. For trail-stubbornness, our results depict strategies involving only T_1. For T_2 and hybrid strategies involving T_2, we checked a large portion of the parameter space and did not observe a case where T_2 outperformed T_1. Therefore, trail-stubborn strategies T_j for $j \geq 2$ are likely effective only for tiny regions of the parameter space.

Result 2. *Selfish mining is not optimal for a large fraction of the parameter space.*

Specifically, Figure 7a shows the region where selfish mining outperforms honest – this was observed by Eyal et al. [9]. Interestingly, Figure 7b shows that selfish mining is the dominant strategy (among the strategy space we consider) only for a narrow range, i.e., the region labeled 2 in Figure 7b.

4.3. How much revenue can a miner earn using our new strategies?

Figure 8 shows the gain of the best stubborn strategy compared to baseline strategies (honest mining or selfish mining).

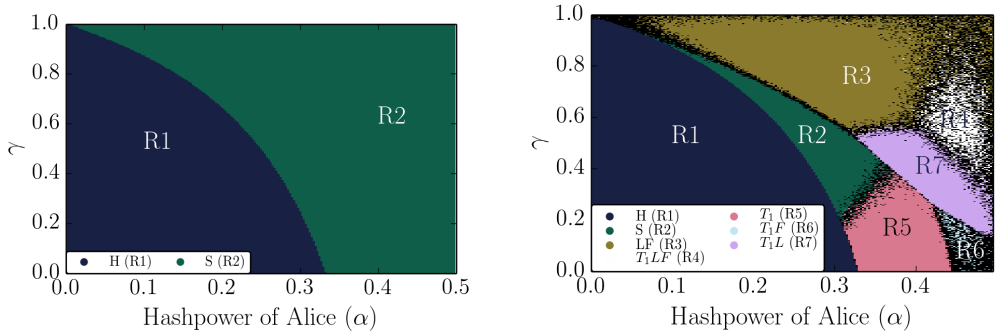
We highlight several observations:

Result 3. Stubborn mining strategies can perform up to 25% better than selfish mining for many reasonable values of α and γ. For example, at $\alpha = 0.4$, $\gamma = 0.9$, the best stubborn strategy outperforms selfish mining by 23% and outperforms honest mining by 63%.

As seen in Figure 8b, for a typical value of $\gamma = 0.9$, selfish mining strategy is outperformed by other stubborn strategies at $\alpha \geq 0.15$. At the time of this writing, all the Bitcoin miners together earn ~$800K per day. Thus, a 23% increase in computation power corresponds to $800K $\times$ 0.23 $\times$ 0.4 = $73K of additional revenue per day.

Result 4. Although seemingly counter-intuitive at first sight, for $\alpha > 0.33$ trail stubbornness outperforms non-trail-stubborn counterparts by up to 13%.

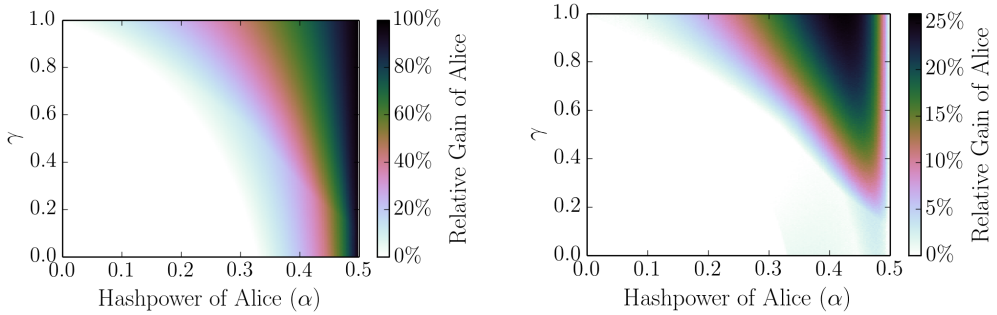
When Alice's private chain is lagging behind, the conventional wisdom is that Alice should give up and mine



(a) Selfish mining [9] outperforms honest in region 2.

(b) Stubborn mining outperforms selfish mining in regions R3-R7.

Figure 7: **Dominant strategies for different values of α and γ.** The black dots indicate regions with no dominant strategy at 95% confidence – this typically happens at the boundary of strategies.



(a) Compared to honest mining.

(b) Compared to selfish mining.

Figure 8: **Relative gain of Alice's best stubborn strategy.**

on the public's chain. However, a trail-stubborn miner does not give up. By not giving up, although it is less probable for Alice to overtake the public again, her rewards are larger when she does manage to overtake the public. Trail stubbornness is effective for reasonably large values of α.

In Figure 7b, regions R4-R7 depict when trail-stubbornness helps. In Region R5 of Figure 7b, T_1-stubborn mining is the best strategy, and leads to 1.44% additional gains relative to selfish mining, and 94.35% gains relative to the honest miner.

Trail stubbornness is also desirable as part of hybrid strategies (e.g., regions R4, R6, and R7 in Figure 7b). In these regions, Strategy LT_1 has up to 12.43% relative gain over L, FT_1 up to 2.73% over F, and LFT_1 up to 8.4% over LF strategy.

