Double-spend Attack Models with Time Advantage for Bitcoin

Carlos Pinzón and Camilo Rocha

Escuela Colombiana de Ingeniería Julio Garavito, Bogotá, Colombia
Pontificia Universidad Javeriana, Cali, Colombia

Abstract. Bitcoin is a digital currency in which the need for a trusted third party is voided. Instead, this digital currency is based on the concept of ‘proof of work’ allowing users to execute payments by digitally signing their transactions. Since electronic files can be duplicated, fraudulent transactions in the form of double-spend attacks – where users spend the same money at least twice – can happen. This paper is about attack models that can assign possible time advantage to attacker agents in the Bitcoin network. In particular, this paper presents: (i) two attack models in which partial advancement towards block production can be influenced by time and not only by the hashrate used to produce blocks of hashes, and (ii) algorithmic experimentation comparing these models against existing well-known hashrate-based attack models that do not consider time advantage. As a conclusion, this paper presents evidence on the fact that advantages are not negligible for cases in which an attacker has had enough time for secretly mining fraudulent blocks. Also, the models presented in this paper help in contrasting some previous claims about analyzing and detecting double-spend attacks in Bitcoin.

1 Introduction

Bitcoin is a digital currency. As of May 2016, it is worth US $7200+ million in the form of 15.8 million bitcoins, approximately; this digital currency is expected to reach 5+ million users by 2019. Bitcoin uses a scheme in which the need for a trusted third party is voided: instead, this digital currency is based on the concept of ‘proof of work’ allowing users to execute payments by digitally signing their transactions using hashes through a distributed time-stamping service. Since electronic files can be duplicated and Bitcoin does not have a trusted third-party that can verify whether a digital coin has been spent, fraudulent transactions with users spending the same money at least twice can happen. These fraudulent schemes are known as double-spend attacks and have taken place already in the Bitcoin network [6].

Some mathematical models have been proposed to analyze how feasible really are double-spend attacks in Bitcoin. In particular, S. Nakamoto [7] and M. Rosenfeld [9] have proposed hashrate-based attack models in which, overall, an attack is successful whenever the number of confirmations of a honest transaction is surpassed by the number of confirmations of a fraudulent one. In this
setting, hashrate means that the model considers how fast a block header can be hashed, while the number of confirmations refers to the number of blocks of transactions claiming a transaction to be valid. In particular, these two models are used to answer the following question:

1. What is the probability of a double-spend attack with $K$ confirmations given that an attacker has mined $n$ blocks?

The models of S. Nakamoto and M. Rosenfeld produce very similar answers for Question (1), despite certain differences between them at the technical level. One important aspect of the models of S. Nakamoto and M. Rosenfeld is the fact that blocks of transactions are considered either non-existent or complete, but never partially complete. That is, these models do not consider blocks to be partially built, neither honest nor attack ones. More importantly, these models lack the expressiveness needed to analyze situations in which an attacker has probably already built fraudulent blocks because it had some time advantage. These features seem important in an attack model given the fact that, at some point, a honest block might just start being built from scratch while competing against an attacker that may have been already mining for a certain period of time. In this case, for instance, such a situation could result in an undetected attack in the attack models of S. Nakamoto and M. Rosenfeld.

This paper addresses the following more general research question:

2. What is the probability of a double-spend attack with $K$ confirmations given that an attacker has mined $n$ blocks and, in addition, has been mining for $t$ time units?

For answering Question (2), this paper presents two double-spend attack models for Bitcoin in which an attacker can be assigned some advantage in terms of time units. The first model presented in this paper results from generalizing the model of M. Rosenfield by adding an extra parameter denoting some time advantage assigned to the attacker, assumed to be used in producing and mining fraudulent blocks. The second model, following a completely different approach, is a purely time-based model that takes into account the times at which both honest and attacker nodes last mined a block, in addition to their relative progress in terms of the length of chains already mined. In this paper, the former model is called the generalized model and the latter is called the time-based model.

Both the generalized and the time-based models are hashrate-based attack models in which partial advancement towards block production is influenced by time, and not only by the hashpower used to produce blocks of hashes. The equations governing these models use an Erlang probability distribution, which is specially suited for modeling waiting times in queueing systems [12]. This contrasts with the models of S. Nakamoto that uses Poisson probability distribution and of M. Rosenfeld that uses a negative binomial probability distribution. The choice of this distribution can be justified because waiting times between occurrences of the event of producing a block can be naturally considered to be Erlang distributed, even if these events occur independently with some average
rate that can be modeled with a Poisson process. Furthermore, since partial block production is directly tied to the amount of time spent on mining a block, it is more of a continuous process than a discrete one. Because of these reasons, the Erlang probability distribution seems well-suited for both models.

This paper presents some results of extensive experimentation performed on the generalized and time-based attack models. As a consequence of this experimentation, there is strong evidence indicating that these two attack models coincide and behave very much like the attack model by M. Rosenfeld, but produce more information because of the time advantage inclusion. That is, on the one hand, this paper presents evidence that partial block production is not negligible for analyzing and detecting double-spend attacks in Bitcoin. On the other hand, experimental data supports the claim that the time-based model presented by the authors is correct with respect to the existing attack model proposed by M. Rosenfeld, which is surprising given the intrinsic differences between these attack models. Furthermore, the models contributed by the authors in this paper witness the observation already made by M. Rosenfeld stating that the double-spend attack model proposed by S. Nakamoto was not entirely accurate [9].

In summary, this paper contributes two double-spend attack models for Bitcoin, more general than existing models commonly used in the analysis of double-spend attacks, and reports on extensive experimentation performed on them. To the best of the authors’ knowledge, double-spend attack models for Bitcoin that incorporate time have never been reported before in the literature. Furthermore, the experimentation performed on both the generalized and the time-based models presented in this paper reinforce the believe that double-spend attacks are very unlikely in practice, even when an attacker has some reasonable time advantage.

*Paper outline.* The paper is organized as follows. Section 2 presents a technical overview of the Bitcoin digital currency network. Section 3 briefly explains what a double-spend attack is. Section 4 presents the double-spend attack models of S. Nakamoto and M. Rosenfeld, while Section 5 presents the generalized and time-based attack models proposed by the authors. Section 6 gathers some information obtained from experimentation on the double-spend attack models. Finally, Section 7 gathers some concluding remarks and future work.

## 2 Bitcoin Overview

This section presents a summary of Bitcoin and its dynamics. The user is referred to [7] and [3] for a more comprehensive explanation.

### 2.1 Bitcoin: the Network and the Currency

The term *Bitcoin* is used for two different notions [3]: (i) a payment service based on a decentralized network and (ii) the name of the currency used in the Bitcoin payment service (often abbreviated as BTC). Bitcoin’s dynamics rely on
the work of the members of a P2P network called the Bitcoin network whose purpose is to update and maintain the stability of a public database called the blockchain. The blockchain plays the role of bookkeeping by registering all the transactions that have ever taken place in the Bitcoin network. This network does not rely on a trusted third-party that can verify whether a digital coin has been spent. Instead, it relies on a ‘proof of work’ principle in which users can execute payments by digitally signing their transactions and registering them in the blockchain.

