Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-SIT

Seidlgasse 22 / 9, 1030 Wien
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

Inffeldgasse 16a, 8010 Graz
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

SECURITY OF BLOCKCHAIN-BASED CRYPTOCURRENCIES

Version 1.0 of 16.04.2019

Alexander Marsalek – Alexander.Marsalek@a-sit.at

Abstract: This document describes a selection of attacks on blockchain-based cryptocurrencies. It covers attacks like the 51% attack, different block withholding strategies, as wells as censoring and implementation attacks. After the introduction of various attacks, the document gives a short overview of selected security-analysis technologies and security insights.

Table of Contents

Table of Contents 1
1. Introduction 2
2. Attacks 2
2.1. Majority Attack / 51% Attack 2
2.2. Withholding Attacks 3
2.2.1. Selfish Mining 3
2.2.2. Block Withholding Attack 3
2.2.3. Fork After Withholding Attack 3
2.2.4. Finney Attack 4
2.3. Goldfinger Attack 4
2.4. Feather-Forking 4
2.5. Flood Attack 5
2.6. High Fee Attack 5
2.7. Transaction Malleability 5
2.8. Dust Attack 6
2.9. Programming Errors 6
2.9.1. Integer Overflow 6
2.9.2. Accidental Rule Change 6
2.9.3. DOS – Inflation Bug - CVE-2018-17144 7
3. Security Analyses 7
3.1. Whitepaper Calculations 7
3.2. Game-Theoretic Model 9
3.2.1. P + ε Attack 9
3.3. Robust Transaction Ledger 10
3.3.1. Chain Growth Property 10
3.3.2. Chain Quality 10
3.3.3. Common Prefix 10
3.3.4. Persistence 10
3.3.5. Liveness 11
4. Conclusions 11
References 11

1. Introduction

In 2018, the total market capitalization of cryptocurrencies surpassed 800 billion Dollar (over 700 billion Euro) [1]. Despite this capitalization there is little work published related to analyzing or proving the security of cryptocurrencies.
This document describes selected attacks on blockchain-based cryptocurrencies, with a focus on the cryptocurrency Bitcoin. Several of these attacks can also be applied to other cryptocurrencies. This document assumes that the reader is familiar with the principles of Bitcoin and its actors. Section 2 introduces different attacks and weaknesses of Bitcoin, while Section 3 introduces techniques that can be used to analyze the security of cryptocurrencies. So far, no model, covering every aspect, is known.

2. Attacks

This section describes a selection of attacks against cryptocurrencies as well as implementation errors that could be misused by attackers. Note that this list is not comprehensive. Not all introduced attacks have the same impact. Some can be used to overtake the network or disrupt the service, while others allow the attacker to unfairly enrich herself.

2.1. Majority Attack / 51% Attack

The possibility for a 51% attack was already discussed in the Bitcoin Whitepaper written by a person or group of persons under the pseudonym Satoshi Nakamoto [2]. Nakamoto decided to use a proof-of-work algorithm, which essentially follows the paradigm one-CPU-one-vote. If the majority of CPU power is controlled by honest nodes, the honest chain will be the one with the greatest proof-of-work. If an attacker controls the majority of CPU power, we call it a Majority attack, 51% attack or 51% cartel attack. Using a Majority attack an attacker can, e.g., censor transactions, do double spending attacks or perform denial of service attacks, but the attacker cannot create coins out of thin air or give herself a higher block reward.

In June 2014, the Ghash.io pool controlled more than 50% of the network's computation power. After this incident, Ghash.io promised that it will never launch a 51% attack and that it will not exceed 39,99% of the overall Bitcoin hash rate in future – and asked other pools to do the same.

It is often assumed that there is an incentive not to build a coalition controlling the majority of computational power to attack a cryptocurrency because this would endanger the long-term stake and the health of the currency. This assumption is mainly based on the idea that miners had to invest a lot in mining hardware and that it would be uneconomic to risk an attack. However, this assumption does not longer hold, as attackers can rent computation power or bribe other miners. One of these bribery attacks on Bitcoin was proposed by Bonneau [3]. He showed that an attacker could, for example, create a pool with negative mining fees, meaning miners would get overpaid for their effort. This would likely motivate profit-oriented miners to switch to the attacker's pool.

The website "PoW 51% Attack Cost" [4] lists the theoretical costs for a 51% attack for various cryptocurrencies. It shows that the theoretical costs to mount a 51% attack are rather small for various smaller cryptocurrencies. Table 1 shows an excerpt of selected cryptocurrencies with a market capitalization of 500.000 Dollar or more. The last two columns are the most relevant. They list the costs for a 51% attack per hour and how much of the necessary computation power can be rented via the popular crypto-mining marketplace Nicehash [5].

Name	Symbol	Market Cap	Algorithm	Hash Rate	1h Attack Cost	NiceHash-able
Expanse	EXP	$1,59 M	Ethash	95 GH/s	$60	6393%
Dubaicoin	DBIX	$1,75 M	Ethash	136 GH/s	$86	4461%
Paccoin	PAC	$3.10 M	X11	14 TH/s	$26	3986%
Metaverse	ETP	$43,79 M	Ethash	506 GH/s	$318	1201%
Karbo	KRB	$607504	CryptoNight	30 MH/s	$12	1047%
Callisto	CLO	$5,59 M	Ethash	732 GH/s	$460	830%

Table 1: Theoretical costs for a 51% attack (16.4.2019, Source: [4]).