Result 5. The relative gain from stubborn mining (relative to honest mining) is greater when either γ or α increases.

As seen from Figure 8a, the gains increase with increasing value of one of γ and α for fixed value of the other.

4.4. Comparison with Sapirshtein et al.[23]

Concurrent to and independent of our work, Sapirshtein et al.[23] also observe that selfish mining is suboptimal. They define a broad strategy space and use a combination of analytic bounds and numeric solvers to compute approximately-optimal ("eps-OPT") strategies given α and γ. However, they do not consider how stubborn mining attacks can be composed with eclipse attacks (Section 5).

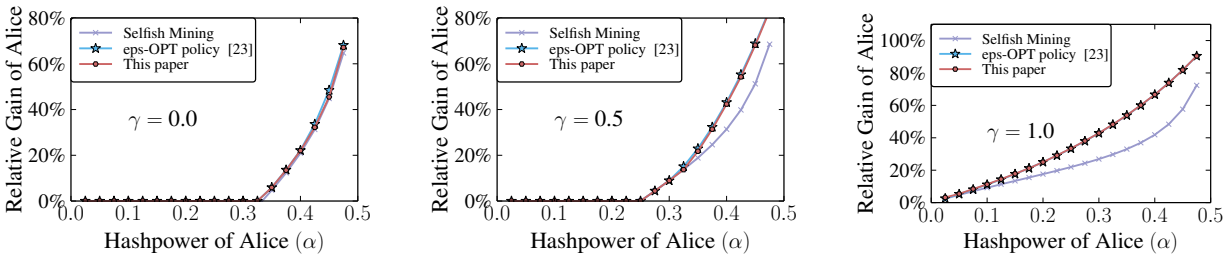
In Figure 9, we compare the revenue of our best stubborn strategies with those reported by Sapirshtein; our stubborn mining strategies perform very similar (at worst within 1.4% relative gain versus honest mining) to theirs.

5. Exploiting an Eclipsed Miner

In this section, we investigate how stubborn mining strategies can be (non-trivially) composed with eclipse attacks.

We use the terminology eclipse attack to describe network attacks where the adversary essentially partitions the honest miners into Bob (the public), and Lucy (the eclipsed) as shown in Figure 10. Alice controls the communication between Bob and Lucy.

Prior works demonstrating eclipse attacks. The feasibility of eclipse attacks (either short-term or long-term) have been demonstrated in a couple of earlier works [13], [11]. These works show how to exploit flaws in Bitcoin's networking stack to facilitate such eclipse attacks without having to



(a) $\gamma = 0.0$ (b) $\gamma = 0.5$ (c) $\gamma = 1.0$

Figure 9: **Comparing stubborn mining with the results of Sapirshtein et al. [23]** Relative gain due to stubborn mining strategies perform very similar to eps-OPT.

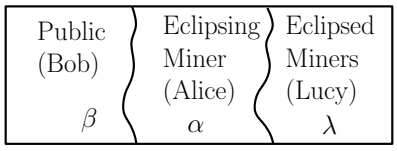


Figure 10: An eclipse attack where the adversary Alice partitions the honest network into Bob (the public) and Lucy (the eclipsed). We use the notation β, α, λ to denote the percentage of hashpower owned by Bob, Alice, and Lucy respectively.

deploy a large number of corrupted peers. Moreover, these works suggest the possibility of combining the eclipse attack with a selfish mining attack to allow an adversary to gain a higher advantage in the mining game.

Composing stubborn mining with an eclipse attack. In the remainder of this section, we systematically explore how a stubborn mining attack can be composed with an eclipse attack. We first describe a few naïve methods for combining the two attacks (including "destroying" or "colluding with" the eclipsed victim Lucy). We then show how the adversary can increase its payoff by composing stubborn mining and eclipse attacks in non-trivial ways. Surprisingly, we show that in some cases, the attacker's best strategy leads to a benefit for the eclipsed nodes.

5.1. Extended Model with an Eclipsed Miner

We extend the basic model from Section 2 with a third party, Lucy, who represents a mining entity accounting for a λ fraction of the network's hashpower (where $\alpha+\beta+\lambda = 1$) (see Figure 10). Lucy is the victim of an eclipse attack mounted by Alice. That is, Alice partitions the network and controls all of the connections between Lucy and Bob.

Our goal is to find strategies that lead to the highest gains for Alice. We only concern ourselves with Alice's steady state revenue in the view of the public network's (i.e., Bob's) blockchain; a private chain kept only between Alice and Lucy does not count.[3]

3. In reality, a private chain would allow Alice to perform double-spend attacks against Lucy (if Lucy consists of any merchants or exchanges).

