Bitcoin's Security Model Revisited

Yonatan Sompolinsky and **Aviv Zohar**
The Hebrew University of Jerusalem, Israel
{yoni_sompo,avivz}@cs.huji.ac.il

Abstract

Cryptocurrencies like Bitcoin provide probabilistic assurances to merchants that payments made to them will not be reversed. We extend the study of payment-reversal (aka *double spending*) attacks by considering more sophisticated behaviours of attackers. We show that attackers who wish to reverse payments face a decision problem: at each point in time, they must decide whether to continue the attack or to abandon it and launch a new one. Merchants can use our computed optimal attacks to set and adjust their transaction acceptance policies. We analyze and compute optimal attack policies for both single-shot attacks and long term persistent ones. We show that when one considers the decision problem induced by long term attacks, payments can be confirmed faster than previously thought, because an optimal attack involves frequent resets of the attack.

Our analysis utilizes an MDP construction adapted from [Sapirshtein *et al.*, 2015].

Finally, we demonstrate how the attack strategy changes if the merchant does not relay blocks to other Bitcoin nodes, and show that relaying blocks to others strictly improves the security of payments accepted by the node.

1 Introduction

Bitcoin is a novel decentralized system, invented in 2008 by Satoshi Nakamoto [Nakamoto, 2008], which allows users to transfer money to one another by publicly recording payments made with the currency. *Transactions* are aggregated in batches called *blocks*, and blocks in turn are sequentially organized in a chain collectively known as the *blockchain*. Once a transaction has been embedded in a block within the chain, it will be considered a part of the valid history of transactions. The security of a bitcoin transaction depends directly, therefore, on the probability that the block containing it remains part of the blockchain forever. In this work we show that this connection is more complex than commonly described, and we define precisely in what sense bitcoin transactions can be considered secure. We show that attackers face a decision problem, when aiming to reverse as many payments as possible, and we compute optimal attack policies using tools from AI.

The Bitcoin protocol. In order to create a new block, a node (aka *miner*) is required to solve a difficult cryptographic puzzle. Each new block contains a pointer to its predecessor – usually, the tip of the current chain – effectively extending it with every such additional block. The difficulty of the puzzle regulates the block creation rate, and ensures that the blockchain is extended approximately once every 10 minutes (every 2016 blocks the difficulty of the puzzle is adjusted to keep the growth rate of the chain constant).

In case several chains of blocks form, the Bitcoin protocol dictates that nodes extend the longest chain only (or the one they received first, in case of a tie), and discard and ignore blocks outside this chain. In particular, if an attacker deliberately creates a secret fork and manages to create a longer branch than the current public one, he can publish his branch and thereby replace all blocks in the public chain (after the fork), effectively reversing all payments embedded within them. This scheme is called a *double spending* attack.

Fortunately, the computational hardness of block creation ensures that a node with less than 50% of the computational power is unlikely to create more blocks than the rest of the nodes, over a long period of time. Consequently, (assuming that blocks propagate much faster than their creation rate), it is highly unlikely that a block buried deep enough in the longest chain would later be removed. *Merchants and payment recipients are thus advised to wait for several blocks (aka* confirmations*) to extend the chain above the block containing their transactions, before considering the payment as finalized.*

The classic security analysis. Satoshi in his original work [Nakamoto, 2008], as well as additional works that follow [Rosenfeld, 2014; Sompolinsky and Zohar, 2015], offer several acceptance policies whose security guarantees are given by a theorem of the following "flavour":

Theorem 1 (informal). *As long as the attacker holds less than 50% of the computational power, the probability of a transaction being reversed decreases exponentially with the number of confirmations the block containing it has received.*

For instance, a merchant who waits for 4 confirmations before accepting payments is safe against an attacker with 10%

of the computational power, with probability $\approx 1-0.00099$.[1]

Problems with the classic analysis. Alas, these analyses apply only when regarding naïve attack strategies, where the attacker tries to create a secret chain only after he broadcasts the transaction to the network. In contrast, consider the following strategy of a 10% attacker against a merchant which regularly waits for 4 confirmations: (i) The attacker continuously tries to create secret extensions to the public chain and to gain a lead of 2 blocks (in Bitcoin, he will usually be successful within 24 hours or less);[2] we term this preparatory stage *pre-mining*. (ii) Once he gains such a lead, the attacker broadcasts the transaction he aims to double spend. The merchant waits for 4 confirmations to appear in the public chain, and then accepts the payment. (iii) Meanwhile, the attacker continues to try and extend his secret chain. (iv) If his chain is longer than the public one at any moment in time after the merchant accepted, he releases his secret chain. This strategy is successful with probability ≈ 0.02728,[1] i.e., it is 27 times more likely to succeed than the attack considered by others.

It is easy to generalize this strategy and demonstrate that, by controlling the timing of the publication of the victim transaction, the attacker is able to guarantee himself an arbitrarily high probability of success in reversing transactions. Figure 1 illustrates such a pre-mining attack.

Admittedly, increasing the success-probability of the attack comes at the expense of long waiting times before launching it, and of wasting more blocks during the pre-mining stage. However, security guarantees given in the above form (Theorem 1) pretend to provide security even against *irrational* attackers that do not necessarily aim at maximizing their profit. Thus, pre-mining refutes a naïve reading thereof.

