On Cancellation of Transactions in Bitcoin-like Blockchains

Önder Gürcan[1] and Alejandro Ranchal Pedrosa[1,2,3]
and Sara Tucci-Piergiovanni[1]

[1] CEA LIST, Point Courrier 174, Gif-sur-Yvette, F-91191 France
[2] LIP6 – Sorbonne Université, Paris, France
[3] EIT Digital, Stockholm, Sweden
{Onder.Gurcan, Alejandro.Ranchal-Pedrosa, Sara.Tucci}@cea.fr

Abstract. Bitcoin-like blockchains do not envisage any specific mechanism to avoid unfairness for the users. Hence, unfair situations, like impossibility of cancellation of transactions explicitly or having unconfirmed transactions, reduce the satisfaction of users dramatically, and, as a result, they may leave the system entirely. Such a consequence would impact significantly the security and the sustainability of the blockchain. Based on this observation, in this paper, we focus on *explicit* cancellation of transactions to improve the fairness for users. We propose a novel scheme with which it is possible to cancel a transaction, whether it is confirmed in a block or not, under certain conditions. We show that the proposed scheme is superior to the existing workarounds and is implementable for Bitcoin-like blockchains.

1 Introduction

Bitcoin, introduced by Satoshi Nakamoto [13], is the core of decentralized cryptocurrency systems. Participants following this protocol can create together a distributed, economical, social and technical system where anyone can join and leave. User participants create and broadcast transactions across the network for being confirmed. Miner participants try to confirm them as a block by solving a computational puzzle (mining). Successful miners broadcast their block to the network to be chained to the blockchain, being rewarded for their success. In addition, all participants validate all data (transactions and blocks) broadcast across the network. It is a very attractive technology, since it maintains a *public*, *immutable* and *ordered* log of transactions which guarantees an *auditable* ledger, accessible by anyone.

The security and sustainability of blockchains, however, are not trivial and require increased participation, since each participant validates the diffused data, and keeps a replica of the entire blockchain. Participants consider worthwhile to join and stay in the system over time only if they find it *fair* [7]. Miner participants find the system fair if they are able to create blocks as they expected, and user participants find the system fair if they manage to cancel their transactions

and/or their transactions are confirmed as they expected. Considering miners, several formal studies have been conducted so far [6, 5, 18, 15], concluding that Bitcoin-like blockchains are not promoting participation of miners.

In [7], it has been for the first time shown that Bitcoin-like blockchains are unfair for user participants. It has also been discussed that for the time being it is not possible to explicitly cancel a transaction. There are only some workarounds that a user can try in order to cancel its transaction. One of these workarounds consist of trying to replace an old transaction with the new one with higher fees (Replace-By-Fee (RBF)[4]). This way, it is possible to create another transaction attempting to spend the same inputs but sending the money to the issuer itself. This is a workaround for cancellation of unconfirmed transactions by *implicitly* marking them to prevent their further use. However, once a transaction is confirmed, there is no workaround to reverse the situation. Since double spending is not allowed, the miners will not put both transactions in their blocks. However, there is no guarantee that the double spending transaction will arrive to the miners before the confirmation of the first transaction inside a block. Since it is like that, once a user decides to issue a transaction, s/he can never abandon this decision. Considering that a user is a rational agent that aims to maximize its utility by choosing to perform the actions with the optimal expected outcomes [17], this implies the utility is going to be minus infinity [7]. In the decision theory terms, this would mean assuming a user having an infinite interest on a transaction, which is hard to assume in realistic settings.

To this end, we propose a novel scheme where it is possible to cancel a transaction by rolling back its state whether it is confirmed in a block or not under certain conditions. This way, the utility of user agents can be maximized and, consequently, their willingness to leave the system decreases.

The contributions of this paper are as follows:

- A novel scheme based on a novel type of transaction that enables *explicit* cancellation of transactions.
- The implementation of such mechanism in Bitcoin-like blockchains.

The remainder of this paper is organized as follows. Section 2 gives the related work. Section 3 provides a formalization of the existing Bitcoin-like blockchain data structure considered in this paper. Section 4 provides a high-level Bitcoin-like blockchain protocol description as a rational multi-agent model where agents are using the aforementioned data structure. The proposed scheme for cancellation of transactions is presented in Section 5. Section 6 presents an analysis of the proposed scheme and, finally, the discussion and conclusions are provided in Sections 7 and 8, respectively.

[4] `https://en.bitcoin.it/wiki/Replace_by_fee`, last access on 16 July 2018.

2 Related Work

This study is based on our previous study on user (nodes that do not participate to the mining) fairness in blockchain systems [7]. The closest works in blockchain systems to our study are [8, 4, 1, 12].

Herlihy and Moir in [8] study the user fairness and consider as an example the original Tendermint[5] [3, 9]. The authors discussed how processes with malicious behaviour can violate fairness by choosing transactions [4] then they propose modifications to the original Tendermint to make those violations detectable and accountable. Helix [1] and HoneyBadgerBFT [12], on the other hand, attempt to design consensus protocols focusing on assuring a degree of fairness among the users by being resilient to transaction censorship where initially encrypted transactions are included in blocks, and only after their order is finalized, the transactions are revealed.

Another notion of fairness applied to the user side concerns the fair exchange in the e-commerce context [2] which is extended to the Bitcoin-Like scenario in [10] more in the sense that if there are two players performing an exchange then either both of them get what they want or none of them.

3 Blockchain Model

We model a blockchain ledger as a dynamic, append-only tree $B = \{b_0 \xleftarrow{\#_0} b_1 \xleftarrow{\#_1} ... \xleftarrow{\#_{l-1}} b_l\}$ where each block b_i $(0 < i \leq l)$ contains a cryptographic reference $\#_{i-1}$ to its previous block b_{i-1}, $l = |B|$ is the length of B, b_0 is the root block which is also called the *genesis block*, b_l is the furthest block from the genesis block which is referred to as the *blockchain head*, $h = |b_i|$ is the height of b_i (the length of the path from b_i block to b_0)[6] and $d = |B| - |b_i|$ is the depth of b_i (the length of the path from b_i block to b_l).

