The Knowledge Engineering Review, Vol. 35, e24, 1 of 19.
doi:10.1017/S0269888920000193

On the impact of architecture design decisions on the quality of blockchain-based applications

DIEGO MARMSOLER and LEO EICHHORN

Department of Computer Science, Technische Universität München, München, Germany
e-mails: diego.marmsoler@tum.de, leo.eichhorn@tum.de

Abstract

In software architectures, architectural design decisions (ADDs) strongly influence the quality of the resulting software system. Wrong decisions lead to low-quality systems and are difficult to repair later on in the development process. As of today, little is known about the impact of certain ADDs for the development of architectures for blockchain-based systems. Thus, it is difficult to predict the outcome of certain ADDs when developing architectures for such systems. In the following, we propose a simulation-based approach for blockchain architectures in which the impact of certain ADDs on certain quality attributes can be simulated. To this end, we first implemented a simulation environment for blockchain architectures. The simulation environment was then used to execute a series of experiments from which we derived a set of hypotheses about the impact of certain ADDs on quality attributes for blockchain architectures. Finally, we tested the hypotheses using statistical analyses and derived an empirical model for blockchain architectures based on the outcome of the analysis. The model can be used by architects to predict the effect of certain decisions in the design of blockchain architectures before implementing them.

1 Introduction

With the emergence and increasing popularity of decentralized cryptocurrencies such as Bitcoin (Nakamoto, 2008) and Ethereum (Buterin, 2013), blockchain architectures (Xu *et al*., 2017) have become more important. Their use seems promising also for other domains, such as the medical one (Azaria *et al*., 2016), land management (Chavez-Dreyfuss, 2016), or even identity management (Yurcan, 2016). The design of blockchain architectures requires many different architectural design decisions (ADDs) to be taken, each of which with a strong impact on the quality of the resulting system (Staples, 2017). Thus, wrong design decisions may have dramatic consequences on the resulting system and once the system is implemented, such decisions are difficult to repair (Garlan, 2000).

As of now, however, little is known about the impact of decisions in the design of blockchain architectures on the quality of the resulting system. Thus, it is difficult, if not to say impossible, to predict the outcome of certain ADDs on the quality of the resulting blockchain system before implementing it.

To address this problem, we propose a *simulation-based* approach to study the effects of certain ADDs on certain quality attributes for Bitcoin-based *Proof of Work (PoW)* blockchain architectures (Nakamoto, 2008). Figure 1 depicts an overview of our approach: as a first step, we implemented a framework to simulate certain properties for such architectures with certain ADDs. To this end, we studied existing literature to identify classes of ADDs which might influence certain quality attributes of blockchain architectures. Then, we implemented a simulation environment which can simulate properties for PoW-based blockchain systems built according to certain ADDs. In the second step, we used the

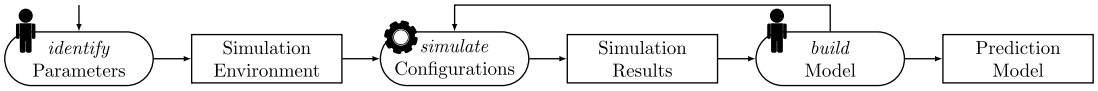


Figure 1 Simulation-based approach showing manual (stick figure) and automated (gear wheel) activities (rounded rectangles) and corresponding artifacts (rectangles)

framework to simulate the effect of certain ADDs on these properties. To this end, we configured the framework according to a set of ADDs, ran the simulation, and recorded the outcome. In the last step, we used the outcome of the simulations to propose an empirical model which relates ADDs with quality attributes. This is an iterative step: we first 'guessed' a potential model based on our observations of the simulation and then we run the simulation again to test the model's predictions against the simulated output.

The major contribution of this work is twofold: (1) we provide a simulation environment which can be used to simulate the impact of five classes of ADDs on two types of quality attributes for blockchain systems. (2) We provide an empirical model which can be used to estimate ADDs based on desired quality attributes for blockchain systems. This paper is an extension of previous work presented at the 3rd Symposium of Distributed Ledger Technology (SDLT) (Marmsoler & Eichhorn, 2018). To this end, the paper provides additional results as well as a detailed analysis of all the results. Moreover, the paper presents a thorough discussion about limitations of the approach and a detailed review of related work.

In the following, we first provide some background for our work (Section 2). Then, in Section 3, we describe our simulation-based approach, and in Section 4, we present the results obtained from our experiments. In Section 5, we interpret these results, and in Section 6, we then propose an empirical model based on our results. Finally, we discuss limitations of our work (Section 7), discuss related work (Section 8), and conclude in Section 9 with an outlook which leads to a discussion of future work.

2 Background

In the following, we provide some background for our work.

2.1 *Blockchain*

A blockchain is a public[1], distributed database used to record, identify, and verify contracts, transactions, or other shared data between multiple parties. The resulting data records are stored in a continuously growing list, which is locally maintained and updated by each individual member (node) of the network. Entries of the list (blocks) are cryptographically linked by including a hash of the previous block as a unique identifier in each newly added entry. More specifically, altering contents of a block changes its unique identifier, forcing a recalculation of every following block currently in the list in order to retain integrity of the ledger. Newly created blocks are broadcasted to all members to keep local blockchain copies synchronized. By adhering to a distributed consensus protocol, the participating nodes validate potential extensions of their blockchain copy in a peer-to-peer manner, thereby eliminating the need for an intermediary, trusted authority.

2.2 *Proof of work*

Since our work is based on the PoW consensus algorithm, we discuss this algorithm in more detail. PoW is a consensus algorithm which requires miners to solve a cryptographic puzzle, in order to mine a new block of the blockchain. It is based on the assumption that the longest chain (in terms of number of confirmed blocks) is the 'correct' one, since it requires the most computational power to be computed. Although it requires a lot of computation power (and thus electricity), it is still widely used in contemporary blockchain architectures, such as Bitcoin (Nakamoto, 2008).

[1] In this paper, only permissionless blockchains are considered. For permissioned blockchains, see Vukolić (2017).

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

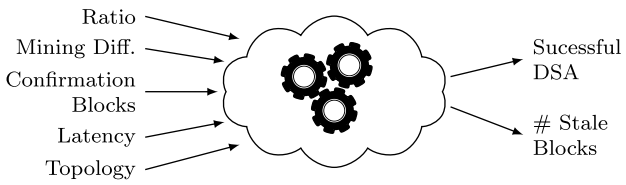


Figure 2 Simulation-based approach to blockchain architectures

2.3 Double-spend attack

Double-spend attacks (DSAs) are attempts to modify already confirmed entries of a blockchain. To this end, an attacker (or a group of attackers) tries to work on an alternative extension of the blockchain, which they hide from the rest of the network. After a block is confirmed by the network, the attackers release their alternative extension of the chain which does not contain the confirmed block (Rosenfeld, 2014; Pinzón & Rocha, 2016). In order to be consistently successful, DSAs usually require the attackers to own more than 50% of the network's computing power (Rosenfeld, 2014; Pinzón & Rocha, 2016; Grunspan & Pérez-Marco, 2017; Ozisik & Levine, 2017). A blockchain architecture's resistance against DSAs is an important property of such architectures. It is influenced by different design decisions, such as the size of the attacking network, the number of required confirmation blocks, the difficulty of extending the blockchain, as well as the topology and latency of the underlying networks.

2.4 Stale blocks

When using PoW consensus algorithms, miners usually compete to solve the next block and sometimes it might happen that different miners solve different blocks at the same time. This leads to two alternative versions of the blockchain. After a new block is solved on top of one of the versions, this version is assumed to be the correct one and the block of the other chain is considered a stale or orphaned block (in the following denoted as SB). SBs actually represent a 'waste' of computing power and ideally their number should be kept low (Decker & Wattenhofer, 2013; Courtois & Bahack, 2014).