Most interesting regions of the parameter space. Through the rest of this paper we will primarily focus on the case when $\alpha + \lambda < \beta$; that is, Alice and Lucy together still comprise less than half the network's total hashpower. As we shall soon see, under these conditions the best strategy depends heavily on the exact parameters and involves strategically delaying messages, the eclipsing miner can exploit the victim for greater profit. Other regions are either degenerate or not as realistic. For example, $\alpha > \beta$, Alice should simply deliver no messages between Lucy and Bob, as this degenerates to a 51% attack where Alice overwhelms Bob and earns 100% of the mining reward. The remaining regions of $\alpha + \lambda > \beta$ may be interesting, but are less likely in realistic scenarios since it requires the adversary to wield and eclipse a large fraction of the network.

5.2. Strategies Exploiting Eclipse Attack Victims

Naïve compositions. We first describe some naïve (i.e., extreme) ways through which Alice can combine stubborn mining with an eclipse attack.

- *No eclipsing*: Even if Alice controls all the connections between Lucy and Bob, she allows full communication of blocks between Bob and Lucy and does not exploit the attack. Alice may perform any of the stubborn strategies against the union of Lucy and Bob. Clearly, for every strategy of this class, Alice's revenue corresponds to her revenue in the simpler model without eclipse attacks after correcting the parameters to include both Lucy and Bob. This strategy is represented by the top-most layer of Figure 11 which we will discuss in detail further on.
- *Destroy the eclipsed victim*: Alice ignores all of Lucy's mined blocks, even if Alice has no private fork. This has the effect of destroying Lucy's computation power, and therefore increases Alice's effective gain. This is represented as the layer of strategies marked with 'Destroy Lucy (D)' in Figure 11. For example, if Alice follows selfish mining and decides to destroy Lucy, then her gain would be equal to

$$\text{gain}_D(\alpha, \lambda) = \text{selfishGain}(\frac{\alpha}{\alpha+\beta})$$

where $\text{selfishGain}(x)$ is the function defining the gains of a selfish miner with computation power x.

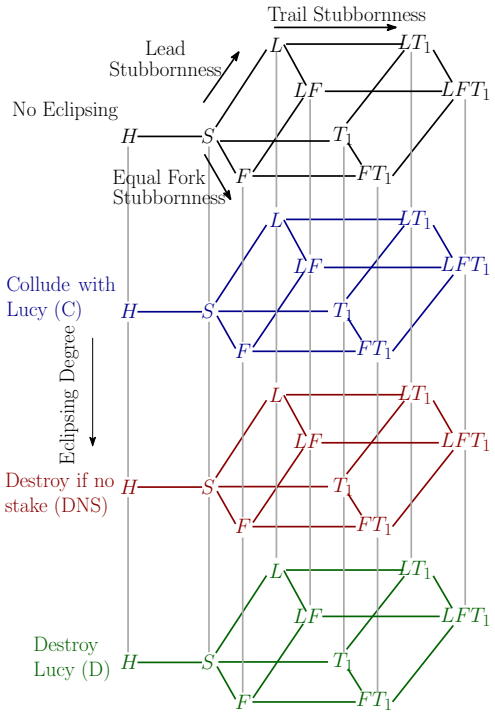


Figure 11: **Strategy Space.** The strategies we consider are derived by combining several heuristic components. Along the vertical axis are different ways for the attacker to interact with the "eclipse attack" victim (i.e., to collude with them or to "destroy" them by permanently isolating them from the network). The other three axes correspond to other forms of "stubbornness" (i.e., conditions under which the selfish miner continues to mine on an apparently losing block)

- *Collude with the eclipsed victim*: Alice can collude with Lucy and force her to cooperate as a stubborn miner. Alice and Lucy would maintain a single private blockchain, and Alice would accept all the blocks mined by Lucy. However, Alice would transmit blocks to Bob according to one of the stubborn mining strategies. This is represented by the layer marked as 'Collude with Lucy(C)' in Figure 11.
For example, if Alice follows selfish mining and she decides to collude with Lucy, then her gain would be given by:

$$\text{gain}_C(\alpha, \lambda) = \frac{\alpha}{\alpha + \lambda} \times \text{selfishGain}(\alpha + \lambda)$$

where selfishGain(x) is the function defining the gains of a selfish miner with computation power x.

Non-trivial composition. There are many possible moderate strategies that strike some compromise between Collude and Destroy. Below we describe a strategy that we find to be particularly effective called "Destroy if No Stake" (DNS).

- *Destroy if No Stake* (DNS): In the DNS strategy, assume that Alice always reveals the head of her own (private) chain to Lucy. Lucy can be mining either on Alice's chain or a private chain of Lucy's own. When Lucy mines a block, Alice will decide whether to accept it based on the following strategy: Alice accepts a block mined by Lucy only if Lucy's block builds on top of Alice's private chain. Otherwise, when Alice has no private blockchain or when Alice and Lucy are working on their separate blockchains - Alice does not accept the blocks mined by Lucy.
This strategy can be thought of as conditionally colluding with or destroying Lucy, depending on the state of Alice's and Lucy's blockchains.

Organizing the infinite strategy space. As we mentioned, in conjunction with each of the three naïve eclipsing strategies (*No Eclipse (N)*, *Collude (C)*, and *Destroy (D)*), as well as *Destroy-if-No-Stake (DNS)*, the eclipse attacker can employ any of the stubborn strategies we discussed in Section 3. We can therefore classify each strategy according to its "eclipse" behavior component and its "stubborn mining" component, as illustrated in Figure 11. The X axis corresponds to "stubbornness" as described in Figure 6 and the Y axis roughly corresponds to the amount of communication permitted between Alice and Bob.

