When cryptocurrencies mine their own business[⋆]

Jason Teutsch, Sanjay Jain, and Prateek Saxena

School of Computing
National University of Singapore
Singapore 117543
teutsch@comp.nus.edu.sg
sanjay@comp.nus.edu.sg
prateeks@comp.nus.edu.sg

Abstract. Bitcoin and hundreds of other cryptocurrencies employ a consensus protocol called Nakamoto consensus which rewards miners for maintaining a public blockchain. In this paper, we study the security of this protocol with respect to rational miners and show how a minority of the computation power can incentivize the rest of the network to accept a blockchain of the minority's choice. By deviating from the mining protocol, a mining pool which controls at least 38.2% of the network's total computational power can, with modest financial capacity, gain mining advantage over honest mining. Such an attack creates a longer valid blockchain by forking the honest blockchain, and the attacker's blockchain need not disrupt any "legitimate" non-mining transactions present on the honest blockchain. By subverting the consensus protocol, the attacking pool can double-spend money or simply create a blockchain that pays mining rewards to the attacker's pool. We show that our attacks are easy to encode in any Nakamoto-consensus-based cryptocurrency which supports a scripting language that is sufficiently expressive to encode its own mining puzzles.

1 Introduction

Hundreds of cryptocurrencies are in use today, and investments in cryptocurrencies continue to increase steadily [1]. Some cryptocurrencies, such as Bitcoin and Ethereum, aim to serve as underlying substrates for financial applications beyond simple distributed ledgers and payment services. Nearly all cryptocurrencies share a protocol known as the Nakamoto consensus protocol as their backbone. The security of the Nakamoto consensus protocol has recently been rigorously analyzed, under the assumption that a majority of the miners follow the protocol honestly [11]. Does this backbone remain secure when miners purely try to maximize their financial payoffs? In this paper, we study this question from the lens of cryptocurrencies which permit expressive transaction semantics.

[⋆] J. Teutsch and P. Saxena's research is supported by Singapore Ministry of Education Grant No. R-252-000-560-112. S. Jain is supported in part by NUS grant Nos. R252-000-534-112, R146-000-181-112 and C252-000-087-001.

Cryptocurrencies often allow applications and users to encode semantic operations in blockchain transactions. For example, Bitcoin and Ethereum both permit transaction scripts which allow users to specify conditions, or contracts, which corresponding transactions must satisfy prior to acceptance. Transaction scripts can encode many useful functions, such as validating that a payer owns a coin he is spending or enforcing rules for multi-party transactions.

Scriptable cryptocurrencies allow clients to outsource computational tasks or puzzles [13]. For instance, assuming a sufficiently expressive scripting language, a client might post a computational puzzle and a transaction that together contractually commit him to pay prize money to the first party that conveys a correct solution. Let us call a computational puzzle encoded via a cryptocurrency transaction script a *script puzzle*. Script puzzles are one way for clients to trade computation power directly with coins. We now ask: what are the security implications of allowing script puzzles in a cryptocurrency? When analyzing this question, we will assume that miners in such cryptocurrencies always try to optimize their expected financial gains. This assumption differs from the folklore assumption that the majority of miners always honestly follow the mining protocol, irrespective of whether it gives them a financial advantage. If we assume that miners are rational instead, then is Nakamoto-consesus based crytocurrency secure against a minority of miners deviating from the protocol for financial gain? Given the financial investments in cryptocurrencies today, it is reasonable to ask whether the core protocol is open to manipulation by rational minorities.

In particular, this paper investigates the consequences of casting proof-of-work mining problems as script puzzles. Miners often have dedicated hardware for solving proof-of-work problems, and so they can easily reuse their hardware for solving such script puzzles. In short, miners engage in a game which incentivizes them to split their computation resources between solving script puzzles and mining. We analyze this game and find that, even when the language does not support script puzzles, an attacker with 38.2% of the computation capacity can not only subvert the consensus protocol — effectively reducing the classic 51% attack to a 38.2% attack — but can also increase his share of mining reward per block without double-spending. The expected minimum financial capacity for such an attacker, beyond its cost invested in mining power, is less than a dozen times the reward for mining a new block when the attacker's power exceeds 39.1% of the network. Several mining pools have enjoyed such large share of computation power in the Bitcoin network recently.

2 Background

Bitcoin and several other cryptocurrencies use similar mechanism for maintaining consensus about a distributed ledger of transactions. The distributed ledger is maintained as a hash-chain of transaction blocks known as the *blockchain* ledger. The consensus protocol used in nearly all blockchain-based cryptocurrencies, known as *Nakamoto consensus*, achieves eventual consistency assuming an honest majority[1]. *Miners* in a cryptocurrency network solve “mining puzzles” — a cryptographic proof-of-work puzzle (e.g. “inverting” a SHA2 hash [10]) — in exchange for cryptocurrency mining rewards. When a miner successfully broadcasts the solution of a proof-of-work puzzle, he proves that he has spent the necessary computation power to merit appending his new set of transactions to the distributed ledger. This step awards the miner a set of newly minted coins. Miners solve the next proof-of-work puzzle using the longest blockchain, which implicitly embodies the majority of the network’s computational effort with overwhelming probability.