Alternative security model. Still, these analyses prove that in some sense it is (exponentially) difficult to reverse blocks, and they can be correctly interpreted as follows:

> On average, the attacker is able to reverse transactions embedded in only an exponentially small fraction of the blocks created within this period.

By the Strong Law of Large Numbers, there is an immediate connection between the success-probability of a single-shot attack and the fraction of overall successful ones: If an attack carried out in an arbitrary point in time (with no selective timing, as previous analyses assume) succeeds with probability upper bounded by ϵ, then the overall fraction of blocks reversed by a persistent attacker converges to a limit upper bounded by ϵ.

However, by carefully considering the dynamics of a long-run attack, specifically by *modeling it as a decision problem*, we show that in fact it is possible to produce tighter security guarantees that work in favour of the merchant. Indeed, an attacker that attempts to maximize the fraction of attacked blocks must actively decide when to give up on the attack on a specific block, if the odds are not in his favour, so that he may attack other more recent blocks instead.

Using a Markov Decision Process (MDP), we precisely define the decision problem and the resulting security model (see Section 2 and 3). We compute optimal attack strategies using an adaptation of the MDP-based algorithm from [Sapirshtein *et al.*, 2015].

Correcting the classic analysis. Arguably, the fractional "on average" model suggested above fits a merchant that engages frequently in bitcoin transactions best, and aims to minimize the number of successful attacks over a long period of time. In contrast, a merchant who uses the blockchain rarely, would probably be focused on defending a payment in a specific block, which corresponds to the security model suggested by the classic analysis.

Unfortunately, as argued above, securing a specific transaction in the blockchain is problematic: The attacker can engage in pre-mining efforts and wait until his lead ensures him a definite success, and only then publish the payment to the victim. Previous analyses which ignored this factor implicitly assume that the attack is carried out in an arbitrary point in time. This assumption might be justified in some scenarios, such as periodic and pre-scheduled payments, or when the attacker is not the entity that initiated the payment.

Still, even under such assumptions, an attack may involve pre-mining, which was not accounted for in previous analyses. We correct this by evaluating the optimal attack analytically. As a separate contribution, we repeat this analysis for the case where the merchant runs a lightweight "Simplified Payment Verification" (SPV) node and does not relay blocks that it receives to others in the network. We show that this provides the attacker with an advantage, and adjust the analysis of the optimal attack strategy in this case.

Our contributions can be summarized as follows:

- We define precisely in what sense bitcoin transactions can be considered secure, taking into account pre-mining and selective timing of transaction publication.
- We use an MDP-based algorithm, adapted from [Sapirshtein *et al.*, 2015], to provide tight upper-bounds on optimal attack strategies. Merchants can use our results and tools to reason about the security of their acceptance policies and adjust them accordingly.
- We correct the classic analysis of Bitcoin's security by accounting for pre-mining. We do so both for ordinary Bitcoin nodes and for merchants running lightweight clients that do not broadcast blocks they receive. The attacks we analyze are essentially a generalization of the Finney attack [Finney, 2011], and of the (somewhat lesser known) Vector76 [2011] attack.

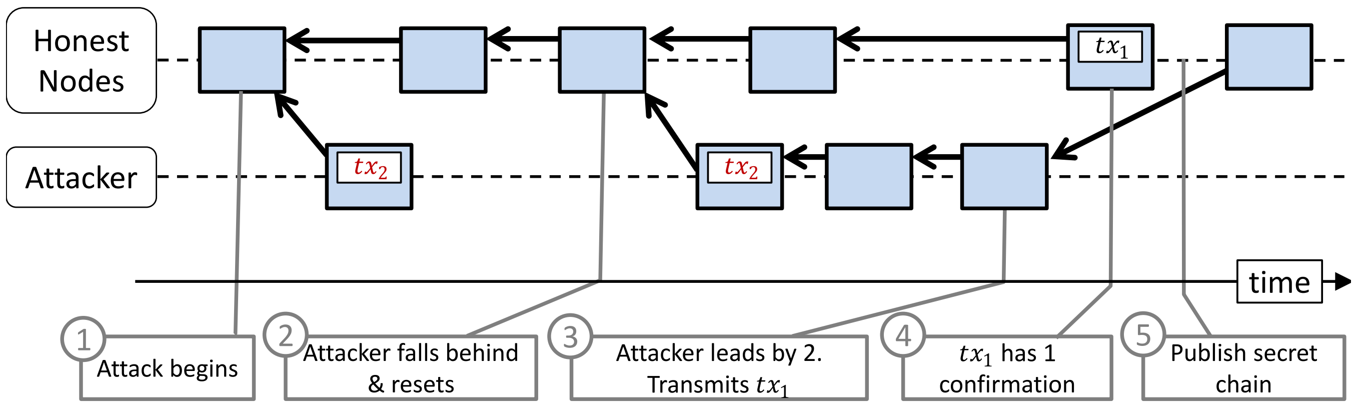
  Our results apply to other blockchain-based systems as well, such as Ethereum [ETH, ; Wood, 2014].