Further variations not considered. There exist an infinite number of strategies that lie between Collude and DNS and between DNS and Destroy, and therefore we do not simulate all of them. For instance, between Collude and DNS, we can define strategies in the following way: Alice colludes with Lucy only if it has a block in the first j blocks of the private chain. If not, it ignores Lucy until it merges with the public. $j = 1$ corresponds to DNS where $j = \infty$ corresponds to Collude. As j increases from 1 to infinity, the degrees of eclipsing decreases, i.e. the strategies move closer to the collude strategy.

Since there are infinitely many such strategies, finding an analytical solution to determine the dominate strategy for any parameter choice is an interesting direction for future research.

5.3. Modeling Eclipse Mining Strategies

In order to model the revenue of the eclipse attacker under various mining strategies, we must extend our state diagram in a non-trivial way. We describe in detail the state diagram model for the *DNS-F* (i.e., Destroy-if-No-Stake and Equal-Fork Stubborn strategy), as shown in Figure 12.

In this strategy, Alice maintains a private chain as in the selfish mining strategy. As the eclipsing strategy is DNS, if Alice mines a block before Lucy does, Alice shares its private chain with Lucy and they mine together on the private chain until the chain is released to the public. However, if Lucy mines the first block, Alice decides to ignore her (destroy it) until she is ahead of Lucy again. Moreover, since the stubbornness degree is F, whenever the concerned party (Alice or Lucy, while colluding, or only Alice otherwise) finds a block in case of a tie with Bob, the block is not released and mined on privately.

The state diagram (in Figure 12) is divided into four regions:

Region 1 represents the scenario where Alice, Bob and Lucy mine on the same block.

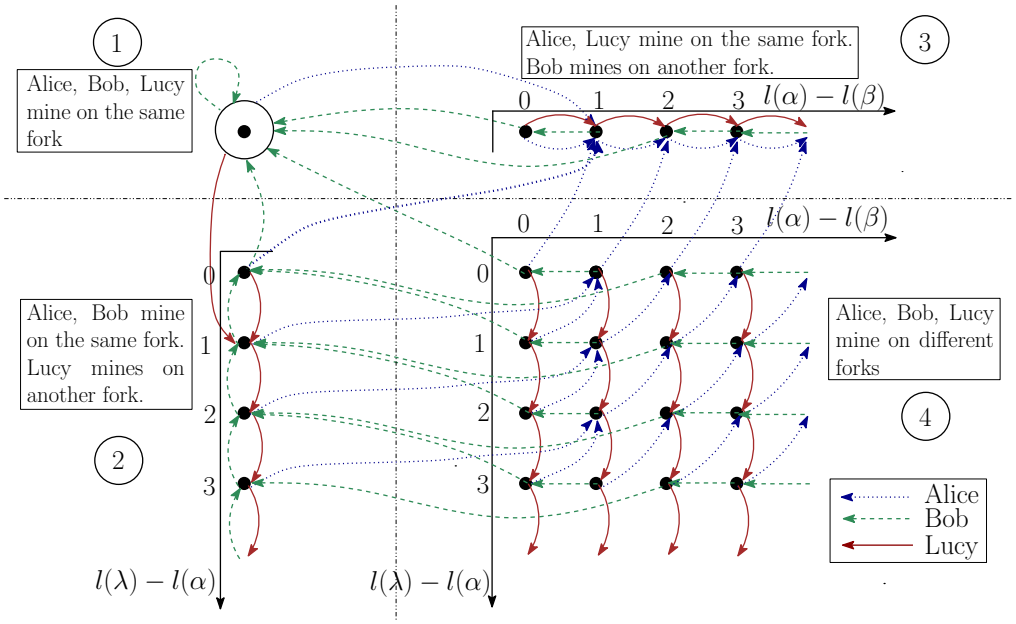


Figure 12: **The markov chain diagram of a combination of eclipsing and stubborn strategies: Destroy if No Stake (DNS) and F-stubborn strategy.** $l(\alpha), l(\beta), l(\gamma)$ represent the lengths of chains of Alice, Bob and Lucy, respectively. Dotted blue arrow indicates transitions when Alice mines a block. Solid red arrow indicates transitions when Lucy mines a block. Dashed green arrow indicates transitions when Bob mines a block.

Region 2 For all states in Region 2, Alice and Bob mine on the same block, whereas Lucy mines on a different block. In DNS, Lucy mines on a different block only if it deviated from the public chain and Alice has no stake in it; hence, Lucy essentially leads Alice and Bob. Hence, the states represent the relative lead maintained by Lucy over Alice.

Region 3 Similarly, in Region 3, Alice and Lucy mine on the same block, whereas Bob mines on a different block. This is the phase where Alice and Lucy mine together on the same chain and are maintaining a lead over the public. The states represent the lead of this private chain over the public.

