6-22-2020

Double-Spending Analysis of Bitcoin

Kaylash Chaudhary
The University of the South Pacific, Kaylash.chaudhary@usp.ac.fj

Vishal Chand
The University of the South Pacific, vishal.chand@usp.ac.fj

Ansgar Fehnker
University of Twente, Netherlands, ansgar.fehnker@utwente.nl

Double-Spending Analysis of Bitcoin

Completed Research Paper

Kaylash Chaudhary
The University of the South Pacific
Suva, Fiji
kaylash.chaudhary@usp.ac.fj

Vishal Chand
The University of the South Pacific
Suva, Fiji
vishal.chand@usp.ac.fj

Ansgar Fehnker
University of Twente
Enschede, Netherlands
ansgar.fehnker@utwente.nl

Abstract

Bitcoin is a distributed online payment system that facilitates anonymous transactions using a peer-to-peer network without a central trusted authority. Every peer in the Bitcoin network keeps the collection of all transactions which is referred to as a ledger. This public ledger will work effectively for honest peers, however, one well-known attack is the fifty-one percent or majority attack. This paper provides an Uppaal model of the Bitcoin protocol focusing on its three important components namely transactions, blocks and the blockchain. It presents a probability analysis for two scenarios of the fifty-one percent attacks. Two Phase Proof-of-Work (2PPoW) is a proposed solution to address attacks of this type, and we will extend the model to include 2PPoW and calculate the probability of a successful attack. The analysis shows that a traditional fifty-one percent attacks can be successful even if the attacker has less than a majority of the processing pool.

Keywords: Bitcoin, Double-spending, Model, Verification, Probability

Introduction

In circulation since 2009, Bitcoin is a well-known cryptocurrency [Nakamoto 2009]. At the end of 2017, there were around 16,774,500 Bitcoins in circulation [Blockchain.com. (2020)] with a total value equivalent to USD 237,670,437,837, making it the most popular digital currency. Bitcoin's popularity is a result of its capability to eliminate the need for a trusted third party such as a broker or a bank to process payments.

Bitcoin operates using a peer-to-peer network. Every peer in the Bitcoin network keeps the collection of all transactions which is referred to as a ledger. This ledger is organised into separate blocks all which are linked to their immediate predecessor forming a chain. The protocol uses a proof-of-work solution to induce a unique order on blocks, a process also known as mining. This work was initially done by peers, called miners. Since the difficulty of finding a solution to the proof-of-work increased over the years, peers started working together in groups to solve the challenges. Such a group is called a pool.

This paper will provide a formalisation of the Bitcoin protocol focusing on its three important components, namely transaction, blocks and blockchain. We use Uppaal for its ability to perform statistical model checking [David et al. 2015] to analyse the success probability of fifty-one percent attacks depending on the relative size of a mining pool. The attack succeeds, i.e. an attacker can spend a Bitcoin twice if it manages to retroactively change the order of blocks. This is possible if a pool encompasses more than half of the global processing power. We will analyse this type of attack for different sizes of the mining pool, and two scenarios of a majority attack. The two-phase proof-of-work (2PPoW) was introduced as counter measure for the majority attack [Eyal et al. 2014]. We will extend the model to include 2PPoW and analyse whether it successfully reduces the probability of a successful attack.

The next section will discuss related work. Section 4 will introduce the Bitcoin protocol, the proof-of-work mechanism, and the scenarios for the fifty-one percent attack. Section 5 will introduce the Uppaal Model, and present the analysis results. Section 6 will present the model and analysis results for the two-phase proof-of-work. The paper concludes with Section 7.

Recent 51% Attack Cases

There have been some recent attacks which were caused by the processing the power of the mining pools. In January 2020, Bitcoin Gold (BTG) blockchain encountered two 51% attacks withing 6 hours, resulting in over $70,000 worth of BTG being double spent [Cointelegraph (2020)]. These attacks involved hitting the hard-fork of Bitcoin by deep reorganization of over 10 blocks.

In May 2019, two mining pools, BTC.com and BTC.top, combined to stop a malicious miner who created a fork in the main chain [Investopedia, 2020] [99 Bitcoins. (2020).]. The intention was to steal coins but the malicious miner has to race against the network and make other mining pools start constructing blocks on its chain. The two mining pools combined their processing powers to increase their chance of success. Although, this shows that 51% attack was done to help the network, it also shows that 51% attack is still possible.

In January 2019, another cryptocurrency, Ethereum Classic (ETC), suffered a double spending attack [99 Bitcoins. (2020)]. Coinbase, one of the biggest cryptocurrency companies around 42 countries, detected a blockchain reorganisation which resulted in the company to suspend all ETC transactions.

In May 2018, approximately $18 million were stolen through exchange companies. Bitcoin Gold suffered a 51% attack from an unknown malicious person [Forum.bitcoingold.org. (2020)].

Related Work

Andrychowicz et. al. modelled Bitcoin contracts using timed automata [Andrychowicz et al. 2014]. It considers that in the presence of distrusting parties, the protocol should be able to ensure fairness. They showed that an honest party does not lose any Bitcoins no matter how the dishonest party behaves. Bergstra et. al. identified research questions related to Bitcoin and other information money [Bergstra et al. 2013]. Ron et. al. performed a quantitative analysis of the Bitcoin transaction graph [Ron et al. 2012]. Herrmann implemented and evaluated a double-spending attack [Herrmann 2012], however, they did not include blockchain forking.

A precursor to this work was presented in [Chaudhary et al. 2015]. It focused on blockchain forking and included a detailed model of the blockchain. This paper abstracts from the identity of a block, allowing us to analyse more elaborate attacks, and the 2PPoW scheme. In [Fehnker et al. 2018] a simplified version of the model was presented to analyse a type of majority attack.

Rosenfield presents the success probabilities of double-spending attack in the standard Bitcoin protocol using stochastic processes [Rosenfeld 2014]. The results suggest that if the attacker controls more hashrate than the honest network then no confirmation depth will reduce the success rate of double-spending.

Bae and Lim presents a random mining group selection technique to reduce the probability of successful double-spending attacks [Bae et al.].

Lee et al. presents a new approach for avoiding double-spending attacks via the concept of recipient-oriented concepts in private blockchain networks [Lee et al.]