2 The model

We adopt the original setup analyzed by Satoshi Nakamoto [2008] that has become a standard model of Bitcoin's operation. The set of all miners creates blocks with

[1]This calculation is obtained by adapting the formula from [Rosenfeld, 2014]. In the original one, a pre-mining of 1 block was assumed.

[2]At any given moment, the chance that the next 2 consecutive blocks will be mined by the 10% miner is $0.1^2 = 0.01$. In 24 hours there are roughly 144 blocks, hence it's likely that an event with probability 0.01 will occur at least once, every 24 hours.



Honest Nodes
Attacker
tx_2
tx_2
tx_1
time
1 Attack begins
2 Attacker falls behind & resets
3 Attacker leads by 2. Transmits tx_1
4 tx_1 has 1 confirmation
5 Publish secret chain

Figure 1: The progression of a pre-mining attack on a 1-confirmation merchant. *(1) The attacker starts working on a secret chain with tx_2 inside its first block. (2) If the attacker's chain is shorter than the public one, the attacker gives up and restarts the attack. (3) The attacker manages to gain a lead of 2 blocks. (4) He then transmits the transaction he wishes to double spend, which is included in a block. The transaction gains enough confirmations (here $k = 1$), and the merchant delivers to the attacker the commodity he paid for. (5) The attacker publishes his secret chain and successfully reverses the payment.*

exponential inter-arrival times, with parameter λ (in Bitcoin, $\lambda = 1/600$ blocks/second).

Each block contains a reference to a single predecessor block (a cryptographic hash). The entire history of blocks created up to time t (including blocks not in the longest chain) thus forms a tree, which we denote by $\mathcal{T}_t$. We assume that, in accordance with the Bitcoin protocol, honest participants only keep track of the longest chain they have been given. We further assume that blocks propagate in the network very fast relative to $1/\lambda$; under this assumption the honest network's chain at every point t in time is uniquely determined, and we denote it by $\mathcal{C}_t$. The length of the chain at time t is denoted $height(\mathcal{C}_t)$. The height of a block B, $height(B)$, is defined as the number of blocks between it and the first *genesis* block.

The attacker is assumed to own a fraction α of the computational power, and the rest $(1 - \alpha)$ is owned by honest nodes. Thus, the attacker creates blocks at a rate of $\alpha \cdot \lambda$, and the honest participants at a rate of $(1-\alpha) \cdot \lambda$. Following [Eyal and Sirer, 2014], we assume that in case of a tie between the attacker's chain and the public chain, the attacker's communication capabilities are such that a fraction γ of the nodes receive his block first and adopt it; for such a race to take place, the attacker must release his matching block immediately after the public chain has extended. A pair α, γ thus characterizes the capabilities of the potential attacker against which the merchant wishes to defend himself. For each such pair, $\sigma_{\alpha,\gamma}$ denotes the merchant's acceptance policy, i.e, the (constant) number of confirmations the merchant waits for before accepting the transaction.

ϵ-fractional-robust policies. Given an acceptance policy $\sigma_{\alpha,\gamma}$, we define the set $\mathcal{A}^t(\sigma_{\alpha,\gamma}) := \{B \in \mathcal{T}_t \setminus \mathcal{C}_t : \exists s < t$ s.t. $B \in \mathcal{C}_s \wedge height(\mathcal{C}_s) - height(B) \geq \sigma_{\alpha,\gamma} - 1\}$; it is the set of blocks which were part of the longest chain at some point in history, had $\sigma_{\alpha,\gamma}$ confirmations in it (including themselves), but were later removed from it due to a successful attack. For simplicity, we assume that every block contains one payment from the attacker to the merchant.

Definition 1. *An acceptance policy $\sigma_{\alpha,\gamma}$ is said to be ϵ-*fractional-robust *iff for any attacker with parameters α and γ, and under any attack policy:*

$$\lim_{t \to \infty} \frac{|\mathcal{A}^t(\sigma_{\alpha,\gamma})|}{|\mathcal{C}_t|} < \epsilon \tag{1}$$

The space of possible attack policies will be defined in Section 3. Importantly, note that $|\mathcal{C}_t|$ grows at a constant rate in time: in Bitcoin, the chain is extended approximately once every 10 minutes. Therefore, (1) essentially measures the number of blocks attacked per unit of time, on average. Consequently, *in order to bound the overall damage a persistent attacker can cause the merchant on average, the merchant should cap the amount paid to him in each block, and use an ϵ-fractional-robust acceptance policy.*

ϵ-arbitrary-robust policies. Let tx be a transaction that appears in some block B in the blockchain. It is possible to define the robustness of tx under the assumption that the attacker was not involved in the choice of B, or, in other words, that the attack was performed in an arbitrary point in time.

Definition 2. *An acceptance policy $\sigma_{\alpha,\gamma}$ is said to be ϵ-*arbitrary-robust *iff for any attacker with parameters α and γ, and under any attack policy:*

$$\Pr\left(\exists s > t : B \notin \mathcal{C}_s \mid height(\mathcal{C}_t) - height(B) = \sigma_{\alpha,\gamma} - 1\right) < \epsilon, \tag{2}$$

where $B \in \mathcal{C}_t$ is a block in the chain whose choice was independent of the attacker's actions.