2.2. Withholding Attacks

This section describes different block withholding attacks. In some cases, the attacker's goal is to enrich herself, while other attacks aim at reducing the profit for others. In this section, the Selfish Mining, the Block Withholding, the Fork After Withholding and the Finney attacks are introduced. There are also other forms of block withholding attacks, like "Cannibalizing pools" [6].

2.2.1. Selfish Mining

Eyal et al. [7] proposed in 2014 the selfish mining attack. In this attack, the attacker does not immediately propagate a block after finding it. Instead, she withholds the blocks and creates a private branch. In the meantime, the honest miners try to extend the longest chain they know. Ideally, shortly before the honest miners find a new block the attacker releases her block. Now the honest miners will switch to the attacker's branch. All efforts spent on the previous chain are wasted.
Furthermore, while the honest miners waste their resources, the attacker can secretly try to extend her chain. As the attacker cannot know when the honest miners will find a block, she has to estimate the optimal block withholding duration. This can be done based on the length of her chain, the length of the honest chain, her chosen risk and other factors like her relative computation power. This attack targets other miners and pools.

2.2.2. Block Withholding Attack

In this form of block withholding attack, the attacker must mine together with other miners, e.g., on a large pool. The idea is to withhold the solution (the block), if the attacker finds it. This way all miners lose their shares of this block reward. Although the attacker sabotages the pool, she still gets paid whenever other miners find the solution, as she submits all her solutions that do not allow to create a new block. This works, because pools assign all members much easier puzzles to solve than needed to create a new block. The idea is that one member will accidentally find a much better solution than needed and thus solve the block puzzle. This attack was introduced by Rosenfeld [8].

2.2.3. Fork After Withholding Attack

In a fork after withholding attack the reward for an attacker is always equal or greater than in a block withholding attack [9]. This attack combines selfish mining and a block withholding attack. The idea is to split the computation power into innocent and malicious mining. If the attacker finds a block as an innocent miner, she publishes it and gets her reward. If the attacker, as a malicious miner, finds a block she does not drop the result, as in the block withholding attack. Instead, she withholds the results. Afterward, three possible paths can be taken.

- First, if a different miner, not participating in the target pool, finds a block, the attacker immediately sends her result to the target pool. The pool will likely publish the result and generate a fork.

- Second, if an honest miner in the target pool finds a new block, the attacker discards her result.
- Third, if the attacker finds another block as an innocent miner, she discards the block found as a malicious miner.

In the first case, the attacker tries to generate a fork. If the other block is chosen by the network, she makes the same profit as with the block withholding attack. If her block is chosen, she gets a reward from the pool, which increases her profit. In the second case, the attacker makes the same profit as in the block withholding attack. In the third case, the attacker will get the block reward from the innocent mined block, thus making more profit than with a block withholding attack. In summary, the fork after withholding attack is always at least as profitable as the block withholding attack for the attacker.

2.2.4. Finney Attack

In a Finney attack, the attacker creates a block and includes a transaction, transferring some of her coins to a second address controlled by her. Instead of broadcasting this block she buys some goods from a merchant, accepting unconfirmed transactions. The attacker pays for these goods with a double spending transaction, spending the same coins as the transaction included in the not yet broadcasted block. After receiving the goods from the merchant, she immediately broadcasts her block and thus double spends her coins. This attack only works if she manages to find a block, buy goods from a merchant and publish her block before the honest network finds a competing block. This attack could, e.g., work against an automated online store, where the attacker controls the time of purchase and quickly gets her goods, e.g., by downloading them [10].

2.3. Goldfinger Attack

In this attack, the attacker wants to destroy a cryptocurrency or has some other incentive outside of the cryptocurrency's economy. Such an attacker could, for example, be a holder of a competing cryptocurrency, a government or someone holding a short position. This attack is named after the James Bond villain Auric Goldfinger, who tried to increase the value of his gold holdings by making the gold reserves in Fort Know radioactive. This attack was first described by Kroll et al. [11]. In this attack, the attacker is willing to invest and lose money or resources in order to destroy a cryptocurrency. This attack is not specific to Bitcoin or cryptocurrencies in general; It is a form of market manipulation. The key difference is that while market manipulation is prohibited in most countries, the cryptocurrency market still lacks regulations. Thus, an attacker can manipulate the cryptocurrency market in ways that would be illegal on other markets, like the stock market.

