Proceedings of the First Workshop on Formal Methods for Blockchains, FMBC, 2019

Néstor Cataño, E-mail: nestor.catano@gmail.com
Diego Marmsoler, E-mail: diego.marmsoler@tum.de
Bruno Bernardo, E-mail: bruno@nomadic-labs.com

Foreword

We are glad to present the articles of the 1st Workshop on Formal Methods for Blockchains (FMBC), which is part of the Formal Methods congress that is held this year in the city of Porto, Portugal. We expect to hold this workshop in the coming years.

This pre-proceedings consists of 10 articles with a conditional acceptance for publication in the post-proceedings, 2 lighting talk articles. The versions hereby submitted are preliminary versions of the final ones that will be published in the post-proceedings.

Acknowledgements

We would like to express our most sincere gratitude to the FM chairs, all the PC members, the external reviewers, the Invited Speaker, and all the authors who submitted their work and made this workshop possible.

We particularly would like to express our gratitude to our sponsor Nomadic Labs for its generous support for the realization of FMBC 2019.

A Distributed Blockchain Model of Selfish Mining

Dennis Eijkel and Ansgar Fehnker

d.j.eijkel@student.utwente.nl and ansgar.fehnker@utwente.nl

University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands

Abstract. Bitcoin is still the most widely used cryptocurrency. A big part of Bitcoin's appeal is that it maintains a distributed ledger for transactions known as the blockchain. Miners receive a fee for every block of transactions that they mine, and should expect a reward proportional to the computational power they provide to the network. Eyal and Sirer introduced *seflish mining*, a strategy timing the publication of blocks to give them a significant edge in profits. This paper models the behavior of honest and selfish mining pools in Uppaal, and analyses properties of the mining process in the presence of network delay. This shows what effects selfish mining would have on the share of profits, but also on the number of orphaned blocks in the blockchain. This analysis allows us to compare those results to known results from literature and to real world data. This analysis shows that it is essential to take into account that there does not exist a single view of the blockchain.

Keywords: Bitcoin, Bitcoin mining, Selfish mining, Uppaal

1 Introduction

Bitcoin [3,11] is at the time of writing the most used cryptocurrency [5] by market capitalisation. Miners in the Bitcoin network are incentivised by the reward that they receive for validating new blocks of transactions. The aim is that every miner receives its fair share of said reward for the computational effort they perform for the network.

The Bitcoin protocol does not specify when miners must publish their newly found blocks. The most basic strategy is to publish them immediately after the miner finds them. This is referred to as the *honest* strategy.

Eyal and Sirer introduced a strategy for publishing newly found blocks called *selfish mining* [8]. It forces honest miners to waste computational power by waiting strategically and responding to what other miners in the network find and publish. Eyal and Sirer provide in [8] pseudo-code for selfish mining, along with a mathematical model of the forking behaviour of the blockchain, and an additional model for the rewards. They compute the expected rewards in the steady state, i.e. in the long run, depending on the share of the selfish pool in the computational power, and the share of races the selfish pool will win in case there are competing forks. For this, they computed a threshold for which selfish mining will increase the profit of the miner. Below this threshold, selfish mining will actually incur a penalty for the selfish miner.

This paper presents an Uppaal-SMC model for selfish mining. It models a blockchain network as a network of nodes, each with their own copy of the blockchain. The model includes stochastic network delays, which means that on average it will take a while before new blocks are adopted by the network. These aspects are absent from the Eyal and Sirer models. Uppaal-SMC can then analyse the behaviour of the network and the evolution of the blockchain over the simulation time – one day – and compare this with historical data obtained from the real blockchain. In particular, how selfish mining affects the number of expected forks, and how this is distinguished from the frequency of forks in the presence of selfish miners.

Chaudary et al. used Uppaal in [7] to model majority attacks. Their paper focuses on blockchain forking and included a detailed model of the blockchain. In [9] the same authors present a simplified version of the model presented in this paper to analyse a particular type of majority attack, intended to enforce a new Bitcoin standard. Uppaal was also used by Andrychowicz et al. to verify the security of Bitcoin contracts, and to repair several issues in the protocol [6].