A block b_{i-1} can have multiple children blocks, which causes the situation called a $fork$. The *main branch* is then defined as the longest path l from any block to b_0 and is denoted as $B^\star$ where $|B^\star| = l$ and $B^\star \subseteq B$ such that $|B^x| < |B^\star|$ for all branches $B^x \subset B$ where $B^x \neq B^\star$. All branches other than the main branch are called *side branches*. If at any time, there exists more than one longest path with a length l (i.e. there are multiple heads), the blockchain ledger B is said to be *inconsistent* and thus $B^\star = \emptyset$. This situation disappears when a new block extends one of these side branches and creates $B^\star$. The blocks on the other branches are discarded and referred as *stale blocks*.

3.1 Block Model

We denote a block as $b_i = \langle \mathtt{h}_i, \Psi_i \rangle$ where $\mathtt{h}_i$ is the block header and Ψ_i is the block data. The block data Ψ_i contains a set of transactions organized as a Merkle tree

[5] Jae Kwon and Ethan Buchman. Tendermint. `https://tendermint.readthedocs.io/en/master/specification.html`, last access on 25 July 2018.

[6] For each transaction tx inside b_i, the block height $|tx|$ is the equal to h also.

[11]. The set of transactions θ_m are selected by the miner from its memory pool. Here it is important to note that, blocks have limited sizes[7] and thus the total size of the selected transactions can not exceed this limit[8].

3.2 Transaction Model

We model a transaction as $tx = \langle I, O \rangle$ where I is a list of inputs ($I \neq \emptyset$) and O is a list of outputs ($O \neq \emptyset$). Each input $i \in I$ references to a previous unspent output for spending it. Each output then stays as an Unspent Transaction Output (UTXO) until an input spends it. If an output has already been spent by an input, it cannot be spent again by another input (no double spending). We model the outputs as $o_i = \langle s_i, ¢_{o_i} \rangle$ where s_i is a set of tuples $s_i = \{(n_i, conds_{n_i})\}$ that define the conditions $conds_{n_i}$ for the receiver $n_i \in N$ to become owner of the coin $¢_{o_i}$ ($¢_{o_i} \geq 0$). The input that wants to spend the particular coin must satisfy at least one list of conditions $conds_{n_i}$, since an output (a coin) might be spendable by two different receivers $n_i, n_j \in N$ independently, although it will ultimately be spent by only one, on a first-come, first-served basis. All inputs of a transaction have to be spent in that transaction and the total input coins $¢_I$ has to be greater than or equal to the total output coins $¢_O$. The fee f_{tx} of a transaction tx is then modeled as $f_{tx} = ¢_I - ¢_O$. Depending on the fee to be paid, if there are still some coins left to be spent, the sender can add an output that pays this remainder to itself.

4 Network Model

In this section we provide a high-level Bitcoin protocol description as a rational multi-agent model[9] where rational agents chooses their actions/behavior with respect to their perceptions in order to maximize their utility.

We model the blockchain network as a dynamic directed graph $G = (N, E)$ where N denotes the dynamic rational agent (vertex) set, E denotes dynamic directed link (edge) set. A link $\langle n, m \rangle \in E$ represents a directed link $n \to m$ where $n, m \in N$, n is the owner of the link and n is the neighbor of m.

4.1 Agent Model

Each agent $n \in N$ has a list of its neighbors N_n where $N_n \subseteq N$ and $\forall m \in N_n | \langle n, m \rangle \in E$. An agent n can communicate one or more of its neighbors by exchanging messages of the form $\langle n, msg, d \rangle$ where n is the sender, msg is the type and d is the data contained. Using such messages, the (user) agents

[7] The current maximum block size in Bitcoin is 1 MB. See `https://bitcoin.org/en/glossary/block-size-limit`, last access on 13 July 2018.

[8] Average block size for Bitcoin is given in `https://blockchain.info/charts/avg-block-size`, last access on 13 July 2018.

[9] This description is based on the system and rational models given in [7].

issue transactions (by creating transactions messages and diffusing them to the network) to send coins to each other.

Each agent n has a memory pool Θ_n in which it keeps *unconfirmed transactions* that have input transactions, an orphan pool $\bar{\Theta}_n$ in which they keep *unconfirmed transactions* that have one or more missing input transactions (orphan transactions) and a blockchain ledger B_n in which they keep confirmed transactions where $\Theta_n \cap \bar{\Theta}_n = \emptyset$, $\Theta_n \cap B_n = \emptyset$ and $\bar{\Theta}_n \cap B_n = \emptyset$ always hold.

A (user) agent n can turn to be a miner agent if it chooses to create blocks for confirming the transactions (mining) in its memory pool Θ_m, and n is said to be a miner node if it started mining but has not stopped yet. The set of miner agents is then denoted by M where $M \subseteq N$. In order to mine, $n \in M$ has to solve a cryptographic puzzle (i.e. Proof of Work) using its hashing power. The successful miners are awarded by a fix amount of reward plus the totality of the transaction fees.

4.2 Behavior Model

A rational agent behaves according to its local perceptions and local knowledge, models uncertainty via expected values of variables or actions, and always chooses to perform the actions with the optimal expected outcome (among all feasible actions) for maximizing its utility [17]. Each rational agent $n \in N$ has a set of actions A_n and a utility function $\mathcal{U}_n$. Using A_n and $\mathcal{U}_n$, n uses a decision process where it identifies the possible sequences of actions to execute. We call these sequences as rational behaviors of n and denote as β. The objective of n is to choose the behaviors that selfishly keep $\mathcal{U}_n$ as high as possible.

We model the utility function of a rational agent $n \in N$ as

$$\mathcal{U}_n = u_0 + \sum_{i=1}^{k} \mathcal{U}(\beta_i) \tag{1}$$

where u_0 is the initial utility value, $k \geq 0$ is the number of behaviors executed so far and $\mathcal{U}(\beta_i)$ is the utility value of the behavior β_i. An agent $n \in N$ finds a system (i.e. the blockchain network) G *fair*, if the total satisfaction of its expectations $\mathcal{U}_n$ is above a certain degree τ_n where $\tau_n < u_0$ [7].