This paper uses a model checker, Uppaal, to calculate probabilities of double-spending attack in the standard Bitcoin and also the 2PPoW Bitcoin protocol. All papers presented in this section does not calculate the probability of double spending based on 51% attack except work presented by Rosenfield [Rosenfield 2014] which mathematical equations with assumptions. This difference is that this paper uses actual values of hashrate and difficultly to calculate probabilities of success.

The Bitcoin Protocol

The Bitcoin protocol proposed by Nakamoto is described as a peer-to-peer electronic cash system, that is decentralised and based on cryptographic principles [Nakamoto 2009]. In this online payment system, a payer sends payment directly to a payee, without any involvement of trusted third party, such as a bank or a broker that is used in traditional payment systems to guard against double-spending. There exists no central trusted authority. The validity of transitions is ensured by public/private key cryptography, while unique order of transactions is determined by a so-called *mining* process.

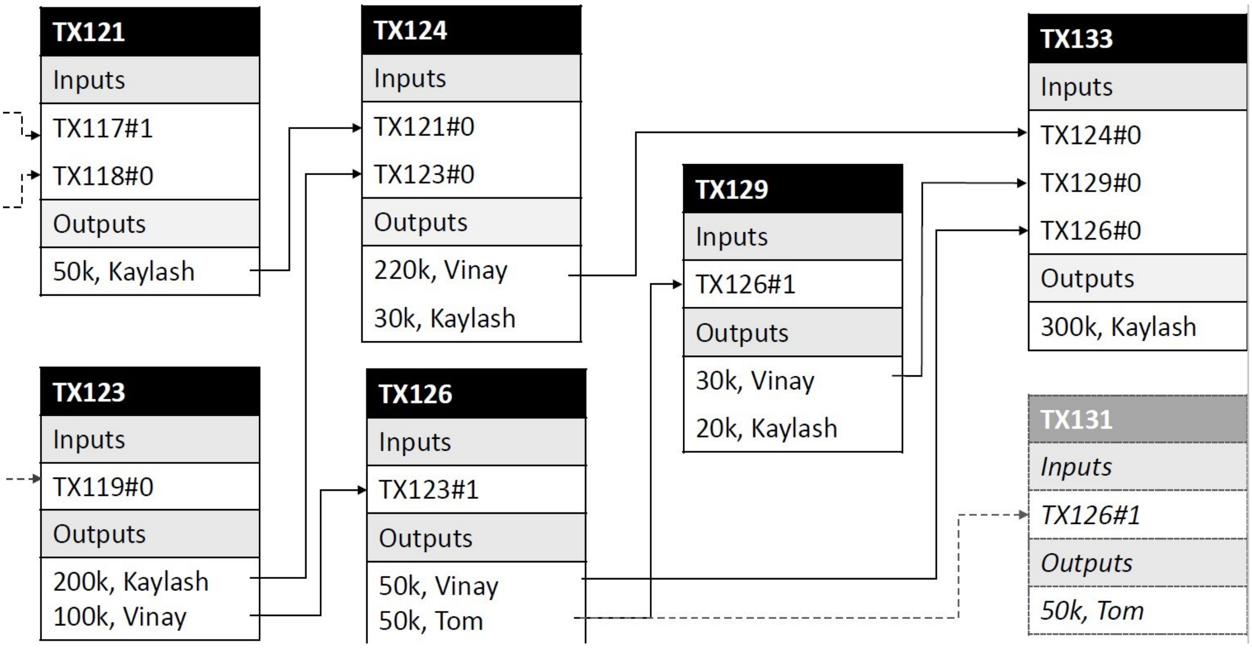


Figure 1. Transaction Graph

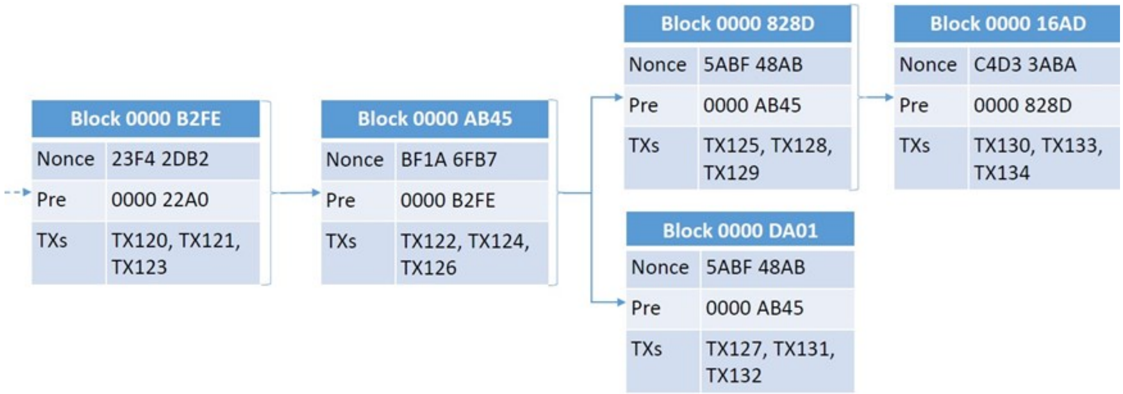
Transactions

There are two types of transactions: coin-base and regular transactions. Coin-base transactions are used for new Bitcoins, whereas regular transactions are used for transferring existing Bitcoins from one user to another. In this paper, we consider regular transactions.

Each transaction has one or more transaction inputs and one or more outputs. A transaction input is a reference to an output of a previous transaction, which proves that senders possess the Bitcoins they claim to have. Each transaction output specifies an amount of Bitcoins and a recipient of that amount. A transaction thus specifies a number of previous transactions, the amount received in those transactions, and how the amount received will be distributed among the recipients. A transaction can include a fee for the miner, a detail which we leave out in the remainder. It is assumed that the sum of the amounts in the inputs (plus the fee) is equal to the sum of the amount in the outputs. If a user wants to keep the change, the change is one of the outputs, where the user is also the recipient.

Figure 1 gives an example of a transaction graph. In this example transaction *TX124* has two inputs: a reference to the first output of transaction *TX123* and one to the only output of transaction *TX121*. The total amount these transactions to user Kaylash is 250k. The amount is expressed in Satoshis (1 BTC is 100 000 000 Satoshi). Transaction *TX124* has two outputs, the first is for 220k for user Vinay, and the second for 30k for user Kaylash. The first output is in turn an input of transaction *TX133*, and thus spent. The second has not been used as input and is thus unspent. Note, that in the Bitcoin protocol references to outputs are hashes, while the identity of a user is a public key. The example uses for simplicity integers and strings to identify transactions and names for user.