Sapirshtein et al. mathematically investigate bounds for which selfish mining is profitable and optimize the original strategy [13]. They show that selfish mining can be optimized, such that the threshold above which the strategy is profitable is lower than described in the original paper [8]. Heilman et al. used Monte-Carlo simulation to investigate eclipse attacks and proposed countermeasures that will reduce the chances of such attacks to succeed [10]. Neudecker presents a full-scale simulation model of Bitcoin to study partition attacks [12].

The next section will describe *selfish mining* and its pseudo-code implementation. Section 3 describes the Uppaal-SMC model, and Section 4 the results of the analysis. Section 5 will conclude with a discussion of future work.

2 Selfish Mining

2.1 Bitcoin Mining Process

Bitcoin is a distributed and decentralized cryptocurrency [3,11] with a shared ledger of transactions which is stored in an append-only chain of *blocks* called the *blockchain*. A block contains a group of transactions, the hash of the preceding block, and a nonce. Since the block also includes the hash of the preceding block it defines a chain of blocks.

Nodes in the peer-to-peer Bitcoin network run a process, known as *mining*, to validate blocks of transactions, as well as to induce an order on transactions. Validation entails finding random nonce such that the hash value of the block falls below a certain threshold. Finding such a nonce can be considered to be

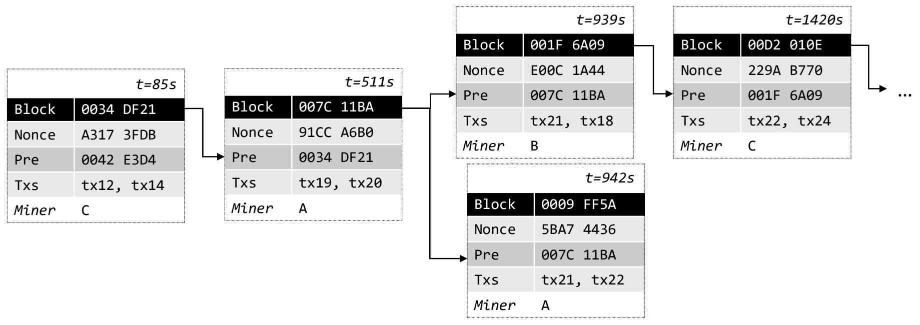

Fig. 1. Illustration of the Blockchain as hash-chain of blocks of transactions. For simplicity each block contains only two transactions.

a stochastic process with an exponential distribution, and is called the *proof-of-work challenge*. The threshold is regularly updated and agreed upon by the entire network such that a new block will be found on average every 10 minutes.

Figure 1 illustrates a blockchain. It starts with a block found by Miner C at `t=85s`, followed by a successor found by Miner A at `t=511s`. Due to the distributed nature of the network two pools may find a block at about the same time: in the example Miner B at `t=939s`, and Miner A at `t=942s`. If Miner A would have received the block of Miner B before it found its own, it would have abandoned its effort and switched to the `Block 001F 6A09`. The example assumes instead that Miner A found its own block first.

At this point, both blocks have been successfully mined as potential successors of `Block 007C 11BA`. Miners will continue with the block they receive first, and due to the distributed nature of the network, different pools may continue with mining different blocks, giving rise to so-called *forks*. It could take some time to resolve a fork and during that time, different views of the blockchain will exist. Blocks that fall outside of this longest chain are called *orphaned* blocks.

The race in Fig.1 is resolved as soon as the next block is found; here Block `001F 6A09`. Once this happens the protocol stipulates that the blocks in the longest chain become part of the authoritative blockchain. Only miners of blocks in the longest chain will receive the rewards attached to mining.

2.2 Selfish Mining Process

The Bitcoin protocol [3,11] does not specify when miners must publish their newly found blocks. The most basic strategy is to publish them immediately after they are mined. This is referred to as the *honest* strategy. Eyal and Sirer introduced a strategy for publishing newly found blocks called *selfish mining* [8].