The nodes of the Bitcoin network are called miners because their main job is to group new transactions together in packets called blocks and to mine them. That is, miners try to solve the difficult problem of finding an appropriate nonce according to block’s information, which requires a lot of computational power. In this sense, miners compete to mine blocks. Once a miner fully mines a block, its solution is broadcasted to the network and it receives 25 BTC as reward for the successful mining task. This is how the Bitcoin network issues currency and promotes miners to work. Overall, once a block is mined, most of the miners will stop mining and will start mining fresh new blocks (even if the transactions in these new blocks are exactly the ones in previously mined blocks).

2.2 Accounts and Transactions

Anyone can create a Bitcoin account, even offline or without the purpose of becoming a miner. Because accounts can be created offline, it is not possible to know how many user accounts are there. BTCs can be sent to any account, even to those created offline with the network confirming, in this case, that such a Joe Doe is entitled to spend that money. With an account, an user can create a transaction. Transactions can have more than one input account and more than one output account; they can also include a small tip for the miner that fully mines a block containing it.

Technically, a Bitcoin account is a public-private key pair based on asymmetric cryptography [8]. Each one of the keys plays the following roles:

- A public key acts as an account number or account identifier. BTCs can be sent to an account by just putting its corresponding public key as an output account in a specific transaction.
- A private key is the digital representation of ownership of an account, that is, it is what entitles someone to be the owner of an account. Consequently, the holder of an account’s private key is the one with the ability to spend its BTCs by signing transactions with the private key.

The balance of BTCs in a Bitcoin account is not represented by a single number. Instead, any balance is represented by pointing to the previous transactions in the blockchain that justify the ownership of the money. That is, a balance is a trace to the time in which each of its BTCs was first created.

For example, consider transaction 987 depicted in in Figure 1, in which Alice wants to pay 1.7 BTC into Bob’s account $B$. Alice uses two accounts, namely,
$A_0$ and $A_1$. She has received 1.3 BTC in account $A_0$ (transaction 198), 0.35 BTC in account $A_1$ (transaction 432), and 0.2 BTC in $A_1$ (transaction 258). Bitcoin requires that transactions consume all the inputs. Therefore, Alice must add one of her accounts to the output list of the transaction in order to have back the remaining change. Now, in transaction 1005, which depends on transaction 987, Alice and Bob want to pay Charlie 0.2 BTC each.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Account</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>198</td>
<td>$A_0$</td>
<td>1.30</td>
</tr>
<tr>
<td>432</td>
<td>$A_1$</td>
<td>0.35</td>
</tr>
<tr>
<td>258</td>
<td>$A_1$</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Fee for the miner who mines this first: 0.01
Digitally signed by $A_0$ and $A_1$

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Account</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>987</td>
<td>$A_0$</td>
<td>0.24</td>
</tr>
<tr>
<td>987</td>
<td>$B$</td>
<td>1.70</td>
</tr>
</tbody>
</table>

Fee for the miner who mines this first: 0.0
Digitally signed by $A_0$ and $B$

Fig. 1: Example of transactions, with transaction 1005 depending on transaction 987.

In Bitcoin transactions, the fee is an optional tip for promoting miners to include the transaction in the block they solve so that it will likely be added faster to the blockchain. However, as shown in transaction 1005 in Figure 1, there can be no fee included for the miner in the transaction. In this case, the miner who first mines the block will receive 25 BTC from the network. This kind of reward is likely to change in the future, when no more BTCs are to be issued by the system.

Finally, note that the system somehow needs to identify if the output of a transaction has been already spent. Otherwise, an user could create multiple transactions from the same source of income. In the Bitcoin network, miners will check that information from the blockchain before mining a block with such a transaction.

2.3 Blocks and Mining

A block mainly consists of the following fields:
A body: a set of transactions, including one that pays 25 BTC to an account of the miner.

A header: a “hashpointer” to a previous block, a place for putting a nonce and the hash of the block’s body.

<table>
<thead>
<tr>
<th>Pointer to a block already in the blockchain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash of block’s body</td>
</tr>
<tr>
<td><img src="nonce" alt="" /></td>
</tr>
<tr>
<td>Transaction 1465</td>
</tr>
<tr>
<td>Transaction 1504</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Transaction 1433</td>
</tr>
<tr>
<td>Transaction 1505</td>
</tr>
<tr>
<td>Transaction 1492</td>
</tr>
<tr>
<td>Special transaction to the miner</td>
</tr>
</tbody>
</table>

Fig. 2: Main parts of a block.

Mining a block consists in finding a nonce (i.e., a number to be put in the nonce place) that causes the hash of the block header to have many leading zeros (around 17 these days).

2.4 The Blockchain

The blockchain contains all transactions and it plays the role of bookkeeper of the Bitcoin network. Technically, the blockchain is a tree of blocks with a very long branch, which makes it look like a single chain. In order to avoid confusion, the main branch in this tree will be called the valid chain, while other branches will be called branches (from the valid chain). Blocks in the valid chain are called valid blocks. Branches may appear because of delays in the network, users trying different scripts, or because of attacks. Nevertheless, the length of these branches is very small when compared to the length of the valid chain. Figure 3 shows an example of a blockchain.

Fig. 3: Example of a blockchain with its original block (0), its valid chain (filled), and two branches of length one each (shallow).
One important observation about the blockchain in Figure 3 is that the last filled block is considered valid. This is because valid nodes form a larger branch having 2 nodes when compared to invalid nodes which have length 1. In general, when there is a tie, the valid block is chosen to be that whose last block was announced first. Also, due to time lag, it can happen that some nodes consider the wrong branch to be the valid one. However, this is very unlikely.

As mentioned earlier in this section, mined blocks contain a hashpointer to their previous block. Therefore, the blockchain can be seen as a collection of linked lists with hashpointers acting as back-pointers. Note that this helps in preventing any updates from happening easily to the internal tree structure of the blockchain.

3 Double-spend Attacks: An Overview

This section presents an overview of double-spend attacks to the Bitcoin network. For more details about double-spend attacks, the reader is referred to [4] and for more information about attacks, in general, to [5].

It is known that falsifying a digital signature is a hard computational problem [8]. Therefore, it is practically useless to try to modify a valid transaction that has been signed. However, there’s still a technique that can be used to deceive someone about the state of a transaction. This kind of attack to the Bitcoin network is called a double-spend attack. Even though it requires a tremendous computational power and it is very likely to fail, it may be profitable. As a matter of fact, it has happened already [6].