Region 4 In Region 4, all three parties mine on different blocks. Similar to the reasoning mentioned earlier, for all states in this region, Alice maintains a lead over the public and Lucy maintains a lead over Alice. The states are parameterized by two values representing the two leads.

6. Eclipsed Mining Results

In this section, we discuss the insights we gain from our experiments with stubborn and eclipsing strategies discussed in Figure 11.

Result 6. No one strategy is the best at all parameter values.

Figure 13 shows the strategy that is most profitable for Alice under different parameters. We show the cases for $\gamma \in \{0, 0.2, 0.5, 0.9\}$ here.Similar to Section 4, we simulated all the strategies and report the best strategy if it is significantly more than the runner-up at a 95% confidence interval.

For extreme values of γ (near 1 or 0), naïve strategies (collude or destroy) perform best. For intermediate values of γ, the conditional DNS strategy performs best.

Result 7. Compared to naïve strategies, an eclipse attacker using our best strategies can earn significantly more revenue. For example, when $\alpha = 0.4$, Alice can achieve up to 30% more gains in comparison to naive combination of eclipsing and selfish mining strategies.

Figures 14 and 15 show how much the attacker's revenue increases when she performs her best strategy, compared to selfish mining and naïve strategies, respectively.

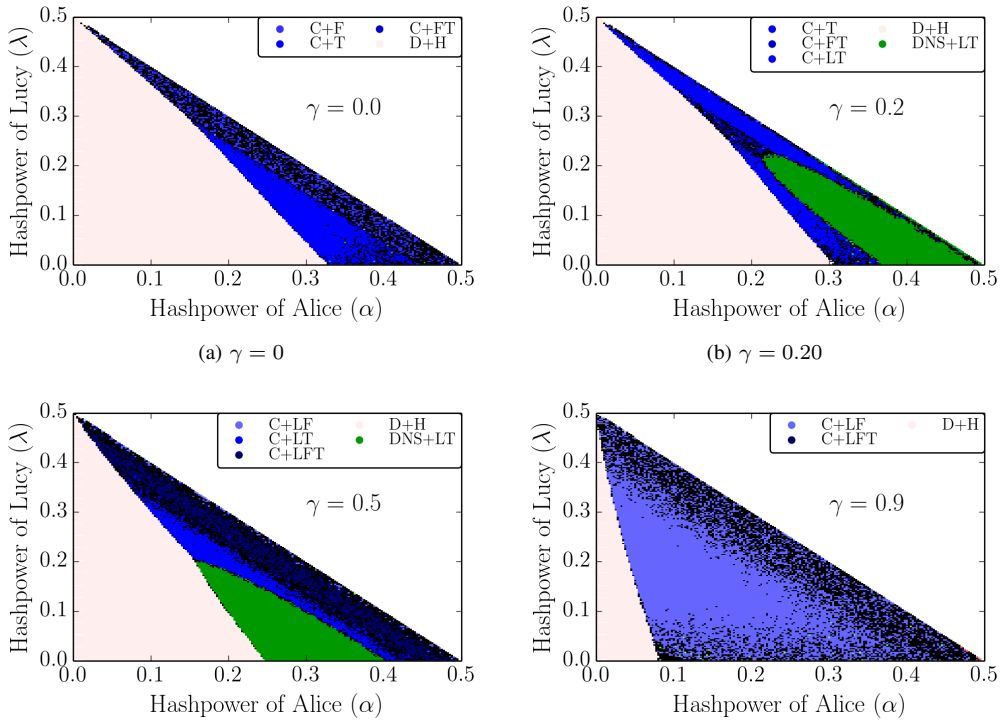
Using the combination of eclipsing and stubborn strategies, Alice can gain considerably more compared to their naive combination.

These results show that an attacker should keep in mind the full range of strategies we discuss, and should choose a strategy based on estimated network parameters in order to maximize revenue.

Result 8. Surprisingly, for some parameter values, the attacker's best strategy *benefits* the eclipse attack victim.

Let us look at Lucy and Bob's gain when Alice performs her best strategy. Their respective gains are represented in Figures 16 and 17.

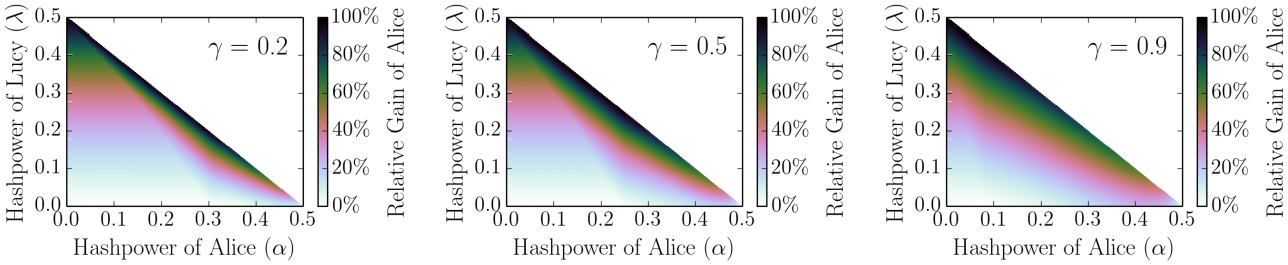
Compared to an ordinary network (one with no attacker), Lucy's revenue increases whenever Alice "colludes" with her and uses any stubborn mining strategy. Intuitively, Alice and Lucy both benefit from this collusion at Bob's expense,



(a) $\gamma = 0$ (b) $\gamma = 0.20$

(c) $\gamma = 0.50$ (d) $\gamma = 0.90$

Figure 13: **Alice's dominant strategy for each value of α and λ.** The green regions represent where non-trivial compositions of mining and eclipse attack outperform other strategies. For clarity, we only show the best strategies in the legend, and not all of them.