3 Simulation approach

Figure 2 depicts an overview of our approach for simulating blockchain architectures: the framework is configured using five types of parameters, which were identified by reviewing literature concerning blockchain architectures (Nakamoto, 2008; Decker & Wattenhofer, 2013; Rosenfeld, 2014; Antonopoulos, 2017) and are assumed to influence the amount of successful double-spends during operation. It then simulates a blockchain system with a corresponding architecture and records the occurrence of successful DSAs as well as the number of SBs. In the following, we provide some details about the simulation. The complete framework is available online and can be downloaded from Eichhorn (2018).

3.1 Simulation model

The simulated system is based on the PoW consensus algorithm (Section 2.2). Its architecture consists of a network of nodes, each of which stores a private copy of the blockchain. Nodes may receive copies of other nodes' blockchains at every point in time, while they are also able to broadcast copies of their own blockchain versions themselves. Moreover, a node can try to mine a new block, in which case it generates random numbers until it obtains one which is below a certain target value. In order to simulate DSAs, we distinguish between *trusted* and *untrusted* (or attacking) nodes. Trusted nodes strictly adhere to the protocol: (1) they always take the longest blockchain as the 'correct' one. Thus, they replace their own copy with a new blockchain, whenever a longer one is obtained. (2) They always try to extend their blockchain and new blocks are mined on top of it. Untrusted nodes, however, may deviate from the protocol in two ways: (1) They may not replace their copy of the blockchain with a longer chain obtained from the trusted network. (2) They may also drop elements from the chain. Figure 3 shows an exemplary

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

Table 1 Possible simulation parameters

Par.	Description	*p*-Value	Default
R	Ratio of untrusted nodes, compared to total nodes	[0,1]	$^1/_3$
T	Mining difficulty target	[0,1]	0.00001
C	Number of confirmation blocks	Integer	6
	Network latency		
L_T	Average latency in the trusted network (ms)	Integer	10
L_A	Average latency in the untrusted network (ms)	Integer	10
L_C	Average latency between trusted and untrusted network (ms)	Integer	10
	Network topology		
D_T	Ratio of existing edges in the trusted network, compared to the maximum amount of possible edges	[0,1]	0.8
D_A	Ratio of existing edges in the untrusted network, compared to the maximum amount of possible edges	[0,1]	0.8

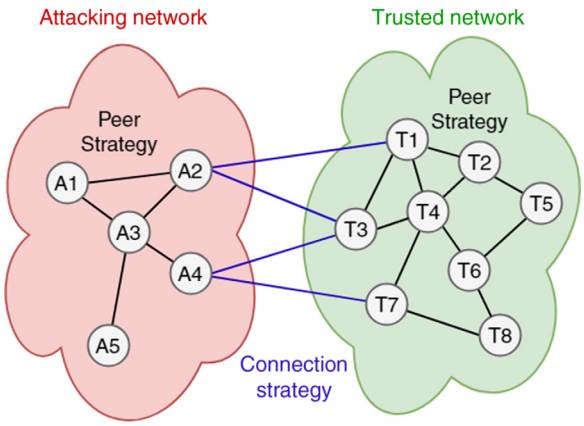


Figure 3 Exemplary simulation network of five attacking and eight trusted nodes

simulation network, consisting of two subnetworks: a trusted network consisting of eight trusted nodes and an attacking network which consists of five untrusted nodes.

3.2 Input parameters

Currently, the simulation supports five types of parameters listed in Table 1. Parameter R is a value of the interval 0 to 1, representing the networks' estimated ratio of untrusted nodes to total nodes. For example, if we expect 500 trusted nodes and 200 attacker nodes, R would be set to 2/7. Parameter T is also a value between 0 and 1, which is used to set the difficulty of mining a new block. In the simulation, miners will generate random numbers between 1 and 0. Thus, whenever a number below T is obtained, a new block can be created. C is the number of required confirmation blocks. A simulated DSA is only possible if it is able to modify a block which was already confirmed by C confirmation blocks. We provide three parameters to simulate network latency: L_T, L_A, and L_C to set the average latency between trusted nodes, attacking nodes, and between the two subnetworks, respectively. Finally, network topology can be influenced with parameters D_T and D_A, representing the number of edges between trusted nodes and untrusted nodes, respectively.

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

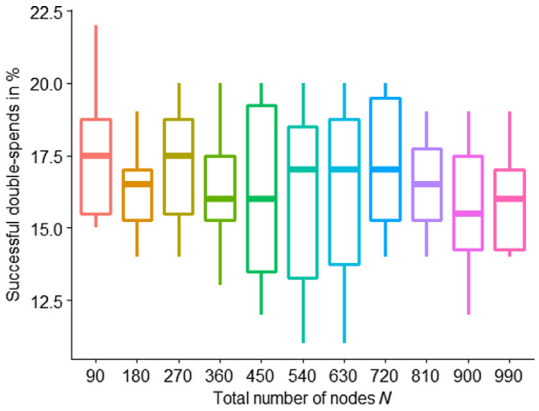


Figure 4 Successful DSAs for different sizes of the network

3.3 Simulation outcome

Currently, two aspects of a blockchain architecture can be simulated: successful DSAs and the number of SBs. Thereby, a DSA is assumed to be successful, whenever the attacking network is able to modify an entry in the chain, which was already confirmed by a certain number of confirmation blocks. By calculating the percentage of successful attempts, a measure describing the simulated architecture's resistance against DSAs is created. The number of SBs, on the other hand, is obtained by measuring the amount of blocks in both subnetworks that were added to obsolete and thereby shorter blockchain copies. This value is again described as a percentage of all blocks mined in total.

4 Results

To identify factors affecting a blockchain architecture's resistance against DSAs, we will now explore the effects of the input parameters discussed in Section 3 on successful DSAs (in the following abbreviated as PDS) and amount of produced stale blocks (PSBs). For every experiment, we performed 100 simulations with an ϵ value of 0.00001. Unless otherwise stated, parameters that are not being tested during an experiment will remain fixed to their default values provided in Table 1. The sampled data, representing successful DSAs, will be gathered in form of scatter plots. To provide a general trend of the data, simple exponential models of the form

$$M(p) = Ae^{Bp} \tag{1}$$

are fitted using nonlinear regression functions (Nash, 2016) and visualized in terms of graphs.

4.1 Impact of network size on DSAs

To determine whether the total number of nodes in the network has an effect on the probability of a successful DSA, multiple simulations have been run, while gradually increasing the number of nodes in both networks: the trusted and the attacking network. By increasing the number of attacking nodes by 30 and the number of trusted nodes by 60, the ratio between attacking and trusted nodes remains at a constant level of $\frac{1}{3}$. Runs for each treatment have been replicated five times. Figure 4 represents the sampled data using a box plot, visualizing treatment levels up to 990 nodes in total. As can be observed from the figure, the average PDS is always in between 15% and 18%.

4.2 Impact of ratio on DSAs

To analyze the impact of the ratio between trusted and attacking nodes on the probability of successful DSAs, we simulated a blockchain architecture with different values for the ratio R of untrusted notes. Figure 5 depicts the outcome of these experiments and shows that the PDS grows exponentially with an

