

Blockchain Network Threats: The Case of PoW and Ethereum

Artem Shurov, Daniil Malevanniy(✉), Oleg Iakushkin, and Vladimir Korkhov

Saint-Petersburg State University,
7/9 Universitetskaya nab., St. Petersburg 199034, Russia
rukmarr@gmail.com, o.yakushkin@spbu.ru

Abstract. Developers do not properly take into account possible problems on the Internet. This fact can cause problems in the operation of blockchain systems. The subject of our analysis was Ethereum due to its short block creation time and, as a result, a strong reaction to faults in the network. We checked the possible threats from the miners during the unreliable work of the Internet.

For the study, we conducted many simulations of the work of Ethereum with different delays in the propagation of blocks and various mining power allocations. The results showed that if the duration of the troubles is more than 10 h, a miner with 26% can capture the entire Ethereum network, but this requires the intervention of people controlling other nodes. This is possible because it will be more cost-effective for them.

Keywords: Ethereum · Propagation delay

1 Introduction

A blockchain is a distributed database whose nodes store copies of it independently of each other. Information in such a database is presented in the form of an ever-growing list of ordered records, called blocks. The blocks are in a continuous sequential chain, built according to certain rules. Each block contains a set of different transactions, a link to the previous block and other supporting information [1]. Any node (miner) can create a new block by solving a cryptographic puzzle and receiving an award for it. Because of the distributed nature of finding new blocks, several nodes can add different blocks with the same number before they learn about the existence of other blocks, which leads to branching of the chain [2]. In order for all nodes to agree on a single-chain condition, they use a single strategy that selects the only chain of blocks that is correct for all. This strategy is called a consensus algorithm, and different blockchains use different consensus algorithms [3].

There are many researches aimed at analysing the network attacks against blockchains with PoW, but they are aimed at obtaining benefits through double

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 606–617, 2019.
https://doi.org/10.1007/978-3-030-24296-1_49

spending or rewriting transaction history [4–7]. Researches also show that Internet parameters affect the state of blockchain systems [8]. We investigated the possible behavior of miners (more precisely, mining pools) when problems occur on the Internet, if one of them also wants to benefit from the situation.

Our investigation reveals that large mining pools in the Ethereum can capture the entire network (as if they have 50% + power) using the arising delays in the propagation of new blocks between several large parts of the network. But this requires decision-making by a person, otherwise the honest mining of all nodes merely decreases the reliability of the network.

2 Why Ethereum

The complexity of solving a cryptographic puzzle is different for various implementations of the blockchain and is chosen in such a way that the block is created at certain intervals [9]. For example, for the popular blockchain platforms, the time intervals are as follows:

1. Bitcoin - 10 min
2. Litecoin - 2.5 min
3. Dogecoin - 1 min
4. Ethereum - about 15 s.

The data above show that the Ethereum blockchain has a small block finding time and most often faces the concept of forking - when several blocks are located at the same chain height. To reduce the negative impact of this factor, it implements a modified GHOST algorithm [10], the main idea of which is the inclusion of so-called uncle-blocks in the chain.

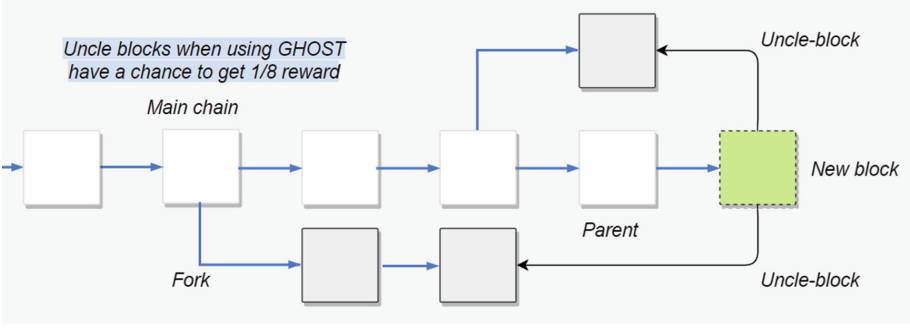

Fig. 1. Modified GHOST implementation.