A utility value $\mathcal{U}(\beta_i)$ can also be interpreted as the *degree of satisfaction* experienced by the realization of β_i. The utility value $\mathcal{U}(\beta_i)$ is calculated as

$$\mathcal{U}(\beta_i) = \mathcal{R}(\beta_i) - \mathcal{C}(\beta_i) \tag{2}$$

where $\mathcal{R}(\beta_i)$ is the overall reward gained and $\mathcal{C}(\beta_i)$ is the overall cost spent for the execution of β_i.

When an agent needs to choose a behavior for execution, it needs to calculate its expected utility value. The expected value $\mathcal{E}(\beta_i)$ depends on the probabilities of the possible outcomes of the execution of β_i. We model the expected value as

$$
\begin{aligned}
\mathcal{E}(\beta_i) &= \sum_{j=1}^{m}(p_j \cdot \mathcal{U}(\beta_i^j)) \\
&= \sum_{j=1}^{m}(p_j \cdot (\mathcal{R}(\beta_i^j) - \mathcal{C}(\beta_i^j))) \\
&= \sum_{j=1}^{m}(p_j \cdot \mathcal{R}(\beta_i^j)) - \sum_{j=1}^{m}(p_j \cdot \mathcal{C}(\beta_i^j)) \\
&= \mathcal{R}^{\mathcal{E}}(\beta_i) - \mathcal{C}^{\mathcal{E}}(\beta_i)
\end{aligned}
\tag{3}
$$

where $m > 0$ is the number of possible outcomes, $\mathcal{U}(\beta_i^j)$ is the utility value of the possible jth outcome β_i^j, p_j is the probability of this outcome such that $\sum_{j=1}^{m} p_j = 1$, and $\mathcal{R}^{\mathcal{E}}(\beta_i)$ and $\mathcal{C}^{\mathcal{E}}(\beta_i)$ are the expected gain and the expected cost of β_i respectively.

In the following, we list the user and miner agents behaviors important for this study[10] conforming to the above description. To formalize the behaviors, we model a round based approach (like Garay et al. [6]) in which miner agents start creating a new block with at the beginning of the round and a round ends when a new block is successfully created by one of the miners. Both user and miner agents make their decisions on a roundly basis. This round-based model implicitly assumes that the block sent at at the end of the round is immediately delivered by all participants, i.e. communication delay is negligible with respect to block generation time.

Miner Agent Behaviors are as follows:

– *Selecting transactions.* When creating the next block, there is no required selection strategy and no known way to make any particular strategy required, but there are two transaction selection strategies popular among miners to include them into their blocks: (1) selecting the transactions with the highest fees to attempt to maximize the amount of fee income they can collect[11]. Since the size of blocks is limited, *miners could decide to deliberately exclude an unconfirmed transaction that has already been received with a lower fee.* This behavior would obviously delay the confirmation time of that transaction and affects its confirmation probability. (2) selecting the transactions with highest amount of coins moved to attempt to maximize the market value of the cryptocurrency [14]. Since the size of blocks is limited, *miners could decide to deliberately exclude an unconfirmed transaction that has already been received with a lower amount.* In this study, it is assumed that miners are using the 1st selection strategy and this behavior is modeled as $P(f)$ probability function.

[10] A detailed list of behaviors, along with their pseudo-codes, can be found in [7].

[11] `https://en.bitcoin.it/wiki/Transaction_fees`, last access on 20 July 2018.

User Agent Behaviors are as follows:

- *Issuing transactions.* We model issuing a transaction with a specific fee f as with the action of the form $issueTransaction(b,¢)$ where $b \in N$ is the receiver and ¢ is the amount of coins (Algorithm 1). For simplicity, it is assumed that a users agent has an ordered set of fees $\{f_1, f_2, \ldots, f_k\}$ to use. It is also assumed that $f_i < f_{i+1}$ where $0 < i < k$ and $0 \leq P(f_i) < P(f_{i+1}) \leq 1$ where $P(f)$ is the probability of a transaction with a fee f to be confirmed.
- *Leaving because of unfairness.* If at any time, an agent a finds G *unfair* ($\mathcal{U}_a \leq \tau_a$), it may decide to leave G if from its points of view it will not be possible to increase its overall utility above τ_a by calculating the expected values of its possible future behaviors. In other words, a may decide to leave G if $\mathcal{U}_a + \sum_{j=k}^{m} \mathcal{E}(\beta_j) \leq \tau_a$ where β_k, ..., β_m are sufficiently enough desired future behaviors of a.

Algorithm 1 The $issueTransaction(b,¢)$ action of a user agent a where a wants to send ¢ to b using an input set I by paying a fee f and returning the remaining $¢_r$ to itself. Note that each output o_i is required to be signed by the public key pk of the receiver. After creation, the transaction tx is diffused to the neighbors N_a. For more details see [7].

1: ***action issueTransaction(***b, ¢***)***
2: $I \leftarrow selectUnspentTransactionOutputs(B_a, ¢)$
3: $o_1 \leftarrow \langle (b, pk_b), ¢ \rangle$
4: $f \leftarrow estimateFee(I, \{o_1\})$
5: $¢_r \leftarrow ¢_I - ¢ - f$
6: $o_2 \leftarrow \langle \{a, pk_a\}, ¢_r \rangle$
7: $tx = \langle I, \{o_1, o_2\} \rangle$
8: $\Theta_a \leftarrow \Theta_a \bigcup \{tx\}$
9: $sendMessage(\langle a, \text{"inv"}, \mathcal{H}(tx) \rangle, N_a)$

In the next section, we present our proposed improvements to allow cancellation of transactions.

5 Cancellation of Transactions

In this section, we improve the models given in Section 3 and Section 4 to enable cancellation of transactions. We first make necessary definitions (Section 5.1), then provide a new model for canceling transactions (Section 5.2) and finally provide user agent behaviors for canceling transactions (Section 5.3).

5.1 Definition

We define cancellation of transactions as *intentionally marking a transaction by its issuer to prevent its further use*, i.e. negating the effect of a transaction

by its issuer and thus creating a new state as if the issuer of the transaction has never sent money to the recipients[12]. Here it should be emphasized that such cancellation can only be performed by the issuer of the transaction, since the receiver can always create another transaction that returns the money back to the original account and sign it. However, such a case is not considered as *cancellation* but as *refunding*.