A double-spend attack can be performed as follows:

1. The attacker A wants a service or a product from B.
2. A creates two transactions: one paying to B and another paying to himself, using the same input for the transactions.
3. A publishes the “A to B” payment and secretly starts mining a block containing the “A to A” payment. Once the latter mining task succeeds, it continues to add blocks after it.
4. B gives the product or service to A, since the payment was confirmed or B did not wait long enough.
5. A is lucky and the fraudulent branch becomes longer than the valid one.
   It publishes all the blocks in the new branch and all the nodes agree on considering them as the valid ones since the branch is longer than the current valid one.
6. B gave out the product or service to A without receiving any payment. At this point, B can not find A because it is anonymous or has left.

Figure 4 depicts some stages of a successful double-spend attack. Stage (a) depicts the initial state of the blockchain. In Stage (b), honest nodes are extending the valid chain by putting valid blocks, while the attacker secretly starts
mining a fraudulent branch. In Stage (c), the attacker succeeds in making the fraudulent branch longer than the honest one. Finally, in Stage (d), the attacker’s branch is published and is now considered the valid one.

4 Two Existing Attack Models

This section presents an overview of the double-spend attack models by S. Nakamoto [7] and M. Rosenfeld [9], together with some notation and observations. Some of these notation and observations are also used in Section 5 for presenting the generalized and time-based double-spend attack models proposed by the authors.

The attack models by S. Nakamoto and M. Rosenfeld use similar notions and definitions:

- Quantity $q \in [0, 1]$ is the probability of the attacker nodes mining a block first than the honest nodes when they all start mining at the same time. This is equivalent to saying that $q$ is the proportion of attacker’s computation power with respect to the total computational power in the network (see Proposition 1).
- Quantity $K \in \mathbb{N}$ is the minimum number of confirmations required to accept a block and its transactions as valid. This quantity is set by each seller and not by the network itself.
Quantity $\tau \in \mathbb{R}_{>0}$ is the average time in seconds required for the whole network (i.e., honest and attacker nodes) to mine a block.

Additionally, functions exclusively used in the model of S. Nakamoto will have $N$ as a subscript, while those exclusively used in the model by M. Rosenfeld, will have $R$ as a subscript. Functions shared between these two models can be written without any subscripts.

Each one of the attack models in this paper, including the models proposed in Section 5, is introduced in terms of three measures. Namely, each model is described in terms of the probability of an attacker successfully performing a double-spend attack under some assumptions ($DS$), a potential ($P$) progress function, and a catch-up function ($C$). Function $DS$ depends on functions $P$ and $C$, and it intuitively measures the double-spend attack vulnerability of the network as a probability. The potential progress function $P$ describes the expected attacker’s branch length once the valid branch is long enough, while the catch-up function $C$ describes the probability of performing a double-spend attack given the expected attacker’s branch length. The ultimate goal of all these models is to observe the behavior of function $DS$ in terms of its parameters, as follows:

1. $DS_N(q, K)$ and $DS_R(q, K)$ represent in the models of S. Nakamoto and M. Rosenfeld, respectively, the probability of an attacker successfully performing a double-spend attack given that the attacker node controls $q$ percent of the network and the honest nodes just mined the $K$’th block.
2. $P_N(q, m, n)$ and $P_R(q, m, n)$ are the the progress functions in the models of S. Nakamoto and M. Rosenfeld, respectively; they represent the probability of an attacker mining exactly $n$ blocks once the honest nodes mine the $m$’th block.
3. $C(q, z)$ represents the probability of an attacker’s branch ever becoming longer than the honest one given an initial disadvantage of $z$ blocks.

Since function $C$ is the same for S. Nakamoto’s and M. Rosenfeld’s models, no subscript is used.

4.1 The Model of S. Nakamoto [7]

The double-spend attack model of S. Nakamoto computes the probability of a double-spend attack by combining the probability of the attacker having mined exactly $n$ blocks once the honest nodes mine the $K$’th confirmation block, with the probability of catching-up from such a $K – n$ block difference. The double-spend attack model of S. Nakamoto in [7] considers that the attacker has exactly mined 1 fraudulent block before starting the attack, but it can be easily modified to handle $n$ fraudulent blocks.

**Attacker’s potential progress function** In this model, the potential progress function corresponds to a Poisson distribution given by:

$$P_N(q, m, n) = e^{-\lambda} \lambda^n / n!,$$
where $\lambda = mq/(1 - q)$.

The catch-up function The catch-up function is based on a random walk in which success and failure are given to attacker or honest nodes mining a block, respectively.

Both S. Nakamoto and M. Rosenfeld agree on this function to be given by

$$C(q, z) = \begin{cases} \left(\frac{q}{p}\right)^{z+1}, & \text{if } q < 0.5 \land z > 0 \\ 1, & \text{otherwise,} \end{cases}$$

where $q$ represents the computational power of the attacker, $p = 1 - q$, and $z$ is the initial disadvantage of the attacker.

Double-spend attack probability The probability of a double-spend attack in S. Nakamoto’s model is the probability of the attacker progressing from 1 pre-computed block to $n$ blocks and then catching up from a difference of $K - n$ blocks:

$$DS_N(q, K) = \sum_{n=0}^{+\infty} P_N(q, K, n)C_N(q, K - n - 1)$$

$$= 1 - \sum_{n=0}^{K} P_N(q, K, n)(1 - C_N(q, K - n - 1)).$$

4.2 The Model of M. Rosenfeld [9]

The double-spend attack model of M. Rosenfeld uses the same approach used by S. Nakamoto. It also considers an initial advantage of 1 fraudulent block.

Attacker’s potential progress function M. Rosenfeld’s potential progress function corresponds to a negative binomial distribution given by:

$$P_R(q, m, n) = \begin{cases} 1, & \text{if } m = n = 0 \\ \binom{m + n - 1}{n} q^n (1 - q)^m, & \text{otherwise,} \end{cases}$$

where $P_R(q, m, n)$ is the probability of the attacker having mined $n$ blocks once the honest nodes mine the $m$’th block.

Catch-up function M. Rosenfeld uses the same catch-up function $C$ originally proposed by S. Nakamoto.
Double-spend attack probability  The double-spend attack probability in this model is denoted $DS_R$ and it is given by:

$$DS_R(q, K) = \sum_{n=0}^{+\infty} P_R(q, K, n) C_R(q, K - n - 1)$$

$$= 1 - \sum_{n=0}^{K} P_R(q, K, n)(1 - C_R(q, K - n - 1)).$$

Finally, it is shown that in these two models (as it is also the case for the attack models proposed in Section 5) there is a direct relation between computational power and the probability of first mining a block.

**Proposition 1.** The probability of mining a block first than the others given a proportion of computational power of $q$ with respect to the total computational power in the network is exactly $q$.