Uncle-block is a block that is not part of the main chain, but is taken into account by it (Fig. 1). If uncle-block is included in the new block, miners get rewarded for work: 1/32 of the regular award for the miner of the new block and

1/8–7/8 to the miner of the uncle-block (depends on how many generations back the uncle forked). A miner can include up to 2 uncle-blocks, and they must be up to 7 generations back in the main chain.

Different consensus algorithms implemented in various blockchains choose the main chain differently. In bitcoin, the longest branch is selected. This strategy, however, has its drawbacks, since it simply wastes all the blocks that are not present in the main chain [11]. If an attacker can find new blocks quickly enough to create his local chain faster than the rest of the network, then ultimately the entire network accepts its chain as the main one. In particular, by increasing the time propagation of blocks in the network, you can increase the number of orphan blocks and, consequently, slow down the growth of the longest chain of the network. Such a possibility poses a serious risk for the blockchain, since the attacker does not even need to own most of the computing power of the network to be able to control the network [12].

In Ethereum, the chain is selected on the basis of the total complexity of the chain—the sum of the complexities of all the blocks in this chain. The GHOST protocol should solve the problem of delays in the propagation of a new block in the network. In the description of its implementation in Ethereum it is said that uncle-blocks participate in the calculation of the total complexity [13]. But in the client's geth code, when calculating the total complexity, only the block itself is taken into account and the uncle-blocks that he added are uncounted, but uncle-blocks are nonetheless awarded. This is also confirmed by a simple check of the numbers in the blocks created. Perhaps this is done for security reasons: the inclusion of uncle-blocks in the calculation of the total complexity complicate the process of choosing a main chain and possibly creates conditions for micro-attacks (sometimes add an attacker block, although it was found later) [14]. For example, the miner honestly found the block could not get data about the possible uncle-block and spread its block, another miner who found the block a few seconds later and already knew about the uncle-block, adds it to his block, thereby making a chain with his new block more, hence its chain becomes the main.

The difficulty of finding a new block in the network of Ethereum is recalculated every new block according to the formula [15]:

$$diff = (parentDiff + (parentDiff/2048 * max(2\ if\ len\ parent.uncles\ else\ 1 - ((timestamp - parent.timestamp)\ //\ 9), -99))) + 2^{periodCount-2}$$

where $periodCount = (numberCurrentBlock - bomdDelay)/100,000$

But despite all this, uncle-blocks do not participate in the calculation of which block has the largest total proof of work backing it, although this is the main part of the GHOST protocol. Of course, they are taken into account when calculating the new complexity of finding a block, but this contribution does not compare with the calculation that should be.

Because of the fact that the Ethereum needs a small amount of time to find the block and also considering that the GHOST protocol does not use uncle-blocks in calculating the total chain complexity, delays in the propagation of the blocks can adversely affect the reliability of the created chain.

3 Simulation of Ethereum Network

To analyze the effect of network latency on the Ethereum, a python script was created that simulates Ethereum network operation. The parameters of this simulation are:

1. List of all miners (part of them in total network)
2. New block propagation time.

Also, the mining process is simplified: a number is chosen at random, and it is determined whether it is small enough to find a new block (according to its complexity).

Implemented algorithm:

Listing 1.1. The simulation algorithm

```
miners = create miners with network power percentage

for 1 to the simulation time:
    for each miner in miners:
        r = random number
        if r is small enough to find a block:
            block = creates a new block
            adds this block to miners chain
            t = current time + delay
            add this block to the queue with t

    for all blocks in the queue with the current time:
        for each miner in miners:
            if block can be added:
                adds a block to its chain
```

The limitations of this simulation are:

1. Time step = 1 s
2. The block propagation delay is the same for all miners.

The simulation starts with block number 7,360,216 and the rest of the data corresponds to this point in time: the mining rewards, the difficulty of finding the block, the average block propagation time, and the average network hashrate [16].