5.2 Cancellation-enabled Transaction Model

In this subsection, we improve the transaction model given in Section 3.2 to allow cancellation of transactions, as defined in Section 5.1.

We consider that a transaction can be canceled by its issuer before a predefined *cancellation timeout* r_c. Such a condition impacts directly in the required amount of time to consider that a transaction is final[13]. We define a *final transaction* as a transaction that is not cancellable any more. However, it should be noted that this definition is relative to a branch of the blockchain. If a *side branch* that does not contain a transaction becomes longer than the current *main branch* that contains that transaction, technically that transaction is said to be rolled back, however this is different from canceling it.

Being $tx = \langle I, O \rangle$ a valid transaction issued by $n \in N$, a *cancellation transaction* txc is a transaction issued by n that gives ownership of the coins back to the user $n \in N$ that issued tx, although perhaps in a different address. In reality, our proposed solution gives more freedom than this definition, as we discuss further on.

5.3 Cancellation Behaviors of User Agents

Consider a user agent $a \in N$ that issues a transaction tx at round r_0 for sending ¢ coins to user node $b \in N$, with a fee of f_{tx}, has an interest $\mathcal{I}$ on tx and a waiting cost $C(¢)$.

Assuming that tx is confirmed at most at round n, the cumulative expected value $\mathcal{E}$ is:

$$\mathcal{E}(\beta_0) = \sum_{r=1}^{n} \overline{P(f)}^{r-1} \cdot P(f) \cdot (\mathcal{I} - f) - \sum_{r=1}^{n} \overline{P(f)}^{r-1} \cdot C(¢)^{r-1} \tag{4}$$

where $\mathcal{R}(\beta_0) = (\mathcal{I} - f)$ and $\mathcal{C}(\beta_0) = C(¢)$ (based on Equation 1 in [7]).

Now suppose user agent a decides to cancel tx at round r_1 ($r_0 \leq r_1 < r_c$). There are two possible user agent behaviors to cancel tx:

– β_1*: Canceling with cancellation transaction*,

[12] It is important to note that the fee paid to the miner is not considered.

[13] This complies with previous usage of transactions that are 'non-final':`https://bitcoin.stackexchange.com/questions/9165/whats-are-non-final-transactions`, last access on 13 July 2018.

- *Explicit* cancellation mechanism for transactions that are even confirmed, proposed by us in this paper (see Algorithm 2).

– β_2*: Canceling with a Replace-By-Fee (RBF) transaction*
- *Implicit* cancellation mechanism for only unconfirmed transactions, proposed by Bitcoin.

where user agents calculate the expected values considering the gain and the cost of each behavior as in [7]. For simplicity, and without loss of generality, we consider that fees are not changing per round.

β_1: Canceling with cancellation transaction User creates a transaction txc at round r_1 with a fee f_{txc}, where f_{txc} has a value such that txc is supposed to be appended to the blockchain in less than round $r_c + r$, being r the round in which tx hits the blockchain.

The expected gain $\mathcal{R}^{\mathcal{E}}(\beta_1)$ of behavior β_1 is then as follows:

$$\begin{aligned}\mathcal{R}^{\mathcal{E}}(\beta_1) &= \left(\sum_{r=r_0}^{n} \overline{P(f_{tx})}^{r-1} P(f_{tx}) \cdot \Big(\sum_{s=r+1}^{r+r_c} \overline{P(f_{txc})}^{s-1-r} P(f_{txc}) \Big) \right) \\ &\quad \cdot (I - f_{tx} - f_{txc}) \\ &= \left(\sum_{r=1}^{n} \overline{P(f_{tx})}^{r-1} P(f_{tx}) \cdot \Big(\sum_{s=1}^{r_c} \overline{P(f_{txc})}^{s-1} P(f_{txc}) \Big) \right) \\ &\quad \cdot (\mathcal{I} - f_{tx} - f_{txc})\end{aligned} \tag{5}$$

where $\mathcal{R}(\beta_1) = (\mathcal{I} - f_{tx} - f_{ftxc})$, and r and s ($r_0 < r \leq s$) are the rounds in which tx and txc are appended to the blockchain, respectively. Note that, for $r_1 < r$, the transaction txc is orphan and thus it can be appended in the blockchain only at $r = s$ or $r < s$. Furthermore, if $r_1 > s$, then the best option is to use this behavior (since RBF transactions cannot be used anymore). Therefore, we compare for cases where $r_0 \leq r_1 \leq r$. For further simplification, without loss of generality, we use $r_0 = 1$.

The series given in Equation 5 follow two nested geometric distributions $X \backsim (p_{tx}), Y \backsim (p_{txc})$, with probabilities of success $p_{tx} = P(f_{tx})$ and $p_{txc} = P(f_{txc})$. Therefore, considering $x = q_{tx} = 1 - p_{tx}$ and $y = q_{txc} = (1 - p_{txc})$, we have the following solution to the series:

$$\mathcal{R}^{\mathcal{E}}(\beta_1) = \left(\sum_{r=1}^{n} x^{r-1}\, p_{tx} \cdot \Big(\sum_{s=1}^{r_c} y^{s-1} p_{txc} \Big) \right) \cdot (\mathcal{I} - f_{tx} - f_{txc}) \tag{6}$$

We can extract all constant values into $c = p_{tx} \cdot p_{txc} \cdot (\mathcal{I} - f_{tx} - f_{txc})$, and consider the solution to

$$g_{\beta_1}(n) = \sum_{r=1}^{n} x^{r-1} \cdot \sum_{s=1}^{r_c} y^{s-1} \cdot c = \sum_{r=0}^{n-1} x^{r} \cdot \sum_{s=0}^{r_c-1} y^{s} \cdot c \tag{7}$$

As such, notice that, applying basics of geometric convergent series, the inner series $\sum_{s=0}^{r_c-1} y^s$ converges to $\frac{1-y^{r_c}}{(1-y)}$ and thus the outer one:

$$\begin{aligned} g_{\beta_1}(n) &= \frac{(1-x^n)(1-y^{r_c})}{(1-x)(1-y)} \cdot (1-x)(1-y)(\mathcal{I} - f_{tx} - f_{txc}) \\ &= (1-x^n)(1-y^{r_c})(\mathcal{I} - f_{tx} - f_{txc}) \end{aligned} \tag{8}$$