Figure 2 illustrates one of the basic steps of selfish mining, intended to increase the number of forks. In this example, Miner C finds a block at `t=7s`. This block will be received by Miner A at `t=8s`, and by Miner B at `t=11s`[1]. All three

[1] Note, that in general, the network does not have access to a shared global time.

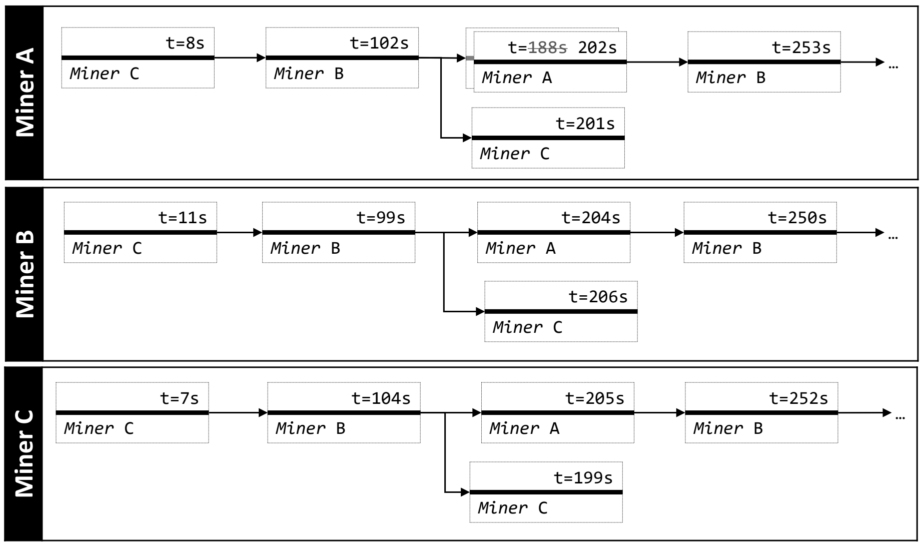

Fig. 2. Illustration of forks as races between different blockchains in a distributed network. Pool A is selfish miner, and postpones publication of a block found at `t=188s` until `t=202s`. This example omits for simplicity the hash values, nonces, and transactions.

miners will continue mining with this block. At `t=99s` Miner B finds a block and publishes it. It will be received by Miner A and C at `t=102s` and `t=104s`, respectively. Again all three miners will continue with this block. Up to this point, all miners employ honest mining.

Assume that Miner A employs the selfish strategy. If it finds a block at `t=188s`, it will not publish it immediately, but wait. If it receives a block by one of the other miners – in the example a block of Miner C at `t=201s` – it will publish its own block immediately, which intentionally creates a fork. The gamble is that its own block arrives at the others miners before the block of Miner C. In the example Miner B receives the block of Miner A before the block of Miner C, and thus continues mining the block of Miner A. If Miner B then finds a new block at time `t=250s` it will orphan the block Miner C found previously. Miner C's computational power from `t=104s` until `t=252s` – when it received the block of Miner B – was effectively wasted.

The question is if this can actually be beneficial for the selfish miner. In the example, Miner A forwent a certain reward for the block it found at `t=188s` to enter a race with pool C at `t=202s`. This looks superficially like a disadvantageous strategy. However, Figure 2 describes only one step of selfish mining, namely the step that intentionally introduces forks.

The following gives a full list of steps for the selfish miners. It assumes that the selfish miner always mines at the end of its own private chain.