Comparison of the results of the simulation and this network can be found in Table 1:

Errors are most likely caused by the simplified concept of mining and the constant power of the network and mining pools.

These graphs show the results of simulations of 24 h of Ethereum network with miners, whose power are related to the power of pools in the real network. Pool - miner association for finding blocks, motivated by the receipt of a constant small profit of a relatively large rare win.

Table 1. Reality vs Simulation

Data for 24 h	Reality	Simulation
Number of all blocks	6400	6350
Number of uncle-blocks	450	500
Uncle-rate	7%	8%
Propagation time	0.5 s	1 s

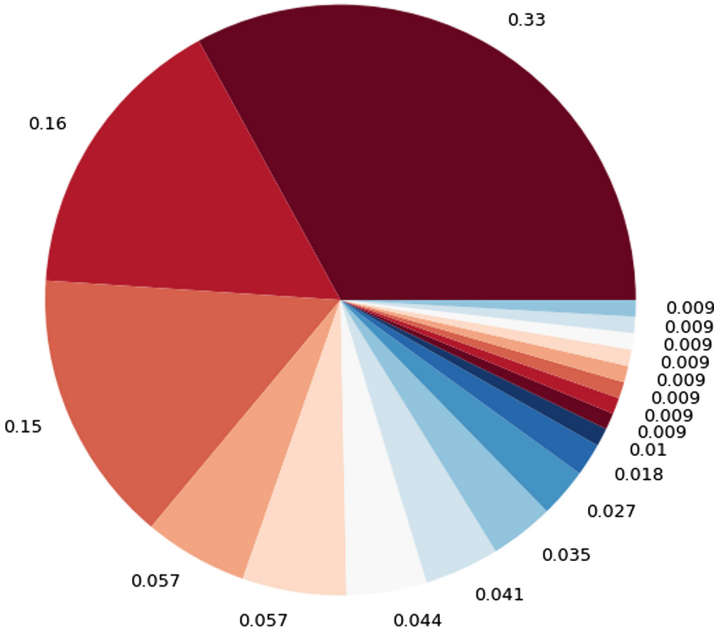

Fig. 2. Percentage distribution of mining pools.

The first graph (Fig. 2) shows the distribution of power miners in the network. The second graph (Fig. 3) shows the general condition of the chain:

1. How many blocks does the chain take into account
2. How many blocks in main chain
3. The number of uncle-blocks in the chain = (1. - 2.).

The third graph (Fig. 4) shows the results of work for each miner separately:

1. How many blocks did he find
2. How many of these blocks are included in the main chain
3. How many blocks are not taken into account at all (orphan)
4. The number of uncle-blocks is (1. - 2. - 3.).

The fourth graph (Fig. 5) shows the number of uncle-blocks and the number of orphan blocks (valid block, but not taken into account in the chain). The graph shows that the number of orphan blocks grows linearly relative to the propagation delays.

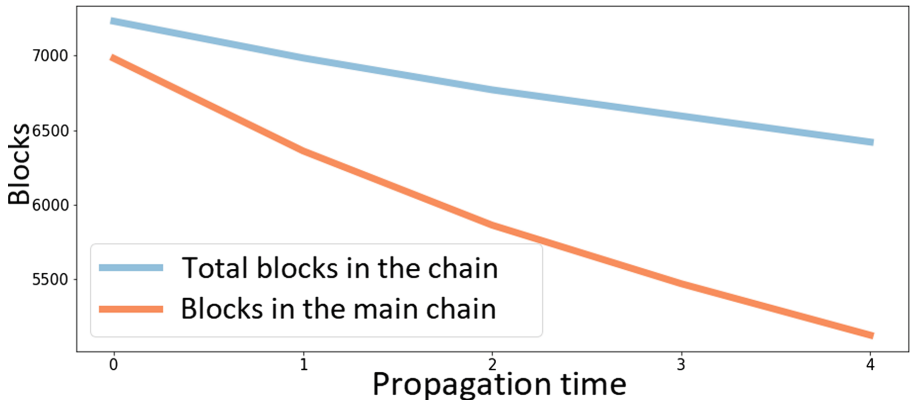

Fig. 3. Main chain.