In a blockchain-based cryptocurrency, the cryptographic hash-chain guarantees integrity of accepted transactions. Thus anyone can query the blockchain for the presence of a transaction. Miners on the network race to extend the blockchain in each time epoch (e.g. 10 minutes in the Bitcoin protocol), and this race effectively generates randomized lottery to elect a leader in each epoch.

2.1 The 51% attack

In principle, the blockchain mechanism only ensures consensus with overwhelming probability. A transaction that appears in the longest, current blockchain can be omitted in a future blockchain which is longer — however, creating such a “forked” blockchain that omits a transaction requires an adversary with computation power larger than half the computation power of the entire network. The backtracking mechanism which permits this kind of double-spending is known as a *51% attack* [16]. Satoshi Nakamoto [2] was aware of this attack when he introduced Bitcoin in 2009. His proof-of-work mechanism ensures correct blockchain consensus under the assumption that the majority of miners are honest. If, for example, the powerful miner spent some money in another fork of the blockchain at the time when that fork appeared to be longest, he would not necessarily have to carry that transaction over to his own fork and could thereby double-spend the money.

2.2 Smart contracts

Bitcoin and several cryptocurrencies allow transactions to specify conditions as scripts. A transaction is deemed valid only if the linked condition holds. Bitcoin’s scripts have limited expressiveness, however emerging cryptocurrencies support expressive scripts that enable development of a variety of powerful decentralized applications. The Ethereum cryptocurrency [9] introduced *smart contracts* in which versatile scripts specify whether or not the network should accept given transactions. Ethereum is *Turing-complete* in the sense that one can encode any algorithm in its scripting language.

One can encode a smart contract which pays a reward for solution to a hash inversion puzzle in many different ways. Figure 1 provides an example of how one might realize such a contract in Ethereum. The protocol in this figure permits

[1] It further assumes a favorable broadcast network between miners.

```
init:
    #Record the initiator, reward and data
    contract.storage[0]=msg.sender
    contract.storage[1]=0
    contract.storage[2]=msg.value
    #record the difficulty of the puzzle
    contract.storage[3]=msg.data[0]
code:
    #if the puzzle is solved
    if contract.storage[4] == 0:
        return (1)
    if msg.datasize == 1:
        #if everything is fine, send the reward
        if sha256(sha256(msg.data[0])) < contract.storage[3]:
            send(1000,msg.sender,contract.storage[2])
            contract.storage[4]=0 #update status
            contract.storage[5]=msg.data[0] #store result
            return (2)
        else:
            return (1)
```

Fig. 1. Code snippet of a contract which asks users to solve a hashing puzzle. The contract will verify the correctness of the result before sending out the reward.

a puzzle giver to post a blockchain transaction containing a public puzzle such that any prover who notices the puzzle can post a solution on the blockchain. If the miners on the network deem the prover's solution to be correct, the prover may receive the puzzle giver's advertised reward in exchange for the solution.

Luu, Teutsch, Kulkarni, and Saxena [13, Section 5.2] showed how one can modify a puzzle script, such as the one in Figure 1, so as to achieve fairness via a commitment scheme. Their protocol rewards the first solution posted and resolves ties as follows. Every potential prover posts a hash of his solution to the puzzle so as both to notify the network that his solution is ready and to prevent him from later changing his answer. The first solutions to appear effectively enter a lottery. Once the network has confirmed the winner of this lottery, the prover publicly reveals his solution. If the network finds that his solution is correct, then he receives the puzzle's reward for his solution. If the solution is incorrect, another prover's solution may be considered.

Any transaction puzzle with suitable computational complexity and monetary reward which incentivizes miners to repurpose their hardware suffices for the attacks in this paper. Ethereum miners use GPU hardware, so in practice one would need to fine tune the puzzle in Figure 1 into a GPU-friendly form in order to achieve the desired effect.

2.3 Assumptions

Our theoretical attack will suppose the following specific conditions about miners' behavior, the attacker's capability, and the underlying cryptocurrency.

1. *Miners always mine on the longest chain.* This is a fundamental and standard assumption for Nakamoto consensus.

2. *Expected time between new blocks is constant.* The network calibrates the difficulty of mining blocks so as to maintain a fixed expected time between new blocks. Both the Ethereum and Bitcoin protocols include periodic adjustments based on the cumulative computational power of active miners.

3. *Miners are rational.* Non-attacking miners on the network distribute their computational resources between mining and puzzle solving in order to maximize expected profits. They do not withhold blocks.

4. *Attacker has limited resources.* The attacker has a fixed amount of capital and computational power at his disposal with which to execute his attack. The graphs at the end of this paper show sufficient capital and computational power as well as the trade-offs between these two parameters.

We will elaborate on and further discuss the details of these assumptions in subsequent sections. Empirical testing may shed light on the practical validity of the above assumptions and remains as valuable future work. Finally, for ease of presentation, our analysis will focus on expected values rather than probabilities for various possible outcomes.