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

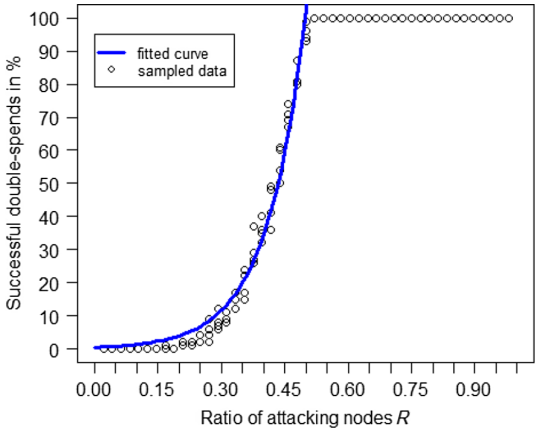
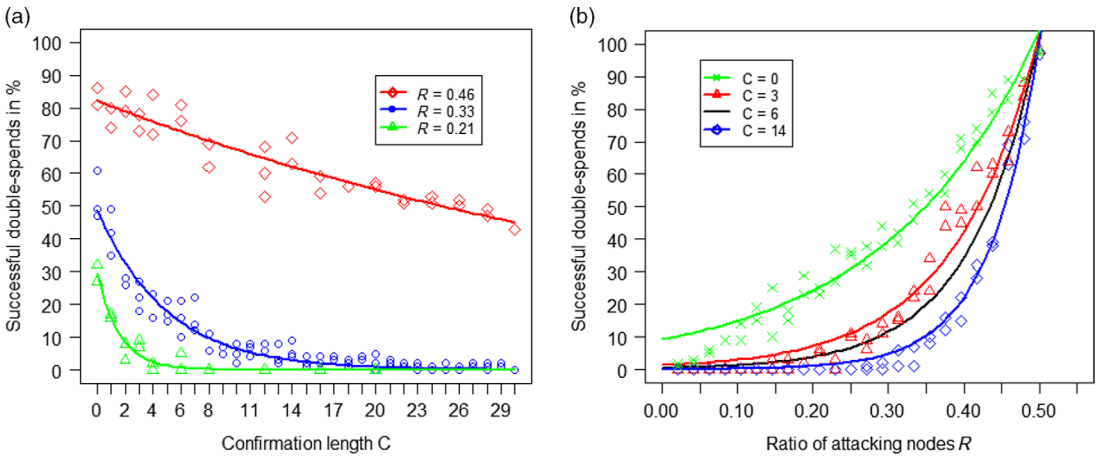


Figure 5 Successful DSAs for different node ratios



Impact of conf. length on successful DSAs. Impact of node ratio on successful DSAs.

Figure 6 Successful DSAs for different numbers of confirmation blocks

increasing proportion of attacking nodes. Moreover, the data confirm the prevalent assumption that for blockchain architectures with a ratio of $R > 0.5$, DSAs always succeed.

4.3 Impact of confirmation length on DSAs

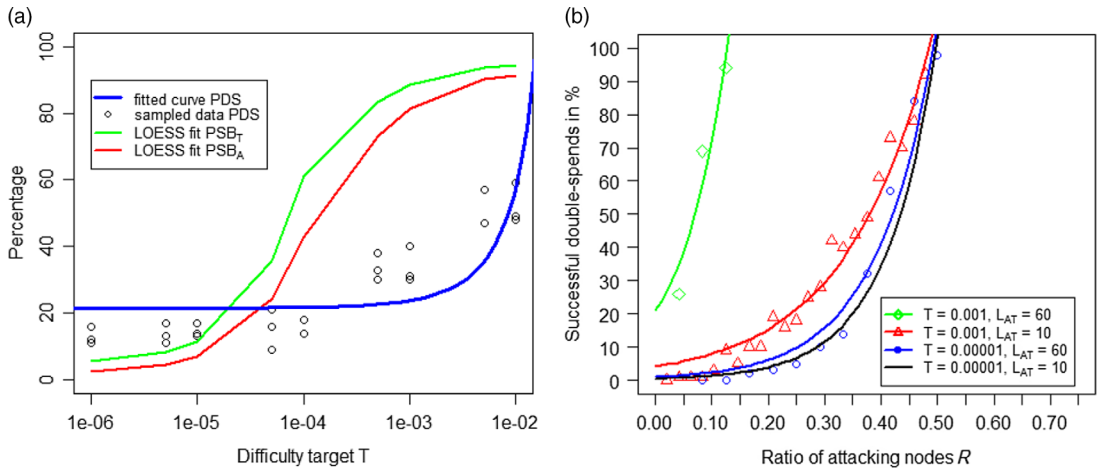
Next, we simulated a blockchain architecture with different values for ratio R and number of confirmation blocks C. The results are depicted in Figure 6: Figure 6(a) shows the impact of the number of confirmation blocks on the PDS for different node ratios, while Figure 6(b) shows the impact of the ratio between attacking and trusted nodes on the PDS for different choices of confirmation blocks. The figures show that the PDS drops exponentially when increasing the number of confirmation blocks. Moreover, it seems that the strength of the impact of node ratio on successful DSAs depends on the chosen number of confirmation blocks, that is, a low confirmation length leads to a lower exponential factor, whereas a high confirmation length leads to a larger exponential factor.

4.4 Impact of mining difficulty on DSAs and SBs

Another set of experiments investigated the impact of mining difficulty T on the PDS and the PSB[2]. To this end, Figure 7(a) shows the outcome of simulating blockchain architectures for different mining

[2] The percentage of SBs is modeled and visualized using locally estimated scatterplot smoothing fits (Cleveland, 1979; R Core Team, 2017).

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press



Impact of difficulty on successful DSAs. Impact of node ratio on successful DSAs.

Figure 7 Successful DSAs for different mining difficulties

difficulties T. The figure shows a clear positive effect of mining difficulty on both the probability of successful DSAs and the number of SBs for corresponding blockchain systems. However, while the PDS remains constant until it then increases exponentially, the PSB increases from the beginning and then flattens out logarithmically. On the other hand, Figure 7(b) shows the impact of node ratio on the PDS for different mining difficulties and latencies, while the number of confirmation blocks and network densities are kept constant. Again, it seems that mining difficulty influences the strength of the impact of node ratio on the PDS: a higher difficulty leads to a lower exponential factor, while a lower difficulty leads to a higher exponential factor.

4.5 Impact of latency on DSAs and SBs

Our next set of experiments investigated the effect of network latency on the PDS and the PSB. As mentioned above, we distinguish between latencies within the trusted and the attacking network and latencies between these two subnetworks.

4.5.1 Latencies within subnetworks

Figure 8 shows the outcome of simulating a blockchain architecture for different latencies L_T and L_A for the trusted and attacking subnetwork, respectively.

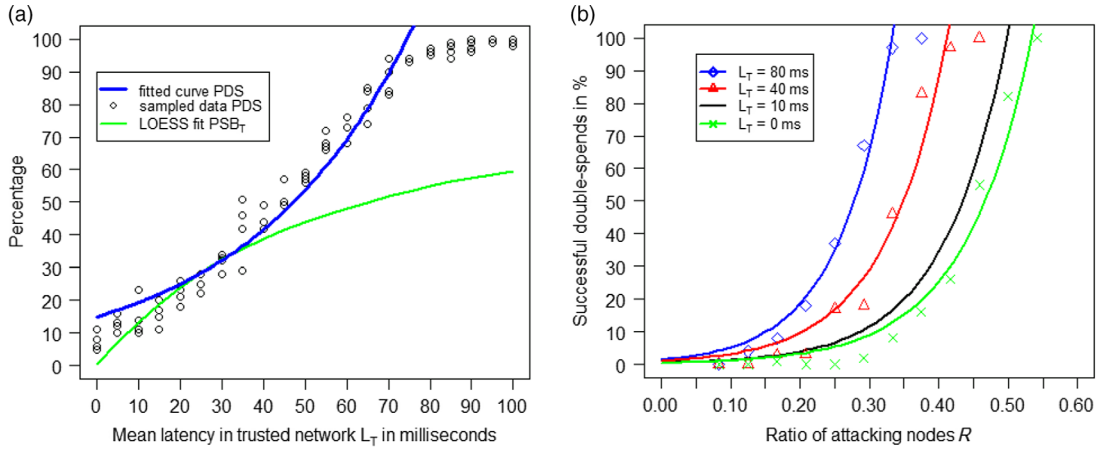
Figure 8(a) and (c) show the impact of average latency in the subnetworks on the PDS as well as the PSB for networks with constant ratio. It can be observed that the expected latency in the trusted subnetwork indeed impacts both the PDS (exponentially) and the PSB (logarithmic). Similarly, we can observe a negative, logarithmic effect of latency in the attacking network on the PSB. However, compared to the positive effect observed in Figure 8(a), the effect of latency in the attacking network on PDS is exponentially negative.

As shown in Figure 8(b) and (d), network latencies influence the impact of node ratio on successful DSAs. While a higher latency within the trusted network decreases the effect of ratio on successful DSAs, the effect is increased by increasing the latency of the attacking network.