1. **The selfish miner finds a block.**
   (a) **There is a fork, and both branches have length 1.** In this case, the selfish miner found a block to decide the race in its favour. The selfish miner appends the block to its private chain and publishes it. The selfish miner intends to orphan the single block in the public branch, and secure the rewards of its own branch.
   (b) **Otherwise.** The new block will be appended to its private chain, without publishing it. This includes cases where the private chain is two or more blocks ahead of the public chain.
2. **The selfish miner receives a block.** Provided that it actually increases the height of the public chain, the selfish miner will proceed as follows:
   (a) **If there is no fork.** This means the public and private chains are identical. The received block is appended to the public chain, and the public chain is adopted as the private chain. The other miner will receive the rewards.
   (b) **There is one unpublished block in the private branch.** The received block is appended to the public chain. The unpublished block is published. This is the scenario depicted in Figure 5.
   (c) **There are two unpublished blocks in the private branch.** The private chain is published. Since the public chain should still be one block behind, this would secure all rewards in the private branch for the selfish miner. After this, there is no fork.
   (d) **Otherwise.** This is the case when the selfish miner is more than two blocks in the lead. The selfish miner will publish the first unpublished block. While the private chain is at least two blocks ahead, the public branch and the portion of the private branch that has been published have the same height. To other miners, a race is ongoing, even though the selfish miner already has the blocks to decide the race in its favour.

To implement this strategy the selfish miner needs to maintain a record of the head of the public chain, of the head of the private chain, the head of the portion of the private chain that has been published, and the block where the private and public chain fork. It should be noted that the *public* chain is the local view that the selfish miner has of the blockchain. As discussed previously, in general, different miners may have different views.

Eyal and Sirer have shown that a miner using selfish mining will gain more rewards than would be proportional to their computational power, under the assumption that the other miners use the honest strategy. This result depends on the share α of computational power the selfish miner has in the network and the fraction γ of miners that adopt the block of the selfish miner in case of a fork. They discovered that selfish mining gives an increased reward if $(1-\gamma)/(3-2\gamma) < \alpha$. This means, for example, that if a quarter of the other nodes adopt the block of the selfish miner, i.e. $\gamma = 0.25$, then the selfish mining strategy will pay off if the network share satisfies $\alpha > 0.3$.

3 Uppaal Model

The Uppaal-SMC model consists of three templates: one for modelling the behaviour of an honest miner, one for a selfish miner, and one for modelling the propagation delay between miners. A fourth template is added to observe the blockchain, but this node does not take part in the protocol. This section will describe the important global variables and templates in detail.

Global variables and constants. The model includes two arrays of broadcast channels, `sendBlock[POOLS]` and `recvBlock[POOLS]`, for miners to send and receive blocks, where `POOLS` is the number of miners. A block is defined as a struct of the `height`, a bounded integer `BlockIndex`, and array `rewards[POOLS]`. If a miner with ID `id` mines a new block, it increments `height` and `rewards[id]`.

Global variable `syncBlock` is used as an auxiliary to copy blocks between processes. Important constants are integer `PDELAY` for the expected network delay, and integer array `POOL_RATES[POOLS]`, which contains for each miner the rate at which it finds blocks. The model uses as basic time unit 1 second; a rate of 1200 means that a miner finds on average one block every 1200 seconds.

Network links. The network link between any two miners is modelled as a one-place buffer with delay. For any pair of IDs `in` and `out`, the model will include one instance of the link template, depicted in Fig. 3. From the initial state it will synchronize on channel `sendBlock[in]` with Pool `in`, and copy the received block in global variable `syncBlock` to its local variable `blockBuffer`. It then enters the location to the right. In this location it will synchronize on channel `recvBlock[out]` with Miner `out` at a rate of 1 in `PDELAY` seconds. This transition will copy the value of the buffer to `syncBlock`. If it receives another block from Miner `in`, it will store that block in the buffer. Note, that the model will include for any pair of miners one link, i.e. for a network with 10 miners, 100 links, each with its own buffer.

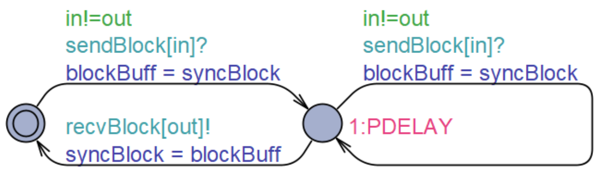

Fig. 3. Parameters of the link template are the `ID` of sender `in` and receiver `out`.