3 A persistent attack as a decision problem

We are now ready to formalize the decision problem that the attacker faces, and describe how we compute the optimal policies. The attack is carried out over the entire history of blocks mining. We assume that every block contains one payment from the attacker to the merchant; this is possible, e.g., if the merchant is an online currency exchange. The attacker's goal is to carry out as many successful double-spending attacks as possible, by overriding the public chain after payments in it were confirmed by the merchant. At each step, he needs to decide whether to extend his secret chain, to publish it, or to abandon it and begin a new one. A successful attack takes place whenever the attacker publishes its secret chain, this chain is longer than (or sometimes, equal to) the public chain, and the public chain is at least k blocks long; indeed, without the latter condition, the merchant hasn't accepted yet any payment in the public chain, hence overriding it does not harm him.

3.1 The underlying Markov Decision Process

We follow [Sapirshtein *et al.*, 2015] and define the attacker's attack policy as a function that determines his action at every possible state. A state of the MDP is characterized mainly by l_a, the length of the attacker's secret chain, and by l_h, the length of the public chain; these lengths are counted from the latest block which both chains agree on. The possible actions are:

- $adopt$– abandoning the current secret chain, so that future chains of the attacker will contain the current tip of the public chain.
- $override$– publishing the current secret chain, in case it is longer than the public one, thereby overriding it.
- $match$– publishing the current secret chain, in case it is the same length as the public one; by our model (following [Eyal and Sirer, 2014]), a fraction γ of the honest nodes will adopt the attacker's chain.
- $wait$– waiting for the next block creation without any additional action.

Every state-transition corresponds to the creation of a new block, either by the attacker (with probability α) or by an honest node (with probability $1-\alpha$). In case of a $match$, the next block will extend the attacker's chain with probability $\alpha+\gamma\cdot(1-\alpha)$, because apart from the attacker there is additionally a fraction γ of honest nodes that adopts his chain. Since the attacker can only $match$ on time if the new block was created by an honest node (otherwise all honest nodes received the honest block first and adopted it), we must encode this information in the state as well. The state is therefore represented by a 3-tuple $(l_a, l_h, fork)$, where $fork$ might take the values rel, $irrel$, and $active$; rel implies that the $match$ action is now feasible (i.e., the last block was created by an honest node hence the attacker can publish immediately a matching chain), $irrel$ implies that $match$ is currently infeasible, and $active$ means that the attacker already matched previously and the honest nodes are already split between his chain and the previous honest chain.

To compute an optimal attack correctly, one must normalize by the length of the chain, as in (1), and not by the total number of blocks, because in the long run only the chain growth is kept constant (via adaptation of the block creation

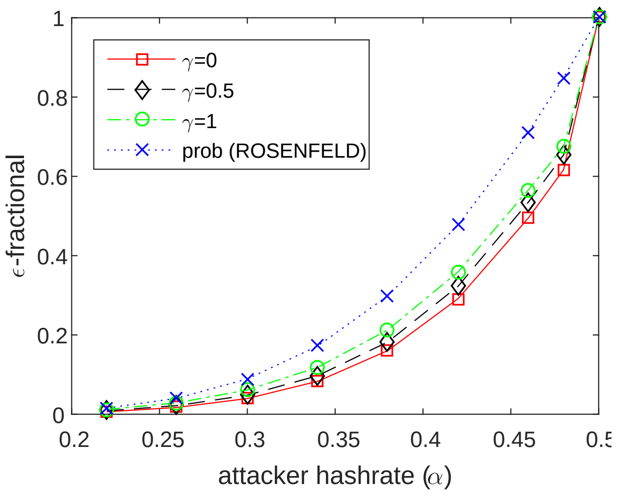


Figure 2: The fraction of accepted blocks that an optimal attacker can double spend, on average, against the acceptance policy $\sigma \equiv 6$, as a function of the attacker's hashrate α. The different curves correspond to different values of γ. The probabilistic bound under the classic model (from [Rosenfeld, 2014]) is also plotted for comparison.

rate). We adapt a technique from Saprishtein *et al.* who define a reward system with two coordinates, and use a binary-search-based algorithm to compute the optimal policy. We refer the reader to [Sapirshtein *et al.*, 2015], for a more comprehensive description. A succinct representation of the transition and reward matrices appears in Table 1 below.

The main difference from Sapirshtein *et al.*'s algorithm is that the latter used this technique to compute optimal selfish mining attacks, and to maximize the number of attacker blocks in the chain. In contrast, we reward the attacker differently (as its objective here is different): he is rewarded 1 unit for every block that the merchant accepted – i.e., that had k confirmations – and that was later removed from the chain.