2.4. Feather-Forking

Miller [12] proposed an attack called Feather-Forking, where the attacker's goal is to blacklist a list of transactions. For this attack, the attacker has to be a miner, who publicity announces that she will not accept any block including one of the blacklisted transactions. Instead, she will fork the chain to prevent the inclusion of the blacklisted transaction. The attacker will continue the fork until it outraces the main branch or it falls more than k blocks behind. If the attacker falls too many blocks behind, she will concede the publication of the transaction. An attacker controlling less than 50% of the network's computation power will likely lose money, but at the same time will succeed in blocking the transaction with a nonzero probability. If the attacker can convince other miners that she is willing to lose money to prevent the inclusion of the blacklisted transactions, other miners could be willing to also censor these transactions in order to minimize their risk. Honest miners would lose their block reward, if the attacker succeeds with this attack. Thus, an attacker could be able to blacklist transactions at no cost, as long as other miners believe the attacker is willing to invest resources and perform a costly fork.

2.5. Flood Attack

A flood attack is a denial of service attack, where the attacker sends thousands of transactions to fill the mempool of Bitcoin nodes and subsequently fill blocks [13]. The idea is to fill blocks with spam transactions to prevent or at least slow down the processing of other transactions. This attack works because of the limited block size. Bitcoin specifies a maximal block size of one megabyte. This attack is typically expensive, as the attacker must pay a transactions fee, which is sufficiently high to ensure that her transactions are included into new blocks. Miners typically try to maximize their profit and include the transactions most lucrative to them. In 2015 an unknown attacker did a massive "stress test" on the Bitcoin network [14]. The attack resulted in over 80000 unprocessed transactions in the mempool at one time. For 2015 this was a huge amount of transactions. Figure 1 shows the number of unconfirmed transactions starting from May 2016. It can be seen that in 2016 the maximum number of transactions at one time was about 60000. Only recently in 2017, the number of transactions in the mempool reached its maximum with more than 180000 transactions waiting for their confirmation.

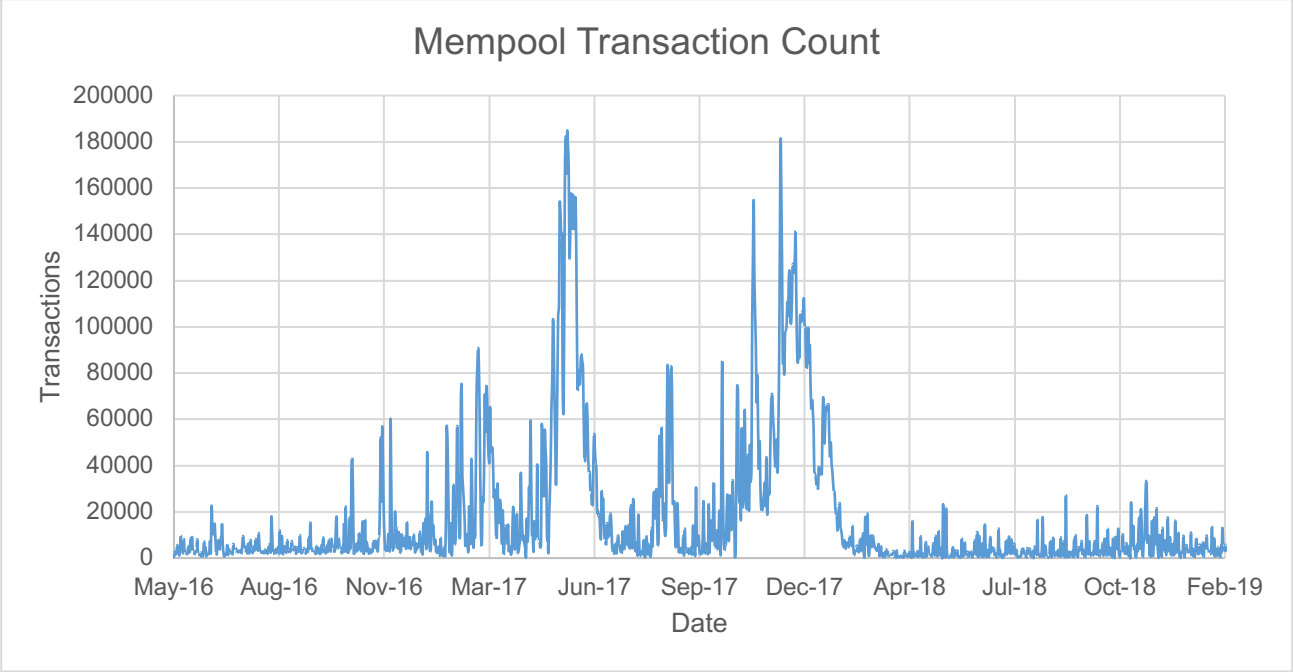

Figure 1: The number of transactions waiting to be included into a block (Source: blockchain.com [15])

2.6. High Fee Attack

In a high fee attack, an attacker creates a transaction T with a very high fee [16]. The idea is that at some point in time this transaction will be included into a block B by a miner M. Now, if the fee is high enough, it is more profitable, for all miners except M, to fork the current blockchain and replace block B with a self-mined block including the transaction T. Then again, for all other miners it is more profitable to fork the chain. This theoretical attack is quite expensive for the attacker but could disrupt the network.