*Proof.* Since the probability of mining a block in a single trial is constant and independent of previous trials, the time needed for mining a block follows an exponential distribution. Let $T$ denote the random variable describing the time needed to mine a block when using all the power available in the network. The density function for $T$ is given for $x \geq 0$ by

$$f(x) = \frac{1}{\tau} e^{-\frac{x}{\tau}},$$

where $\tau$ is the expected time, while the probability of mining a block per unit time is $\frac{1}{\tau}$. Since the power in the network is proportional to the amount of hashes computed per unit time, the probability of mining a block per unit time is also proportional to such a power. Therefore, the probabilities of mining a block per unit time for the attacker and for the honest nodes will be $\frac{q}{\tau}$ and $\frac{p}{\tau}$, respectively. Let $T_p$ and $T_q$ be random variables denoting the time needed by the attacker nodes and the honest nodes to mine a block, respectively. Then, the probability that the attacker nodes mine a block faster than the honest nodes is given by:

$$P(T_q < T_p) = \int_{0}^{\infty} P(T_q = x) P(T_p > x) dx$$

$$= \int_{0}^{\infty} \frac{q}{\tau} e^{-\frac{x}{\tau}} e^{-\frac{x}{\tau}} dx$$

$$= q \int_{0}^{\infty} \frac{1}{\tau} e^{-\frac{x}{\tau}} dx$$

$$= q.$$
5 Two New Attack Models

This section presents the generalized model and the time-based model for double-spend attacks proposed by the authors. Some of the notation, conventions, and functions used in this section are borrowed from Section 4. In addition, functions defined exclusively for the time-model have $T$ as a subscript.

5.1 The Generalized Model

This is the first double-spend attack model proposed by the authors. This model results from generalizing the model of M. Rosenfield by adding an extra parameter denoting some time advantage assigned to the attacker, i.e., time in which the attacker has been secretly trying to mine blocks; hence, it is called the generalized model.

Attacker’s potential progress function It is represented by function

$$P(q, m, n, t)$$

which generalizes the potential progress functions in Section 4. Function $P$ denotes the probability of an attacker mining exactly $n$ blocks once the honest nodes mine the $m$’th block, assuming that the attacker has been mining secretly during $t\tau$ seconds. The extra parameter $t$ represents the time advantage to be assigned to the attacker nodes towards fraudulent block production.

In order to define the progress function $P$ for this model, it is mandatory to consider the possible number of blocks that could have been mined during $t\tau$ seconds. Let $a(q, t, n)$ denote the probability of mining exactly $n$ blocks if constantly mining during $t\tau$ seconds with a proportion of $q$ hashpower respect to the total in the network. Proposition 2 proposes a closed formula for computing $a$.

**Proposition 2.** If $t > 0$ or $n > 0$, then

$$a(q, t, n) = \frac{(qt)^n}{n!} e^{-qt}.$$  

**Proof.** Note that $a(q, t, 0)$, which is the probability of mining 0 blocks in $t\tau$ seconds, represents the probability of mining the first block in a time greater than $t\tau$ seconds. That is

$$a(q, t, 0) = \int_t^{+\infty} q e^{-qs} ds = e^{-qt}.$$  

Consider $a(q, t, n) = \frac{(qt)^n}{n!} e^{-qt}$. The goal is to show that

$$a(q, t, n + 1) = \frac{(qt)^{n+1}}{(n+1)!} e^{-qt}.$$
If \( s \) is the moment at which the next block is mined (in case there is a next block), then:

\[
a(q, t, n + 1) = \int_0^t q e^{-qs} a(q, t - s, n) ds
\]
\[
= \int_0^t q e^{-qs} \frac{(q(t - s))^n}{n!} e^{-q(t-s)} ds
\]
\[
= \int_0^t q e^{-qt} \frac{(q(t - s))^n}{n!} ds
\]
\[
= q e^{-qt} \frac{q^n}{(n+1)!} ((t-0)^{n+1} - (t-t))^{n+1}
\]
\[
= \frac{(qt)^{n+1}}{(n+1)!} e^{-qt}.
\]

Thanks to Proposition 2, function \( P \) can be defined as

\[
P(q, m, n, t) = \sum_{z=0}^{n} a(q, t, z) P_R(q, m, n - z),
\]

where

\[
a(q, t, n) = \begin{cases} 
1 & \text{, if } t = n = 0 \\
0 & \text{, if } t <= 0 \\
\frac{(qt)^n}{n!} e^{-qt} & \text{, otherwise.}
\end{cases}
\]

Note that functions \( P_R \) and \( P_N \) differ from \( P \) only in that they assume \( t = 0 \). Thus, it can be expected that function \( P \) satisfies

\[
P_N(q, m, n) \approx P_R(q, m, n) = P(q, m, n, 0).
\]

**Catch-up function** It uses the catch-up function \( C \) originally proposed by S. Nakamoto (see Section 4).

**Double-spend attack probability** It is modeled by function

\[
DS(q, K, n, t),
\]
denoting the probability of an attacker successfully performing a double-spend attack given an initial advantage of \( n \) blocks and \( t \tau \) seconds over the honest nodes.

Function \( DS \) is defined by:

\[
DS(q, K, n, t) = \sum_{z=0}^{\infty} P(q, K, z, t) C_R(q, K - n - z).
\]
\[
= 1 - \sum_{z=0}^{K-n} P(q, K, z, t) (1 - C_R(q, K - n - z)).
\]
Note that functions $DS_N$ and $DS_R$ (introduced in Section 4) differ from $DS$ only in that they assume $t = 0$ and an initial advantage of $n = 1$ blocks. In Section 6, it is shown that function $DS$ satisfies

$$DS_N(q, K) \approx DS_R(q, K) = DS(q, K, 1, 0).$$

### 5.2 The Time-based Model

This section presents the time-based model, the second double-spend attack model proposed by the authors. On the one hand, this model is as flexible as the generalized model in Section 5.1, and thus more general than the models of S. Nakamoto and M. Rosenfeld in Section 4. On the other hand, it uses a completely different approach to compute the probability of an attacker successfully performing a double-spend attack.

In the time-based model, states are identified by the length of the valid and fraudulent branches, which are assumed to have the same length, and the time difference at which both the honest and attacker nodes mined their last block. Namely, a state in the time-based model consists of two values $t$ and $n$ denoting the time difference $t$ at which both the honest and attacker nodes mined their $n$'th block. It is key to note that the states in the time-based model differ from the previous attack-models in that the latter represent the states in terms of the length of the attacker's and the honest nodes' branches since they can be different. In the time-based model, sometimes it is enough to focus on the time parameter $t$, leaving the length parameter $n$ to represent the maximum such length so that both the attacker and the honest nodes have mined their $n$'th block.

**Attacker’s potential progress function in time** It is represented by function

$$P_T(q, m, n, t)$$

denoting the probability of the time at which the $n$'th block mined by an attacker node is exactly (in the sense of a probability density function) $t\tau$ seconds after the time at which the $m$'th block is mined by a honest node.