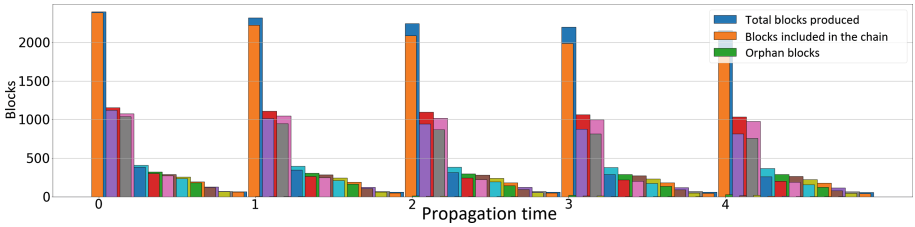

Fig. 4. Blocks created by mining pools.

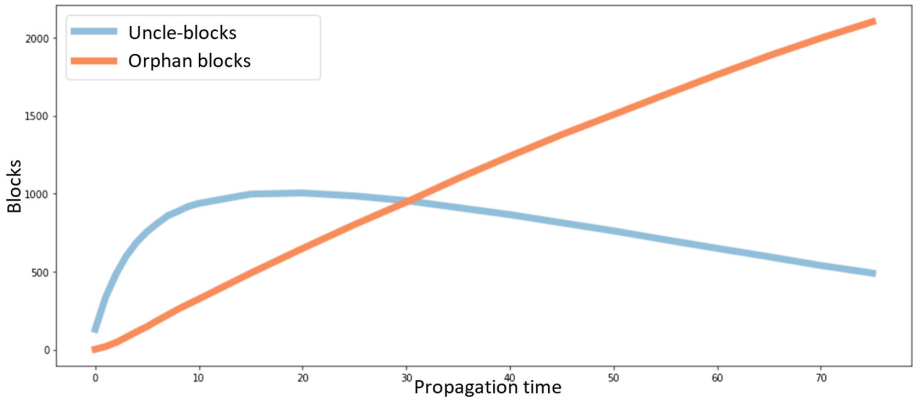

Fig. 5. Loss of Ethereum network.

Already in the second graph (Fig. 3), you can see how strongly the delays affect the Ethereum network: an increase in the block propagation time from 1 s to 3 reduces the number of blocks in the main chain by 14% and increases the number of uncle-blocks by almost 2 times.

The simulation is quite accurate, so using this simulation, we research the operation of the network with different propagation time and network power distributions.Based on the information obtained earlier, it can be concluded that delays can affect the Ethereum network quite a lot.

4 Possible Attack

Suppose that an attacker have the opportunity to influence the operation of the Internet Network: a break in communication, delays and various noises. Possible attack version:

The entire Ethereum network divided into two parts: 51% and 49% of capacity (51% - A part, 49% - B part). The propagation time of blocks between these parts rises to a critical value that must be calculated (hereinafter, we consider delays from 0 to 200 s). The attacking side has 26% (C-part) of the total power in the A part, that is, within the A part, the attackers have 51%, the remaining 25% belong to the D part. The network diagram during the attack is depicted in Fig. 6.

This threat variant is slightly different from the 51% attack [17]: which is that, having 50% + power, you can create a chain of blocks of greater complexity than the other part of the network. In the Ethereum consensus algorithm, like many other blockchain implementations, a chain of greater complexity is always selected (the creation of which used more power). Most complexity in different algorithms is understood in its own way, for example, for Bitcoin it is a chain of greater length, and for Ethereum it is a chain with a greater total complexity of blocks in it.