Honest mining. Figure 4 shows the template for an honest miner with ID `id`. It has a single location with two transitions. The first models successfully mining a block. It calls method `outputBlock` which increments the height of the head of its private chain and the rewards for itself. The other transition models receiving a block which calls method `updateBlock` which will adopt the new block if it improves on the height of the head of its private chain.

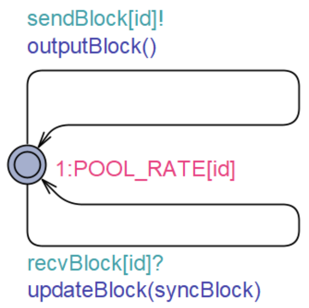

Fig. 4. The honest mining template has as parameter the id of the miner.

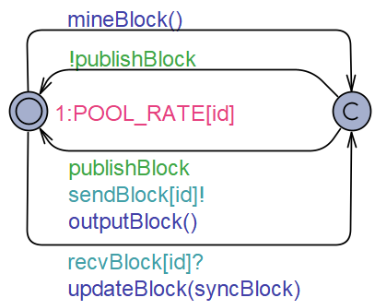

Fig. 5. The selfish mining template has as parameter the id of the miner.

Selfish mining. The selfish miner keeps a record of four blocks: the head of the private chain privateBlock, the head of the public chain publicBlock, the most recently published block publishedBlock, and the block where the public and the private chain fork, forkBlock. In addition, it uses a local Boolean publishBlock which encodes whether a block should be published.

The top-most edge models mining a block (case 1 on Page 5). It calls method mineBlock() at a rate of 1 in POOL_RATE[id] seconds. It decide whether to publish (part of) its private chain if it mines or receives a block.

The bottom-most edge models receiving a block (case 2 on Page 5). It synchronizes on channel recvBlock[id], and calls updateBlock which decides whether to append it to the private chain, or whether to publish a part of the private chain. It sets Boolean publishBlock, depending on whether a block should be published, or not.

The committed location in the mining template in Fig. 5 completes the process. If mineBlock() or updateBlock set publishBlock to false, the selfish miner returns silently to the initial location. If it was set to true method outputBlock will copy the block that is meant to be published to syncBlock, but also to publishedBlock and publicBlock. The code for methods mineBlock() and updateBlock is given in Listing 1.

System composition. The analysis in Section 4 uses a model with 10 miners and 100 links. It considers the 6 sets of network shares, as given in Table 1. If the model includes a selfish miner it would be Miner A. Miner B has a share of 20% in all experiments to make the results comparable. A share of 20% would correspond to finding a block once every 3000 seconds, assuming a network rate of one block every 600 seconds. These rates are simplified but still largely similar to the distribution of hash rates in the real world [4].

Uppaal-SMC simulated each scenario 1000 times for one day of simulation time, i.e. for 86400 seconds. The simulation of one single scenario takes about 80 seconds on an Intel Core i5-5200 with 2 cores at 2.2GHz.

```
void mineBlock() {//case 1
    if (privateBlock.height == publicBlock.height &&
        privateBlock.height-forkBlock.height == 1) {//case 1.(a)
        privateBlock.height++;
        privateBlock.rewards[id]++;
        outputBuffer   = privateBlock;
        forkBlock      = privateBlock;
        publishBlock   = true;
    }
    else{                                         //case 1.(b)
        privateBlock.height++;
        privateBlock.rewards[id]++;
        publishBlock   = false;
    }
}

void updateBlock(Block newBlock) {                //case 2
    if (newBlock.height>publicBlock.height) {
        if(newBlock.height>privateBlock.height){    //case 2.(a)
            privateBlock   = newBlock;
            forkBlock      = newBlock;
            publishedBlock = newBlock;
            publicBlock    = newBlock;
            publishBlock   = false;
        }
        else
        if(newBlock.height == privateBlock.height) {//case 2.(b)
            outputBuffer   = privateBlock;
            publishBlock   = true;
        }else                                     //case 2.(c)
        if(newBlock.height == privateBlock.height-1) {
            outputBuffer   = privateBlock;
            forkBlock      = privateBlock;
            publishBlock   = true;
        }
        else {                                    //case 2.(d)
            publishedBlock.height++;
            publishedBlock.rewards[id]++;
            outputBuffer   = publishedBlock;
            publishBlock   = true;
        }
    }
}
```

Listing 1. Essential methods of the selfish miner.