(a) $\gamma = 0.2$ (b) $\gamma = 0.5$ (c) $\gamma = 0.9$

Figure 14: **Alice's relative gain compared to selfish mining [9].**

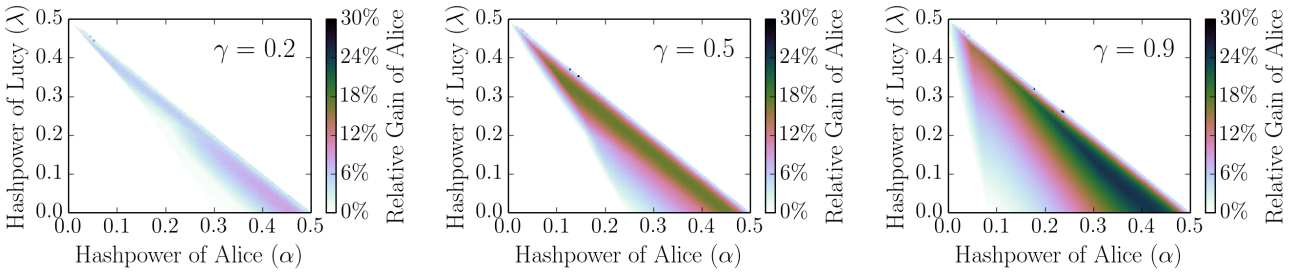
and Lucy unwittingly becomes a selfish miner. Lucy's gains can even double in some cases, when $\alpha + \lambda$ is close to 0.5.

Since the victim benefits from being eclipsed, the attacker and the victim effectively form a symbiotic relationship, where eclipsed node shares in the increased profits of the attacker. Since our empirical results suggest that "colluding" strategies are optimal for the attacker at some system parameters, this implies that there are equilibria where the eclipsed node prefers to stay eclipsed, even if the cost of defending is zero.

7. Discussion

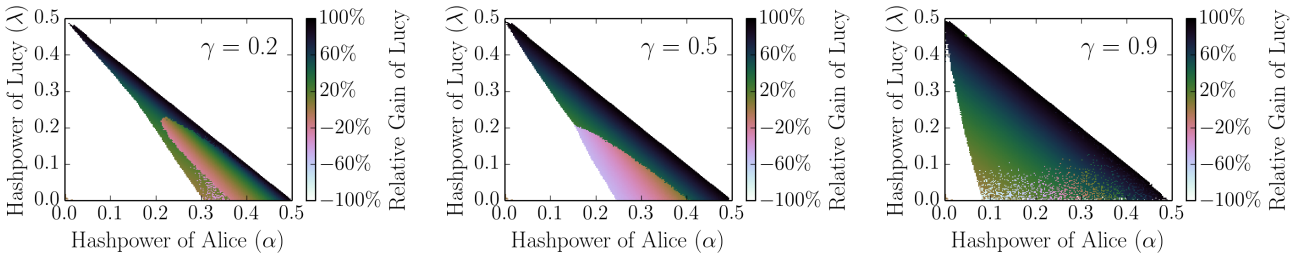
Detecting and inferring attacks. Eclipse attacks and stubborn mining can likely be detected if they occur in practice. One way is by observing the stale block rate – a stale block is one that has valid transactions and proof-of-work, but is ultimately excluded from the main chain [3].[4] Stale blocks occur by chance in ordinary operation, even when every miner is honest. When all miners are honest, the relative rate of stale blocks is the same for all miners. However, if a miner is stubborn and/or performs an eclipse attack, as we

4. Stale blocks are often (but inaccurately) referred to as "orphans" [18]



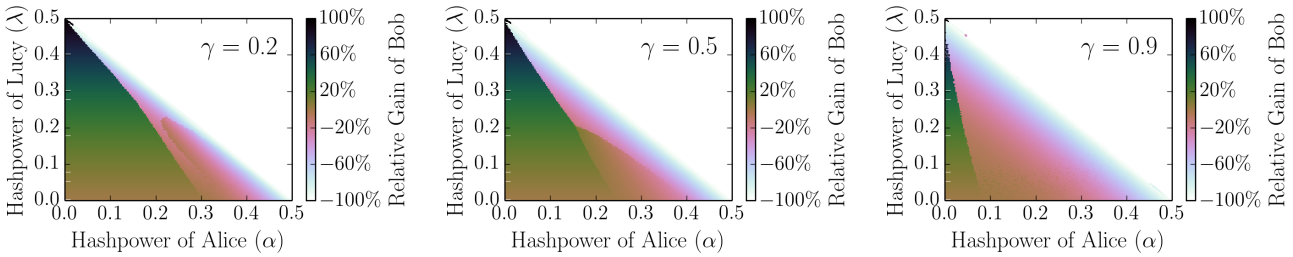
(a) $\gamma = 0.2$ (b) $\gamma = 0.5$ (c) $\gamma = 0.9$

Figure 15: **Alice's relative gain compared to to the best naïve strategies.** Naïve strategies consist of naïve compositions of eclipsing and selfish mining (see Section 5.2).