4.5.2 Latency between subnetworks

Figure 9 shows the outcome of simulating a blockchain architecture with different latencies L_c *between* the attacking and the trusted network. Figure 9(a) shows the impact of average latency between the two subnetworks on PDS as well as PSB for a network with constant ratio. Similar to the experiments with the latency in the attacking network, we can observe an exponentially negative effect on the PDS. Again, Figure 9(b) shows a similar effect on the impact of node ratio on successful DSAs, as already observed for trusted and attacking latencies.

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press



Impact of tr. latency on successful DSAs. Impact of node ratio on successful DSAs.

Impact of at. latency on successful DSAs. Impact of node ratio on successful DSAs.

Figure 8 Successful DSAs for different network latencies

4.6 Impact of topology on DSAs and PSBs

Finally, we investigated the impact of network topology on the PDS and the PSB. As mentioned above, topology is measured in terms of graph density (ratio of existing and maximum amount of edges in the network).

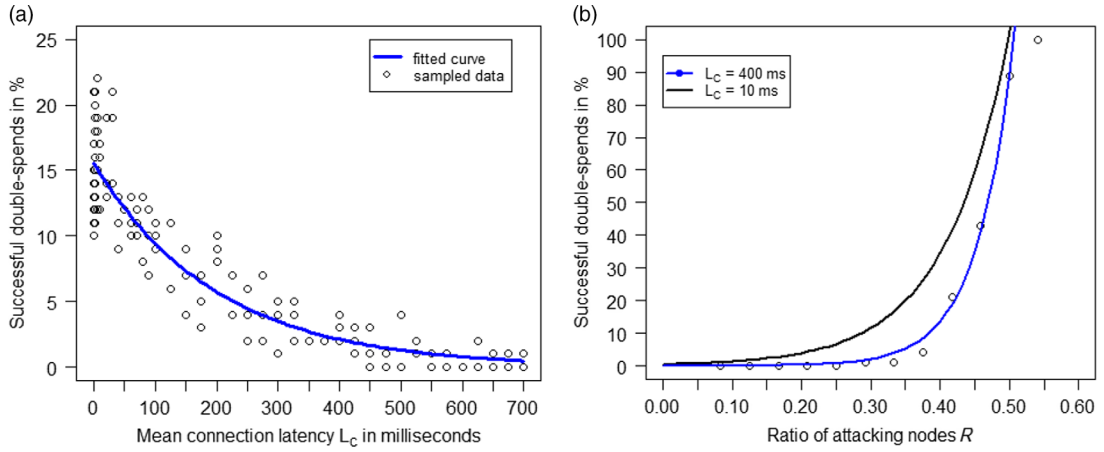
Figure 10 depicts the outcome of simulating a blockchain architecture with different densities for the trusted and attacking subnetwork: Figure 10(a) and (c) depict the percentage of successful DSAs for different densities of the trusted subnetwork D_T and the attacking subnetwork D_A for latencies of 100 ms (green) and 10 ms (blue). For the latter (10 ms), they also show the impact of density on the amount of PSBs (red plot). We can observe a negative impact of the density of the trusted network on both PDS and PSB. On the other hand, the effect is inverted for the density of the attacking network. For both cases, however, the effect is intensified when increasing network latency.

Similar as for network latencies, as shown in Figure 10(b) and (d), there is also a clear influence of network densities on the impact of node ratio on PDS. Again, a higher density of the trusted network decreases the effect of ratio on successful DSAs, while the effect is increased by increasing the density of the attacking network.

5 Discussion

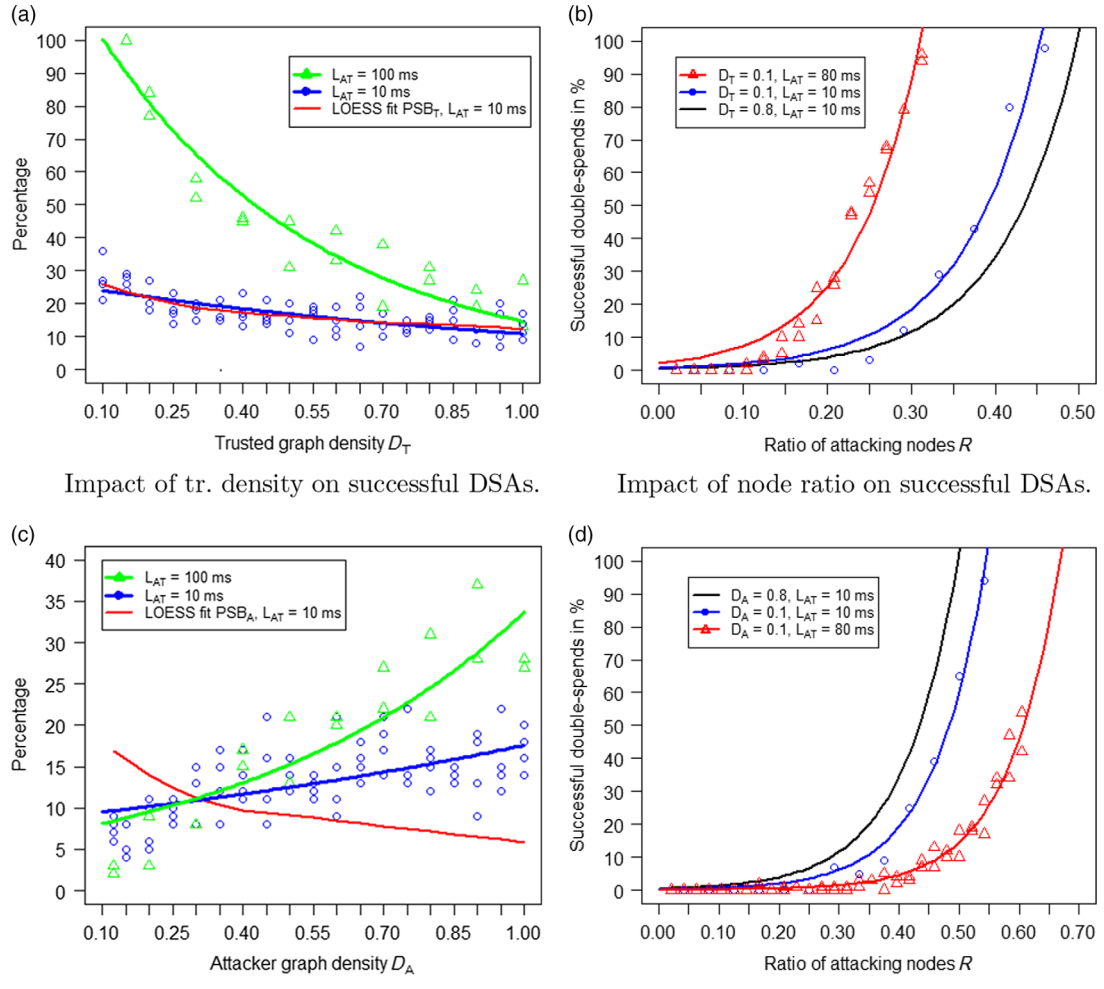
In the following, we interpret the results of our experiments presented in Section 4 with the aim to provide evidence whether or not each of the independent variables has an impact on the amount of successful

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press



Impact of latency on successful DSAs. Impact of node ratio on successful DSAs.

Figure 9 Successful DSAs for different latencies between the trusted and the attacking subnetwork



Impact of tr. density on successful DSAs. Impact of node ratio on successful DSAs.

Impact of tr. density on successful DSAs. Impact of node ratio on successful DSAs.

Figure 10 Successful DSAs for different network densities

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

Table 2 Results of Kruskal–Wallis tests for $H_{0,p}$

p	Df	χ^2	p-Value
N	10	2.7562	0.9866
R	46	164.69	2.909e-15
C	30	81.106	1.368e-06
L_C	37	120.36	9.409e-11
L_A	20	70.515	1.501e-07
L_T	20	81.603	2.095e-09
D_A	18	57.391	5.364e-06
D_T	18	34.8	0.01001
T	8	21.637	0.005634

DSAs. Thus, for each independent variable p, the data presented in the last section will be tested against null hypotheses of the form

$$H_{0,p}: \text{ Parameter } p \text{ has no effect on DSAs} \tag{2}$$

using the Kruskal–Wallis rank sum test (Kruskal & Wallis, 1952; R Core Team, 2017) and a confidence level of 95%. The results are shown in Table 2 and discussed in the following subsections.