And thus, being the expected gain when $g_{\beta_1}(n)$ tends to infinity:

$$\lim_{n\to\infty} g_{\beta_1}(n) = (1-y^{r_c})(\mathcal{I} - f_{tx} - f_{txc}) \tag{9}$$

As for the cost resulted from the Time Value of Money (TVM)[14], both transactions use the same amount money, but we consider that the TVM remains for as long as the transaction is not fully canceled. As such, the expected cost of behavior $\mathcal{C}^{\mathcal{E}}(\beta_1)$ of β_1 is modeled as follows:

$$\mathcal{C}^{\mathcal{E}}(\beta_1) = \sum_{r=1}^{\infty} \overline{P(f_{tx})}^{r-1} \cdot \Big(\sum_{s=r_1}^{r+r_c} \overline{P(f_{txc})}^{s-1-r_1} \cdot C(¢_{tx})^{s-1} \Big) \tag{10}$$

Where $C(¢)$ represents the waiting cost derived of the time value of money, per round. These series are identical to $g_{\beta_1}(n)$ applying $x = q_{tx}$, $y = q_{txc} \cdot C(¢_{tx})$ and the constant c by $d = C(¢_{tx})^{r_1}$, therefore, we define $h_{\beta_1}(n)$:

$$h_{\beta_1}(n) = \sum_{r=1}^{n} x^{r-1} \cdot \sum_{s=1}^{r+r_c-r_1} y^{s-1} \cdot d = \sum_{r=0}^{n-1} x^{r} \cdot \sum_{s=0}^{r+r_c-r_1-1} y^{s} \cdot d \tag{11}$$

For which the result is analogously obtained as for $g_{\beta_1}(n)$ (note that $|xy| < 1$ to guarantee convergence):

$$h_{\beta_1}(n) = \Big(\frac{1-x^n}{(1-y)(1-x)} - y^{r_c-r_1} \cdot \frac{1-(xy)^n}{(1-y)(1-xy)} \Big) \cdot d \tag{12}$$

And thus, again, being the cost when $h_{\beta_1}(n)$ tends to infinity:

$$\begin{aligned} \lim_{n\to\infty} h_{\beta_1}(n) &= \Big(\frac{1}{(1-y)(1-x)} - y^{r_c-r_1} \cdot \frac{1}{(1-y)(1-xy)} \Big) \cdot d \\ &= \Big(\frac{1}{(1-q_{txc}C(¢_{tx}))(1-q_{tx})} - (q_{txc}C(¢_{tx}))^{r_c-r_1} \\ &\quad \cdot \frac{1}{(1-q_{txc}C(¢_{tx}))(1-q_{tx}q_{txc}C(¢_{tx}))} \Big) \cdot C(¢_{tx})^{r_1} \end{aligned} \tag{13}$$

[14] The concept that indicates that money available at the present time worths more than the identical sum in the future due to its potential earning capacity.

Algorithm 2 The $issueCancellableTransaction(b,\ ¢,\ r_c)$ and $issueCancellationTransaction(tx)$ actions of a user agent a. The former contains the necessary constraints (e.g., txc is only valid if current block-height $|B_a|$ is less or equal than $|tx| + r_c$) for the latter action to work.

1: ***action issueCancellableTransaction(***$b,\ ¢,\ r_c$***)***
2: $I \leftarrow selectUnspentTransactionOutputs(B_a,\ ¢)$
3: $o_1 \leftarrow \langle\{(b, sk_b), (a, sk_a\ and\ |B_a| \leq |tx| + r_c)\}, ¢\rangle$
4: $f \leftarrow estimateFee(I,\ f_{tx})$
5: $¢_r \leftarrow ¢_I - ¢ - f$
6: $o_2 \leftarrow \langle\{a, pk_a\}, ¢_r\rangle$
7: $tx = \langle I, \{o_1, o_2\}\rangle$
8: $\Theta_a \leftarrow \Theta_a \bigcup \{tx\}$
9: $sendMessage(\langle a, \text{"inv"}, \mathcal{H}(tx)\rangle, N_a)$
10:
11: ***action issueCancellationTransaction(***tx***)***
12: $i_1 \leftarrow tx.o_1$
13: $o_1 \leftarrow \langle(a, pk_a), ¢\rangle$
14: $f \leftarrow estimateFee(i_1,\ f_{tx})$
15: $¢_r \leftarrow ¢_I - ¢ - f$
16: $txc = \langle i_1, o_1\rangle$
17: $\Theta_a \leftarrow \Theta_a \bigcup \{txc\}$
18: $sendMessage(\langle a, \text{"inv"}, \mathcal{H}(txc)\rangle, N_a)$

β_2: Canceling with a Replace-By-Fee (RBF) transaction User agent issues a transaction txr with a greater fee than the fee of tx, hoping to replace it before tx hits the blockchain (i.e. $r_c = 0$). Such *implicitly* tells miner agents to ignore tx. In this case, we model the reward as follows:

$$\mathcal{R}^{\mathcal{E}}(\beta_2) = \left(\sum_{r=r_0}^{\infty} \overline{P(f_{tx})}^{r-1} P(f_{tx}) \cdot \Big(\sum_{s=r_1}^{r} \overline{P(f_{txr})}^{s-1-r_1} P(f_{txr}) \Big) \right) \cdot (\mathcal{I} - f_{txr}) \tag{14}$$

and its expected cost:

$$\mathcal{C}^{\mathcal{E}}(\beta_2) = \sum_{r=r_0}^{\infty} \overline{P(f_{tx})}^{r-1} \cdot \Big(\sum_{s=r_1}^{r} \overline{P(f_{txr})}^{s-1-r_1} \cdot C(¢_{tx})^{s-1} \Big) \tag{15}$$