To guard against double spending each output can only be used once as an input. This means that transactions *TX129* and *TX131* cannot both be part of the transition graph, since they both spend the second output of *TX126*. The protocol needs a mechanism to decide which transaction came first, and which of these transactions needs to be disregarded. Due to the distributed nature of the Bitcoin network it cannot be assumed that all nodes agree on an order of events. To impose an order on transactions, and thus to prevent double spending, the Bitcoin uses a so-called *blockchain*.



Block 0000 B2FE	
Nonce	23F4 2DB2
Pre	0000 22A0
TXs	TX120, TX121, TX123

Block 0000 AB45	
Nonce	BF1A 6FB7
Pre	0000 B2FE
TXs	TX122, TX124, TX126

Block 0000 828D	
Nonce	5ABF 48AB
Pre	0000 AB45
TXs	TX125, TX128, TX129

Block 0000 16AD	
Nonce	C4D3 3ABA
Pre	0000 828D
TXs	TX130, TX133, TX134

Block 0000 DA01	
Nonce	5ABF 48AB
Pre	0000 AB45
TXs	TX127, TX131, TX132

Figure 2. Transaction Graph

Blockchain

A block contains a (reference to a) set of transactions, a header, and the hash of the predecessor block. This hash serves as reference to the predecessor in the chain. Transactions in the same block can be considered to have happened at the same time. Transactions in different blocks are ordered using the predecessor relation between blocks. Transactions are only confirmed if they appear in some block, unconfirmed transactions are kept in the *transaction pool*.

Any node can select transactions from the transaction pool, and create a block to append it to the end of the blockchain. Provided that it manages to complete a so-called *proof-of-work*.

For the *proof-of-work* the node has to randomly select a *nonce* for a block. This nonce is part of the block header. The hash of the block (including this nonce) is calculated and the result is compared with a target value. If the hash of the block header is lower than the target value, then the proof-of-work is completed; otherwise the node repeats this process until it finds a nonce that results in a hash value lower than the target value. The target value can be thought of as a required number of leading zeros of the hash value. The target value determines the expected amount of work required to complete the proof-of-work. The entire process is called *mining*. Once a nonce is found, the block becomes valid and is broadcasted to the network.

Due to the distributed nature of the network, with different *miners* working on different blocks, the blockchain may have forks. There is a possibility that two blocks are created and broadcasted over the network simultaneously. Some peers might receive the first block first, and other peers the second one. In this situation, peers will continue building the chain on the block they received first.

The longest path is considered to be the path containing the confirmed transitions. Transactions in blocks that are not in the longest chain are added back to the pool of transactions, and they can be used to build new blocks. Usually, a miner will try to extend the longest chain, since they will only be rewarded for adding new blocks to the longest chain.

Although discouraged, selection of an older block is possible and will result in a shorter branch. There is a possibility that a branch can become the longest, and thus change the validity of transactions. However, this is only likely if the fork happens towards the end of the chain since miners that work on the shorter branch have to compete with the rest of the network that is working on extending the longest chain. The transactions in the block at the end of the chain should not be considered confirmed as there is a non-negligible possibility that another chain will become the longest. The distance of a block to the end of the chain is called the confirmation depth, and it is advised to wait for a confirmation depth of 6 before considering a transaction in a block to be confirmed [Bonneau et al. 2015].

Figure 2 gives an example of a blockchain that includes the transaction of Fig. 1. Transaction *TX126* is included in *Block 0000 AB45*, and transaction *TX129*, which uses an output of 50k for Tom from *TX126*, is in *Block 0000 828D*. Transaction *TX131* cannot be included in this block, or any of its successors, since any output can only be used once. It would however be possible that another miner includes *TX131* in a block that forks o of *Block 0000 AB45*. If this fork become the longest, it would mean that *TX131* would be considered valid, instead of *TX129*, thus that the 50k would have gone to Tom, instead of being shared between Vinay and Kayash. However, for this to happen the miners would have to successfully outcompete the rest of the network that works on extending the last block of the longest chain, *Block 0000 16AD*.

Hash-Rate

Initially, ordinary peers could participate in the mining process. Since the difficulty of finding a solution to the proof-of-work increased over the years, peers started working together in groups to solve the challenges. Such a group is called a pool. Today, there are a number of pools that participate in the Bitcoin network.

The network *hash-rate* (hashes per second) is a measure for the processing power of the Bitcoin network. The *target value* of the proof-of-work is automatically adapted after every 2016 blocks, to ensure that it is not too easy for the network to solve the challenges with the current network hash-rate. A decrease in target value means an increase in difficulty. The aim is to have a confirmation time of about 10 minutes for the entire network. In 2017, the actual confirmation time was about 12 minutes [Blockchain.com. (2020)].

The expected time taken to find a solution by a particular pool is based on the hash-rate of that pool. The hash-rate of a pool is relative to hash-rate of the network. A hash-rate of r 2 [0,1], means that the pool alone would find 1 block in 12/r minutes.

In April 2018, the largest pool *BTC.com* had a hash-rate of 20%, which means that the expected time for this pool to find a solution would be about 50 minutes. The second-largest pool is currently *AntPool*, with a hash-rate of 15%. These numbers are subject to fluctuation; in 2014 pool *Ghash.io* neared at some point in time a hash rate of 50% of the network hash-rate [Cawret 2014].

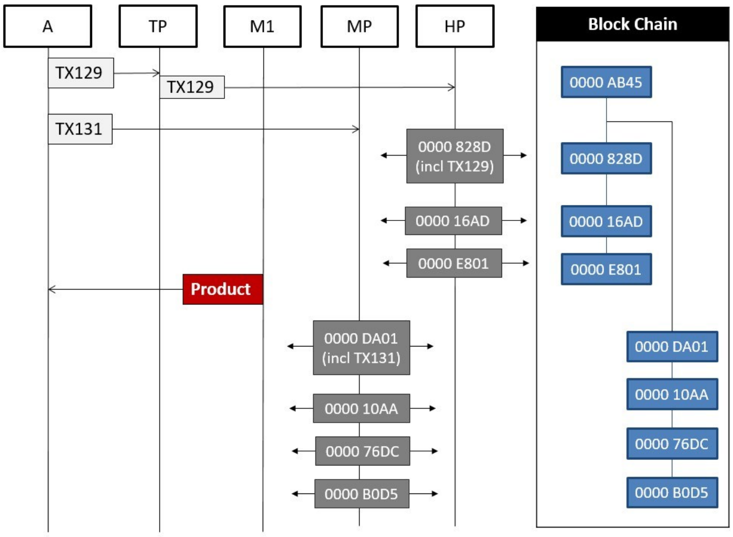


Figure 3. Fifty-one Percent Attack

Fifty-one percent attack

The fifty-one percent attack is a double spending attack, and also known as majority attack. 51% (or majority) refers to the share a pool has of the total hash rate. If the attacker controls more than half of the network, no value for the confirmation depth will be able to stop an attacker from succeeding. The reason behind the attacker's success is that the attacker can generate blocks faster than the honest pools.