5.1 *Number of nodes N*

Figure 4 suggests that the means of the factor levels for N are roughly equal and no underlying trend of the data can be observed immediately. This assumption is validated by the Kruskal–Wallis test in Table 2. Since $p > 5\%$, the differences in the means of factor levels are not significant at a level of 5%. Therefore, the null hypothesis $H_{0,N}$ cannot be rejected. We conclude that the amount of total nodes N has no significant effect on successful DSAs. Since all nodes of the simulator possess the same computing power, this observation is sensible. We assume that especially the proportion of attacking nodes compared to trusted nodes results in a change of successful DSAs. As this proportion was kept at a constant level in Figure 4, no effects can be observed.

5.2 *Ratio of attacking nodes R*

The trend of Figure 5 is comparable to the plots of hashrate-based models by Rosenfeld (2014) or Pinzón & Rocha (2016), whereas the hashrate possessed by an attacker corresponds to the ratio of nodes under an attacker's control in our model. The probability of successful DSAs seems to be rising exponentially with increasing R, which is also confirmed by the fitted curve. Once the majority of the network is controlled by an attacker, DSAs appear to be always successful. An important difference and interesting observation is that at exactly 50% control of the attacker, successful DSAs are not guaranteed. This observation is contrary to the mathematical models proposed by Nakamoto (2008), Rosenfeld (2014), and Pinzón and Rocha (2016) and might be a result of the network parameters whose effects have not been evaluated at this point. Another reason for this observation could be seen in the end conditions of our simulator. If the trusted network is able to maintain a steady lead over the attackers' blockchain fork without triggering the maximum lead parameter to end the simulation, the attempt will fail after a maximum blockchain length is reached. This scenario is especially likely if the attacking network's computing power is very similar to the computing power controlled by the trusted nodes, since this results in both networks having equal chances of mining the next block. In reality, this observation might correspond to an attacker being unable to maintain the necessary resources of conducting a DSA over a long time frame. The attacker might be forced to abort the DSA, since mining on a secret chain does not generate any revenue. Courtois (2014) argues in a similar way, in that an attacker will only be able to maintain such a significant amount

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

of hashing power over a short period of time. According to Table 2 and the observable pattern in Figure 5, the differences between data means are significant at a level of 5%. We therefore reject $H_{0,R}$.

5.2.1 Confirmation length C

According to Figure 6(a), an increasing amount of confirmations reduces the probability of successful DSAs. Regardless, the effect decreases significantly with increasing R and, looking at Figure 6(b), the confirmation length has no effect at all once an attacker gains control over the majority of computing power in the network. This observation seems plausible and confirms the results of Nakamoto (2008) and Rosenfeld (2014), describing that every trusted lead can be overcome by an attacker once the majority is mining on the malicious chain. Overall, a merchant should aim to wait for as many confirmations as possible to reduce the risk of DSAs. The concrete amount of confirmations could be determined by the value of the merchant's product. Combining the data of the three plots in Figure 6(a) enables us to perform a Kruskal–Wallis test (Table 2). Using the generated results, we successfully reject $H_{0,C}$ at a significance level of 5%.

5.3 Connection latency L_C

At a first glance, effects of increasing the connection latency L_C seem to be similar to an increase in the confirmation length. The amount of successful DSAs is reduced by longer latency times, but once an attacking ratio of 50% is reached, the parameter shows no effect (Figure 9(b)). This phenomenon can be explained by regarding the nature of the connection latency. Assume Alice plans a DSA and experiences a latency time of L_C to every trusted node. For her attack to be successful, she needs to gain a sufficient lead with her secret blockchain branch in order to still be ahead of the trusted nodes' chain once her malicious branch reaches the targeted network. During L_C, the trusted network might be able to catch up to Alice's branch, despite her possibly being in the lead at the time she sent her copy. However, if Alice controls more computing power than the defending network, she will always be able to generate an arbitrarily large lead over the trusted nodes' chain, thereby successfully overcoming L_C. The connection latency could be increased by the trusted network by not revealing concrete addresses of peers in order to hide them from potential attackers and be more resistant against DSAs. An attacker will always aim to gather as much information as possible about trusted peers to reduce the maximum L_C. The Kruskal–Wallis test results in Table 2 indicate that L_C does in fact have a significant effect on the number of successful DSAs, allowing us to reject H_{0,L_C} at our confidence level of 95%.

5.4 Network latency times L_A, L_T

The visible effects of L_A and L_T on successful DSAs depicted in Figure 8 can be explained by observing the percentage of SBs generated during the experiments. At 0 ms of latency between peers, no SBs are mined. This is plausible since block propagation is instant and the participating nodes are in sync at all times. However, with increasing latency times, chances of two nodes mining a block before being notified by each other increase as well, therefore raising the number of SBs. As SBs do not contribute to the length of the agreed blockchain, their creation indicates a waste of computing power, especially when defending against DSAs. Networks with a high number of SBs consequently possess less *effective* computing power than networks producing fewer SBs and the same number of nodes (hashrate). Figure 8(c) shows how an increase in L_A raises the number of SBs for the attacking network, which in turn reduces the effective computing power possessed by this network, resulting in a lower probability of successful DSAs. Since the distribution of computing power is represented by R in our model, a translation along the R-axis for different L_A can be observed in Figure 8(d). Consequently, an increase in L_A leads to the decrease in R, whereas an increase in L_T corresponds to an increase in R. We conclude that it is not sufficient to only consider the number of nodes or the hashrate controlled by each network when analyzing the distribution of computing power between attacking and defending networks. Instead, it is also important to consider how well nodes of a network are working together on a single consented blockchain and how many hashes are wasted toward the computation of SBs. The number of SBs is therefore indicative

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

of a network's mining efficiency which is affecting their capability of extending the blockchain. To maximize resistance against DSAs, the trusted network should aim to reduce the overall latency in their network. It should also be considered that an attacker might use the same infrastructure and protocol as the trusted nodes. Additionally, the attacker might deliberately target the trusted node's communication with other attacks in order to increase L_T and consequently the chance of DSAs. The results of the Kruskal–Wallis tests shown in Table 2 allow us to reject H_{0,L_A} and H_{0,L_T} at a confidence level of 95%, confirming the significant effect of the parameters on successful DSAs.

5.5 *Graph density D_A, D_T*

Similar to L_A and L_T, the graph density influences the number of SBs mined by the networks, producing comparable effects on successful DSAs. A network with low graph density contains less edges between nodes, which increases the average latency time between peers. Nodes of dense networks consequently work together in higher harmony, resulting in less computational power being wasted toward the mining of SBs. Therefore, denser networks excel at extending their consented blockchain by increasing efficiency of sharing new blocks. Again, the effects on the functions of R in Figure 10(b) and (d) can be roughly described as a translation along the R-axis. An important observation can be seen in the fact that a higher latency time appears to amplify the effects of the graph density on *PDS* and vice versa. This is plausible, since adding edges to a high latency network reduces overall latencies to a higher degree compared to adding edges in an already low latency network. To increase resistance against DSAs, the trusted network should gather information about as many peers as possible in order to increase density of the network and reduce latency times. This might again be influenced by an attacker using denial of service or similar attacks on the nodes' communication. The results of performing two Kruskal–Wallis tests on the data depicted in Figure 10(a) and (c) are listed in Table 2. Since both p-values are below 5%, the graph density can be considered to have a significant effect on successful DSAs and we conclude by rejecting H_{0,D_A} and H_{0,D_T}.