For both, the process is analogous to extract $g_{\beta_2}(n)$ and $h_{\beta_2}(n)$, obtaining:

$$\begin{aligned} g_{\beta_2}(n) = \Big(\frac{1 - x^n}{1 - x} - y^{r_1} \cdot \frac{1 - (xy)^n}{1 - xy} \Big) \cdot (1 - x)(\mathcal{I} - f_{txr}), \\ with\ x = q_{tx},\ y = q_{txr} \end{aligned} \tag{16}$$

$$\lim_{n\to\infty} g_{\beta_2}(n) = \Big(\frac{1}{1-q_{tx}} - q_{txr}^{r_1} \cdot \frac{1}{1-q_{tx}q_{txr}}\Big) \cdot (1-q_{tx})(\mathcal{I} - f_{txr}) \tag{17}$$

and

$$\begin{aligned} h_{\beta_2}(n) &= \Big(\frac{1-x^n}{(1-y)(1-x)} - y^{r_1} \cdot \frac{1-(xy)^n}{(1-y)(1-xy)}\Big) \cdot d, \\ &with\ \ x = q_{tx},\ y = q_{txr} \cdot C(¢_{tx}),\ d = C(¢_{tx})^{r_1},\ |xy| < 1 \end{aligned} \tag{18}$$

$$\begin{aligned} lim_{n\to\infty} h_{\beta_2}(n) &= \Big(\frac{1}{(1-y)(1-x)} - y^{r_1} \cdot \frac{1}{(1-y)(1-xy)}\Big) \cdot d \\ &= \Big(\frac{1}{(1-q_{txr}\cdot C(¢_{tx}))(1-q_{tx})} - (q_{txr}\cdot C(¢_{tx}))^{r_1} \\ &\quad \cdot \frac{1}{(1-q_{txr}\cdot C(¢_{tx}))(1-q_{tx}q_{txr}\cdot C(¢_{tx}))}\Big) \cdot C(¢_{tx})^{r_1} \end{aligned} \tag{19}$$

In the next section, we will compare β_1 and β_2 by quantitatively analysing the equations formulated in this section.

6 Analyses and Results

We analyzed the rational behaviors of user agents proposed in Section 5.3 using gnuplot 5.2.4[15]. It is assumed that the initial utility values u_0 of all agents are the same and high enough from the threshold τ. In the following, we provide results of these analyses employing synthetic data.

Recalling Equation (4), it is clear that, depending on the values of $\overline{P(f)}$ and $C(¢)$, the expected value $\mathcal{E}$ may or may not converge to $-\infty$. For the geometric distributions of $\mathcal{R}^{\mathcal{E}}(\beta_1)$ and $\mathcal{R}^{\mathcal{E}}(\beta_2)$, we can see that they always converge, since all three $\overline{P(f_{tx})}$, $\overline{P(f_{txc})}$ $\overline{P(f_{txr})}$ are positive, and strictly less than 1. For the expected costs, $\mathcal{C}^{\mathcal{E}}(\beta_1)$ converges to the value listed in equation(13) if and only if $|q_{txc}\cdot C(¢_{tx})| < 1$. Analogously, $\mathcal{C}^{\mathcal{E}}(\beta_2)$ converges to the value of equation (19) if and only if $|q_{txr}\cdot C(¢_{tx})| < 1$.

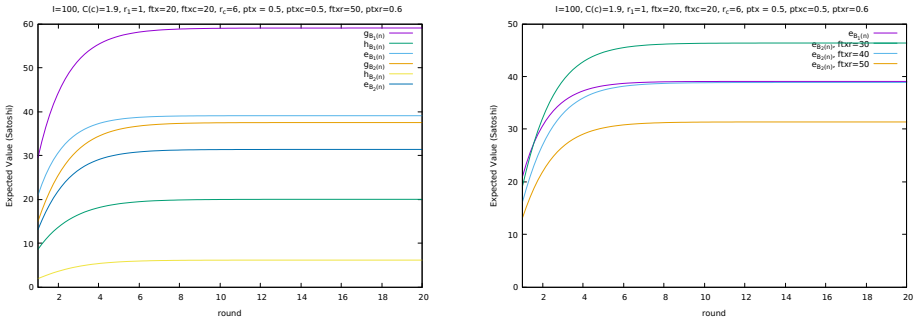
Obviously, many values will have a strong impact in the results. Specifically, the correlation between the fees of a transaction f_{tx} and how much they increase the probability of such transaction to be included in each round $P(f_{tx})$. There are some online results on the average fees in Bitcoin and the average amount of blocks (rounds) a transaction takes with each fee [16]. Again, it is easy to see that the average amount of blocks is the expected value of a geometric distribution of which the probability p is the one we are looking for, and, therefore, one can obtain this value solving the series. However, this is out of the scope of this document. Furthermore, we show further on that the cancellation transaction behavior can be better even under optimistic values for this probability, in which a small increase of the fee f_{tx} incurs in a big increase in the probability $P(f_{tx})$.

Figure 1 left shows the gain, cost, and expected values $\mathcal{E}$ of both behaviors, as functions on the number of rounds, and fixing the rest of the variables. Notice

[15] `http://www.gnuplot.info/`, last access on 24 July 2018.

[16] `https://bitcoinfees.earn.com/`, last access on 24 July 2018.

that here we consider that 20 Satoshis give a probability of $P(20) = 0.5$, while $P(50) = 0.6$. In this case, we can see how our approach is better, regardless of the round. To the right, we compare our approach with different assumptions on the increase required in the fee to increase the probability of a transaction being included. We can see how our approach is not always the best, and for example when only 10 Satoshis are required to increase the probability to $P(30) = 0.6$, using a Replace-By-Fee transaction is a better behavior.



(a) (b)

Fig. 1: Expected reward g, cost h, and value functions e as functions on the number of rounds, comparing both behaviors (a). Also, comparison of different assumptions on the relationship between f_{tx} and $P(f_{tx})$ for canceling with cancellation transaction behavior β_1 and canceling with a RBF transaction behavior β_2 (b).