Scenario	A	B	C	D	E	F	G	H	I	J
#1	1%	20%	20%	15%	15%	10%	10%	5%	2%	2%
#2	10%	20%	20%	15%	15%	10%	5%	2%	2%	1%
#3	20%	20%	15%	15%	10%	10%	5%	2%	2%	1%
#4	30%	20%	15%	10%	10%	5%	5%	2%	2%	1%
#5	40%	20%	10%	10%	5%	5%	5%	2%	2%	1%
#6	50%	20%	10%	5%	5%	2%	2%	2%	2%	2%

Table 1. Network shares for different scenarios.

4 Analysis Results

This section will present for a 24h period the expected mining rewards and the expected number of orphaned blocks. The former allows a comparison with results by Eyal and Sirer, the latter with data obtained from the publicly available Bitcoin blockchain.

4.1 Mining Rewards

Fig. 6 depicts the height and rewards in different views of the blockchain. First is the number of *blocks mined* over the 24 hours period. It is around 144 blocks, as expected for a network that finds on average one block every 10 minutes.

Not all of these blocks will become part of the longest chain. Fig. 6 gives the blockchain height and the reward of the selfish and first honest miner, reward selfish and reward honest, respectively. Each of these three come in two versions depending on whether it is part of the private chain of the selfish miner, or the chain as known by the network.

These results show that as the network share of the selfish miner increases, it decreases the height of the blockchain, and increases the rewards for the selfish miner. For a miner with a 50% share the height is 89.4 and the reward 68.9, in the private blockchain of the selfish miner. In the blockchain of the first honest miner – Miner B in Table 1, who has a network share of 20% – the height is only 81.7, and the reward of the selfish miner is only 57.7. The difference is partly due to network delay, but mostly because the selfish miner has a buffer of 7.6 unpublished blocks in its private chain.

Fig. 7 translates these numbers to shares in the rewards. It also includes the nominal share these miners should achieve; the selfish miner proportionally to its network share, and the honest miner 20%. The results show that selfish mining becomes profitable once the network share of the selfish miner exceeds 30%.

To compare these to Eyal and Sirer's result, we need the probability that other miners adopt the block of the selfish miner above a competing block. It depends on two steps succeeding from the moment that the competing block is found. First the selfish miner has to receive the competing block before the

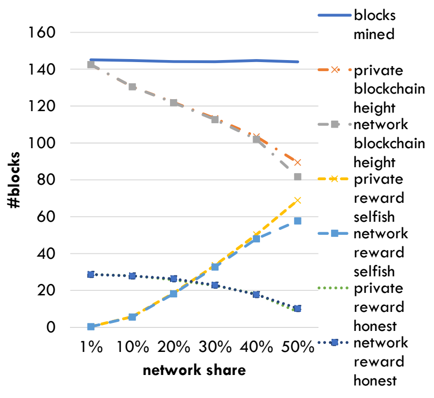

Fig. 6. Height and rewards of selfish and honest miner after 24 hours.

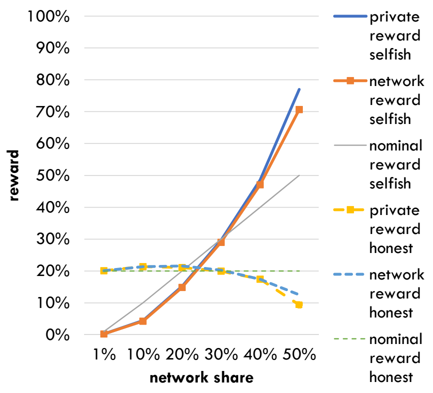

Fig. 7. Share of rewards for selfish and honest miner after 24 hours.