3.2 Results

We constructed the MDP described above, for various acceptance policies of the merchant, truncating the state-space to consider chains of length up to 50 blocks. We used the MDP solver from [Chadès *et al.*, 2014], where we utilized the relative value iteration routine to obtain the optimal average reward under an undiscounted average reward scheme. The value of the optimal attack, as computed by the MDP algorithm, defines the degree of fractional-robustness achieved by the corresponding acceptance policy. Table 2 presents the average percentage of blocks that the attacker is able to successfully attack, for several numbers of confirmations. Each cell was computed separately with its own optimal policy. A merchant willing to tolerate a fraction ϵ of successful double-spends, against an attacker with computational power at most α, should choose an acceptance policy such that the corresponding cell in the table is ϵ or less.

Figure 2 depicts the ϵ-fractional-robustness of the policy $\sigma_{\alpha,\gamma} = 6$, as computed by the algorithm, for different γ's. The results of Rosenfeld for the success-probability of an at-

Table 1: The transition and reward matrices of the MDP. The first coordinate of 'Rewards' accumulates the attacker's rewards, and the second one is used for proper normalization.

State (from) $\times$ Action	**State (to)**	**Probability**	**Reward**
$(l_a, l_h, \cdot), adopt$	$(1, 0, irrel)$	α	$(0, l_h)$
	$(0, 1, irrel)$	$1 - \alpha$	
$(l_a, l_h, \cdot), override^{\dagger}$	$(l_a - l_h, 0, irrel)$	α	$(l_h - k + 1, k)^{\S}$
	$(l_a - l_h - 1, 1, rel)$	$1 - \alpha$	
$(l_a, l_h, irrel), wait$	$(l_a + 1, l_h, irrel)$	α	(0,0)
$(l_a, l_h, rel), wait$	$(l_a, l_h + 1, rel)$	$1 - \alpha$	(0,0)
$(l_a, l_h, active), wait$	$(l_a + 1, l_h, active)$	α	(0,0)
$(l_a, l_h, rel), match^{\ddagger}$	$(l_a - l_h, 1, rel)$	$\gamma \cdot (1 - \alpha)$	$(l_h - k + 1, k - 1)^{\P}$
	$(l_a, l_h + 1, rel)$	$(1 - \gamma) \cdot (1 - \alpha)$	(0,0)

†Feasible only when $l_a > l_h$. ‡Feasible only when $l_a \geq l_h$. §If $l_h < k - 1$, then the reward is $(0, l_h + 1)$ since no block was actually attacked. ¶If $l_h < k - 1$, then the reward is $(0, l_h)$.

$\alpha \backslash conf$	1	2	3	4	5	6	7	8	9	10
2%	0.08%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%
6%	0.69%	0.12%	0.03%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%
10%	1.89%	0.52%	0.16%	0.05%	0.02%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%
14%	3.70%	1.34%	0.53%	0.23%	0.10%	0.05%	0.02%	$\approx$ 0%	$\approx$ 0%	$\approx$ 0%
18%	6.16%	2.75%	1.34%	0.69%	0.36%	0.20%	0.11%	0.06%	0.04%	0.02%
22%	9.37%	4.92%	2.80%	1.66%	1.02%	0.64%	0.41%	0.27%	0.18%	0.12%
26%	13.47%	8.12%	5.34%	3.63%	2.52%	1.78%	1.28%	0.92%	0.67%	0.49%
30%	18.71%	13.20%	9.63%	7.19%	5.48%	4.23%	3.30%	2.60%	2.07%	1.66%
34%	26.46%	20.57%	16.36%	13.29%	10.99%	9.17%	7.69%	6.49%	5.51%	4.71%
38%	36.54%	31.04%	26.95%	23.60%	20.77%	18.37%	16.39%	14.66%	13.16%	11.84%
42%	50.32%	46.42%	42.99%	39.91%	37.17%	34.73%	32.49%	30.43%	28.56%	26.84%
46%	69.53%	67.65%	65.84%	64.06%	62.33%	60.64%	59%	57.38%	55.79%	54.23%
48%	81.48%	80.59%	79.66%	78.72%	77.75%	76.77%	75.76%	74.73%	73.67%	72.59%
50%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 2: The fraction of the network's blocks that an attacker with a given hashrate (α) successfully attacks, when using an optimal attack policy, given the number of confirmations the acceptance policy waits for ($conf$).

tack on a single block are included for comparison. Interestingly, the fraction of blocks that can be attacked, in the long term, is in fact *lower* than implied by Rosenfeld. This is because his analysis (and Satoshi's as well) considers an attack on a single block that goes on infinitely—the attacker is assumed to never give up and to try to catch up with the chain no matter how far behind he is. In contrast, an attacker that aims to maximize the fraction of blocks he successfully attacks must occasionally give up and restart the attack if he is far behind. This effect is demonstrated in these results (note that, on the other hand, our model allows the attacker to double spend several blocks at once. The effective ϵ is lower nonetheless). A similar effect occurs under different sizes of attacker or different numbers of confirmations.

3.3 Optimal attack policies

We now present the optimal attack policies returned by the algorithm, in two particular setups. Tables 3,4 describe the attack policy for an attacker with $\alpha = 0.25$ and with $\gamma = 0$ or $\gamma = 0.5$. The row numbers correspond to the length of the attackers branch l_a and the columns to the length of the honest network's branch l_h. Actions are abbreviated to their initials: $\boldsymbol{a}dopt, \boldsymbol{o}verride, \boldsymbol{m}atch, \boldsymbol{w}ait$, while the token '$*$' represents an unreachable state. When $\gamma > 0$ the $match$ action becomes feasible (see Table 1). Accordingly, each entry in the bottom table contains a string of three characters, corresponding to the possible values of the $fork$ variable.