2.7. Transaction Malleability

In Bitcoin, transactions are identified by their hash values. A transaction consists of data like the transaction inputs, transactions outputs, release conditions and a signature. Bitcoin uses ECDSA signatures. The problem is given a valid signature *sig = (r,s), sig'=(r,-s mod N)* is also a valid signature. This and other techniques allow to change the content of a transaction and thus its hash value without breaking the signature. In practice, this means every node or miner could change a given transaction T to a valid transactions T' with a different identifier. While this attack does not steal money and does not allow to spend someone else's Bitcoin, it still can become an issue in

certain situations or with buggy software. One example is the claimed attack on the crypto exchange MtGox were about 850000 bitcoins got lost or stolen.
When a user withdraws Bitcoin from MtGox, MtGox creates and publishes a transaction sending the desired amount of bitcoins to the user's address. MtGox used the transactions ID (hash) to track if a transaction was successfully included into a block. If not, the user got recredited the amount to her wallet at MtGox. By manipulating the send transaction, it was possible to receive the desired amount and at the same time, trick the exchange to recredit it to the user's wallet on the exchange. This way an attacker could withdraw her deposits multiple times. While MtGox claims that it lost about 850000 bitcoins, researchers concluded that it is unlikely. They concluded that only about 386 bitcoins could have been stolen from MtGox using the transaction malleability attack [17].
There have been several proposals for fixing this issue, but in the long term, Segregated Witness is supposed to solve this issue by not including the signature in the transaction ID calculation [18].

2.8. Dust Attack

The term "dust" refers to a tiny amount of cryptocurrency. This amount is usually so small that users and miners tend to ignore it. The amount is usually so small in value that it takes more fees to spend it than it is worth [19]. The smallest unit in Bitcoin is 1 Satoshi, which equals 0.00000001 Bitcoin. Values around a few couple of hundreds of Satoshi are often seen as dust.
In a dust attack, spammers send dust to many addresses and monitor these addresses for movements [20]. The goal of this attack is to deanonymize addresses by linking them and eventually determine the company or user behind the address.
Some cryptocurrency wallets allow marking the received dust transaction (the Unspent Transaction Output) as "do not spend" to prevent this attack. An example is the Samourai Wallet [21]. Furthermore, Peter Todd created a script to get rid of the dust [22]. Discussions about this attack date back to 2013 [23].

2.9. Programming Errors

Besides the previously introduced attacks, there were also incidents related to programming errors in Bitcoin implementations.
In the following section, we will introduce a selection of programming errors.

2.9.1. Integer Overflow

In 2010 someone created a transaction that sends in total 184,467,440,737.09551616 Bitcoins to two addresses and left 0.01 BTC for the miner [24] [25]. Each of these two addresses received, 92233720368.54277039 Bitcoins, which equals 9223372036854277039 Satoshi. This value is almost the maximum number a 64-bit Integer can hold. Thus, when the sum of these two outputs was calculated an Integer overflow occurred. As a result of the overflow, the sum of the outputs was smaller than the input and the transaction was considered valid. At that time, the Bitcoin software did not check for overflows and therefore accepted this malicious transaction. This incident was resolved using a soft-fork, that checked transactions for overflows and also made sure that no transaction has an output of more than 21 million Bitcoins [24].

2.9.2. Accidental Rule Change

Another programming error caused a fork in March 2013, at block 225430 [26]. This fork lasted for 6 hours (24 blocks) and resolved itself after block 225454. The reason for this fork was an accidentally rules change in the, back then latest release of the Bitcoin daemon. In this release, the database was switched to LevelDB. Earlier versions used the less efficient BerkeleyDB. The older Bitcoin daemons specified a default limit of 10000 locks for the BerkeleyDB. Block 225430 affected the status of so many transactions, that the defined lock limit prevented the processing of the block on older Bitcoin daemons while the back then newest version could process this Block, because the LevelDB had no such restriction. This led to an accidental fork, as older software versions did not accept this block, while the back then current version accepted it.

When the fork was detected the developers settled on the chain of the old nodes, because this chain could be processed by new and old nodes. After the developers made the decision, the fork was resolved by asking mining pools to downgrade to the older version. Thus, after some time the chain of the old nodes overtook the other chain. After that time also, the new nodes switched to the old chains, as it was the one with the most combined proof of work.

2.9.3. DOS – Inflation Bug - CVE-2018-17144

In September 2018 a severe denial-of-service (DOS) bug was discovered that could have been used to take down nodes or at worst temporarily crash parts of the network [27]. This bug could only be exploited by miners, which would risk losing their block reward. At this time the block reward was worth more than 65000 Euro. The bug was introduced in 2017 in version 0.14.0 of Bitcoin Core. Later it was announced that the bug was even more severe. It also allowed inflating the Bitcoin supply [28].
The root cause for both issues was an optimization which was added to Bitcoin Core 0.14. This optimization prevented a costly check, whether the same inputs are used multiple times in a transaction, during initial block re-relay validation. This check was added in 2012. Version 0.14 checked this via a sanity check assertion, which resulted in a crash if a transaction output is double spend in the same transaction. In Bitcoin Core 0.15 the tracking of the unspent transaction outputs was simplified and a resource exhaustion bug was corrected. During this redesign, an assertion was changed to verify only that the output exists. Before the redesign, the assertion also ensured that the output was previously unspent. Because of this redesign, a malicious miner could have double spent an output, already spent in the previous block, because then the transaction output would still be stored in an internal map, with the dirty bit set, but the double spend would not be detected as the assertion does not check the dirty bit. The denial of service bug and the inflation bug were fixed in version 0.16.3 and 0.17.0rc4 respectively.