3 The double-spend attack

An attacker with sufficient capital, regardless of computational power can double-spend in any cryptocurrency with sufficiently expressive scripting language via a modification of the backtracking 51% attack discussed in Section 2. We expect that block-sized hash inversion script puzzles can distribute rewards fairly because the time required to verify the correctness of a solution is small [13].[2]

Attack 1. Let M be a miner with p fraction of the network's computation power[3]. Let b be the fixed reward for mining a new block. Starting from the current block, M begins, using his full power p, privately mining new blocks on a fork which is unknown to other miners. Meanwhile the following is repeated until M's private blockchain is longer than the public one:

> Once per (public) block, M posts a transaction with a hash-inversion puzzle[4] whose solution requires exactly the same amount of work as

[2] Since the time required to solve these script puzzles is modest, Ethereum's `gaslimit` function does not hinder their execution.

[3] In the following discussions, we assume that the total number of processors on the network is fixed.

[4] The puzzle M chooses may be identical to the nonce he needs to solve in order to extend his private blockchain, and this choice may help M to mine faster on his private chain. We do not attempt to quantify the advantage of implementing this strategy, however, as latency from network broadcasts and puzzle reward commitment schemes make the benefit difficult to estimate.

mining a new block[5]. M offers a prize for its solution exceeding

$$\frac{1-2p+\epsilon}{p-\epsilon} \cdot b \tag{3.1}$$

for some fixed value $0 < \epsilon < p$.

Since M is free to add or not add transactions from the public blockchain into his private blockchain, he may double-spend when the loop finally terminates by revealing his private blockchain.

Each time the miner M from Attack 1 posts a puzzle transaction, the other processors on the network have two options: work on the transaction puzzle or try to mine a new block. Each processor will work on the puzzle with some probability a and hence will mine with probability $1-a$.[6] We can view this process as a game in which each processor tries to select the value a which optimizes his expected profits. The processors' set of strategies for choosing a are said to form a *Nash equilibrium* when no individual processor has financial incentive to deviate from his current strategy given that the other strategies are fixed. By definition, *rational* processors choose the value a which satisfies the Nash equilibrium.

Lemma 2. *Let $t > 0$ and let $0 < a < 1$. If the miner in Attack 1 offers a reward of $a \cdot t$ to solve his puzzle and the reward for mining a new block is $(1-a)t$, then rational processors on the network will puzzle-solve with probability a and mine with probability $1-a$.*

Proof. Suppose a given processor R has $0 < q < 1$ fraction of the total computational power of the network. When every processor on the network works on the puzzle with probability a and mines with probability $1-a$, R's expected gain is the sum of his expected fraction of total work on the puzzle times the puzzle prize plus his expected fraction of total work on mining times the mining prize:

$$\frac{aq}{aq + a(1-q)} \cdot at + \frac{(1-a)q}{(1-a)q + (1-a)(1-q)} \cdot (1-a)t = qt.$$

By symmetry, no processor can expect to obtain more than his share of the reward, so an expected reward of qt is optimal on a purely rational network. Suppose that R were to deviate from this strategy by puzzle-solving with probability $a + \delta$ where $0 < \delta < 1 - a$. Then his expected reward would be

$$\frac{(a+\delta)q}{(a+\delta)q + a(1-q)} \cdot at + \frac{(1-a-\delta)q}{(1-a-\delta)q + (1-a)(1-q)} \cdot (1-a)t$$
$$= \frac{aq + \delta q}{a + \delta q} \cdot at + \frac{q - aq - \delta q}{1 - a - \delta q} \cdot (1-a)t = \frac{a - a^2 - 2\delta aq + \delta q - \delta^2 q}{(a+\delta q)(1-a-\delta q)} \cdot qt, \tag{3.2}$$

[5] For simplicity of calculation, we assume that the hardness of the puzzle that M posts in a given block is equally hard compared to the mining problem in the current block.

[6] For the purposes of our calculations, it is equivalent to assume that the miner devotes a fraction of his computational resources to puzzle solving and $1-a$ fraction to mining.

which is less than qt whenever $0 < \delta q < 1 - a$, and in particular this inequality holds for our choice of $\delta < 1 - a$. Indeed for such δ, the denominator of the rightmost fraction of (3.2) is positive and exceeds its numerator by $\delta^2 q - \delta^2 q^2$. By symmetry, R gains no advantage by biasing himself towards mining, either. Hence the given strategy yields a Nash equilibrium. □

We now show that Attack 1 succeeds when the other processors on the network are rational.

Theorem 3. *If a miner M has sufficient initial capital, possesses p fraction of the network's computing power and the other processors on the network are rational, then the loop in Attack 1 eventually terminates. Hence M can double-spend any money spent since the beginning of the attack.*

Proof. Let $x = (1-2p+\epsilon)/(p-\epsilon)$ as in (3.1) so that the reward for solving M's puzzle is xb. Then the sum of the rewards between mining in a given block and solving the puzzle is $(1+x)b$. Thus the fraction of reward devoted to mining is at most:

$$\frac{b}{(1+x)b} = \frac{1}{1+\frac{1-2p+\epsilon}{p-\epsilon}} = \frac{p-\epsilon}{p-\epsilon+1-2p+\epsilon} = \frac{p-\epsilon}{1-p}.$$

It follows that a rational processor on the network will mine with probability less than $(p-\epsilon)/(1-p)$ and puzzle-solve with probability greater than $1-[(p-\epsilon)/(1-p)]$ as the Nash equilibrium is achieved with these parameters (Lemma 2). Since at most $(1-p)\cdot(p-\epsilon)/(1-p) = p-\epsilon$ fraction of the network power is devoted to extending the public blockchain and p fraction of the network power is devoted to extending M's private chain, M's private chain will eventually become longer than the public one. □