The network attack is based on the fact that part of the created valid blocks will not have time for propagating across the blockchain network and will not

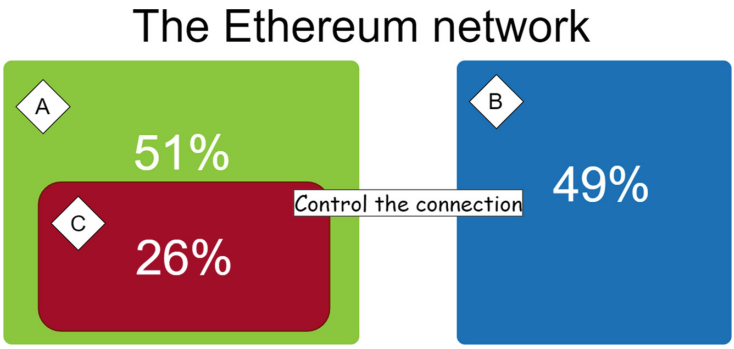

Fig. 6. Network partitioning.

be included in the main chain, despite the fact that the blocks are absolutely correct and can be included in the chain. The meaning of a possible attack is that attackers can control the entire Ethereum network by having far from 51% power, and even if there are mining pools larger than them. The goal is to earn more ETH when mining than usual, as well as impose their fees and manage the execution of transactions.

Firstly, attackers must force the remaining players in part A (D part) to follow their rules. Secondly, together they must get a chain of greater complexity than a chain of the B part. To do this, it is necessary to consider a number of cases and show that these requirements can be met.

Obviously, if A part will be mine without taking into account the chain created by B part, its chain will be the main one, since it has more power. But attackers own only part C (26%), and they need to convince the D part to mine together, without working on the chain B part. For this, the economic benefit for D to work on the chain together with C must be higher than together with B. In this case, we do not consider the bribing option.

5 Analysis

To find out the economic benefits of the D part, two cases were simulated with propagation delays from 0 to 200 s:

1. 25% and 49%
2. 25% and 26%.

Next, we take a closer look at the first situation at 25% and 49% of the power of each part, considering them as pools with corresponding powers. Figure 7 shows the characteristic of the chain relative to each pool:

1. How many blocks each pool produced
2. How many blocks were included in the main chain
3. The number of orphan blocks
4. The number of uncle-blocks is not specified, but it is (1. - 2. - 3.).

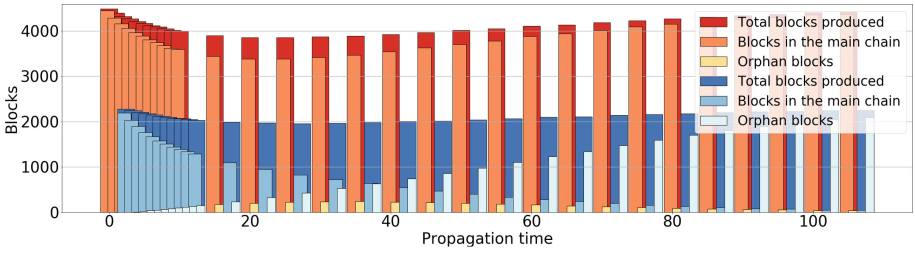

Fig. 7. Mining pools: 25% and 49%.