3. Security Analyses

Analyzing the security of a cryptocurrency, or Bitcoin, in particular, is a very hard task. Among others, it is necessary to show that the used cryptography is secure, that the protocol is secure, that the majority of used implementations is secure, that the system is stable and robust but also to study the incentives. Bitcoin was built or at least published without a formally defined specification and a well-defined security model [29]. Satoshi calculated the probability that an attacker can generate an alternative chain faster than the remaining honest nodes. These results will be presented in Section 3.1. In Section 3.2 a game-theoretic model is introduced and in Section 3.3 we will introduce the results of a formal analysis of the backbone protocol of Bitcoin.

3.1. Whitepaper Calculations

Nakamoto [30] calculated in his whitepaper the probability that an attacker can catch up with the chain of the remaining honest nodes, depending on three variables. These variables are:

- ***p*** = the probability that an honest node finds the next block,
- ***q*** = the probability that the attacker finds the next block, and
- ***z*** = the number of blocks the attacker's chain is behind the honest chain.

Nakamoto calculated the probability q_z that an attacker will ever catch up after being ***z*** blocks behind. If the attacker succeeds she could perform denial-of-service attacks or e.g. double spend her coins but she cannot create coins out of thin air or take money from other users.

$$q_z = \begin{cases} 1 & if\ p \leq q \\ (p/q)^z & if\ p > q \end{cases}$$

Assuming that the majority of the network is honest, meaning $p > q$, the probability of success drops exponentially with the number of blocks the attackers has fallen behind. However, for users of Bitcoin, it may be more relevant to know how long they have to wait before being sufficiently sure

that the sender cannot change the transaction. For this calculation, Nakamoto assumed a Poisson distribution to model the attacker's potential progress. For more details, we refer to the Whitepaper [30]. Figure 2 shows the necessary confirmation depth to reduce the attacker's success probability to less than 0,1% dependent on the resources available to the attacker.

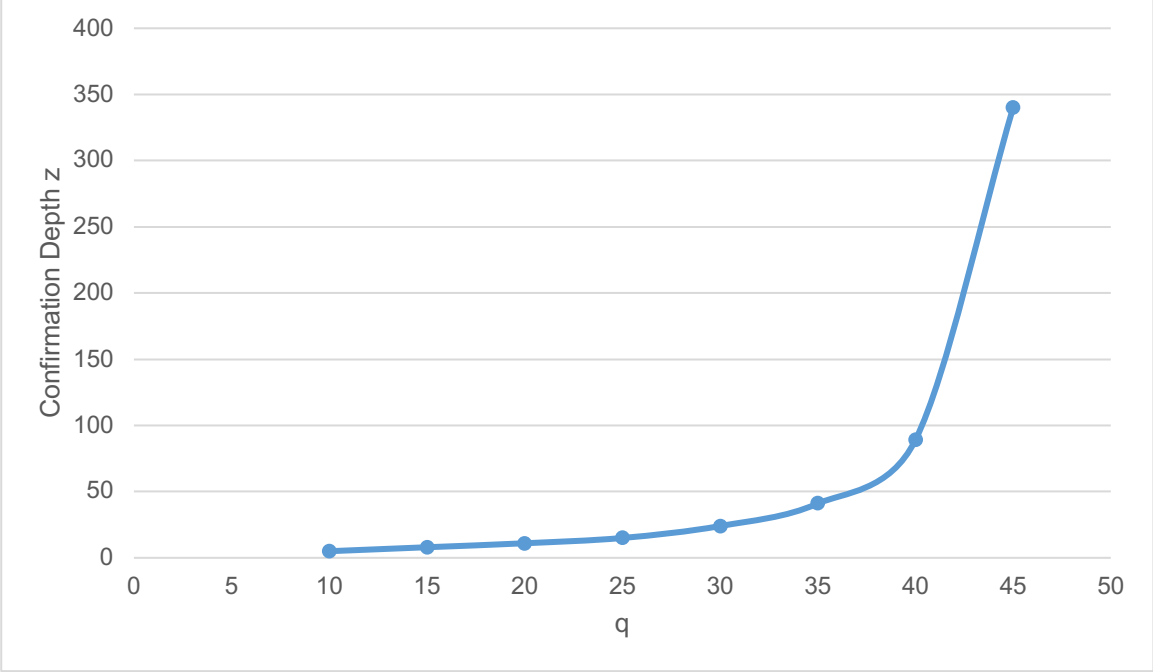

Figure 2: Necessary confirmation depth to reduce the probability of a successful attack to less than 0,1% depending on the attacker's relative computation resources ***q***

The results show that assuming an attacker that controls 10% of the networks computation power it is sufficient to have five confirmation blocks on top of the block containing the attacker's transaction. The more computation power the attacker controls, the more confirmations blocks are required to reduce the success probability of an attack to less than 0,1%. Figure 3 shows the success probability of a successful attack depending on the confirmation depth for an attacker controlling 10% of the network's computation power.