5.6 *Mining difficulty*

The effects of changing the difficulty T of mining new blocks are again closely related to the number of SBs generated by the attacking and trusted networks. If T is small, chances of finding a valid PoW decline, making it more difficult to mine a new block. This results in less blocks being mined on average, which decreases the chances of two blocks being mined at roughly the same time, creating a SB. If T is increased, more blocks are mined overall, resulting in a higher percentage of SBs. Figure 7(a) shows the number of SBs for both networks as a function of T. At the lowest tested mining difficulty, barely any SBs are created. However, after raising the target, over 90% of all blocks generated by both networks are stale. Since the number of SBs produced by the trusted and attacking network is comparably high, both networks are wasting similar amounts of computing power on the creation of SBs. But despite both networks being equally affected by the high number of SBs, the attacking networks seem to benefit in their attempts to double-spend. Because of the outstanding majority of blocks being stale, each node in the networks is effectively mining on its own. Only rarely will a node receive blockchains longer than its own copy, since latency times are too long in relation to the difficulty target and the node has already mined more blocks on its own chain when receiving its peers' blockchains. If each node is mining on its own, the computing power of the attacking and trusted networks consists of effectively one node. In the worst case, this leads to 50% of the computing power being controlled by an attacker, increasing the chance for DSAs significantly. Effects of the mining difficulty are obviously always related to the latencies in the networks. We therefore distinguish between *high* mining difficulty, leading to high amounts of SBs and vulnerability against DSAs and *low* mining difficulty leading to a lower number of SBs and therefore higher resistance against DSAs. More specifically, a low mining difficulty in a low latency network could resemble a high mining difficulty in a high latency network, since more SBs will be generated. Ultimately, the mining difficulty should always be chosen in a way that the smallest number of SBs is produced.

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

Figure 7(b) visualizes the relation between latency and mining difficulty. While increasing the mean latency in both networks by 50 ms at a mining difficulty of 0.00001 has almost no effect on successful *DSAs*, increasing the latency for $T = 0.001$ leads to a significantly higher amount of successful DSAs. While 0.001 might be considered a low difficulty for a mean latency of 10 ms, it is not low enough in networks experiencing higher latencies. The goal of the trusted network is to increase resistance against DSAs. Therefore, it would be advantageous to establish a low mining difficulty in regard to the latency and density of their network. Since these network parameters often depend on communication architecture or hardware, they might be hard to influence. However, by choosing a suitable mining difficulty, deficits in L_T and D_T might be compensated, maximizing the probability of agreement between nodes.

The significance of the mining difficulty T w.r.t. successful DSAs is validated by the Kruskal–Wallis test in Table 2, allowing us to reject $H_{0,T}$.

6 Toward an empirical prediction model for blockchain architectures

The simulation environment presented so far can be used to simulate DSAs and PSBs for different ADDs. However, it is not yet possible to estimate necessary ADDs for a blockchain architecture with a desired resistance against DSAs. To this end, we performed several additional experiments to systematically develop an empirical model, based on the results of these experiments. In total, over 4000 data points were created, allowing us to perform regressions to fit the resulting empirical constants of the proposed model.

6.1 Derivation

The challenge of finding a mathematical model, sufficiently representing the simulator's characteristics, relies on identifying a suitable basis function to fit to the sampled data of all experiments. To this end, we followed an iterative approach: we estimated an initial basis function which we then gradually refined by iteratively including the remaining parameters of the simulator. The resulting empirical constants for each parameter are fitted to our data using nonlinear regressions (Nash, 2016) after each iteration.

Since the ratio of computing power controlled by the attacking network seems to be of special relevance to estimate the probability of successful DSAs, we chose the function depicted in Figure 6(b) as our basis model. The remaining parameters are now included by analyzing the direct effects of the different parameters on successful DSAs (left columns of Figures 6, 7, 8, 9, and 10), as well as their indirect influence (right columns of Figures 6, 7, 8, 9, and 10). The influence of C and L_C on successful DSAs seems to be independent of the effects created by other parameters. They are therefore chosen as the next parameters to be included in the model after the initial factor R. Since the effects of latency and density parameters amplify each other, the corresponding parameters are added to the formula simultaneously during the fourth iteration. The parameter describing T is included during the last step, since it directly influences the effective impact of the latency and density parameters, respectively.

6.2 Model

The final model formalizes the probability of a successful DSA in terms of a function of the estimated ratio R, mining difficulty T, amount of required confirmation blocks C, latencies L_T, L_A and L_C, and network topologies D_T and D_A:

$$PDS = 100\, exp\left(\left(a\left(R - \frac{e_1\,(e_2 T + e_3)\,L_A}{d_1\,(D_A + d_2)} + \frac{e_1\,(T + e_3)\,L_T}{d_1\,(D_T + d_2)}\right) - \frac{a}{2}\right) \cdot (b_1 C + b_2)\,(c_1 L_C + c_2)\right)$$

The values of the corresponding empirical constants are summarized in Table 2.

To test our model, we compared its predictions to the outcome of our experiments. Although our model is not entirely accurate, the general profile of the plotted formula matches the individually fitted curves of the data.

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

The model can be used to estimate certain ADDs, given a desired resistance against DSAs for the resulting blockchain architecture.

7 Limitations

Although simulations and experiments are generally accepted as ways to analyze and explore the behavior of diverse and dynamic systems (Wohlin *et al*., 2012), limitations of our results should be mentioned.

7.1 External validity

During development of the simulator, simulation runs, and the generation of results, some threats to the validity of our experiments were identified which will be summarized in the following sections.

7.1.1 Miner scheduling

The simulation runs were carried out on two machines using *Windows 7* and *Ubuntu 16.04* operating systems equipped with *Intel i5* processors. Since each miner is represented by a single thread, scheduling was required to simulate concurrency. Throughout the experiments, it was realized that sometimes certain mining threads were given access to the processing unit for too long, resulting in multiple blocks being mined by a single node before other nodes were given the chance to generate a PoW. This phenomenon is especially relevant during simulations with high mining difficulty, which leads to many blocks being mined in quick succession. Although access to the processing unit between trusted and attacking nodes converges over time, some nodes were able to generate a significant advantage by generating multiple blocks early during the simulation run. This issue has been addressed by synchronizing the mining threads after each node has generated 10 random numbers toward their next PoW. This way, each node can only generate a maximum of 10 random numbers ahead of its peers, reducing the likelihood of multiple blocks due to the scheduling bias. Nevertheless, the phenomenon might still persist, especially when simulating with a very high mining difficulty ($T > 0.1$).

7.1.2 Constant connection strategy

The connection strategy used for the simulations assumes that each attacking node is aware of every trusted node in the system and manages to establish a connection of constant latency to all targeted nodes. Although this is the optimal strategy an attacker should strive for, since it reduces the time needed to send fraudulent blockchains to the trusted nodes, some trusted peers could be masked by the network and remain unknown to an attacker. A connection strategy using a *connection density* parameter is missing in our simulation, therefore such a scenario was not examined. However, it is assumed that reducing the density of an attacker's connection to the trusted network produces similar effects as increasing the connection latency L_C.

7.1.3 Random graph creation

The pseudorandom graph created for our simulations is not perfectly random and introduces a bias to certain topologies. Our algorithm initially creates a pseudorandom spanning tree of $n - 1$ edges and n nodes by iterating through all nodes in the network. During each iteration, a random node of already visited and connected nodes is chosen to create an edge with a new node, which is not yet included in the graph. Since for each iteration only visited nodes are considered for one end of a new edge, earlier nodes are more likely to receive a higher degree. This could alter effects of parameters regarding the network topology, especially latency and graph density. Nevertheless, these effects are assumed not to be significant. Instead, our algorithm creates graphs that are more closely related to real computer networks by increasing the likelihood of graphs with centralized nodes (Miller *et al*., 2015).

7.1.4 Post hoc analysis

The null hypotheses presented in Section 5 were tested using the Kruskal–Wallis rank sum test. The results of this test only confirm the existence of a significant difference between groups of the same

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

treatment level, but do not explain which specific pairs of groups are different. This information could be obtained by performing a *post hoc* analysis using, for example, the Wilcoxon rank sum test (Wilcoxon *et al*., 1970). Looking at our data, such an analysis could be important, since some of the independent variables seem to not show any effect for specific treatment levels. This phenomenon can be observed in the data samples presented in Figure 6(b), which remain 0 for multiple treatment levels when testing a confirmation length of 14. Similarly, the value of successful DSAs seems to be unchanged after reaching 100 in Figure 5. A *post hoc* analysis might be necessary to determine significant intervals of the tested parameters.