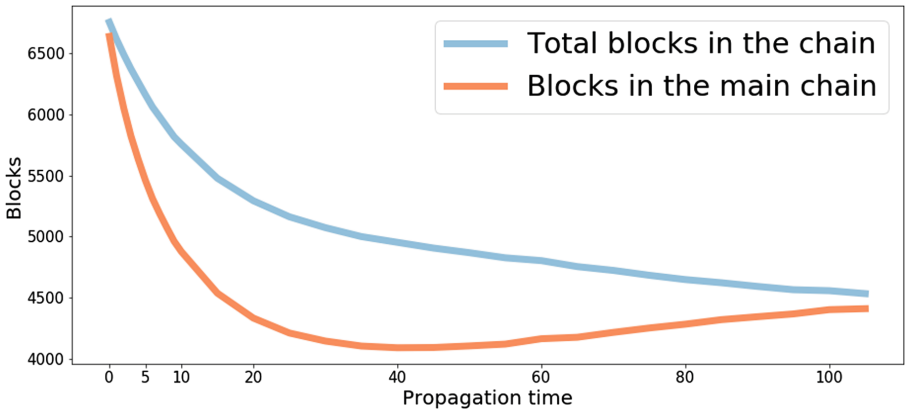

Fig. 8. Mining pools: 25% and 49%.

Figure 8 shows the general condition of the chain:

1. How many blocks does the chain take into account
2. How many blocks in the main chain.

Delays between two parts of the network can be controlled, but delays within each part of the network remain the same. Therefore, we can consider in more detail what will happen to the D part during mining only inside the A part, and mining with the B part (then we control the delay between them).

Below is a graph for the D part (Fig. 9), with its two decisions:

1. Mine with 26% in A parts (brown)
2. Mine with 49% in the B part (blue).

The graph shows that the first solution is most beneficial for the miner (D part), even during normal network operation. As a result: either he creates about 2700 blocks for a chain, or less than 1000 already with a delay of 20 s (1700 blocks is about $ 230,000). The difference almost three times should represent a weighty argument for making a decision of the D part in favour of the attackers.

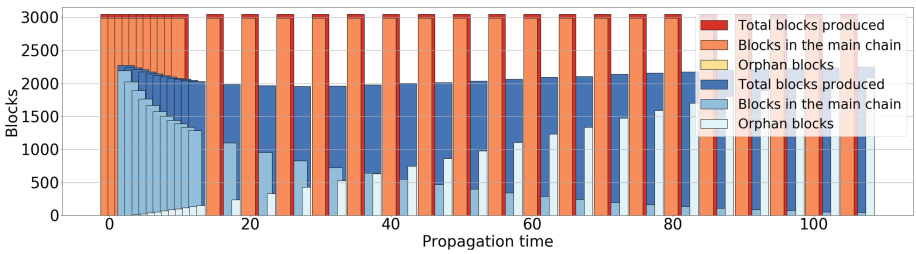

Fig. 9. Two decisions for 25%.

It should also be noted that in the presented simulation there were miners with powers of 49% and 25% represented by a single pool, which cannot be in reality. And their power will be a little less, which should not affect the results in any way. To prove this, another option was researched: 40% and 25%. As a result, the graphics coincided qualitatively, differing only in a slightly smaller number of blocks in all indicators.

It remains to be determined whether the chain made by A part (51%) is larger than that of B (49%). This is not entirely obvious because of the fact that the complexity of finding a new block is constantly recalculated and it is worth checking what happens if the power of the entire network drops twice (since both parts will start mining different chains independently of each other).

Several cases were considered, and all of them are presented in Table 2. The data was collected for the network operation within 24 h, the first column shows the data of the A part, which is taken as the basis for comparison. The first line shows the number of blocks in the main chain, and the second - how many blocks you need to create in order to achieve the same chain complexity as the A part. Because of the fact that the complexity of finding a new block is constantly recalculated, it is impossible to draw conclusions about the power of the chain simply by the number of blocks in it. The difference in blocks is calculated as the difference in the total chain difficulty divided by the complexity of the last block created.

Table 2. Chain comparison

Data for 24 h	$25+26$	49	$10\times4+9$	$32+10+7$	$32+17$	7×7	1×49
Blocks in the chain	5786	6127	5518	5732	5769	5468	5380
Blocks up to 25% + 26%	0	−40	428	228	200	448	541

In a situation where in the B part there is only one miner, who owns the entire power of this part, the attack fails. But having such large pool is unlikely and threatens the entire network due to the 51% attack that was mentioned earlier. In the real situation, when the B part consists of several miners, the network attack is successful. Of course, the 25% D parts also consist of several miners, but in the general distribution of forces is maintained as 51% and 49%, respectively, and makes the chain A parts more powerful than the other chain.