As it is expressed in Proposition 1, and since the probability of mining a block in a single trial is constant and independent of previous trials, the time needed for mining a block follows an exponential distribution with expected value $\lambda = \frac{1}{q\tau}$. Consequently, the time $T_{q,n}$ needed to mine $n$ blocks with hashpower $q$ is given by the sum of the outcome of $n$ independent exponential distributions, which follows an Erlang distribution [12] (i.e., a Gamma distribution with integer parameter), with shape $n$ and scale $\mu = \frac{1}{q}$ having probability density function:

$$h_{q,n}(t) = \frac{t^{n-1}q^n e^{-qt}}{(n-1)!}.$$
In this way, an attacker’s potential progress function in time $P_T$ is the probability density function of the random variable $X = T_{q,n} - T_{p,m}$, with independent $T_{q,n}$ and $T_{p,m}$. Adopting this observations, function $P_T$ is defined as (see [2] for more technical details):

$$P_T(q, m, n, t) = \int_{-\infty}^{+\infty} h_{q,n}(x) h_{p,m}(t - x) dx.$$ 

On a side note and given its complexity, for the experiments reported on Section 6, the authors have opted for integrating the expression using numerical packages instead of using directly its closed mathematical form.

**Catch-up function** It is represented by function $C_T(q, t)$ denoting the probability of an attacker’s branch ever becoming longer than the valid one given that the honest nodes mined their last block $t_\tau$ seconds before the attacker mined its last block.

For this model, it is required that function $C_T$ satisfies the following two conditions:

1. $C_T(q, t) = 1$ whenever $t < 0$, covering the case in which an attacker has mined the last block earlier than the honest nodes, meaning that the attacker’s branch was already longer.

2. Since the probability of catching up from a time difference is equal to the probability of catching up with the next block plus the probability of catching up from the time difference once the next block is mined,

$$C_T(q, t) = \int_{-\infty}^{+\infty} P_T(q, 1, 1, x) C_T(x + t) dx.$$ 

A function that satisfies both requirements is:

$$C_T(q, t) = \begin{cases} 
eg p e^{-(p-q)t} & \text{if } t > 0 \\ 1 & \text{otherwise} \end{cases},$$

were $p = 1 - q$, which can be verified from the definition of $P_T$ and given that:

$$P_T(q, 1, 1, t) = \begin{cases} pq e^{-qt} & \text{if } t > 0 \\ pq e^{pt} & \text{otherwise}. \end{cases}$$

**Double-spend attack probability** It is represented by function $DS_T(q, K, n, t)$
denoting, similar to $DS$ in the generalized model, the probability of an attacker successfully performing a double-spend attack given an initial advantage of $n$ blocks and $t\tau$ seconds over the honest nodes.

The double-spend attack probability in the time-based model $DS_T$ can be expressed as the probability of having a time disadvantage of $t\tau$ seconds once the $K + 1$'th block is mined times the probability of catching up from that disadvantage. Note that an attack will conclude with $K + 1$ blocks and not with $K$ blocks.

Function $DS_T$ is defined by:

$$DS_T(q, K, n_0, t_0) = \int_{-\infty}^{+\infty} P(q, K + 1, K - n_0 + 1, t)C_T(q, t - t_0)dt.$$  

6 Model Comparison

This section summarizes experimental results obtained on the four attack models presented in Section 5. Additionally, this section gathers the code snippets necessary to run these experiments. They are written in the Python 3 programming language. The main conclusion of this section, supported mostly by experimental data and regarding the correctness of the proposed models, is twofold:

1. The generalized model proposed in this paper and the hashrate-based model of M. Rosenfeld [9] have the same results when predicting the probability of double-spend attacks in the Bitcoin network.
2. The generalized and the time-based models coincide.

In addition, the experimental data suggests that:

- The probability of a double-spend attack increases as the number of confirmations decreases or the attacker’s power increases, which is a well-known fact about Bitcoin.
- Double-spend attacks are very unlikely in practice, if an attacker has limited time advantage.
- If an attacker gains control of 40% of the network’s computational power, then double-spend attacks are almost impossible to contain. Since this situation is very unlikely, the best containment method for these sort of attacks is to set a high number of confirmations, as the four models suggest.
- The model of S. Nakamoto behaves slightly different to the other three models.

6.1 $P_N(q, m, n) \approx P_R(q, m, n) = P(q, m, n, 0)$

This is actually a fact which requires no experimental data to be shown. The ‘equation’ $P_N(q, m, n) \approx P_R(q, m, n)$ is a consequence of the similarities between the Poisson distribution and the Negative Binomial distribution. Moreover, the equation $P_R(q, m, n) = P(q, m, n, 0)$ follows from
\[ P(q, m, n, 0) = \sum_{z=0}^{n} a(q, 0, z) P_R(q, m, n - z) \]
\[ = a(q, 0, 0) P_R(q, m, n) \]
\[ + \sum_{z=1}^{n} a(q, 0, z) P_R(q, m, n - z) \]
\[ = P_R(q, m, n), \]

given that
\[ a(q, 0, n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise.} \end{cases} \]

6.2 \( DS_N(q, K) \approx DS_R(q, K) = DS(q, K, 1, 0) \)

This is one of the main conclusions of this paper and it is also a fact that can be proven without experimental data. This follows from the definition of functions denoting the probability of an attacker successfully performing a double-spend attack, in the corresponding model, and the equation \( P_R(q, m, n) = P(q, m, n, 0) \) above-stated.

However, in order to give a more precise idea of how these functions behave, we include some experimental data in figures 5 and 6, and in Table 1.

In Figure 5, the plots of \( DS \) vs \( K \) are shown under two different cases, namely \( q = 10\% \) and \( q = 20\% \). Note that the curves representing \( DS_R \) and \( DS \) overlap, while the curve representing \( DS_N \) does not. For more detail, Figure 6 includes a zoomed version of Figure 5 in which the gap between the curves is more noticeable. This supports the claim about the model of S. Nakamoto not being entirely accurate, as previously noted by M. Rosenfeld.

Table 1 includes the values of \( DS_N \), \( DS_R \), and \( DS \) for some values for \( K \) and \( q \), without any time advantage for the attacker nodes.

6.3 \( DS(q, K, n_0, t_0) = DS_T(q, K, n_0, t_0) \)

This is also a main conclusion of this paper. Although it is not yet mathematically proven, the experimental data strongly supports this claim.

Table 2 includes the values of \( DS \) and \( DS_T \) as a function of \( K \) and \( q \) assuming \( n_0 = 0 \) and \( t_0 = 2.5 \). This table summarizes a scenario in which the last block was published \( 2.5\tau \) seconds ago, where \( \tau \) is the average time between blocks, time in which the attacker has been mining but has not published any mined block (in case there is any). The more dense the time axis is in function \( P_T \) (default is \( 30K \), see Section 6.6.5 for more detail), the closer the results are. For lower values of \( q \), the curve \( P_T \) extends rapidly towards the right, increasing the window length, thus diminishing the absolute density of the time axis and the precision of function \( DS \).
Fig. 5: Comparison of the number of confirmations $K$ for accepting a transaction as valid vs. the probability of a double-spend attack in three models ($DS_N$, $DS_R$, and $DS$). The curves appear to be the same but Nakamoto’s are not.