7.2 Internal validity

While building the model proposed in Section 6, additional threads to the validity of the model were identified.

7.2.1 Missing factors

There might be additional factors affecting an architecture's resistance against DSAs which are missing in the implementation of our simulator and are consequently not included in the model definition. One of these potentially missing factors is the connection density D_C already mentioned above. This parameter could lead to an amplifying effect on L_C, similar to the observed relation between D_T and L_T (D_A and L_A). A second parameter not included in the simulator and the model is the amount of lost block transmissions due to packet loss. This phenomenon is present in real blockchain architectures and leads to missing blocks and the generation of orphan blocks in the networks. A node is unable to mine on top of a received orphan block as these blocks are disconnected from the local blockchain copy. Therefore, a potentially longer blockchain might remain unknown to the node until all missing parent blocks are acquired. In the mean time, the node persists on mining on the valid shorter blockchain. If blocks mined onto this shorter chain are then replaced by the orphan block and its delayed parents, the number of generated SBs increases. Since the number of SBs has an effect on successful DSAs (as discussed in Section 5), a packet loss parameter could influence a blockchain architecture's resistance against DSAs. Simulating this parameter would require implementing the transmission of single blocks in our simulator, as opposed to the current propagation of whole blockchain copies.

7.2.2 Empirical constants

Similar to missing factors, the relationships between parameters pose another threat to our model. Although relationships between the latency, density, and difficulty parameters have been identified, their modeling in Section 6 might be inaccurate. Furthermore, the existence of additional relationships which are not yet modeled by our formula should be considered. Moreover, the model's empirical constants (Table 3) were iteratively fitted to the experimental data using nonlinear regressions (Nash, 2016). These constants might be erroneous and rely fundamentally on the quality and amount of the sampled data they are fitted to.

8 Related work

To identify parameters affecting the resistance of blockchain architectures against DSAs and to gain an understanding of existing literature regarding general attacks on blockchains and their modeling, we will review a selection of related work. The following provides an overview of the reviewed literature and summarizes the authors' conclusions that are relevant to our investigations.

8.1 Hashrate-based models for DSAs

Potential attacks on blockchains, especially the Bitcoin protocol, have been conceptualized since the release of Nakamoto's original Bitcoin paper (Nakamoto, 2008). In chapter 11 of his proposal, Nakamoto formulates the first mathematical model to calculate the theoretical probability of successful DSAs on his protocol. This model has since been improved and adapted several times, while similar independent

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

Table 3 Empirical constants for the PDS model

Const.	value
a	10.6572
b_1	0.102625
b_2	0.409421
c_1	0.00273521
c_2	1.02279
d_1	226.506
d_2	0.868318
e_1	50000.2
e_2	0.956131
e_3	0.00001

models have been formulated as well (Rosenfeld, 2014; Pinzón & Rocha, 2016; Grunspan & Pérez-Marco, 2017; Ozisik & Levine, 2017). Similar to Pinzón and Rocha (2016), we will call the collection of these approaches *hashrate-based* attack models. The central premise of a hashrate-based model is splitting the total computing power available to the network (hashrate H) into two parts. To achieve this, Rosenfeld (2014) defines pH as the hashrate controlled by honest nodes adhering to the protocol, while qH is used by malicious nodes trying to attack the blockchain. Since also

$$p + q = 1, \tag{3}$$

it can easily be seen that p and q are precisely the probabilities of the next block being mined by either the honest or the attacking part of the network. Using an adaptation of the Gambler's Ruin problem (Coolidge, 1909), Nakamoto and Rosenfeld (2014) are now calculating the probability Q_z of an attacker successfully catching up with their fork of the blockchain, assuming they are at a total deficit of z blocks compared to the honest nodes. The success of a potential DSA against a merchant waiting for n confirmations can now be formulated as the probability of Q_z, after n blocks have been mined by the honest network. While Nakamoto (2008) is using a Poisson distribution in his formula, Rosenfeld (2014) is achieving similar results with a negative binomial distribution. Combined with Grunspan and Pérez-Marco (2017), the author of Rosenfeld (2014) is also pointing out and correcting an of-by-one error introduced by Nakamoto, who is only calculating the probability of an attacker *catching up* with his fraudulent blockchain fork, while for the DSA to be successful, the length of the honest chain has to be *surpassed*. Pinzón and Rocha (2016) present two similar approaches but considers partial advancement toward block creation as well. The author's first model extends the hashrate-based model formulated by Rosenfeld (2014) and includes an additional parameter, indicating the time an attacker has already spent mining on blocks. The second model is fundamentally based on the time differences at which honest and attacking nodes have mined their last block, but is also relying on hashrates as a measure of computational power. It is interesting to note that all of the previously mentioned models produce similar results despite the differences in their calculations. Therefore, it is less surprising that the authors seem to agree on their general conclusions as well. Those can be summarized as follows:

- If an attacker controls the majority of computational power in the network, DSAs will always succeed.
- Probabilities for successful DSAs decrease exponentially with an increasing number of confirmations.
- Probabilities for successful DSAs increase exponentially with an increasing amount of hashing power controlled by the attacker.
- Although DSAs at the standard of six confirmations are considered to be unlikely, there is nothing special about the number six.
- DSAs are always possible regardless of the number of confirmations and amount of computational power controlled by the attacker.

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

8.2 Simulations

An important remark can be seen in the fact that none of the before mentioned hashrate-based models have been confirmed or supported by experimental data or tests in an exhaustive and realistic way. Instead, the validity of these models relies mostly on mathematical proofs and expertise or the comparison with other similar models. One exception here is the work of Ozisik and Levine (2017). After presenting a detailed explanation of Nakamoto's model for DSAs, the author validates the mathematical approach by performing a Monte Carlo simulation (Mooney, 1997) with different sets of input. In the light of this, the author identifies an error of the model, which is linked to its use of the Poisson distribution. In spite of this success, the Monte Carlo simulation is missing parameters of a real blockchain protocol and 'does not actually mine coin, it simply flips some coins to see whether each miner wins a block as simulated' (Ozisik & Levine, 2017). A more realistic simulation of a different attack on Bitcoin blockchains, the *selfish-mine* attack, is presented by Mwale (2016) and Göbel *et al*., (2015). In these two models of the Bitcoin protocol, random (x, y)-coordinates of a two-dimensional plane are assigned to each node in order to create a simple network topology. By simulating the latency between two nodes as proportional to their euclidean distance in the plane, both authors successfully model block propagation times of a real network. During the simulation, new blocks are again generated as instances of a Poisson process with an average rate of 10 minutes.

8.3 Other attacks

Next to DSAs a wide variety of different attacks against blockchains and the Bitcoin protocol has been conceptualized. The already mentioned *selfish-mine attack* or *block-discarding attack* focuses on invalidating the honest miners' work by selectively publishing premined blocks to the network (Bahack, 2013; Eyal & Sirer, 2013; Göbel *et al*., 2015; Mwale, 2016). According to Bahack (2013) and Eyal and Sirer (2013), this attack can succeed with only a fraction of 25% total hashing power controlled by the attacker. The *whale attack* presented by Liao and Katz (2017) aims to increase an attacker's chances of successful double-spends by publishing transactions with large mining fees to incentivize honest miners to build on a fraudulent blockchain fork. Karame *et al.* (2012) and Sompolinsky and Zohar (2016) further capitalize on the *premining* strategy of secretly mining blocks ahead of the honest blockchain, while also including a reversed transaction to the target. Once the attacker has gained a comfortable lead, the original transaction to the target merchant is released. Now the lead of the attackers' fork only has to be maintained until the transaction has been confirmed, in order to perform a double-spend. The authors demonstrate how this attack can be used to increase the probability of DSAs, when timing of the transaction used as bait is not important.

8.4 Relationship to our work

The studies reviewed in this section mainly focus on the formalization of theoretical attack models against blockchains. To this end, correctness of the proposed models is achieved through mathematical proofs. While this ensures a high degree of rigor in the results, usually some simplifications are needed to allow for such analyses. With the work presented in this paper, we took an alternative approach using simulation-based analyses. Thus, although the results are less rigorous, we could incorporate additional complexities, such as network densities and latencies. Hence, with the work presented in this paper, we actually complement the studies discussed above by providing additional analysis capabilities for blockchain-based applications.