Notice that here the attacker does not override the network's chain until the honest branch is of length 2 or more, as a successful attack requires that the merchant sees 2 confirmations above his transaction before the attack is released. Note further that the attacker does not give up on his attack when he is just slightly behind. If his chain is relatively long, he will not abandon it unless he is at least 2 blocks behind.

4 Revisiting the classic security analysis

As discussed in previous sections, a merchant can consider a specific payment in the blockchain as secure if he is willing to assume that the attack was carried out in an arbitrary point in time. Using this (implicit) assumption, previous works analyzed the security of a transaction tx in terms of the probability that the block containing it, B, will not forever remain in the blockchain; i.e., in terms of Definition (2).

However, even under this assumption, the attacker can still engage in pre-mining, and try to gain an early advance before B's creation. Previous works that analyze the security of payments do not take this into account (with the exception of [Pass *et al.*, 2016]; see Section 6), and we thus correct the analysis. Our main result (Lemma 1) is a probability distribution over the lead that the attacker may have at the time of the

$l_a \backslash l_h$	0	1	2	3	4	5	6	7	8	9
0	w	a	*	*	*	*	*	*	*	*
1	w	w	w	a	*	*	*	*	*	*
2	w	w	w	w	a	*	*	*	*	*
3	w	w	w	w	w	w	a	*	*	*
4	w	w	w	o	w	w	w	a	*	*
5	w	w	w	w	o	w	w	w	a	*
6	w	w	w	w	w	o	w	w	w	a
7	w	w	w	w	w	w	o	w	w	w
8	w	w	w	w	w	w	w	o	w	w
9	w	w	w	w	w	w	w	w	o	w

Table 3: Optimal actions for an attacker with $\alpha = 0.25$ and $\gamma = 0$, against the acceptance policy $\sigma \equiv 2$ confirmations (see also caption of Table 4).

$l_a \backslash l_h$	0	1	2	3	4	5	6
0	w**	a**	***	***	***	***	***
1	w**	*w*	w**	a**	***	***	***
2	w**	ww*	wm*	w**	a**	***	***
3	w**	ww*	www	wm*	w**	a**	***
4	w**	ww*	www	wmw	wm*	w**	a**
5	w**	ww*	www	www	omw	wm*	w**
6	w**	ww*	www	www	*ww	omw	wm*

Table 4: Optimal actions for an attacker with $\alpha = 0.25$ and $\gamma = 0.5$, against the acceptance policy $\sigma \equiv 2$ confirmations. The row and column indices correspond to l_a and l_h, respectively. See legend in text.

creation of B, which terminates the pre-mining attack stage. Using this result, we proceed to calculate in Theorem 2 the probability defined in (2), by augmenting and correcting the analysis from [Rosenfeld, 2014]:

Lemma 1. *Let B be an arbitrary block in $\mathcal{C}^t$, and denote by (l_a, l_h) the respective lengths of the attacker and the public chain at the time of B's creation. Then, for any $l \geq 0$, under the optimal attack policy:* $\Pr(l_a - l_h = l) = \frac{1-2\cdot\alpha}{1-\alpha}\cdot\left(\frac{\alpha}{1-\alpha}\right)^l$.

Theorem 2.

$$\Pr\left(\exists s > t : B \notin \mathcal{C}^s \mid height(\mathcal{C}^t) - height(B) = k-1\right) = \sum_{l=0}^{\infty} \frac{1-2\cdot\alpha}{1-\alpha}\cdot\left(\frac{\alpha}{1-\alpha}\right)^l \cdot$$

$$\left(\sum_{m=0}^{k-l}\binom{m+k-1}{m}\cdot\alpha^m\cdot(1-\alpha)^k\cdot\left(\frac{\alpha}{1-\alpha}\right)^{k+1-m-l} + \sum_{m=k-l+1}^{\infty}\binom{m+k-1}{m}\cdot\alpha^m\cdot(1-\alpha)^k\right)$$

The proofs of Lemma 1 and Theorems 2-4 appear in the full version of the paper. Table 5 shows the difference between the corrected analysis and the original one, for an attacker with $\alpha = 0.3$. The uncorrected analysis is a variation of Rosenfeld [2014]. This result can be used by merchants to correctly choose ϵ-arbitrary-robust acceptance policies.