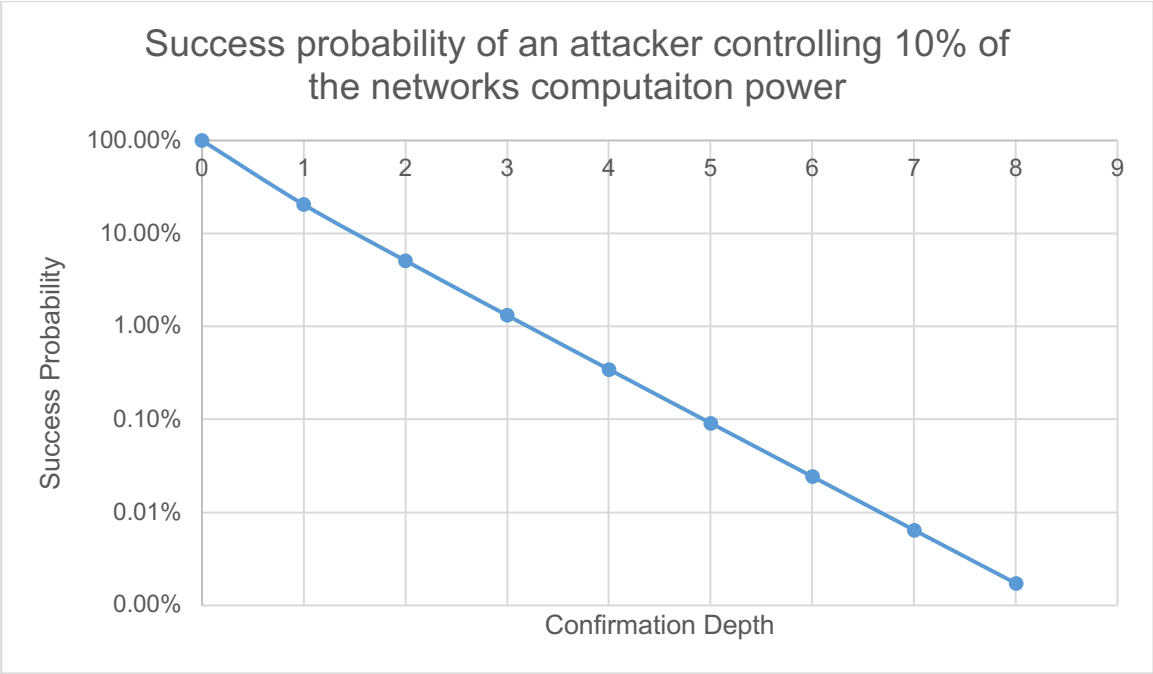

Figure 3: Success probability of an attacker controlling 10% of the network's computation resources to successfully double-spend coins depending on the confirmation depth. Note the logarithmically scaled y-axis.

3.2. Game-Theoretic Model

Game theory is used to model interactions between rational decision-makers. These mathematical models can be applied to different fields of social science as well as computer science. Game theory models consist of at least the following three components:

- **Players** who make decisions.
- A **strategy** backing up the decisions.
- A **payoff**, which represents the outcome of the strategy.

Several researchers used game theory to model certain aspects of Bitcoin. Examples are the works of Dimitrie [31] who modeled Bitcoin mining as a contest and Dey [32], who used machine learning and algorithmic game theory to stop majority attacks.
In the following section, we will introduce a game theoretic model for the so-called P+ε attack. This model is intended to demonstrate the basic idea of game-theoretic models in the context of cryptocurrencies.

3.2.1. P + ε Attack

In this section, an example attack proposed by Andrew Miller [33] is taken up. He looks at the theoretical cryptocurrency SchellingCoin, which relies on the community to determine the correct answer to a question. An example question could be: "Is the earth flat?". Every user can vote yes or no and the majority answer is considered to be the correct answer. To motivate the users to participate the system pays a reward P to users, who voted with the majority, while other users get nothing. If the majority of voters is honest, this model works flawlessly, because each user will vote for the correct answer in order to get the reward and assume that the majority will follow this approach. Table 2 summarizes the payoff or all possible cases.

	You vote **NO**	You vote **YES**
Others yote **NO**	P	0
Others vote **YES**	0	P

Table 2: The payoff in the SchellingCoin game

Next, we introduce a malicious player, which bribes other players. The aim of the attacker is to encourage other players to vote for the wrong answer. In this example, the correct answer is NO. Therefore, the attacker's goal is to bribe users to vote YES. To achieve this, the attacker offers users who vote YES a bribe of P+ε. Table 3 shows the updated payoffs for the game with a bribing attacker.