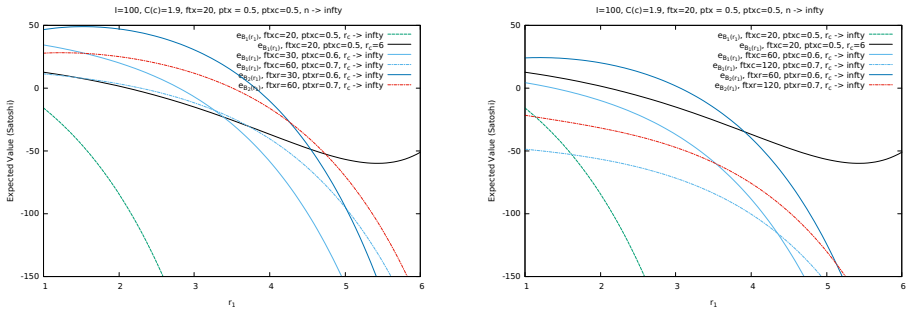
One can note that another important variable fixed in figure 1 is r_1, that is, the round at which the user decides he wants to cancel the transaction tx, issued at round 1. In figure 1, we assume $r_1 = 1$. However, it is important to consider that the user may want to cancel the transaction later than when they issued it. Finally, r_c can also have an impact in the results, since it increases the time in which a transaction can be cancellable by the cancellation transaction behavior β_1, but it also increases the cost of such behavior, since it can lead to a greater waiting time. In the following, we study how the results vary depending on r_1 and r_c.

For such cases where the series converge, we consider the values in rounds in the infinite (that is equations (9, 13, 17, 19)), and tweak other values. Firstly, it is easy to note that behavior β_1 also converges for $r_c \to \infty$:

$$\begin{aligned} \lim_{n\to\infty, r_c\to\infty} \mathcal{E}(\beta_1)(n, r_c) &= \lim_{n\to\infty, r_c\to\infty} g_{\beta_1}(n, r_c) - h_{\beta_1}(n, r_c) \\ &= (\mathcal{I} - f_{tx} - f_{txc}) - \frac{C(¢_{tx})^{r_1}}{(1 - q_{txc}C(¢_{tx}))(1 - q_{tx})} \end{aligned} \tag{20}$$

While $\lim_{n\to\infty, r_c\to\infty} \mathcal{E}(\beta_2)(n)$ does not depend on r_c by construction. It is possible to see, however, how the gain decreases for β_2 when r_1 increases, while in β_1 this is irrelevant. Nevertheless, the cost increases with r_c for β_1, since this leads to higher waiting cost.

Figure 2 left compares several values depending on r_1, when the amount of rounds n tends to infinity. One can see how, for $r_c = 6$, it is a better approach to use β_1 for $r_1 > 4$. Furthermore, even if r_c tends to infinity, and being optimistic in terms on the correlation between fees and probability of hitting the blockchain, β_1 seems to be a better behavior for $r_1 > 5$. Nevertheless, if one decides within the first rounds to cancel the transaction, then it is better to use behavior β_2 and issue a Replace-By-Fee transaction. Recall, however, that figure 2 plots values for $n \to \infty$. For constant number of rounds, with $r_1 = 1$, figure 1 already showed that β_2 is only a better approach when one can be optimistic about the probability of a transaction with a slightly higher fee hitting the blockchain.



(a) (b)

Fig. 2: Expected values for canceling with cancellation transaction behavior β_1 and canceling with a RBF transaction behavior β_2 as a function on r_1, with different values for r_c, f_{tx} and $P(f_{tx})$ where $\mathcal{I} = 100$. To the left, optimistic values for $P(f_{tx})$ (better for β_2).

As a result, we suggest always creating the initial transaction tx as if it can be canceled by a cancellation transaction txc. However, depending on the conditions, the user may choose to issue a Replace-by-Fee transaction txr as well as trying to cancel tx. Nevertheless, a transaction that can be canceled in r_c rounds leaves this transaction as non-final for the first r_c, increasing the block-depth required for a receiver of coins to consider full ownership of such coins. The receiver can however move the coins to a new UTXO. Besides, txr is more flexible and may work even if tx is prepared for being canceled by txc.

In general, it seems as a good approach to firstly issue an RBF transaction, and after round r_1 issue a cancellable transaction such that $f_{txc} + f_{tx} = f_{txr}$. Otherwise, miners will always choose the one that gives higher reward. It is possible that the left side should be greater, $f_{txc} + f_{tx} > f_{txr}$, to account for

more space used by two transactions, in such a way that the reward for the miner equalizes.

7 Discussion

In this section, we discuss the proposed mechanism, and the results obtained from several perspectives: *fairness*, *security* and *implementability*.

7.1 Fairness

To the best of our knowledge, this is the first study focusing on providing an *explicit mechanism for cancellation of transactions* to improve the fairness for *users*. In other words, the proposed mechanism is a first step towards guaranteeing fairness for users, given that fairness is the overall satisfaction of rational agents. Moreover, we showed that the users as rational agents have no incentive to choose the Replace-by-Fee behavior (β_2) since its *cancellation timeout* is much less shorter. As a result, it can be said that the proposed mechanism is superior to the Replace-by-Fee workaround provided by the existing protocol.

7.2 Security

To avoid double-spending attacks and inconsistencies, blockchains need a selection strategy based on a predefined criteria. Bitcoin uses the *longest chain* strategy for selecting the main branch. Moreover, there can be situations where the network is partitioned for some time and then reconnects, with[17] or without any malicious participant. The question is: which branch should be followed in case the *cancellation transaction* exists only in one branch, and there is another transaction that is spending the transaction we wish to cancel in another branch? Should we throw away one or more branches that creates such inconsistencies? Which branch should be followed?

In our opinion, pruning any chain (i.e. reducing the length of the chain) is dangerous and weakens its security level, which is directly proportional to the work that must be done to *replace* such a chain and allow an attacker to target the chain more easily. Thus, we think that the blockchain should remain *append-only*. Furthermore, we claim that the existing longest chain rule should remain the same. Even though with this setting the *cancellation* might be at times ignored, the security of the blockchain is more important than what the user desires. Furthermore, an upgrade to support cancellation of transactions should be backward compatible, not to require a hard-fork for existing Bitcoin-like blockchains.

[17] If there is a malicious participant that is partitioning the network, this is called a man-in-the-middle-attack.

7.3 Implementability

As shown in Section 6, the proposed cancellation mechanism is feasible. This section shows its implementability in Bitcoin-like blockchains. Many such blockchains (Bitcoin, Bitcoin Cash, Litecoin, etc.) already give support to a similar feature as the one described in this document, motivated by the implementation of 2nd-layers in their network, such as the Lightning Network [16].