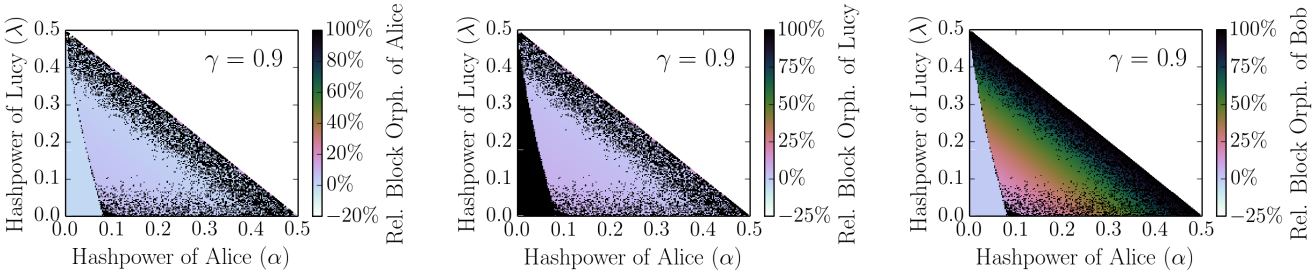
(a) $\gamma = 0.2$ (b) $\gamma = 0.5$ (c) $\gamma = 0.9$

Figure 16: **Lucy's relative gains when Alice is performing her dominant strategy.**



(a) $\gamma = 0.2$ (b) $\gamma = 0.5$ (c) $\gamma = 0.9$

Figure 17: **Bob's relative gains when Alice is performing her dominant strategy.**



(a) Alice (b) Lucy (c) Bob

Figure 18: **The stale block rate of Alice, Lucy and Bob when Alice performs her best strategy.** Stale block rate is the percentage of blocks that each one of them loses as compared to when they are mining honestly. Here $\gamma = 0.9$ in all the cases.

Rate of block orphanage	Eclipsing Component	Stubbornness Component
A = B = L	No	Honest
A < B = L	No	Any strategy but honest
A = L < B	Collude	Any strategy but honest
A = B < L	Destroy/DNS	Honest
A < B < L	Destroy/DNS	Any strategy but honest
A < L < B	DNS	Any strategy but honest

TABLE 4: Summary of inferences drawn about strategies employed by the attacker based on the stale block rates. We assume γ is high, which is practically feasible. The parties are represented by their initial characters.

show in Figure 18, the relative rate of stale blocks depends on the attacker's strategy and the network parameters. Bob can thus detect an attack by counting stale blocks and comparing with a baseline. Several public blockchain services (e.g., blockchain.info) collect and report stale blocks.

Furthermore, the relative impact on the parties involved can depend on the strategy employed by the attacker. For example, an eclipse attack victim may detect the attack comparing with the public average. We summarize these dependencies and the inferences that may be drawn in Table 4.

Are these attacks likely to occur? Why hasn't selfish mining been seen in practice? Stimulated by the selfish mining attack's publication, the Bitcoin community has deployed various services to monitor for evidence of the attack occurring. Despite (or because of!) this widespread awareness, the selfish mining attack has not been observed in practice.

Recall that selfish mining has *not* been shown to be an equilibrium strategy – in fact our models assume that the rest of the network (Bob) is compliant with the reference protocol. In reality, the members of the Bitcoin network may react and change their strategy (for example, launching "vigilante" attacks against Alice [15]) or upgrade the protocol itself [3].

Another explanation is that potential selfish miners may consider it risky to get caught. Several watchdog services (e.g., [22]) already pay close attention to mining pools. Since it takes two weeks for the Bitcoin difficulty to adjust, a stubborn mining attack must be sustained this long before any extra revenue comes in. A "brief" stubborn mining attack, or one that is detected before the difficulty adjusts, would not be profitable.

However, these explanations rely on conditions that may later change. The governance structure of the Bitcoin network is uncertain [3], so adaptive responses may be difficult to coordinate. Furthermore, the broader cryptocurrency "ecosystem" already includes hundreds of rivals using similar proof-of-work mining – these may also be vulnerable.

While our work shows that the potential profit of an attacker is greater than previously known, the likelihood of these attacks remain uncertain overall.