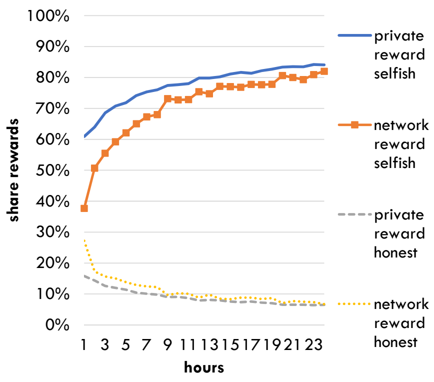

Fig. 8. Share of rewards per hour for the honest and selfish miner over 24 hours.

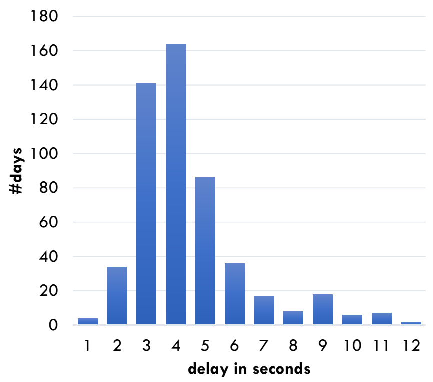

Fig. 9. Histogram of the propagation delays in the selected data set.

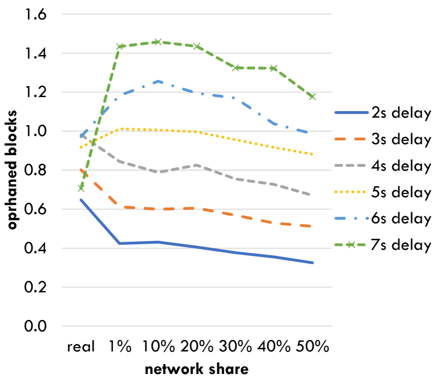

Fig. 10. Number of orphaned blocks in network w/o selfish miner after 24 hours.

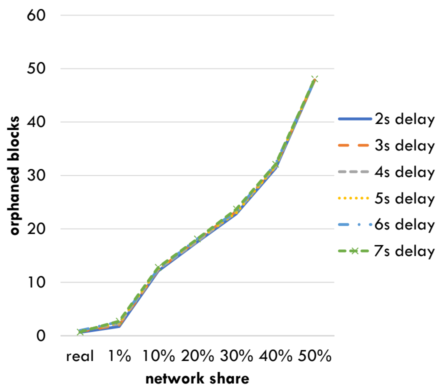

Fig. 11. Number of orphaned blocks in network with selfish miner after 24 hours.

other miners, (2) the block sent by the selfish miner in response has to arrive before the competing block. Given that in this model all delays use the same memoryless distribution both steps succeed with a 50% chance, giving an overall chance of 25%. Eyal and Sirer predict a threshold of 30% for this case, while in Fig. 7 the share of the 30% selfish miner is 29.6%.

Fig. 8 shows how this evolves over a 24 hour period for a selfish miner with a 50% network share. Initially, the selfish miner will appear to have a share that is below its 50% network share, as it is secretly mining blocks. As the day progresses its share will quickly exceed 80%, once it starts publishing blocks from its private chain.

All results of this section are based on a propagation delay of 4 seconds. For the results in this subsection, the propagation delay has little to no influence. The next subsection will discuss different propagation delays in more detail.

4.2 Orphaned Blocks

An essential aspect of selfish mining is to create forks such that other miners waste computational resources on blocks that are bound to be orphaned. To compare the models with data from the actually Bitcoin blockchain, we combined the data on orphaned blocks [2] with data on propagation times [1]. This gave 528 usable data points in the period from 18 March 2014 to 22 March 2017, i.e. days with both data on orphaned blocks and propagation times. Fig. 9 shows the distribution of days over different propagation times, rounded to the nearest integer second. This leaves us with a reasonable data set for propagation delays in the interval from 2 to 7 seconds.