4 Mining advantage without double-spending

We now observe that if a miner controls more than $(3-\sqrt{5})/2 \approx 38.2\%$ of the network's computational power, then he can execute Attack 1 without double-spending and still achieve an overall mining advantage. In the short run, such a miner obtains a per-block advantage over honest mining which, as we discuss at the end of Section 5, translates into a gain per unit time after consecutive repetitions of the attack. Unlike our double-spend attack, in this scenario the attacker can offer his puzzle rewards on an external website or system known to the cryptocurrency miners; the puzzles need not be posted as transactions within the cryptocurrency itself. Thus our analysis establishes the insecurity of Nakamoto consensus against a rational-but-dishonest minority of miners without assuming any scriptability properties for transactions.

Theorem 4. *A miner M with p fraction of the network's power where $(3-\sqrt{5})/2 < p \leq 1/2$ who executes Attack 1 with appropriately chosen $\epsilon > 0$ so that the reward for each of his puzzles, xb, satisfies the additional constraint*

$$x < 1 - p \tag{4.1}$$

expects to gain a mining reward advantage of at least $[1-(x+p)]b$ *per block when other processors on the network are rational. Thus* M *can benefit from the Attack 1 without double-spending or otherwise manipulating public transactions in his private blockchain.*

Proof. Since M's reward satisfies the lower bound from Attack 1, the argument in Theorem 3 shows that the length of M's private blockchain will eventually exceed the length of the public one. In particular, we can assume that by the end of the attack M's private blockchain has at least as many blocks in it as the public blockchain. Note that the reward that M would have expected to receive from mining a block without posting the puzzle transaction is pb, and the reward he receives per block on his private blockchain is b, so the attack is only profitable if the cost of each puzzle is less than $b-pb$, that is, when $x < 1-p$. Since M can win all the mining rewards from the network when $p > 1/2$ using the 51% attack, the miner receives no additional advantage in posting puzzle transactions when $p > 1/2$.

Finally, let us consider when an x satisfying both (3.1) and (4.1) actually exists. For $\epsilon < p$, this happens when:[7]

$$\frac{1-2p+\epsilon}{p-\epsilon} < 1-p,$$

or equivalently $1-2p+\epsilon < p-\epsilon-p^2+\epsilon p$, which simplifies to

$$p^2-(3-\epsilon)p+(1+2\epsilon) < 0. \tag{4.2}$$

Applying the quadratic formula, using the fact that the leading coefficient in the left-hand side of (4.2) is positive, and using that the larger root of this expression is greater that $1/2$, we find that $p \leq 1/2$ is a solution to the inequality (4.2) if and only if

$$p > \frac{(3-\epsilon)-\sqrt{(9-6\epsilon+\epsilon^2)-4-8\epsilon}}{2} = \frac{3-\epsilon-\sqrt{5+\epsilon^2-14\epsilon}}{2}.$$

Taking the limit of this expression as $\epsilon \to 0$, we obtain $p > (3-\sqrt{5})/2$, which means the advantage exists for any such p whenever the attacker offers a reward in (3.1) with sufficiently small parameter $\epsilon > 0$. □

Between 38.2% and 50% the attacker gains an increasing mining advantage from executing the simple attack in Theorem 4. At 51% power, the miner need not award any prize for solving his puzzle because he can outright win all the mining rewards by extending his private blockchain quickly.

[7] A slightly weaker inequality holds here. At the end of Attack 1, the attacker's private chain is a block longer than the public chain, and so the attacker's expected net gain per block actually exceeds $(1-p)\cdot b$ by some positive quantity, namely $[(1-p)\epsilon/(p-\epsilon)]\cdot b$, which tends to zero as $\epsilon \to 0$. In this argument we ultimately care only about what happens as ϵ approaches 0, and so for now we ignore this quantity. We revisit the present calculation in more detail in Lemma 7.

Can one repeat Attack 1 with double-spending more than once? Miners who lose their rewards from solving puzzles may find themselves once bitten, twice shy. Unlike the double-spend attack described in Theorem 3, the *38.2% attack* in Theorem 4 permits miners to keep their rewards for solving puzzles. Thus the deterrent of "once bitten, twice shy" disappears in the latter form of this attack.

Theorem 4 highlights a tragedy of the commons for rational miners. By working on the puzzle posed in Attack 1, miners place the integrity of the network at risk, yet none of them are individually motivated to switch back to mining. The assumption that miners optimize their personal profit ensures that they work on puzzles even while "honest" miners lose all of their mining rewards.

5 How much does it cost?

Both Theorem 3 and Theorem 4 assume that the perpetrator M has "sufficient initial capital" to successfully generate the long private fork described in Attack 1. We now estimate how much initial capital is "sufficient." Following the notation of Attack 1, let p denote the fraction of the network's computational power that belongs to the attacker, let ϵ represent the difference in mining effort during the attack between the attacker and the rest of the network, and let b be the prize offered by the cryptocurrency for mining a new block.