5 The Generalized Vector76 Attack

In this section we present the Generalized Vector76 attack. The attack is aimed against lightweight clients, which typically keep track of the longest chain of blocks but do not relay blocks they receive to other nodes. As a result, even if the chain held by such a node is the longest one, it is not necessarily published—the chain could have originated from an attacker node which hasn't broadcast it yet.[3] The attack proceeds as follows:

1. The attacker starts working on a secret branch of the chain. It embeds the transaction tx_1 (that it later wishes to reverse) in its first block.
2. If the merchant requires $\sigma \equiv k$ confirmations, the attacker needs to build an additional $k-1$ blocks on top of the one containing tx_1 (for a total of k confirmations). He attempts to do so in secret.
3. If his branch of the chain is longer than that of the public chain, at some point *after* he has k confirmations for tx_1, he shows the k confirmations to the lightweight client which then accepts it as the legitimate chain, since it is the longest one.
4. The attacker then transmits a conflicting transaction tx_2 to the public network. As the honest network is not aware of the attacker's chain, the public chain will grow long enough for tx_2 to be accepted by all nodes (and eventually even by the attacked one).

Figure 3 depicts the attack. Again, notice that a crucial stage in the success of the attack is that the honest network does not adopt the block containing tx_1.

The following theorem states that a merchant running a lightweight Bitcoin node is less secure than a full node:

Theorem 3. *Let $\epsilon > 0$. If $\sigma_{\alpha,\gamma}$ is an ϵ-fractional-robust policy for a non-relaying Bitcoin node, then there exists an $0 < \epsilon' < \epsilon$ such that $\sigma_{\alpha,\gamma}$ is an ϵ'-fractional-robust policy for a full Bitcoin node.*

Fortunately, we can still upper bound the success-probability of attacks on lightweight clients, using the following acceptance policy. Let $\sigma^{spv}_{\alpha,\gamma} := \min\{k \in \mathbb{N} : g(k,\alpha) < \epsilon \cdot (1-\alpha)\}$, where $g(k,\alpha) :=$

$$\frac{1-2\cdot\alpha}{1-\alpha}\cdot\sum_{l=0}^{\infty}\left(\frac{\alpha}{1-\alpha}\right)^l\cdot\left(\sum_{n=0}^{k+l}\binom{n+k-1}{n}\cdot\alpha^k\cdot(1-\alpha)^n + \sum_{n=k+l+1}^{\infty}\binom{n+k-1}{n}\cdot\alpha^{n-l}\cdot(1-\alpha)^{k+l}\right).$$

Theorem 4. *For any $\epsilon > 0$, the policy $\sigma^{spv}_{\alpha,\gamma}$ is ϵ-fractional-robust, even when run by a non-relaying Bitcoin node.*

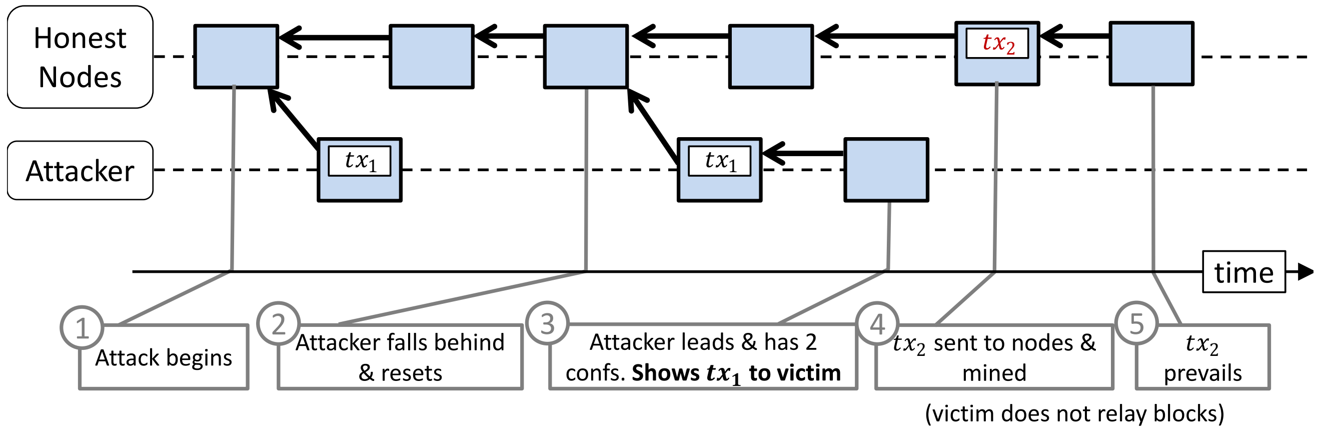
6 Related work

Since Satoshi's analysis in his white paper, several works have dealt with correcting and extending his analysis [Sompolinsky and Zohar, 2015; Garay *et al.*, 2015; Karame *et*

[3]A simpler version was suggested by a user named Vector76 in the bitcoinTalk forums to possibly explain a successful double spending attack against the MyBitcoin e-wallet [Vector76, 2011].

$\# \, conf$:	1	2	3	4	5	6	7	8	9	10
without pre-mining	0.3086	0.2330	0.1801	0.1412	0.1117	0.0891	0.0714	0.0575	0.0465	0.0376
with pre-mining	0.5069	0.3932	0.3097	0.2463	0.1973	0.1588	0.1283	0.1040	0.0846	0.0688

Table 5: The success-probability of an attack on an arbitrary block, for an attacker with $\alpha = 0.3, \gamma = 0$, with and without pre-mining.