Figure 3 shows an example of a fifty-one percent attack. The attacker buys a product from merchant *M1* using transaction *TX129*. It is added to the transaction pool TP, and then used by the honest pool *HP* to build a block. The attacker creates another transaction *TX131* to pay themselves spending the same Bitcoin. This transaction is given directly to the malicious pool *MP*. The honest pool *HP* will then confirm blocks *828D* (which includes *TX129*), *16AD* and *E801*.

The malicious pool waits for the confirmation depth that is used by the merchant, in this example for three confirmations. The merchant will then release the product or service, assuming that the payment is confirmed and final. While waiting, the malicious pool can either wait or secretly mine blocks that will not be broadcasted. Once the product or service is received by the attacker, the malicious pool races against the network to extend its chain and make it longer than the chain constructed by the honest pool. If the attacker succeeds the transaction in this fork will be considered confirmed. Once this happens, transaction *TX129*, which was considered confirmed, will have been invalidated.

If the malicious pool holds more than 50% of the hash-rate, the attack will succeed, no matter the confirmation depth. This paper will investigate the probability of a successful attack, even if the malicious pool has less than 50%. It will consider two scenarios, and also analyse a proposed two-phase proof-of-work solution, intended to reduce the risk of a fifty-one percent attack.

Bitcoin Model and Analysis

This section focuses on the formal modelling of the Bitcoin protocol, developed in the UPPAAL modelling language. It also presents the results of our probability analysis for double-spending attacks. We consider the probability of a successful attack for two scenarios, depending on the computational power of the pools and confirmation depth.

In the first scenario, the malicious pool tries to catch up the main chain from behind. The race starts when the vendor releases the product to the attacker, i.e. after the confirmation depth. The second scenario is where the malicious pool will not wait for the depth of confirmation. Instead, it will construct the chain secretly and will only broadcast this chain when the required confirmation depth has been exceeded.

Uppaal Model

For this model, we will be using Uppaal SMC [David et al. 2015]. The basic model uses Uppaal's networks of timed automata, with control locations, discrete variables, and clocks. Uppaal offers synchronisation via binary handshake or broadcast channels. Uppaal-SMC also includes probabilistic choice, uniformly distributed delay over a range, and stochastic delay using an exponential distribution.

The key components of the Bitcoin protocol are transactions and blocks of transactions. A transaction is modelled as a transaction number, and an input and output transaction. In the Bitcoin protocol a transaction contains more information, like the amount. However, for the probability of a successful attack these are not relevant. Consequently, every wallet of each peer will store only the unique ID of the recipient, and for the input the ID of the recipient. The model will not store blocks, but only the current block number.

Figure 4 shows the Uppaal model for double-spending attack. The model uses one channel, broadcastSolution, to broadcast blocks to pools and peers. It uses one metavariable, to exchange block details when broadcasting a block solution, and the second variable for the malicious peer to place the malicious transaction.

We model the protocol as a composition of an honest pool, a peer, a malicious pool, and a malicious peer. The honest pool has only one control location, except for a committed initial location that is used for initialisation. The honest pool keeps track of the length of main chain and its fork. If the length of the main branch is above the confirmation depth, and the length of the fork longer than the main branch, then the attack succeeds. Variable blockDifference is used to track the difference between the two chains. If the first chain is extended then one is added to the variable blockDifference otherwise one is subtracted from blockDifference.

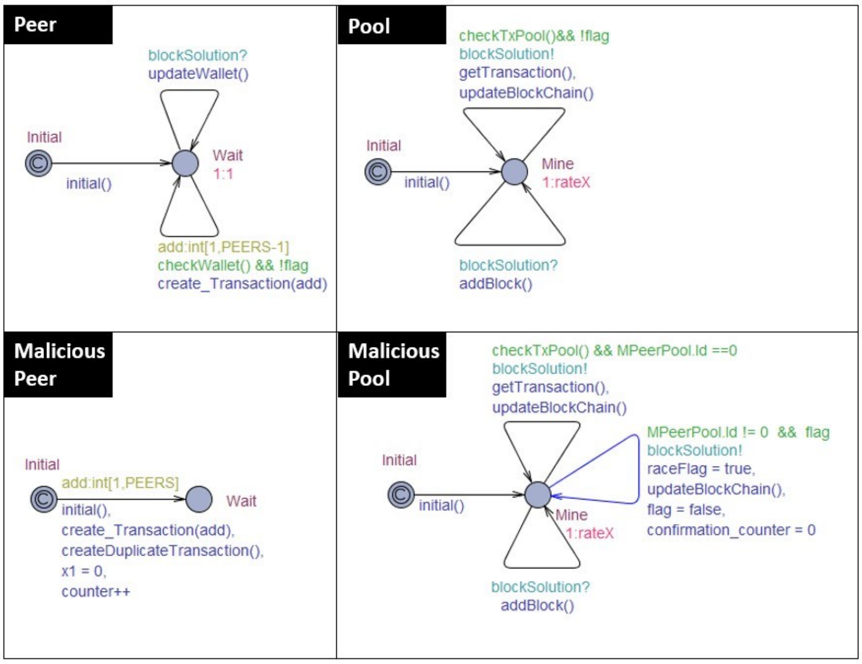


Figure 4. Model for Double Spending Attack

The honest pool has only two edges, one for broadcasting blocks and the other for receiving blocks. The rest is enabled if there is a transaction in the transaction pool (*checkTxPool ()*) and flag is false. Variable flag is global and will be set to true by the pool automaton when the number of blocks built by the honest pools exceeds confirmation depth. This is included for the scenario where the malicious pool starts late. This edge makes two updates: *getTransaction()* and *updateBlockChain()*. The first update copies a transaction from the transaction pool to a local variable. The second updates the block number and the transaction details to a metavariable.