Lemma 5. *A miner who executes Attack 1 using*

- *p fraction of the network's computing power,*
- *with $0 < \epsilon < p$ difference in mining effort between the attacker and the rest of the network, and*
- *with prize b for mining a new block*

expects to spend

$$\frac{1 - 2p + \epsilon}{\epsilon} \cdot bk \tag{5.1}$$

on mining puzzles before his private blockchain becomes k blocks longer than the public one when other processors on the network are rational.

Proof. Let $t(p, \epsilon, k)$ denote the expected number of public blocks which are mined before the length of M's private blockchain exceeds the public blockchain's length by at least k blocks, and let $c(p, \epsilon, b)$ denote the cost per block of the puzzle reward given in (3.1). The expected initial capital required to execute Attack 1 for given parameters p, ϵ, k, and b is then $t(p, \epsilon, k) \cdot c(p, \epsilon, b)$.

Let us assume that the network generates on average one new block per unit time. Then after s units of time, the attacker expects to have generated ps blocks on his private chain, and the rest of the network expects to have generated $(p - \epsilon)s$ blocks on the public chain (see the proof of Theorem 3 for calculation of these estimates). We are interested in the time s at which the difference between these blockchain lengths reaches k, that is $k = ps - (p - \epsilon)s$, or equivalently when $s = k/\epsilon$. Thus

$$t(p, \epsilon, k) = s \cdot (p - \epsilon) = \frac{p - \epsilon}{\epsilon} \cdot k.$$

For a fixed ϵ, we may now estimate the total expected cost of the attack as

$$t(p, \epsilon, k) \cdot c(p, \epsilon, b) = \frac{p - \epsilon}{\epsilon} \cdot \frac{1 - 2p + \epsilon}{p - \epsilon} \cdot bk$$

when $0 < \epsilon < p$, as required in Attack 1 in order to ensure the puzzle prize value is positive. The above quantity simplifies to (5.1). □

We now compute the ϵ which minimizes the cost of a successful double-spending attack.

Theorem 6. *A miner with p fraction of the network can double-spend using Attack 1 with approximately $b/p - b$ initial capital, where b is the reward for mining a new block, when other processors on the network are rational.*

Proof. The attack cost in (5.1) is minimized when ϵ is as large as possible, that is, as ϵ approaches p. Thus the expected cost of the double-spend attack in Theorem 3 can be made arbitrarily close to $(1/p - 1) \cdot bk$. In the notation of Lemma 5, we may assume $k = 1$ because the attacker's private chain need only be longer than the public chain momentarily in order for the double-spend attack to succeed.[8] □

If an attacker were to double-spend *only* the puzzle prize money, then he would gain approximately $(1 - p)b$ more capital per block from executing the double-spend attack compared to honest mining.

Lemma 7. *A miner M with p fraction of the network's power where $(3 - \sqrt{5})/2 < p < 1/2$ who successfully executes a 38.2% attack (that is, Attack 1 without double-spending), using $0 < \epsilon < p$ difference in mining effort from the rest of the network, expects to gain per (private blockchain) block of the attack*

$$\frac{3p - p^2 - 1 - \epsilon}{p} \cdot b \tag{5.2}$$

more than he would from honest mining, where b is the reward for mining a new block, when other processors on the network are rational.

Proof. Applying Lemma 5 with $k = 1$, we obtain that M's total expected expenditures on puzzle prizes during the attack is $[(1 - 2p + \epsilon)/\epsilon] \cdot b$, and he earns $(p/\epsilon) \cdot b$ from his length p/ϵ extension of the private blockchain when it becomes public. His expected reward for mining honestly over these p/ϵ blocks would have been $p \cdot (p/\epsilon) \cdot b$, and therefore his expected net gain over honest mining throughout the course of the attack is

$$\left[(1 - p) \cdot \frac{p}{\epsilon} - \frac{1 - 2p + \epsilon}{\epsilon}\right] \cdot b = \frac{3p - p^2 - 1 - \epsilon}{\epsilon} \cdot b.$$

[8] Since many Bitcoin users do not consider a transaction confirmed until the transaction is at least 6 places deep in the blockchain, one might wish to wait until the private chain extension is at least 6 blocks long before revealing it. This can be done be choosing an ϵ satisfying, in the notation of Lemma 5, $t(p, \epsilon, 1) \geq 6 \cdot (p - \epsilon)/p$, or equivalently $\epsilon \leq p/6$.

It follows that M's per block expected gain over honest mining is

$$\frac{3p - p^2 - 1 - \epsilon}{\epsilon} \cdot b \div \frac{p}{\epsilon},$$

or equivalently, the quantity in (5.2). □

The per block net gain over mining that one can expect to achieve with a 38.2% attack depends on how much initial capital one has available. In order to make the expected gain per block in (5.2) positive, ϵ must be less than $3p - p^2 - 1$, and substituting this value for ϵ into the estimated puzzle prize total puzzle (5.1), we obtain the expected break-even cost given in (5.3) below.

Theorem 8. *Let b be the block mining reward. A miner with p fraction of the network's power where $(3 - \sqrt{5})/2 < p < 1/2$ expects to spend in total*

$$\frac{p - p^2}{3p - p^2 - 1} \cdot b \tag{5.3}$$

on puzzle prizes during a 38.2% attack (without double-spending) in order to break even on the rewards he would have earned from honest mining over the same number of blocks. Moreover, if the miner

- *chooses a target puzzle prize budget for Attack 1 of xb (by setting $\epsilon = (1 - 2p)/(x - 1)$), and*
- *xb exceeds the quantity in (5.3),*

then his expected net gain per block (of private mining[9]) over honestly mining will be:

$$\frac{3p - p^2 - 1 - \frac{1-2p}{x-1}}{p} \cdot b \tag{5.4}$$

when other processors on the network are rational.