Fig.10 shows the number of expected orphans if we have a network without any selfish miner. The figure includes, for reference, the number of orphans from the real data set, labelled *real*. The results show that as the delay increases, the number of orphans increases as well. With the exception of the data point for 7 seconds, the real data falls into the range given by the simulation.

This picture changes once we introduce a selfish miner as depicted in Fig. 11. Even a selfish miner with only a 1% network share leads to more orphans than for any scenario with only honest miners or the real data. This comparison suggests that there is no evidence in the real data of a prolonged presence of a selfish miner with a significant network share.

5 Discussion and Conclusion

In [8], Eyal and Sirer provide a pseudocode algorithm for selfish mining. The analysis uses a separate state transition model that captures the presence and length of a fork. Based on this model they manually derived state probabilities and expected rewards for each state. To validate the overall reward they use Monte Carlo Simulation. Their combination of models assumes a single view of the public chain where blocks are propagated instantly to provide estimates of the rewards a selfish miner can expect in the long run.

This paper presented a single unified modelling artefact. It also includes propagation delays, a block model with rewards, and a distributed blockchain. It does not separate the pseudo code from the transition probabilities, rewards, and the analysis of the evolution of the network over time. This allowed an automated analysis from the perspective of different participants, and compare these to the theoretical results by Eyal and Sirer, as well as to real-world data.

The analysis confirms that selfish mining becomes profitable for networks shares above 30%. The results of this paper show that the presence of a selfish miner may go undetected for the first few hours, but would be obvious after that. Future work would need to investigate how to identify a short-term attack on a blockchain. For this type of analysis, it is especially important to distinguish between the different views of the blockchain of different participants, as it is done in this paper.

All Uppaal-SMC models, simulation data and more detailed results will be available on `https://wwwhome.ewi.utwente.nl/~fehnkera/Q19`.

References

1. Bitcoinstats, network propagation times. `http://bitcoinstats.com/network/propagation/`. Accessed: 06-07-2019.
2. Bitcoin.info, number of orphaned blocks. `https://blockchain.info/charts/n-orphaned-blocks?timespan=all`, 2008. Accessed: 06-07-2019.
3. Bitcoin protocol rules. `https://en.bitcoin.it/wiki/Protocol_rules`, 2019. Accessed: 06-07-2019.
4. Blockchain.info, hashrate distribution. `https://blockchain.info/pools`, 2019. Accessed: 06-07-2019.
5. Coinmarketcap: Cryptocurrency market capitalizations. `https://coinmarketcap.com/`, 2019. Accessed: 06-07-2019.
6. Andrychowicz, Marcinand Dziembowski, S. M. D. M. Ł. Modeling bitcoin contracts by timed automata. In *Formal Modeling and Analysis of Timed Systems* (2014), M. Legay, Axeland Bozga, Ed., Springer.
7. Chaudhary, K., Fehnker, A., van de Pol, J., and Stoelinga, M. Modeling and verification of the bitcoin protocol. In *MARS 2015.* (2015), EPTCS.
8. Eyal, I., and Sirer, E. Majority is not enough: Bitcoin mining is vulnerable. In *FC 2014* (2014), LNCS 8437, pp. 436–454.
9. Fehnker, A., and Chaudhary, K. Twenty percent and a few days - optimising a bitcoin majority attack. In *NFM 2018* (2018), LNCS 10811, Springer.
10. Heilman, E., Kendler, A., Zohar, A., and Goldberg, S. Eclipse attacks on bitcoin's peer-to-peer network. In *SEC'15* (2015), USENIX Association.
11. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. `http://bitcoin.org/bitcoin.pdf`, 2008.
12. Neudecker, T., Andelfinger, P., and Hartenstein, H. A simulation model for analysis of attacks on the bitcoin peer-to-peer network. In *INM 2015* (2015).
13. Sapirshtein, A., Sompolinsky, Y., and Zohar, A. Optimal selfish mining strategies in bitcoin. In *FC 2016.* (2017), LNCS 9603, pp. 515–532.