The second edge does not have any guard, as it models that another pool has confirmed a block. This transition will add the block to the blockchain, i.e. increment the length of the corresponding fork.

The control location *Mining* of the pool waits with an exponential distribution in-between transitions that model broadcasts of new blocks. The rate of this distribution reflects the hash-rate of the pool; the higher the rate, the more frequent the pool is expected to broadcast a solution. We use equation (1) to obtain the rate from the hash-rate.

$$Time = difficulty * 2^{32}/hash - rate \qquad (1)$$

The automaton modelling the peer also has only one main control location. It has two edges from this Wait location. The first edge models that a peer has to update its wallet if a transaction with their ID has been confirmed. The second is used to spend Bitcoins in the wallet, i.e. it adds transactions to the transaction pool (*create_Transaction(add)*), provided there are bitcoins in the wallet (guard: *checkWallet()*).

The automaton for the malicious pool differs in that it includes one extra edge that models construction of the fork. For the first scenario, the construction will start as soon as the main chain reached the confirmation depth. For the second scenario mining for blocks will start immediately. However, in this case, the model uses a different channel, secretSolution, that will not send the solution to honest peers. It will only be added to the blockchain if the main chain reaches the confirmation depth.

The model for the malicious pool also includes a Boolean ag raceSuccess to indicate that the race has been won.

For the second scenario, there are two variables in the malicious pool automaton that keeps track of the length of the chains: lengthChain1 and lengthChain2. Since there will be secret chain construction, success does not depend on the difference between the two chains. It depends on whether the length of chain 2 is greater than chain 1 and that chain 1 length is greater than the number of confirmation depth. Once this happens, variable raceSuccess will be set to true.

The model for the malicious peer only has one edge, modelling a double spending attempt by creating two transactions of the same input transactions (updates: *create_Transaction(add)* and *createDuplicateTransaction()*).

Results

For the analysis, we checked the model for different confirmation depths and different hash-rates. For all instances, we executed the following query for sixty thousand steps:

Pr[#<=60000](<> MPool.raceSuccess)

where Pr is the probability, MPool is the malicious pool and raceSuccess is a boolean variable in malicious pool. The results will give a 95% confidence interval for the success of an attack. Table 1 shows the probability of double-spending for the first scenario, while Table 2 gives the results for the second scenario.

Table 1. Probability of Double Spending – Scenario 1

HashRate (%)			**Depth**			
	1	**2**	**3**	**4**	**5**	**6**
5	[0.02,0.12]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
10	[0.08,0.18]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
15	[0.13,0.23]	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
20	[0.21,0.31]	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
25	[0.28,0.38]	[0.1,0.2]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
30	[0.37,0.47]	[0.13,0.23]	[0.07,0.16]	[0,0.1]	[0,0.1]	[0,0.1]
35	[0.5,0.6]	[0.23,0.33]	[0.13,0.23]	[0,0.1]	[0,0.1]	[0,0.1]
40	[0.63,0.73]	[0.39,0.49]	[0.21,0.31]	[0.13,0.23]	[0.07,0.17]	[0.05,0.14]
45	[0.79,0.89]	[0.59,0.69]	[0.48,0.58]	[0.38,0.48]	[0.36,0.46]	[0.31,0.41]
50	[0.9,1]	[0.9,1]	[0.9,1]	[0.9,1]	[0.9,1]	[0.9,1]

Table 2. Probability of Double Spending – Scenario 2

HashRate (%)			**Depth**			
	1	**2**	**3**	**4**	**5**	**6**
5	[0.02,0.12]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
10	[0.1,0.2]	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
15	[0.2,295007]	[0.05,0.14]	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]
20	[0.26,0.36]	[0.07,0.17]	[0.02,0.12]	[0.01,0.11]	[0,0.1]	[0,0.1]
25	[0.27,371556]	[0.09,0.19]	[0.03,0.13]	[0.03,0.13]	[0,0.1]	[0,0.1]
30	[0.46,0.56]	[0.31,0.41]	[0.22,0.32]	[0.14,0.24]	[0.13,0.23]	[0.09,0.19]
35	[0.57,0.67]	[0.47,0.57]	[0.46,0.56]	[0.36,0.46]	[0.34,0.44]	[0.33,0.43]
40	[0.62,0.72]	[0.58,0.68]	[0.45,0.55]	[0.41,0.51]	[0.37,0.46]	[0.34,0.44]
45	[0.8,0.9]	[0.72,0.82]	[0.69,0.79]	[0.66,0.76]	[0.65,0.75]	[0.6,0.7]
50	[0.9,1]	[0.9,1]	[0.9,1]	[0.9,1]	[0.9,1]	[0.9,1]

As expected, a malicious pool with a 50% hash-rate have a close to 100% ([0.9,1]) chance of a successful attack for any number of confirmation depth. However, a pool with an even lower hash-rate can be successful. With a hash-rate of 45%, there is a reasonable chance of success, even for a confirmation depth of 6. For smaller confirmation depths, even smaller pools can be successful. For example, for a depth of 1, even a pool with 30% has a reasonable chance of success. The probabilities of success achieved using Uppaal mode checker is comparable with results presented by Rosenfeld [Rosenfeld 2014]. The next section will analyse whether these probabilities change for the 2PPoW protocol.