Proof. Suppose $(3 - \sqrt{5})/2 < p < 1/2$, $x > (p - p^2)/(3p - p^2 - 1)$ and let $\epsilon = (1 - 2p)/(x - 1)$. We wish to show $0 < \epsilon < p$ so that we may apply Lemma 7. Now

$$\epsilon < \frac{1 - 2p}{\frac{p - p^2}{3p - p^2 - 1} - 1} = \frac{(1 - 2p)(3p - p^2 - 1)}{1 - 2p} = 3p - p^2 - 1 \leq p.$$

as $p < 1/2$.

Since $p < 1/2$, we have $1 - 2p > 0$, which implies $p - p^2 > 3p - p^2 - 1$, and therefore $x > (p - p^2)/(3p - p^2 - 1) > 1$ as both the numerator and denominator of this fraction are positive whenever $(3 - \sqrt{5})/2 < p \leq 1/2$. Since ϵ is the quotient of two positive reals, we have $\epsilon > 0$.

By Lemma 7, the attacker gains per block $[(3p - p^2 - 1 - \epsilon)/p] \cdot b$, which immediately establishes (5.4) as his total net gain over honest mining through the entire course of the attack. Note that by setting (5.4) equal to 0 and solving for x, we obtain the break-even point (5.3) modulo a factor of b. □

9 In the long run, the private blockchain becomes the main chain.

Since most cryptocurrencies periodically adjust the hardness of their mining problems so that the expected number of blocks mined per unit time remains constant, we can expect that if one repeats the 38.2% attack several times in row, then the expected gain per block eventually becomes equal to the expected gain per unit clock time for mining one block. Thus, in the long run, one can use the same figures to estimate the profit of performing a 38.2% attack relative to honest mining over clock time as well. Since the attacker's private blockchain becomes the main blockchain in the long run, clock time per block eventually corresponds to per block time on the private chain.

At the market rate in September 2015 of approximately $b =$ \$6000 reward per block in Bitcoin, Theorems 6 and 8 show that a miner with 40% of the network power can execute a 38.2% attack for \$36,000. This cost is significantly lower than the amount one would have to spend to purchase the hardware needed to execute a 51% attack, and moreover the attacker gets his initial capital back after a successful attack. See the figures at the end of this paper for other cost and return estimates.

6 Rationality and Nakamoto consensus

When analyzing the ultimate success of Attack 1, we tacitly assumed that miners always mine on the longest available blockchain. Will rational miners actually choose this strategy? We cannot answer this question decisively since many distinct sets of mining strategies achieve Nash equilibrium. The collective strategy which says "everyone mine on the longest chain" is a Nash equilibrium, but so is the collective strategy which says "everyone mine on the same chain as Fred Flintstone." Under the usual convention that the majority of computing power selects valid transactions and blocks, the optimal strategy for an individual miner is always to mine on the chain where most processors are mining.

While one can imagine individuals deciding to mine on chains which are adversarial against Attack 1 or chains which maximize personal capital, miners do have compelling reasons to choose the canonical honest strategy of mining on the longest chain. As an official default strategy, we may view the situation where everyone mines on the longest chain as a *de facto* initial condition. As this initial condition happens to both be a Nash equilibrium and satisfies the desired objective that individuals follow the majority, miners will persist in mining on the longest chain unless some particular influence drives them to change their behavior. Thus, under reasonable assumptions, rational miners will mine on the longest chain.

As established in Section 4, puzzle prizes in a 38.2% attack need not be encoded into transactions but rather may appear anywhere in the network ecosystem. This means that the 38.2% attack is not strike against expressiveness of scripting languages but rather Nakamoto consensus itself. Every cryptocurrency based on Nakamoto consensus, including Bitcoin, is vulnerable to a 38.2% attack. Consequently if miners are rational rather than majority-honest, then Bitcoin may be not only insecure when an adversary controls more than half of the network's computation power, as Nakamoto pointed out in his original paper [14], but even when the adversary controls merely 38.2% of the network's power. Can we achieve secure consensus beyond 38.2% under the assumption of rational miners?

7 Related work

Cryptocurrency incentive structure flaws, such as the ones we pointed out in Ethereum and Bitcoin, can be difficult to detect. Other Bitcoin attacks which also rely on misplaced incentives include selfish mining [8] and mining pool block withholding [7,12]. We outline these attacks below. Although all three of these types of attack appeal to parties' rational behavior, the execution mechanisms vary considerably.

Unlike the 38.2% attack, neither selfish mining nor mining pool block withholding attacks use puzzle transactions. The 38.2% attack creates a distraction whereas the other two attacks focus purely on withholding blocks. Consequently, the 38.2% attack motivates other miners to deviate from protocol, while only the attacker deviates in the other two attacks. In other words, the victims miners in a 38.2% attack are rational rather than honest. In contrast to selfish mining, in which the attacker obtains in the short run only a relative per block advantage, the successful attacker in a 38.2% attack achieves an absolute per block advantage. Here "relative" advantage means that selfish mining hurts the mining rewards of all miners, but the attacker's actions penalize himself less than they penalize other victim miners.