Should mining power be dispersed? Even using our improved strategies, honest mining is an attacker's best strategy for small values of α. This is also true for the strategy space considered by Sapirshtein et al. [23] in concurrent and independent work. For the strategy space we consider, honest mining is therefore a Nash equilibrium strategy as long as no coalition or individual entity exceeds a threshold size (depending on γ). This bolsters the motivation for mechanisms intended to discourage consolidation of mining influence, such as memory-bound proofs of work [24] and coalition-resistant puzzles [20]. Eyal [8] has recently shown that scenarios where one pool attacks another lead to an equilibrium where pools have limited size, suggesting that this property may be self-enforcing.

Eclipse attacks can benefit the victim. As it turns out, the "victim" of an eclipse attack can sometimes even *profit from the attack*, even when the attacker uses the optimal strategy (as shown in Figure 16). In such cases, Alice and Lucy effectively have a mutually beneficial relationship and share their increased revenue relative to Bob.

This implies that users in some cases may have little incentive to detect or defend against eclipse attacks. For example, suppose Lucy has an accurate belief about the attacker's hashpower and the network propagation parameter, and that Lucy's strategy space consists of choosing to invest (or not to invest) in an eclipse-resistant network connection. Then for some values of α and γ (and still assuming that Bob is unconditionally compliant), it is clear that the Lucy's dominant strategy is not to invest, and thus the unique equilibrium involves a successful colluding eclipse attack.

8. Conclusion and Future Work

We show that in decentralized cryptocurrencies such as Bitcoin, mining strategies form a complicated space, and this space can be expanded further by combining mining attacks and network-level attacks in non-trivial manners. Our work leaves open the following challenges:

1) a more complete characterization of the complex strategy space and an analytical method for deriving and proving the optimal strategy given any parameter choice; and
2) designing a *provable* secure consensus protocol whose security is formally founded on rationality assumptions rather than honest majority. By unfolding the complexity of the strategy space, our work suggests that to achieve this goal is likely challenging especially if the formal model also needs to capture realistic network-level propagation.

9. Acknowledgements

We thank the anonymous reviewers for their insightful feedback. We thank Yonatan Sompolinsky for providing us with data corresponding to results in Sapirshtein et al. [23]. This work is funded in part by NSF grants CNS-1314857, CNS-1453634, CNS-1518765, CNS-1514261, a Packard Fellowship, a Sloan Fellowship, two Google Faculty Research Awards, and a VMWare Research Award.

This work was done in part while a subset of the authors were visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/SimonsCollaboration in Cryptography through NSF grant CNS-1523467. Parts of this work is supported by Maryland Procurement Office contract H98230-14-C-0137, ARO grants W911NF11103, W911NF1410358, and W911NF09102.

References

[1] The rise and rise of bitcoin. Documentary, http://bitcoindoc.com/.

[2] A. Biryukov and I. Pustogarov. Bitcoin over tor isn't a good idea. *arXiv preprint arXiv:1410.6079*, 2014.

[3] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies (Extended Version). Cryptology ePrint Archive, Report 2015/452, 2015.

[4] M. Corallo. High-speed bitcoin relay network. http://sourceforge.net/p/bitcoin/mailman/message/31604935/, November 2013.

[5] N. T. Courtois and L. Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014.

[6] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P*, 2013.

[7] S. G. Ethan Heilman. Alison Kendler, Aviv Zohar. Eclipse attacks on bitcoins peer-to-peer network. Cryptology ePrint Archive, Report 2015/263, 2015.

[8] I. Eyal. The Miner's Dilemma. In *IEEE Symposium on Security and Privacy*, 2015.

[9] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2014.

[10] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Eurocrypt*, 2015.

[11] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun. Is bitcoin a decentralized currency? *IEEE Security & Privacy*, 12(3):54–60, 2014.

[12] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun. Tampering with the delivery of blocks and transactions in bitcoin. Cryptology ePrint Archive, Report 2015/578, 2015. http://eprint.iacr.org/.

[13] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. 2015.

[14] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[15] Littleshop. A user ghashthrow has announced that they are selfish mining at ghash.io. https://bitcointalk.org/index.php?topic=651451.0, June 2014.

[16] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor. On power splitting games in distributed computation: The case of bitcoin pooled mining. Technical report, Cryptology ePrint Archive, Report 2015/155, 2015, http://eprint. iacr. org, 2015.

[17] J. Matonis. The bitcoin mining arms race: Ghash.io and the 51% issue. http://www.coindesk.com/bitcoin-mining-detente-ghash-io-51-issue/, 2014.

[18] A. Miller and R. Jansen. Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications.

[19] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. Discovering bitcoins public topology and influential nodes.

[20] A. Miller, E. Shi, A. Kosba, and J. Katz. Nonoutsourceable Scratch-Off Puzzles to Discourage Bitcoin Mining Coalitions (preprint), 2014.

[21] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http://bitcoin.org/bitcoin.pdf, 2008.

[22] organofcorti. Neighborhood Pool Watch. http://organofcorti.blogspot.com/.

[23] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. *arXiv preprint arXiv:1507.06183*, 2015.

[24] J. Tromp. Cuckoo Cycle: a memory-hard proof-of-work system. In *Workshop on Bitcoin Research*, 2015.