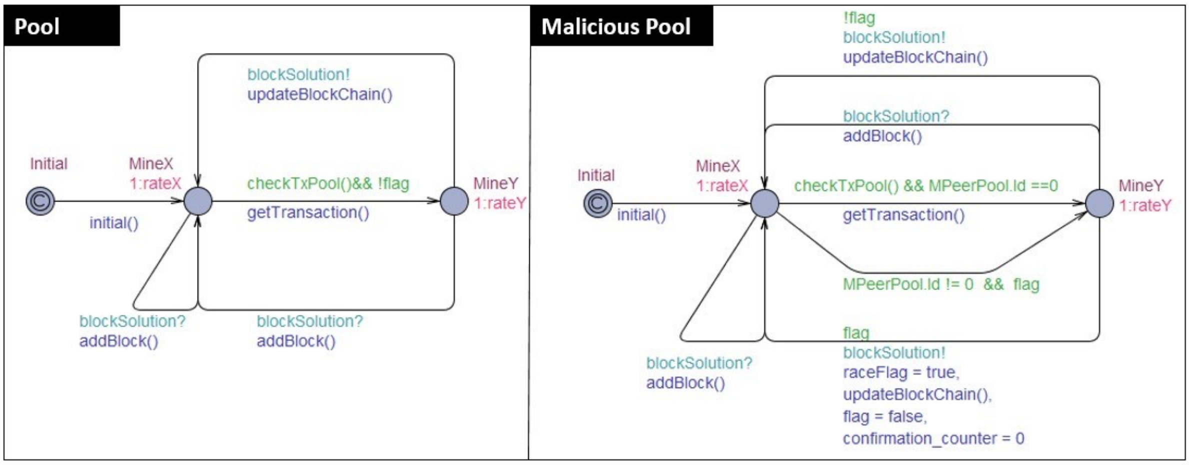


Figure 5. 2PPoW Pool Model

Two-Phase Model and Analysis

Large mining pools increase the probability of double spending in the Bitcoin network. These pools can use their processing power to race against the network and confirm a malicious transaction. To disadvantage large pools, Eyal and Sirer proposed a solution called the *Two Phase Proof-of-Work (2PPoW)* [Eyal et al. 2014]. It exploits the fact that even large pools will only own a fraction of the mining hardware themselves. They propose combining the mining challenge with a second challenge that miners will only want to solve on their own trusted hardware.

The first challenge is the same as the current Bitcoin challenge. A pool has to find a nonce, such that the hash of the header is less than the target value. The mining pools will have to solve this puzzle first. Once the miners have successfully solved the first challenge (found a nonce), the miner will have to sign the block with their private key and produce a hash that is less than the second target value. The private key is the same key that is used for payment, and miners will only solve these on their own trusted mining hardware.

The 2PPoW protocol specifies two target values, X for the first challenge, and Y for the second challenge. By choosing these values it is possible to choose to what extent a miner has to rely on trusted hardware. A high value of Y means that the pools can make use of publicly available resources, and then solve the second part without effort. A low target value of Y means the miner will have to rely more on its own hardware to solve the second more difficult challenge. In this section, we will investigate how the share of trusted hardware will influence the overall probability of success of a 51% double-spending attack.

Uppaal Model

There are four parties involved in the double spending attack, the honest pool and peer, and the malicious pool and peer. The models for the peer and malicious peer automaton remains the same. Since pools will now be solving two challenges, the pool and malicious pool automaton will change.

The pool automaton, shown in Figure 5, has three tasks: solving the first challenge, computing the hash for the second challenge and broadcasting the solution to other pools and peers. It has three locations. Location Initial is where the automaton begins. Location MineX and MineY model the first and second challenge respectively. Each is labelled with an exponential rate to model the time taken to find a solution for the respective challenge. After solving the second challenge, the pool will broadcast the block solution to other pools and peers. Other pools who lost the race, will receive the block solution via broadcast channel blockSolution. The automaton will transit from location Initial to MineX and will initialise the chains declared in Pool template. The automaton will stay at MineX location for some time. We assume here that the first puzzle will be solved (first task). The rate, 1:rateX, given at this location will cater for the time taken to solve the first puzzle. Then the automaton will transit to location MineY. While transiting it will get a transaction from the transaction pool which has been declared globally. Again, the automaton will stay at location MineY for some time (second task). This indicates the time taken to solve the second challenge. The pool will then broadcast the block solution to other pools and peers (third task). Function *updateBlockChain()* updates the block detail to a metavariable in the global declaration. This broadcast will take this pool to location MineX.

Other pools who did not succeed in solving both challenges will receive the block solution via the broadcast on channel blockSolution. The function *addBlock()* updates the required chain using the values present in the metavariable. If a pool was still solving the first challenge, it will make a transition back to the same location, that is, MineX. If a pool successfully solved the first puzzle and was solving the second one, it will make a transition from location MineY to location MineX. This shows the end of the current and start of the new mining process.

The malicious pool automaton includes two extra transitions. One from location MineX to location MineY with guard MPeerPool.Id != 0 && flag. Global variable MPeerPool is used by the malicious peer to place the malicious transaction. The malicious pool retrieves the transaction and resets it. Variable flag handles the turn-based scenario as explained in the previous section. The second transition is from location MineY to location MineX with condition flag. This transition will broadcast the block solution on channel blockSolution. It will update the global metavariable and will set variable raceFlag to true which indicates that the race has started. This transition is used only once and that is when the malicious transaction needs to be included in a block and then into blockchain. Figure 5 depicts the pool automaton for the first scenario. In the second scenario, the malicious pool will start the race in secret immediately, instead of waiting for the confirmation.

Results

In case of 2PPoW, a pool will have untrusted and trusted hardware. Untrusted hardware represents other miners that work for that pool whereas trusted hardware is the pool's own hardware. Suppose the overall network can do 100 hashes per second. The malicious pool has 40 hashes per second in total - 30 hash per second untrusted and 10 hashes per second trusted. Suppose that there are three other pools, each has a total of 20 hash per second, of which 10 hashes per second trusted. In that case, all pools use their total for the first challenge. The malicious pool will use 40 hashes per second and each honest pool will use 20. The hash-rate for the first challenge would be 40%, and 20% respectively. For the second challenge, the malicious pool, and the other 3 pools each use 10 hashes per second. That means all pools have 25% of the trusted hash-rate.

We used the query in Section IV to calculate the probability of double spending for both scenarios.

To see the effects of 2PPoW, we divided the analysis into two parts. In the first part, we analysed double spending by fixing the hash-rate for the second challenge. All pools will have different hash-rate for the first challenge and same hash-rate for the second challenge. Then we analysed double spending by having a variable hash-rate for the second challenge. All pools will have different hash-rates for both first and second challenge.

Second Challenge Fixed Hash-Rate

Tables 4 and 5 show the probabilities of double spending for different hash-rates (first challenge) until a depth of 6 block confirmation. There are four pools: three honest (HP) and one malicious pool (MP). The tables show hash-rate for X parameter difficulty and the probabilities. The hash-rate for Y parameter difficulty is equal for all pools, which is 25%. For example, Table 4 shows that for malicious pool (MP) with 50% hash-rate and honest pools (HP) with 50% (P1 has 15%, P2 has 15% and P3 has 20%) hash-rate, the probability of double-spending for one confirmation depth will be [0.65,0.75].