Finally, unlike a mining pool block withholding attack, the 38.2% attack has no inherent need for mining pools. If the attacker does happen to have a mining pool, however, rational miners in a 38.2% attack might have incentive to join the attacker's pool. Cortois, Bahack, and Rosenfeld analyzed Bitcoin block withholding attacks [5,6,15]. Bonneau, Felten, Goldfeder, Kroll, and Narayanan [4] investigated a variation of our double-spend attack in which one offers monetary bribes, rather than puzzle rewards, to gain control of the network. We remark that the relative profit margins for a 38.2% attack or double-spend attack exceed those of selfish mining and mining pool block withholding.

Selfish mining. Due to sheer luck, a miner who controls a significant portion of the network's computing power will occasionally find himself successfully mining two or three blocks in rapid succession. When this happens, the miner might not immediately announce his blocks but rather continue to mine privately on a private blockchain. At this point, the miner's private blockchain is longer than the public one. Right before the public chain catches up to the private one, the miner reveals his longer chain which the consensus protocol dictates is, in fact, the valid chain. Thus the miner may double-spend any money spent on the public blockchain during the fork. Eyal and Sirer introduced this attack in [8].

Mining pool block withholding. In order to obtain a nontrivial chance of winning the race to mine the next block, miners arrange themselves into "pools" which split both the mining work and rewards among members. According to the pool protocol, a miner who solves the nonce should announce it, and then everyone in the pool shares the reward for mining a new block. However, Luu, Saha, Parameshwaran, Saxena, and Hobor [12] and Eyal [7] demonstrated that miners have financial incentive *not* to reveal their solution. A miner can join multiple pools simultaneously, and by not reporting a block in one he increases his chances of obtaining a reward in the other.

Acknowledgements

We thank Frank Stephan, Loi Luu, and Gregory J. Duck for useful discussions and helpful feedback.

References

1. `http://coinmarketcap.com/`.
2. `http://www.mail-archive.com/cryptography@metzdowd.com/msg09959.html`,.
3. Joseph Bonneau, Edward W. Felten, Steven Goldfeder, Joshua A. Kroll, and Arvind Narayanan. Why buy when you can rent? bribery attacks on Bitcoin consensus. `http://www.jbonneau.com/doc/BFGKN14-bitcoin_bribery.pdf`.
4. Nicolas T. Courtois. On the longest chain rule and programmed self-destruction of crypto currencies. *CoRR*, abs/1405.0534, 2014.
5. Nicolas T. Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in Bitcoin digital currency. *CoRR*, abs/1402.1718, 2014.
6. Ittay Eyal. The miner's dilemma. In *IEEE Symposium on Security and Privacy (SP 2015)*, pages 89–103, May 2015.
7. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, volume 8437 of *Lecture Notes in Computer Science*, pages 436–454. Springer Berlin Heidelberg, 2014.
8. Ethereum Foundation. Ethereum's white paper. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2014.
9. Pedro Franco. *Understanding Bitcoin: Cryptography, Engineering and Economics.* John Wiley & Sons, 2014.
10. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Proceedings of Eurocrypt 2015*, 2015.
11. Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas Hobor. On power splitting games in distributed computation: The case of Bitcoin pooled mining. `http://eprint.iacr.org/2015/155`.
12. Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 706–719, New York, NY, USA, 2015. ACM.
13. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `http://bitcoin.org/bitcoin.pdf`.
14. Meni Rosenfeld. Analysis of Bitcoin pooled mining reward systems. *CoRR*, abs/1112.4980, 2011.
15. Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. `http://eprint.iacr.org/2015/464`.

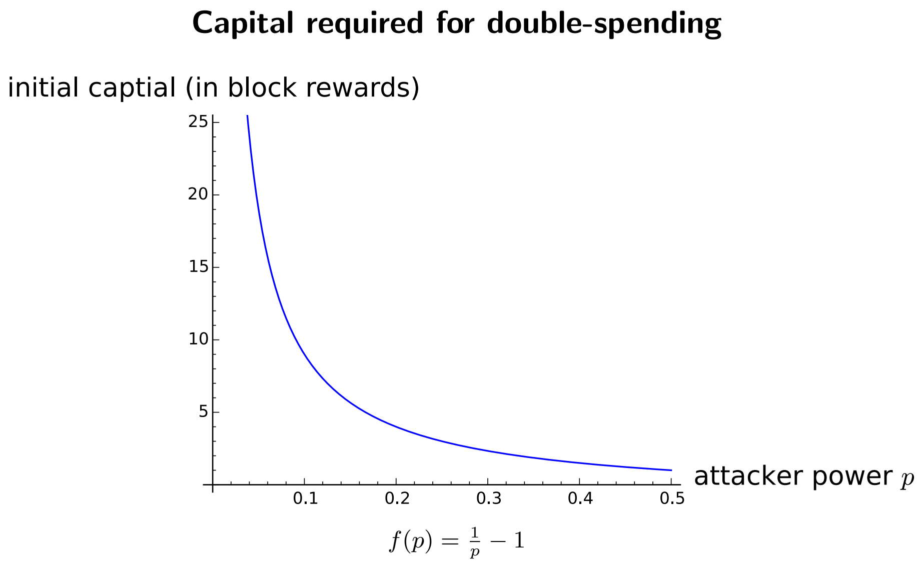


Fig. 2. Expected initial capital required to execute a double-spend attack with p fraction of the network power (for $k = 1$ block advantage).