Fig. 6: Figure 5 partially zoomed. It becomes clearer that Nakamoto’s curves differ from the other two.
Table 1: Probability of a double-spend attack according to three models ($DS_N$, $DS_R$, and $DS$) vs the hashpower $q$ and the number of confirmations $K$ for accepting a transaction as valid.

<table>
<thead>
<tr>
<th>Model</th>
<th>q</th>
<th>Number of confirmations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0%</td>
<td>100.00% 0.00% 0.00% 0.00% 0.00% 0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>Nakamoto</td>
<td>1%</td>
<td>100.00% 2.00% 0.05% ≈ 0 ≈ 0 ≈ 0 ≈ 0 ≈ 0</td>
</tr>
<tr>
<td>Rosenfeld</td>
<td>1%</td>
<td>100.00% 2.00% 0.06% ≈ 0 ≈ 0 ≈ 0 ≈ 0 ≈ 0</td>
</tr>
<tr>
<td>Generalized</td>
<td>1%</td>
<td>100.00% 2.00% 0.06% ≈ 0 ≈ 0 ≈ 0 ≈ 0 ≈ 0</td>
</tr>
<tr>
<td>Nakamoto</td>
<td>5%</td>
<td>100.00% 10.12% 1.26% 0.02% ≈ 0 ≈ 0 ≈ 0 ≈ 0</td>
</tr>
<tr>
<td>Rosenfeld</td>
<td>5%</td>
<td>100.00% 10.00% 1.45% 0.23% 0.04% 0.01% ≈ 0 ≈ 0</td>
</tr>
<tr>
<td>Generalized</td>
<td>5%</td>
<td>100.00% 10.00% 1.45% 0.23% 0.04% 0.01% ≈ 0 ≈ 0</td>
</tr>
<tr>
<td>Nakamoto</td>
<td>10%</td>
<td>100.00% 20.46% 5.10% 1.32% 0.35% 0.09% 0.02% 0.01%</td>
</tr>
<tr>
<td>Rosenfeld</td>
<td>10%</td>
<td>100.00% 20.00% 5.60% 1.71% 0.55% 0.18% 0.06% 0.02%</td>
</tr>
<tr>
<td>Generalized</td>
<td>10%</td>
<td>100.00% 20.00% 5.60% 1.71% 0.55% 0.18% 0.06% 0.02%</td>
</tr>
<tr>
<td>Nakamoto</td>
<td>25%</td>
<td>100.00% 52.23% 31.54% 19.61% 12.35% 7.81% 4.99% 3.19%</td>
</tr>
<tr>
<td>Rosenfeld</td>
<td>25%</td>
<td>100.00% 50.00% 31.25% 20.76% 14.11% 9.79% 6.87% 4.86%</td>
</tr>
<tr>
<td>Generalized</td>
<td>25%</td>
<td>100.00% 50.00% 31.25% 20.76% 14.11% 9.79% 6.87% 4.86%</td>
</tr>
<tr>
<td>Nakamoto</td>
<td>40%</td>
<td>100.00% 82.89% 73.64% 66.42% 60.34% 55.06% 50.40% 46.23%</td>
</tr>
<tr>
<td>Rosenfeld</td>
<td>40%</td>
<td>100.00% 80.00% 70.40% 63.49% 57.96% 53.31% 49.30% 45.77%</td>
</tr>
<tr>
<td>Generalized</td>
<td>40%</td>
<td>100.00% 80.00% 70.40% 63.49% 57.96% 53.31% 49.30% 45.77%</td>
</tr>
<tr>
<td>Nakamoto</td>
<td>49%</td>
<td>100.00% 98.50% 97.77% 97.21% 96.74% 96.32% 95.94% 95.59%</td>
</tr>
<tr>
<td>Rosenfeld</td>
<td>49%</td>
<td>100.00% 98.00% 97.00% 96.25% 95.63% 95.08% 94.59% 94.14%</td>
</tr>
<tr>
<td>Generalized</td>
<td>49%</td>
<td>100.00% 98.00% 97.00% 96.25% 95.63% 95.08% 94.59% 94.14%</td>
</tr>
</tbody>
</table>
Table 3 includes the values of $DS$ and $DS_T$ as a function of $K$ and $q$ assuming $n_0 = 1$ and $t_0 = 0.5$. Notice that the results are similar, specially when $q \neq 0$.

Table 2: Comparison of $DS$ vs $DS_T$ assuming $n_0 = 0$ and $t_0 = 2.5$. 30$K$ points used for time axis.

<table>
<thead>
<tr>
<th>Model</th>
<th>q</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized 0%</td>
<td></td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Time-based 0%</td>
<td></td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Generalized 1%</td>
<td></td>
<td>3.45%</td>
<td>0.11%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Time-based 1%</td>
<td></td>
<td>4.24%</td>
<td>0.17%</td>
<td>0.01%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Generalized 5%</td>
<td></td>
<td>16.40%</td>
<td>2.50%</td>
<td>0.40%</td>
<td>0.07%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Time-based 5%</td>
<td></td>
<td>16.34%</td>
<td>2.52%</td>
<td>0.41%</td>
<td>0.06%</td>
<td>0.01%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Generalized 10%</td>
<td></td>
<td>30.77%</td>
<td>8.97%</td>
<td>2.73%</td>
<td>0.86%</td>
<td>0.28%</td>
<td>0.09%</td>
<td>0.03%</td>
<td>0.01%</td>
</tr>
<tr>
<td>Time-based 10%</td>
<td></td>
<td>30.71%</td>
<td>8.99%</td>
<td>2.70%</td>
<td>0.85%</td>
<td>0.28%</td>
<td>0.09%</td>
<td>0.03%</td>
<td>0.01%</td>
</tr>
<tr>
<td>Generalized 25%</td>
<td></td>
<td>64.32%</td>
<td>40.90%</td>
<td>26.94%</td>
<td>18.24%</td>
<td>12.95%</td>
<td>8.79%</td>
<td>6.20%</td>
<td>4.41%</td>
</tr>
<tr>
<td>Time-based 25%</td>
<td></td>
<td>64.31%</td>
<td>40.88%</td>
<td>26.95%</td>
<td>18.25%</td>
<td>12.57%</td>
<td>8.79%</td>
<td>6.20%</td>
<td>4.40%</td>
</tr>
<tr>
<td>Generalized 40%</td>
<td></td>
<td>87.74%</td>
<td>77.44%</td>
<td>69.58%</td>
<td>63.31%</td>
<td>58.09%</td>
<td>53.61%</td>
<td>49.70%</td>
<td>46.22%</td>
</tr>
<tr>
<td>Time-based 40%</td>
<td></td>
<td>87.74%</td>
<td>77.44%</td>
<td>69.57%</td>
<td>63.30%</td>
<td>58.09%</td>
<td>53.61%</td>
<td>49.69%</td>
<td>46.21%</td>
</tr>
<tr>
<td>Generalized 50%</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Time-based 50%</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Table 3: Comparison of $DS$ vs $DS_T$ assuming $n_0 = 1$ and $t_0 = 0.5$.