The results from both tables show a slight decrease in probabilities, that is, double spending is still possible. This is because all pools had the same percentage hash-rate for the second challenge.

Table 3. Internal Capacity of Pools for 10 Experiments

MP			HP					
#	Rate X	Rate Y	P1-X	P1-Y	P2-X	P2-Y	P3-X	P3-Y
1	5	50	35	20	35	20	25	10
2	10	45	35	20	30	25	25	10
3	15	40	30	25	30	25	25	10
4	20	35	30	25	25	20	25	20
5	25	30	30	25	25	25	20	20
6	30	25	30	25	20	25	20	25
7	35	20	25	30	20	25	20	25
8	40	15	20	30	20	30	20	25
9	45	10	20	30	15	40	20	20
10	50	5	15	40	15	40	20	15

Table 4. Probability of Double Spending in 2PPoW with Same Percentage Hash-Rate Y – Scenario 1

	HP			Depth					
MP	P1	P2	P3	1	2	3	4	5	6
5	35	35	25	[0.03,0.13]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
10	35	30	25	[0.05,0.15]	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
15	35	25	25	[0.15,0.25]	[0.03,0.13]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
20	30	25	25	[0.21,0.31]	[0.07,0.17]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
25	30	25	20	[0.28,0.38]	[0.11,0.21]	[0.04,0.14]	[0,0.1]	[0,0.1]	[0,0.1]
30	30	20	20	[0.29,0.39]	[0.12,0.22]	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]
35	25	20	20	[0.46,0.56]	[0.21,0.31]	[0.06,0.16]	[0.02,0.12]	[0.01,0.11]	[0,0.1]
40	20	20	20	[0.54,0.64]	[0.26,0.36]	[0.12,0.22]	[0.09,0.19]	[0.02,0.12]	[0,0.1]
45	20	15	20	[0.61,0.71]	[0.42,0.52]	[0.22,0.32]	[0.18,0.28]	[0.05,0.15]	[0.03,0.13]
50	15	15	20	[0.65,0.75]	[0.53,0.63]	[0.41,0.51]	[0.28,0.38]	[0.25,0.35]	[0.12,0.22]

Second Challenge Variable Hash-Rate

The 2PPoW assumes that smaller pools own more mining hardware relative to their overall pool size. Table 3 shows the hash-rates for 10 experiments with different internal capacities of pools. The results presented in Tables 6 and 7 are based on the experiments provided in Table 3.

Table 3 shows that for experiment 10, malicious pool has a 50% hash-rate for first challenge (X) while the honest pools have 50% (P1 has 15%, P2 has 15% and P3 has 20%) hash-rate. Also for the second challenge (Y), the malicious pool has 5% hash-rate and the honest pools have 95% hash-rate (P1 has 40%, P2 has 15% and P3 has 15%). As shown in Table 6, the probability for experiment 10 and for one confirmation depth is [0.07,0.17].

Table 5. Probability of Double Spending in 2PPoW with Same Percentage Hash-Rate Y – Scenario 2

	HP			Depth					
MP	P1	P2	P3	1	2	3	4	5	6
5	35	35	25	[0.01,0.11]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
10	35	30	25	[0.07,0.17]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
15	35	25	25	[0.15,0.25]	[0.02,0.12]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
20	30	25	25	[0.22,0.32]	[0.09,0.19]	[0.03,0.13]	[0.01,0.11]	[0,0.1]	[0,0.1]
25	30	25	20	[0.31,0.41]	[0.16,0.26]	[0.06,0.16]	[0.01,0.1]	[0,0.1]	[0,0.1]
30	30	20	20	[0.41,0.51]	[0.16,0.26]	[0.11,0.21]	[0.06,0.16]	[0.02,0.12]	[0,0.1]
35	25	20	20	[0.45,0.55]	[0.31,0.41]	[0.22,0.32]	[0.15,0.25]	[0.10,0.20]	[0.06,0.16]
40	20	20	20	[0.57,0.67]	[0.43,0.53]	[0.27,0.37]	[0.24,0.34]	[0.17,0.27]	[0.13,0.23]
45	20	15	20	[0.62,0.72]	[0.48,0.58]	[0.40,0.50]	[0.31,0.41]	[0.25,0.35]	[0.17,0.27]
50	15	15	20	[0.70,0.80]	[0.53,0.63]	[0.41,0.51]	[0.34,0.44]	[0.26,0.36]	[0.22,0.32]

Table 6. Probability of Double Spending in 2PPoW with Different Percentage Hash-Rate Y – Scenario 1

	MP		Depth					
#	Rate X	Rate Y	1	2	3	4	5	6
1	5	50	[0.06,0.16]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
2	10	45	[0.19,0.29]	[0.06,0.16]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
3	15	40	[0.29,0.39]	[0.15,0.25]	[0.05,0.15]	[0.01,0.11]	[0,0.1]	[0,0.1]
5	20	35	[0.16,0.26]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
5	25	30	[0.33,0.43]	[0.14,0.24]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
6	30	25	[0.33,0.43]	[0.09,0.19]	[0.02,0.12]	[0,0.1]	[0,0.1]	[0,0.1]
7	35	20	[0.3,0.4]	[0.07,0.17]	[0.03,0.13]	[0.02,0.12]	[0,0.1]	[0,0.1]
8	40	15	[0.29,0.39]	[0.06,0.16]	[0.03,0.13]	[0.02,0.12]	[0,0.1]	[0,0.1]
9	45	10	[0.22,0.32]	[0.03,0.13]	[0.02,0.12]	[0,0.1]	[0,0.1]	[0,0.1]
10	50	5	[0.07,0.17]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]

Table 7. Probability of Double Spending in 2PPoW with Different Percentage Hash-Rate Y – Scenario 2