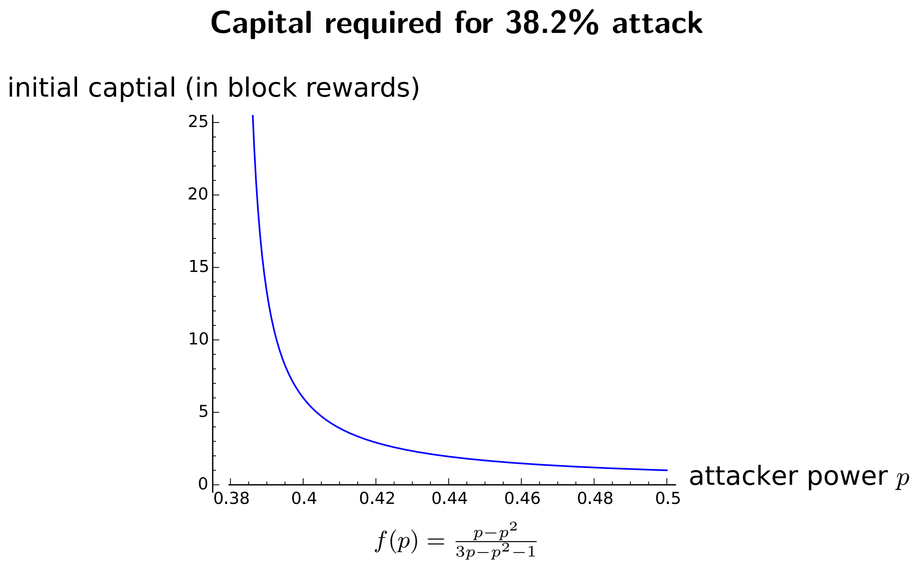
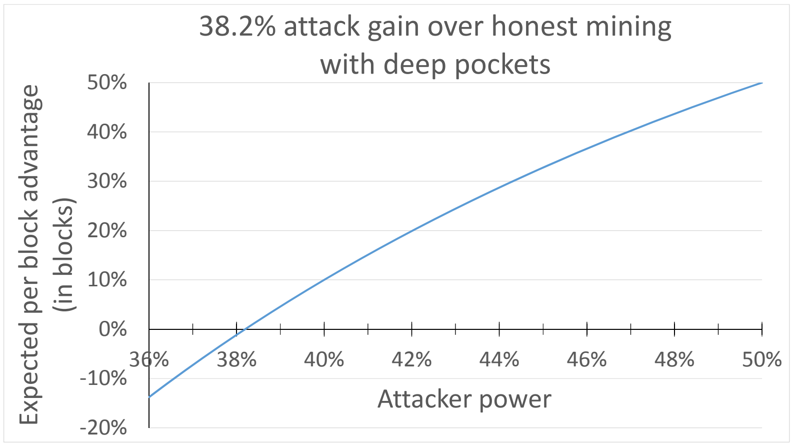
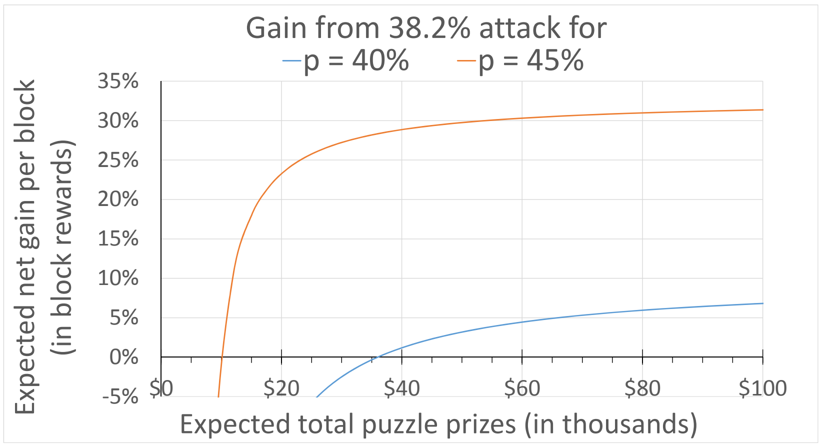


Fig. 3. Expected initial capital required to break even on a 38.2% attack relative to honest mining without double-spending and with p fraction of the network power.



$$f(p) = \frac{3p-p^2-1}{p}$$

Fig. 4. Expected net gain over honest mining for a 38.2% attack without double-spending with infinite budget for puzzle prizes. In general, the puzzle prize investment may be small compared to the attacker's financial investment in mining hardware.



$$f(x) = \frac{3p-p^2-1-\frac{1-2p}{x-1}}{p}$$

Fig. 5. Expected net gain over honest mining for a 38.2% attack without double-spending and with $p = 40\%$ and $p = 45\%$ of the network power. The dollar estimates assume 1 block reward = 25 BTC = \$6000 (Sept. 2015).