<table>
<thead>
<tr>
<th>Model</th>
<th>q</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized 0%</td>
<td></td>
<td>100.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Time-based 0%</td>
<td></td>
<td>100.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Generalized 1%</td>
<td></td>
<td>100.00%</td>
<td>2.49%</td>
<td>0.98%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Time-based 1%</td>
<td></td>
<td>100.00%</td>
<td>4.54%</td>
<td>0.18%</td>
<td>0.01%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Generalized 5%</td>
<td></td>
<td>100.00%</td>
<td>12.22%</td>
<td>1.80%</td>
<td>0.29%</td>
<td>0.05%</td>
<td>0.01%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Time-based 5%</td>
<td></td>
<td>100.00%</td>
<td>12.53%</td>
<td>1.85%</td>
<td>0.30%</td>
<td>0.05%</td>
<td>0.01%</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
</tr>
<tr>
<td>Generalized 10%</td>
<td></td>
<td>100.00%</td>
<td>23.90%</td>
<td>6.78%</td>
<td>2.08%</td>
<td>0.66%</td>
<td>0.22%</td>
<td>0.07%</td>
<td>0.02%</td>
</tr>
<tr>
<td>Time-based 10%</td>
<td></td>
<td>100.00%</td>
<td>23.94%</td>
<td>6.74%</td>
<td>2.06%</td>
<td>0.66%</td>
<td>0.21%</td>
<td>0.07%</td>
<td>0.02%</td>
</tr>
<tr>
<td>Generalized 25%</td>
<td></td>
<td>100.00%</td>
<td>55.88%</td>
<td>35.19%</td>
<td>23.86%</td>
<td>15.94%</td>
<td>11.06%</td>
<td>7.76%</td>
<td>5.49%</td>
</tr>
<tr>
<td>Time-based 25%</td>
<td></td>
<td>100.00%</td>
<td>55.87%</td>
<td>35.19%</td>
<td>23.86%</td>
<td>15.93%</td>
<td>11.06%</td>
<td>7.77%</td>
<td>5.49%</td>
</tr>
<tr>
<td>Generalized 40%</td>
<td></td>
<td>100.00%</td>
<td>83.63%</td>
<td>73.80%</td>
<td>66.61%</td>
<td>60.83%</td>
<td>55.97%</td>
<td>51.77%</td>
<td>48.06%</td>
</tr>
<tr>
<td>Time-based 40%</td>
<td></td>
<td>100.00%</td>
<td>83.63%</td>
<td>73.79%</td>
<td>66.60%</td>
<td>60.83%</td>
<td>55.97%</td>
<td>51.78%</td>
<td>48.07%</td>
</tr>
<tr>
<td>Generalized 50%</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Time-based 50%</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

6.4 Other results

The way in which the model of M. Rosenfeld was generalized in Section 5.1 allows for new properties to be studied. In fact, the generalized model as well
as the time-based model are still consistent if \( n_0 < 0 \), which corresponds to the scenario in which the attacker nodes are not mining at the top of the blockchain, but in a branch \(|n_0|\) blocks behind of the valid branch.

Figure 7 depicts the probability of a double-spend attack as the time advantage increases, which is the main feature of the models proposed in this paper. Note that if enough time has elapsed, it is highly possible that the attacker nodes have already built a fraudulent branch long enough to be considered valid. Also note that the two proposed models experimentally coincide.

![Fig. 7: Probability of a double spend as a function of the time advantage, assuming \( K = 6 \) and \( n_0 = 1 \). The outputs of the proposed models appear to be the same.](image)

Figure 8 shows an extension to negative values of the plots of \( DS \) and \( DS_T \) assuming \( K = 6 \), which is a commonly accepted value, and \( t_0 = 0 \) in order to be comparable with the previous models.

Table 4 considers a scenario in which the attacker has been mining during \( 2.7\tau \) seconds and from the time it started mining, one valid block has been published.

### 6.5 Code Snippets

**Model of S. Nakamoto** First, the code used to model the attack model by S. Nakamoto is presented. The catch-up function is implemented with function \( C \), as follows:

```python
def C(q, z):
    if z < 0 or q >= 0.5:
        return 1
```

Fig. 8: Probability of a double spend as a function of the block advantage, assuming $K = 6$ and $t_0 = 0$. The outputs of the proposed models appear to be the same. Negative values for $n_0$ represent an attacker which is not mining in the top of the blockchain, thus it is already losing the race against the honest nodes.

Table 4: Comparison of $DS$ vs $DS_T$ assuming $n_0 = -1$ and $t_0 = 2.7$.

<table>
<thead>
<tr>
<th>Model</th>
<th>$q$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized</td>
<td>1%</td>
<td>0.07%</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
</tr>
<tr>
<td>Time-based</td>
<td>1%</td>
<td>≈ 0</td>
<td>0.01%</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
</tr>
<tr>
<td>Generalized</td>
<td>5%</td>
<td>1.70%</td>
<td>0.25%</td>
<td>0.04%</td>
<td>0.01%</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
</tr>
<tr>
<td>Time-based</td>
<td>5%</td>
<td>1.62%</td>
<td>0.27%</td>
<td>0.04%</td>
<td>0.01%</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
<td>≈ 0</td>
</tr>
<tr>
<td>Generalized</td>
<td>10%</td>
<td>6.28%</td>
<td>1.80%</td>
<td>0.55%</td>
<td>0.18%</td>
<td>0.06%</td>
<td>0.02%</td>
<td>0.01%</td>
<td>≈ 0</td>
</tr>
<tr>
<td>Time-based</td>
<td>10%</td>
<td>6.27%</td>
<td>1.81%</td>
<td>0.55%</td>
<td>0.17%</td>
<td>0.06%</td>
<td>0.02%</td>
<td>0.01%</td>
<td>≈ 0</td>
</tr>
<tr>
<td>Generalized</td>
<td>25%</td>
<td>31.83%</td>
<td>20.14%</td>
<td>13.38%</td>
<td>9.13%</td>
<td>6.34%</td>
<td>4.45%</td>
<td>3.15%</td>
<td>2.25%</td>
</tr>
<tr>
<td>Time-based</td>
<td>25%</td>
<td>31.78%</td>
<td>20.13%</td>
<td>13.39%</td>
<td>9.13%</td>
<td>6.33%</td>
<td>4.45%</td>
<td>3.16%</td>
<td>2.25%</td>
</tr>
<tr>
<td>Generalized</td>
<td>40%</td>
<td>68.91%</td>
<td>60.93%</td>
<td>55.01%</td>
<td>50.26%</td>
<td>46.26%</td>
<td>42.80%</td>
<td>39.75%</td>
<td>37.03%</td>
</tr>
<tr>
<td>Time-based</td>
<td>40%</td>
<td>68.89%</td>
<td>60.93%</td>
<td>55.01%</td>
<td>50.26%</td>
<td>46.26%</td>
<td>42.80%</td>
<td>39.74%</td>
<td>37.03%</td>
</tr>
<tr>
<td>Generalized</td>
<td>49%</td>
<td>96.57%</td>
<td>95.63%</td>
<td>94.91%</td>
<td>94.30%</td>
<td>93.77%</td>
<td>93.29%</td>
<td>92.85%</td>
<td>92.44%</td>
</tr>
<tr>
<td>Time-based</td>
<td>49%</td>
<td>96.57%</td>
<td>95.63%</td>
<td>94.91%</td>
<td>94.30%</td>
<td>93.77%</td>
<td>93.29%</td>
<td>92.85%</td>
<td>92.44%</td>
</tr>
</tbody>
</table>
else: prob = (q/(1-q)) ** (z+1)
return prob