	MP		Depth					
#	Rate X	Rate Y	1	2	3	4	5	6
1	5	50	[0.06,0.16]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
2	10	45	[0.19,0.29]	[0.06,0.16]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
3	15	40	[0.27,0.37]	[0.15,0.25]	[0.05,0.15]	[0.01,0.11]	[0,0.1]	[0,0.1]
4	20	35	[0.41,0.51]	[0.3,0.4]	[0.15,0.25]	[0.08,0.18]	[0.06,0.16]	[0.02,0.12]
5	25	30	[0.35,0.45]	[0.2,0.3]	[0.08,0.18]	[0.06,0.16]	[0.05,0.15]	[0,0.1]
6	30	25	[0.41,0.51]	[0.16,0.26]	[0.11,0.21]	[0.06,0.16]	[0.02,0.12]	[0,0.1]
7	35	20	[0.31,0.41]	[0.2,0.3]	[0.09,0.19]	[0.02,0.12]	[0,0.1]	[0,0.1]
8	40	15	[0.24,0.34]	[0.18,0.28]	[0.03,0.13]	[0.02,0.12]	[0,0.1]	[0,0.1]
9	45	10	[0.2,0.3]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]
10	50	5	[0.07,0.17]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]	[0,0.1]

The overall result shows that the probability of double spending has been significantly reduced in 2PPoW given that the large pools will own less mining hardware. The purpose of the 2PPoW was to disintegrate large pools. This has been shown by the calculated probabilities. In the standard Bitcoin protocol, the probability was almost 1 when the malicious pool occupied more than 50% of the Bitcoin network. The probability in 2PPoW was almost 0. Even the pool with higher hash-rate was not able to generate many blocks. So there are no incentives for the miners in that pool to stay. Therefore, this pool will definitely break which will in-turn reduce double-spending.

The results so far show that 2PPoW depends on a pools second challenge hash-rate. It will prevent double spending if large pools own less mining hardware. But, 2PPoW will fail if large pools can also have more trusted hardware.

We also investigated the effect of decreasing the difficulty for the second challenge. The results showed that a decrease in the difficulty for the second challenge increases the probability of double spending. If the difficulty is increased, then the time to solve the second challenge increases.

Conclusion

This paper provided a formalisation of the Bitcoin protocol focusing on its three important components namely transaction, blocks and blockchain. We used Uppaal to analyse the probability of success of a fifty-one percent attack, depending on the relative size of a mining pool, and the confirmation pool. The attack succeeds, i.e. an attacker can spend a Bitcoin twice if it manages to retroactively change the order of blocks.

We found that this is possible, with a reasonable chance even if the hash-rate is below the 50% threshold, and especially for smaller confirmation depths.

The two-phase proof-of-work (2PPoW) was introduced as counter-measure for the majority attack [Eyal et al. 2014]. We extended the model to include 2PPoW and found that the 2PPoW does not make that much of a difference unless the second challenge is much more difficult than the rest. But that means to change bitcoin such that the amount of trusted hardware count instead of hardware in general, and it is not clear if this would change the chances of a miner getting too much leverage. The models used in this paper are available on http://repository.usp.ac.fj/id/eprint/10092 .

References

99 Bitcoins. 2020. *51% Attack Explained Simply + Real Life Example (2020 Updated)*. [online] Available at: https://99bitcoins.com/51-percent-attack [Accessed 2 Feb. 2020].

Andrychowicz, M., Dziembowski, S., Malinowski, D., & Mazurek, Ł. 2014. “Modeling bitcoin contracts by timed automata,” In *International Conference on Formal Modeling and Analysis of Timed Systems, Springer, Cham*, pp. 7-22

Bae, J., & Lim, H. 2018. “Random mining group selection to prevent 51% attacks on bitcoin,” In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 81-82. IEEE.

Bergstra, J. A., & de Leeuw, K. 2013. “Questions related to Bitcoin and other Informational Money,”arXiv preprint arXiv:1305.5956.

Blockchain.com. 2020. *Bitcoin Charts & Graphs - Blockchain*. [online] Available at: https://blockchain.info/charts [Accessed 2 Feb. 2020].

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., & Felten, E. W. 2015. “Sok: Research perspectives and challenges for bitcoin and cryptocurrencies,” In *2015 IEEE Symposium on Security and Privacy*, IEEE, pp 104-121

Chaudhary, K., Fehnker, A., Van De Pol, J., & Stoelinga, M. 2015. “Modeling and verification of the bitcoin protocol,” *arXiv preprint arXiv:1511.04173*.

Cointelegraph. 2020. *Bitcoin Gold Blockchain Hit by 51% Attack*. [online] Available at: https://cointelegraph.com/news/bitcoin-gold-blockchain-hit-by-51-attack-leading-to-70k-double-spend [Accessed 29 April. 2020].

David, A., Larsen, K. G., Legay, A., Mikučionis, M., & Poulsen, D. B. 2015. "Uppaal SMC tutorial," *International Journal on Software Tools for Technology Transfer*, *pp* 397-415.

Eyal, I., & Sirer, E. G. 2014. "How a mining monopoly can attack bitcoin," *Hacking, Distributed, June*.

Eyal, I., & Sirer, E. G. 2014. "How to disincentivize large bitcoin mining pools," *Blog post: http://hackingdistributed. com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools*.

Fehnker, A., & Chaudhary, K. 2018. "Twenty percent and a few days–optimising a bitcoin majority attack," In *NASA Formal Methods Symposium*, Springer, Cham, pp 157-163

Forum.bitcoingold.org. 2020. *Double Spend Attacks on Exchanges*. [online] The BTG Community Forum. Available at: https://forum.bitcoingold.org/t/double-spend-attacks-on-exchanges/1362 [Accessed 2 Feb. 2020].

Herrmann, M. 2012. *Implementation, evaluation and detection of a doublespend-attack on Bitcoin* (Master's thesis, ETH Zürich, Department of Computer Science).

Investopedia. 2020. *51% Attack*. [online] Available at: https://www.investopedia.com/terms/1/51-attack.asp [Accessed 2 Feb. 2020].

Karame, G. O., Androulaki, E., Roeschlin, M., Gervais, A., & Čapkun, S. 2015. "Misbehavior in bitcoin: A study of double-spending and accountability," *ACM Transactions on Information and System Security*, pp 1-32.

Lee, H., Shin, M., Kim, K. S., Kang, Y., & Kim, J. 2018. "Recipient-Oriented Transaction for Preventing Double Spending Attacks in Private Blockchain," In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. pp. 1-2. IEEE.

Nakamoto, S. 2009. "Bitcoin: A peer-to-peer electronic cash system," White Paper, 2008.

Ron, D., & Shamir, A. 2012. "Quantitative analysis of the full Bitcoin transaction graph," Cryptology ePrint Archive.

Rosenfeld, M. 2014. "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*.