9 Conclusion

This paper described a simulation-based approach for the design of PoW-based blockchain architectures. To this end, we presented a framework for the simulation of such architectures, which may simulate DSAs and the number of SBs for various types of design decisions: (1) the estimated ratio of untrusted vs. total

https://doi.org/10.1017/S0269888920000193 Published online by Cambridge University Press

nodes, (2) mining difficulty, (3) number of confirmation blocks, (4) network latency, and (5) network topology. Then, we described the outcome of several experiments for various design decisions: (1) the impact of network size, node ratio, number of confirmation blocks, mining difficulty, network latency, and density of nodes, on the probability of successful DSAs and (2) the impact of mining difficulty, network latency, and density of nodes, on the amount of PSBs. Based on the outcome of the experiments, we derived a set of hypotheses which we tested by means of statistical analyses. On top of these analyses, we finally provide an empirical model for the impact of ADDs on a PoW-based blockchain architecture's resistance against DSAs.

The framework can be used to simulate certain design decisions for PoW-based blockchain architectures before implementing them. The empirical model can be used to approximate design decisions for desired qualities. Thus, wrong design decisions (w.r.t. expected properties) could be avoided which reduces effort of fixing them after implementation.

As of today, however, the simulation environment (and corresponding model) supports only blockchain architectures based on PoW. For the future, we want to integrate additional consensus algorithms, such as Proof of Stake (Buterin, 2016) which is used, for example, in the Ethereum blockchain (Buterin, 2013). Moreover, we are working on an extended version of the framework to support additional design decisions and simulate additional properties.

Acknowledgments

The authors would like to thank Manfred Broy and all the anonymous reviewers of *SDLT3* for their comments and helpful suggestions on earlier versions of this paper. Parts of the work on which we report in this paper were funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01Is16043A.

References

Antonopoulos, A. M. 2017. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, Inc.

Azaria, A., Ekblaw, A., Vieira, T. & Lippman, A. 2016. MedRec: Using blockchain for medical data access and permission management. In *International Conference on Open and Big Data (OBD)*, 25–30. IEEE.

Bahack, L. 2013. *Theoretical Bitcoin Attacks with Less Than Half of the Computational Power (Draft)*. *CoRR*, abs/1312.7013. http://arxiv.org/abs/1312.7013.

Buterin, V. 2013. *Ethereum*. https://web.archive.org/web/20150328054135/ https://github.com/ethereum/wiki/wiki/White-Paper.

Buterin, V. 2016. *A Proof of Stake Design Philosophy*. https://web.archive.org/web/20191209165316/ https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51.

Chavez-Dreyfuss, G. 2016. *Sweden Tests Blockchain Technology for Land Registry*, June. https://web.archive.org/web/20161024065806/ http://www.reuters.com/article/us-sweden-blockchain-idUSKCN0Z22KV.

Cleveland, W. S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* **74** (368), 829–836.

Coolidge, J. L. 1909. The Gambler's ruin. *Annals of Mathematics* **10**(4), 181–192. ISSN: 0003486X. http://www.jstor.org/stable/1967408.

Courtois, N. T. 2014. *On the Longest Chain Rule and Programmed Self-Destruction of Crypto Currencies*. *CoRR*, abs/1405.0534. http://arxiv.org/abs/1405.0534.

Courtois, N. T. & Bahack, L. 2014. *On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency*. *CoRR*, abs/1402.1718. http://arxiv.org/abs/1402.1718.

Decker, C. & Wattenhofer, R. 2013. Information propagation in the Bitcoin network. In *IEEE P2P 2013 Proceedings*, 1–10, September. doi: 10.1109/P2P.2013.6688704.

Eichhorn, L. 2018. *Simulation-Based Analysis of Blockchain Architectures*. https://github.com/LeoEichhorn/Blockchain.

Eyal, I. & Sirer, E.G. 2013. *Majority is Not Enough: Bitcoin Mining is Vulnerable*. *CoRR*, abs/1311.0243. http://arxiv.org/abs/1311.0243.

Garlan, D. 2000. Software architecture: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, 91–101. ACM.

Göbel, J., Keeler, H. P., Krzesinski, A. E. & Taylor, P. G. 2015. *Bitcoin Blockchain Dynamics: The Selfish-Mine Strategy in the Presence of Propagation Delay*. *CoRR*, abs/1505.05343, 2015. http://arxiv.org/abs/1505.05343.

Grunspan, C. & Pérez-Marco, R. 2017. *Double Spend Races*. *CoRR*, abs/1702.02867. http://arxiv.org/abs/1702.02867.

Karame, G. O., Androulaki, E., Capkun, S. 2012. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 906–917. Association for Computing Machinery, New York, NY.

Liao, K. & Katz, J. 2017. Incentivizing double-spend collusion in bitcoin. In *Financial Cryptography Bitcoin Workshop*. https://fc17.ifca.ai/bitcoin/schedule.html.

Marmsoler, D. & Eichhorn, L. 2018. Simulation-based analysis of blockchain architectures. In *The 3rd Symposium on Distributed Ledger Technology, SDLT 3, Proceedings*. https://symposium-dlt.org/3rd/index.html.

Kruskal, W. H. & Wallis, W. A. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association* **47**, 583–621.

Miller, A. K., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N. & Bhattacharjee, B. 2015. Discovering *Bitcoin's Public Topology* and influential nodes. University of Maryland, College Park.

Mooney, C. Z. 1997. *Monte Carlo Simulation*, **116**. Sage Publications.

Mwale, M. 2016. *Modelling the Dynamics of the Bitcoin Blockchain*. PhD thesis, Stellenbosch University. http://scholar.sun.ac.za/handle/10019.1/98844.

Nakamoto, S. 2008. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

Nash, J. C. 2016. *nlmrt: Functions for Nonlinear Least Squares Solutions*. https://CRAN.R-project.org/package=nlmrt. R package version 2016.3.2.

Ozisik, A. P. & Levine, B. N. 2017. *An Explanation of Nakamoto's Analysis of Double-Spend Attacks*. *CoRR*, abs/1701.03977. http://arxiv.org/abs/1701.03977.

Pinzón, C. & Rocha, C. 2016. Double-spend attack models with time advantage for bitcoin. *Electronic Notes in Theoretical Computer Science*, **329**, 79–103. ISSN 1571-0661. doi: 10.1016/j.entcs.2016.12.006. http://www.sciencedirect.com/science/article/pii/S157106611630113X. The Latin American Computing Conference.

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org/.

Rosenfeld, M. 2014. *Analysis of Hashrate-Based Double Spending*. *CoRR*, abs/1402.2009. http://arxiv.org/abs/1402.2009.

Sompolinsky, Y. & Zohar, A. 2016. *Bitcoin's Security Model Revisited*. *CoRR*, abs/1605.09193. http://arxiv.org/abs/1605.09193.

Staples, M. 2017. Software engineering research for blockchain-based systems. In *1st Symposium on Distributed Ledger Technology*. http://www.ict.griffith.edu.au/network/sdlt.html.

Vukolić, M. 2017. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, BCC 2017*, 3–7, ACM. ISBN 978-1-4503-4974-1. doi: 10.1145/3055518.3055526.

Wilcoxon, F., Katti, S. K. & Wilcox, R. A. 1970. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected Tables in Mathematical Statistics* **1**, 171–259.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B. & Wessln, A. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated. ISBN: 3642290434, 9783642290435.

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C. & Rimba, P. 2017. A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE International Conference on Software Architecture, ICSA 2017, Gothenburg, Sweden, April 3–7, 2017*. doi: 10.1109/ICSA.2017.33.

Yurcan, B. 2016. *How Blockchain Fits into the Future of Digital Identity*, April. https://web.archive.org/web/20170119054131/ https://www.americanbanker.com/news/how-blockchain-fits-into-the-future-of-digital-identity.