The probability of success of a double-spend attack in the model of S. Nakamoto is implemented with functions $P_N$ and $DS_N$, as follows:

```python
from scipy.stats import poisson

def $P_N(q, m, n)$:
    return poisson.pmf(n, m*q/(1-q))
def $DS_N(q, K)$:
    return 1-
        sum($P_N(q, K, n)*(1-C(q, K-n-1))$ for n in range(0, K +1))
```

**Model of M. Rosenfeld** Next, functions $P_R$, $DS_R$, and an alternative way of computing $DS_R$ using dynamic programming techniques, namely $DS_R_dp$, which corroborates the correctness of the function $DS_R$.

```python
from scipy.stats import nbinom

def $P_R(q, m, n)$:
    if q>=0.5 or ((q==0 or m==0) and n==0): prob = 1
    elif q==0 or m==0: prob = 0
    else: prob = ((1-q)/q) * nbinom.pmf(m-1,n+1,q)
    return prob

def $DS_R(q, K)$:
    return 1-
        sum($P_R(q, K, n)*(1-C(q, K-n-1))$ for n in range(0, K +1))
def $DS_R_dp(q, K)$:
    dp1 = [C(q,n-1) for n in range(K+1)]
    for m in range(K):
        dp2 = [1]
        for n in range(K):
            dp2.append( q*dp2[n]+ (1-q)*dp1[n+1] )
        dp1 = list(dp2)
    return dp1[-1]
```

The function $DS_R_dp$ is based on a recursive way of computing $DS_R$:

$$DS_R(q, K) = dp(q, K, K),$$

where

$$dp(q, m, n) = \begin{cases} 
q \cdot dp(q, m, n-1) + \\
(1-q)dp(q, m -1, n) & \text{if } n > 0 \land m > 0 \\
C(q, m - n -1) & \text{otherwise}
\end{cases}$$
which can be implemented either iteratively as shown above, or recursively as shown below:

```python
from functools import lru_cache
@lru_cache(maxsize=None) # Enables memoizing
def dp(q,m,n):
    if n>0 and m>0:
        ans = (1-q)*dp(q,m-1,n)+q*dp(q,m,n-1)
    else: ans = C(q,n-m-1)
    return ans
```

**Generalized model** The generalized functions $a$, $P$, and $DS$ are implemented as follows:

```python
def a(q,t,n):
    if t==0 and n==0: prob = 1
    else:
        prob = np.exp(-q*t)
        for i in range(n):
            prob *= q*t/(i+1)
    return prob
def P(q,m,n,t):
    return sum(a(q,t,z)*P_R(q,m,n-z) for z in range(0,n+1))
def DS(q,K,n0,t0):
    return 1-sum(P(q,K,n,t0)*(1-C(q,K-n0-n)) for n in range(0,K-n0+1))
```

**Time-based model** The time-based model consists of three functions, namely $C_T$, $P_T$ and $DS_T$, which are presented below.

The function $P_T$ returns two arrays $X$ and $Y$ (with $30K$ samples), representing $P_T(q,m,n,t)$ as a function of $t$.

```python
import numpy as np
from scipy.signal import fftconvolve
from scipy.stats import gamma

def C_T(q,t):
    if t<0 or q>=0.5: prob = 1
    else: prob = q/(1-q)*np.exp((2*q-1)*t)
    return prob
def P_T(q,m,n):
    hi = abs(n/q-m/(1-q)) + 3*(n/q**2+m/(1-q)**2)
    X = np.linspace(-hi,hi,30001)
    Ynq = gamma.pdf(X, n, scale=1/q)
```
Ymp = gamma.pdf( X, m, scale=1/(1-q) )
Y = fftconvolve(Ynq,Ymp[::-1],'same')
Y = Y/np.trapz(Y,X)
return X,Y

def DS_T(q,K,n0,t0):
    if q>=0.5 or n0>K: prob=1
    elif q==0: prob=0
    else:
        T,Y = P_T(q,K+1,K-n0+1)
        Y = [y*C_T(q,t-t0) for t,y in zip(T,Y)]
        prob = np.trapz(Y,T)
    return prob

7 Concluding Remarks

This paper has presented two models for double-spend attacks in the Bitcoin network. Overall, both models consider partial advancement towards block production and how fast a block header can be hashed. Previously existing attack models only considered how fast a block header can be hashed. In this regard, the extension with partial advancement towards block creation is a novelty, which to the best of the authors’ knowledge, has not been previously reported in the literature.

The first model, called the generalized model, extends an attack-model developed by M. Rosenfeld [9] by adding a time parameter, enabling the model to assign time advantage to an attacker trying to achieve a double-spend attack. The second model, called the time-based model, follows a completely different approach by taking into account the times at which the honest and attacker nodes last mined a block, in addition to their relative progress in terms of the length of chains already mined. The two models proposed by the authors are more general than existing double-spend attack models.

The generalized and the time-based have been compared to two well-established hashrate-based models where partial production of a block is not considered. Namely, the attack models proposed by the authors in this paper have been extensively compared to the attack models previously developed by S. Nakamoto [7] and M. Rosenfeld [9]. With the help of experimental data, it is strongly suggested that the probability of catching up from a time difference is very similar to the probability of catching up from the new time difference obtained after the next block is found. That is, the new models presented in this paper, and the models of S. Nakamoto and M. Rosenfeld, surprisingly behave in a similar way. This reinforces the belief that double-spend attacks are very unlikely in practice given the current parameters in the Bitcoin network in terms of the number of confirmations required for a transaction to be valid and the number of leading zeroes required to sign a block of transactions. However, the double-spend attacks become likely when an attacker has enough time advantage towards fraudulent block production.
As usual, some work remains to be done. Proving that the generalized model and the time-based model coincide is important, but seems like a fairly challenging task. Furthermore, it seems promising to specify the attack models in a formal tool and exercise them against temporal formulas, perhaps following some of the ideas in [11]. More precisely, it will be interesting to formally specify the models proposed by the authors in the rewrite-based specification language PMaude [1] and then use a statistical model checker such as MultiVeStA [10] to formally verify probabilistic temporal properties related to possible scenarios for double-spend attacks with time advantage in the Bitcoin network.

References