	You vote **NO**	You vote **YES**
Others yote **NO**	P	P+ε
Others vote **YES**	0	P

Table 3: The payoff in the SchellingCoin game with a bribery attacker

Now every profit-oriented player will vote YES, because this promises a reward in both cases. If the other users vote NO the reward is P+ε and if the other users vote YES the reward is P. Assuming that the users are not dominated by altruists, the majority will vote YES. This means the attacker succeeds with the attack and does not have to pay the bribe, meaning the attacker has no costs for this attack. This simple example demonstrates how game-theoretic models can help to analyze the security of cryptocurrencies, by modeling players as rational actors. As example, game theory is well suited to design and analyze incentive mechanisms.

3.3. Robust Transaction Ledger

Garay et al. [34] and Kiayias et al. [35] [36] have shown that chain growth, chain quality and common prefix are essential requirements for a robust transaction ledger, which ensures Liveness and Persistence. Showing that a proposed blockchain solution fulfills the chain growth, chain quality, and common prefix property is a good and popular technique applied in many security analyses. Typically, it is easier to show that these three properties are fulfilled than directly showing that Liveness and Persistence are always given. In the following sections, we will introduce these properties.

3.3.1. Chain Growth Property

The chain growth property informally says, that if two honest nodes are queried for their chain at two different points in time, then the later queried chain will be longer than the earlier queried. Essentials this says that a blockchain must grow, meaning blocks have to be added.

3.3.2. Chain Quality

In principle, the chain quality states that given any portion of at least *l* blocks of a chain possessed by an honest party, then no more than a fraction of μ blocks have been created by an adversary, where μ is the fraction of computing resources controlled by the adversary.

3.3.3. Common Prefix

The common prefix property states that if two chains are taken from two honest parties at two different time slots, then both chains will share a common prefix. This essentially says that both nodes share a common blockchain up to a block.

3.3.4. Persistence

Persistence means, that once a transaction is considered stable, other nodes, if queried, will either report the transaction in the same block and position or will not report any other transaction in conflict (e.g., double spending transaction) as stable. A transaction is considered stable, if it is part of a block in the ledger that is more than a defined number of blocks deep. Typically, wallets classify a transaction as stable if at least six blocks confirm it.

3.3.5. Liveness

Liveness ensures, loosely speaking, that any valid transaction will be included into the ledger before *t* time slots and that after this time all honest nodes will report this transaction as being stable. For a more formal definition, we refer the reader to Garay et al. [34] and Kiayias et al. [35] [36]. Basically, this property ensures that every valid transaction will eventually be included in the blockchain. Bitcoin is an example of a blockchain that emphasizes on liveness [37].

4. Conclusions

Blockchain-based cryptocurrencies are a relatively new technology that is not yet completely understood. In the last years, several attacks have been found, but none of them had a serious impact on Bitcoin. So far, the Bitcoin community was able to quickly fix bugs and prevent major ramifications. Some smaller cryptocurrencies weren't able to protect themselves and failed. Some other cryptocurrencies are still highly vulnerable to certain attacks like the majority attack [4].
At the same time, many researchers focus on blockchain-based systems and cryptocurrencies. While these systems tend to be complex, it can be assumed that these systems will be understood better and better. So far, most cryptocurrencies were designed without a thorough security analysis. Typically, an existing solution is forked and features are added or adapted. This led to many attacks and as a consequence to failed cryptocurrencies. With recent research insights, it should become easier to create thorough security analyses. Now it is clear that a robust system needs to fulfill at least the chain growth, chain quality, and common prefix property. This will result in improved systems with better security analyses and as a consequence higher security guarantees.

References

[1] A. Marshall, "Combined Crypto Market Capitalization Races Past $800 Bln," 2018. [Online]. Available: https://cointelegraph.com/news/combined-crypto-market-capitalization-races-past-800-bln. [Accessed 15 03 2019].

[2] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System.*

[3] J. Bonneau, "Why buy when you can rent? Bribery attacks on Bitcoin consensus," *FC 2016 Workshops,* vol. LNCS 9604 , pp. 19-26, 2016.

[4] Crypto51, "PoW 51% Attack Cost," 2019. [Online]. Available: https://www.crypto51.app. [Accessed 10 04 2019].

[5] niceHASH, "Largest Crypto-Mining Marketplace," 2019. [Online]. Available: https://www.nicehash.com. [Accessed 15 04 2019].

[6] Blockgeeks, "Hypothetical Attacks on Cryptocurrencies," 2018. [Online]. Available: https://blockgeeks.com/guides/hypothetical-attacks-on-cryptocurrencies/#What_is_Cannibalizing_pools. [Accessed 29 03 2019].

[7] I. Eyal and E. G. Sirer, "Majority Is Not Enough: Bitcoin Mining Is Vulnerable," in *Financial Cryptography and Data Security*, Berlin, 2014.

[8] M. Rosenfeld, "Analysis of Bitcoin Pooled Mining Reward Systems," 2011. [Online]. Available: https://arxiv.org/pdf/1112.4980. [Accessed 09 04 2019].

[9] Y. Kwon, D. Kim, Y. Son, E. Y. Vasserman and Y. Kim, "Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin," *CoRR,* vol. abs/1708.09790, 2017.

[10] bitcoinj, "Understanding the bitcoinj security model," [Online]. Available: https://bitcoinj.github.io/security-model#finney-attacks. [Accessed 20 03 2019].