After it has been shown that the fulfillment of all the conditions for an attack can be fulfilled, it is necessary to know how long it takes to carry out an attack. That is, how much minimum time there should be delays between the two parts of the network. For this, simulations of two parts were performed:

1. 26%, 10%, 8%, 5%, 2% (A)
2. 32%, 10%, 7% (B).

As a result, we compared the total complexities of the chains created by parts A and B and counted the number of blocks for which both chains differ. A significant difference in blocks for a successful attack is achieved with 10 h of work of both parts separately from each other and is equal to about 80 blocks. The attack for more than 10 h always becomes successful.

6 Conclusions

In this paper, we show that large mining pools can take over the network, provided that the entire blockchain network is divided into two or more parts between which the delays will be more than regular. But it all depends on the distribution of power during the separation. Not necessarily the most powerful pool can take control of the entire network. There are several main conditions for network capture during network attacks:

1. The attacker needs to be in the most powerful part of the network
2. They need to have there more than 50% + power
3. It is desirable to have not much more than half the power
4. Delays between the two parts of the network should be such that it becomes more cost-effective for each part to work independently of each other
5. Necessary intervention in the normal operation of independent network nodes (to eliminate interaction with another part of the network).

Also, the simulation results show that even the slightest malfunction in the global Internet network leads to a deterioration in network reliability. This may mean that when creating Ethereum, it was assumed that the Internet connection will work stably all the time.

One of the main disadvantages of our simulation is that we do not take into account the behavior of the network operation within each part of the Internet Network when global problems occur. We also consider miners as absolutely identical entities, in reality, the network of Ethereum consists of nodes that use a huge number of different clients to work in the network [18].

In further work on this problem, it is necessary to conduct modeling with uneven delays between parts of the network [19] and various clients for the operation of the Ethereum nodes.

References

1. Pilkington, M.: Blockchain Technology: Principles and Applications. Research Handbook on Digital Transformations, pp. 225–253. Edward Elgar Publishing, Northampton (2017)
2. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: IEEE P2P 2013 Proceedings
3. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data

4. Natoli, C., Gramoli, V.: The Balance Attack Against Proof-Of-Work Blockchains: The R3 Testbed as an Example (2016)
5. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking Bitcoin: Routing Attacks on Cryptocurrencies (2017)
6. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse Attacks on Bitcoin's Peer-to-Peer Network (2015)
7. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack (2015)
8. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the Security and Performance of Proof of Work Blockchains (2016)
9. Kraft, D.: Difficulty control for blockchain-based consensus systems. Peer-To-Peer Netw. Appl. **9**(2), 397–413 (2015)
10. Sompolinsky, Y., Zohar, A.: Secure High-Rate Transaction Processing in Bitcoin (2015)
11. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 515–532. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_30
12. Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 436–454. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_28
13. Ritz, F., Zugenmaier, A.: The impact of uncle rewards on selfish mining in Ethereum. In: 2018 IEEE European Symposium on Security and Privacy Workshops
14. Werner, S.M., Pritz, P.J., Zamyatin, A., Knottenbelt, W.J.: Uncle traps: harvesting rewards in a queue-based Ethereum mining pool (2019)
15. A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/Ethereum/wiki/wiki/White-Paper
16. "blockchain". https://www.blockchain.com/en/
17. Beikverdi, A., Song, J.: Trend of centralization in Bitcoin's distributed network. In: 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)
18. Kim, S.K., Ma, Z., Murali, S., Mason, J., Miller, A., Bailey, M.: Measuring Ethereum Network Peers (2018)
19. Malevanniy, D., Iakushkin, O., Korkhov, V.: Simulation of distributed applications based on containerization technology. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11620, pp. 587–595. Springer, Cham (2019)