Honest Nodes
Attacker
tx_2
tx_1
tx_1
time
1 Attack begins
2 Attacker falls behind & resets
3 Attacker leads & has 2 confs. Shows tx_1 to victim
4 tx_2 sent to nodes & mined
5 tx_2 prevails
(victim does not relay blocks)

As the attack begins the attacker starts working on a significant secret chain with tx_1 inside its first block (1). If the attacker's chain is too far behind it may restart the attack (2). The attacker manages to gain a lead of 1 blocks, but has the two confirmations on his tx_1 needed to convince the victim (3). He then reveals the secret chain to the victim (that does not relay it), and collects an item in exchange. He then transmits the double spending transactions tx_2 to the network which is then included in a block (4). The network continues to mine atop tx_2 and it eventually prevails (5).

Figure 3: The progression of a generalized Vector76 attack on a 2-confirmation merchant

al., 2012]. Rosenfeld [2014] goes on to correct the analysis, and includes the pre-mining of a single block before it is launched. An important work of Pass *et al.* [2016] proves formally that a double-spending attack fails eventually even when considering pre-mining; in comparison, we provide the precise distribution over the pre-mining lead, which can be used by merchants to choose correct acceptance policies.

In the full version of this paper, which will be made available online, we investigate the selfish mining and double spending strategies that maximize the attacker's utility. In contrast, the focus of our work is to define a security model assuming an arbitrary possibly *irrational* attacker, which provides us with worst-case security guarantees for bitcoin transactions. This difference in the objectives yields different designs of the MDP, both of which are extensions of [Sapirshtein *et al.*, 2015]. The latter provided an algorithm to derive optimal block withholding attacks, improving upon a scheme by Eyal and Sirer [2014]. Our work uses similar techniques to quantify the optimal fraction of successfully attacked blocks. The problem of selective-timing of attacks, the fractional security model, and the resulting decision problem that the attacker faces—were not discussed in previous works on this topic.

Other AI-related works on Bitcoin include [Babaioff *et al.*, 2012] on the incentives to share transactions, [Lewenberg *et al.*, 2015] on game-theoretic dynamics of miners' coalitions, [Eyal, 2015] on a game where some mining nodes sabotage other nodes' mining efforts, and more [Rosenfeld, 2011; Luu *et al.*, 2015; Johnson *et al.*, 2014].

7 Conclusion

In this work we provided a formal security model for bitcoin transactions. We demonstrated that it is not enough to analyze blocks' robustness, namely, the probability that a given block remains forever in the longest chain. Indeed, an attacker can selectively embed transactions in blocks whenever the conditions are in his favour. Specifically, he can wait for his secret pre-mined forks to obtain a sufficient lead over the public chain, before carrying out the attack.

In our fractional security model, transactions are secure if the attacker cannot reverse a significant portion of them. We devised an algorithm to compute the worst-case attacker, under this model. The resulting analysis is more tight, as an attacker must trade-off his current attack with future ones.

We additionally revisited the classic security model which assumes that the attack is carried out in an arbitrary point in time (which was not controlled by the attacker). We've shown that transactions are less secure than previously claimed, due to the pre-mining attack stage.

Finally, we've formalized the Vector76 attack, which proves that Bitcoin nodes that do not relay blocks can more easily be defrauded—an attacker can feed to such a node a chain of blocks which will not become part of the public chain. Thus, such nodes need to wait longer in order to meet the same level of security as full nodes which do relay blocks to their peers.

References

[Babaioff *et al.*, 2012] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM, 2012.

[Chadès *et al.*, 2014] Iadine Chadès, Guillaume Chapron, Marie-Josée Cros, Frédérick Garcia, and Régis Sabbadin. Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography*, 37(9):916–920, 2014.

[ETH,] Ethereum. https://www.ethereum.org/.

[Eyal and Sirer, 2014] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[Eyal *et al.*, 2015] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. *arXiv preprint arXiv:1510.02037*, 2015.

[Eyal, 2015] Ittay Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[Finney, 2011] Hal Finney. The finney attack, 2011. Originally in https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384.

[Garay *et al.*, 2015] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.

[Gervais *et al.*, 2016] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 3–16. ACM, 2016.

[Johnson *et al.*, 2014] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In *International Conference on Financial Cryptography and Data Security*, pages 72–86. Springer, 2014.

[Karame *et al.*, 2012] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917. ACM, 2012.

[Lewenberg *et al.*, 2015] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey 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.

[Luu *et al.*, 2015] Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas Hobor. On power splitting games in distributed computation: The case of bitcoin pooled mining. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 397–411. IEEE, 2015.

[Nakamoto, 2008] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

[Pass *et al.*, 2016] Rafael Pass, Cornell Tech, and Lior Seeman. Analysis of the blockchain protocol in asynchronous networks. 2016.

[Rosenfeld, 2011] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[Rosenfeld, 2014] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[Sapirshtein *et al.*, 2015] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. *CoRR*, abs/1507.06183, 2015.

[Sompolinsky and Zohar, 2015] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. *Financial Cryptography and Data Security*, 2015.

[Vector76, 2011] Vector76. The vector76 attack, 2011. Originally in https://bitcointalk.org/index.php?topic=36788.msg463391#msg463391.

[Wood, 2014] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.