[11] J. A. Kroll, I. C. Davey and E. W. Felten, "The economics of Bitcoin mining, or Bitcoin in the presence of adversaries," in *Proceedings of WEIS*, 2013.

[12] A. Miller, "Feather-forks: enforcing a blacklist with sub-50% hash power," 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=312668.0. [Accessed 10 04 2019].

[13] Bitcoin Wiki, "Flood attack," 2015. [Online]. Available: https://en.bitcoin.it/wiki/Flood_attack. [Accessed 29 03 2019].

[14] Bitcoin Wiki, "July 2015 flood attack," 2017. [Online]. Available: https://en.bitcoin.it/wiki/July_2015_flood_attack. [Accessed 29 03 2019].

[15] Blockchain Luxembourg S.A., "Mempool Transaction Count," 2019. [Online]. Available: https://www.blockchain.com/de/charts/mempool-count?timespan=all. [Accessed 29 03 2019].

[16] S. Dziembowski, "Mining Pools and Attacks," 2016. [Online]. Available: https://www.slideshare.net/vpnmentor/mining-pools-and-attacks. [Accessed 29 03 2019].

[17] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," in *Computer Security - ESORICS 2014*, Cham, 2014.

[18] W. McChesney and A. Menghrajani, "Bitcoin Transaction Malleability in 2018," [Online]. Available: https://www.quaxio.com/Bitcoin_Transaction_Malleability_in_2018.pdf. [Accessed 21 03 2019].

[19] A. Hertig, "Bitcoin Dust: What It Is and Why You Should Get Rid of It," 11 4 2018. [Online]. Available: https://www.coindesk.com/bitcoin-dust-tell-get-rid. [Accessed 07 02 2019].

[20] Binance Academy, "What Is a Dusting Attack?," [Online]. Available: https://www.binance.vision/security/what-is-a-dusting-attack. [Accessed 9 2 2019].

[21] Samourai Wallet, 10 2018. [Online]. Available: https://twitter.com/SamouraiWallet/status/1055345822076936192. [Accessed 9 2 2019].

[22] P. Todd, "Clean your Bitcoin-QT wallet of unwanted dust," [Online]. Available: https://github.com/petertodd/dust-b-gone. [Accessed 9 February 2019].

[23] Varous users on bitcointalk.org, "Someone sending out MilliBits," 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=282238.0. [Accessed 9 February 2019].

[24] Bitcoin Wiki, "Value overflow incident," 2016. [Online]. Available: https://en.bitcoin.it/wiki/Value_overflow_incident. [Accessed 2019 03 21].

[25] J. Garzik, "Strange block 74638," 15 08 2010. [Online]. Available: https://bitcointalk.org/index.php?topic=822.0. [Accessed 21 03 2019].

[26] V. Buterin, "Bitcoin Network Shaken by Blockchain Fork," 2013. [Online]. Available: https://bitcoinmagazine.com/articles/bitcoin-network-shaken-by-blockchain-fork-1363144448/. [Accessed 21 03 2019].

[27] A. Hertig, "The Latest Bitcoin Bug Was So Bad, Developers Kept Its Full Details a Secret," 21 08 2018. [Online]. Available: https://www.coindesk.com/the-latest-bitcoin-bug-was-so-bad-developers-kept-its-full-details-a-secret. [Accessed 21 03 2019].

[28] Bitcoin Core, "CVE-2018-17144 Full Disclosure," 20 09 2018. [Online]. Available: https://bitcoincore.org/en/2018/09/20/notice/. [Accessed 21 03 2019].

[29] J. Lopp, "Bitcoin's Security Model: A Deep Dive," 2016. [Online]. Available: https://www.coindesk.com/bitcoins-security-model-deep-dive. [Accessed 22 03 2019].

[30] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed 02 03 2018].

[31] N. Dimitri, "Bitcoin Mining as a Contest," *Ledger,* vol. 2, pp. 31-37, 2017.

[32] S. Dev, "A Proof of Work: Securing Majority-Attack in Blockchain Using Machine Learning and Algorithmic Game Theory," *International Journal of Wireless and Microwave Technologies(IJWMT),* vol. 5, pp. 1-9, 2018.

[33] V. Buterin, "The P + epsilon Attack," 2015. [Online]. Available: https://blog.ethereum.org/2015/01/28/p-epsilon-attack/. [Accessed 10 04 2019].

[34] J. Garay, A. Kiayias and N. Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," in *Advances in Cryptology - EUROCRYPT 2015*, Berlin, 2015.

[35] A. Kiayias and G. Panagiotakos, "Speed-Security Tradeoffs in Blockchain Protocols.," *IACR Cryptology ePrint Archive,* 2015.

[36] A. Kiayias, A. Russell, B. David and R. Oliynykov, "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol," in *Advances in Cryptology -- CRYPTO 2017*, Cham, 2017.

[37] S. W. Kim, "Safety and Liveness—Blockchain in the Point of View of FLP Impossibility," 25 05 2018. [Online]. Available: https://medium.com/codechain/safety-and-liveness-blockchain-in-the-point-of-view-of-flp-impossibility-182e33927ce6. [Accessed 04 2019].