Bitcoin, for instance, provides support for lightning, thanks to the implementation of Bitcoin Improvement Proposal (BIP) 112[18] in the system. BIP 112 implements the opcode[19] `OP_CHECKSEQUENCEVERIFY` (typically referred to as `OP_CSV`), that *prevents a non-final transaction from being selected for inclusion in a block until the corresponding input has reached the specified age, as measured in block-height or block-time*. In this case, a non-final transaction refers to the fact that it has not reached the specified age. As such, BIP 112 already offers functionality of giving preference to some specific node to spend an UTXO.

Algorithm 3 Node a sends coins to node b in a transaction.

```
1: IF
2:    <pubkey of node b> CHECKSIG
3: ELSE
4:    "30d" CHECKSEQUENCEVERIFY DROP
5:    <pubkey of node a> CHECKSIG
6: ENDIF
```

For example, node a can pay node b in a transaction with the redeem script given in Algorithm 3. In this case, node b can spend the output at any time, while node a needs to wait 30 days. After such time, any of the two can independently spend the output. However, we would like the inverse functionality where node a can still cancel the issued transaction in the first 30 days, if node b has not spent this output (Algorithm 4). Before 30 days, both node a and node b can independently spend the output. After 30 days, only node b can spend it.

As illustrated in BIP 68[20], and by James Prestwich[21], `OP_CSV` compares the top stack item to the input's sequence_no field. Thus, the top stack item is parsed just as the sequence_no field for `nSequence` (the input-level relative time-lock). That is, it interprets 18 of the 32 bits (the remaining 14 bits are still undefined). There are two special flags: the disable and the type flag. The disable flag (bit 31, the 32nd least significant bit) specifies that logs are disabled. The type flag

[18] `https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki`, last access on 28 August 2018.

[19] Operation codes from the Bitcoin Script language which push data or perform functions within a pubkey script or signature script.

[20] `https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki`, last access on 28 August 2018.

[21] `https://prestwi.ch/bitcoin-time-locks/`, last access on 28 August 2018.

Algorithm 4 Node a cancels the transaction.

```
1: IF
2:    <pubkey of node b> CHECKSIG
3: ELSE
4:    "-30d" CHECKSEQUENCEVERIFY DROP
5:    <pubkey of node a> CHECKSIG
6: ENDIF
```

(bit 22, the 23rd least significant bit) specifies the type of information: if set, the remaining 16 least significant bits are interpreted in units of `512` seconds granularity, if not, they are interpreted as block-height. The flag ($1 \leq\leq 22$) is the highest order bit in a 3-byte signed integer for use in bitcoin scripts as a 3-byte `PUSHDATA` with `OP_CHECKSEQUENCEVERIFY` (BIP 112), as detailed in the specification of BIP 68.

At the time of writing, using `OP_CSV` with value `0x00400001` and `0x00C00001` applies the same timelock: a relative locktime of 512 seconds. Moreover, the specification says `OP_CSV` "errors if the top stack item is less than 0". We propose, however, to consider the sign bit as the sign flag, and interpret instead this negative value exactly as detailed in the aforementioned example.

`OP_CSV` is useful for considering a relative time from the inclusion of the transaction in the blockchain. If, instead, one would like to specify an absolute time, or a block-height, this is possible using `OP_CHECKLOCKTIMEVERIFY` (`OP_CLTV`) (see BIP 65[22]). Analogously, we propose the same consideration for `OP_CLTV`. These two simple backward compatible features can be implemented after proposing them in a new BIP. Also, for simplicity, and to comply with the current definition of `OP_CSV` and `OP_CLTV`, we propose referring to each upgrade as `OP_CHECKSEQUENCEVERIFYINVERSE` (`OP_CSVI`) and `OP_CHECKLOCKTIMEVERIFYINVERSE` (`OP_CLTVI`).

As such, we show that it is possible, and convenient, to implement the required opcodes to give support for cancellation transactions in Bitcoin, making use of the sign bit, that was useless until now, although it was respected as the sign bit. We do this in a backward compatible way.

8 Conclusions

In this paper, we proposed a novel *explicit* transaction cancellation mechanism that cancels issued transactions under certain conditions. Such a mechanism increases the fairness for the users and thus increases the security and sustainability of the blockchain system. To avoid security issues and related complex analyses, the proposed mechanism sticks as much as possible to the original Bitcoin protocol, introducing mechanisms to improve the degree of fairness of the

[22] `https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki`, last access on 28 August 2018.

system. To this end, we showed the implementation of our approach for Bitcoin, and consider also its implementability for Bitcoin-like blockchains.

References

1. Asayag, A., Cohen, G., Grayevsky, I., Leshkowitz, M., Rottenstreich, O., Tamari, R., Yakira, D.: Helix: A scalable and fair consensus algorithm. Tech. rep., Orbs Research (2018), https://orbs.com/wp-content/uploads/2018/07/Helix-Consensus-Paper-V1.2-1.pdf
2. Asokan, N.: Fairness in electronic commerce (1998)
3. Buchman, E.: Tendermint: Byzantine Fault Tolerance in the Age of Blockchains. Ph.D. thesis, University of Guelph (6 2016)
4. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 154–167. ACM (2016)
5. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Int. Conf. on Fin. Cryp. and Data Security. pp. 436–454. Springer (2014)
6. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin Backbone Protocol: Analysis and Applications, pp. 281–310. Springer Berlin Heidelberg, Berlin, Heidelberg (2015), http://dx.doi.org/10.1007/978-3-662-46803-6_10
7. Gürcan, Ö., Del Pozzo, A., Tucci-Piergiovanni, S.: On the bitcoin limitations to deliver fairness to users. In: Panetto, H., Debruyne, C., Gaaloul, W., Papazoglou, M., Paschke, A., Ardagna, C.A., Meersman, R. (eds.) On the Move to Meaningful Internet Systems. OTM 2017 Conferences. pp. 589–606. Springer Int. Publishing, Cham (2017)
8. Herlihy, M., Moir, M.: Enhancing accountability and trust in distributed ledgers. CoRR abs/1606.07490 (2016), http://arxiv.org/abs/1606.07490
9. Kwon, J.: Tendermint: Consensus without mining. Tech. rep., Tendermint (2014), https://tendermint.com/static/docs/tendermint.pdf
10. Liu, J., Li, W., Karame, G.O., Asokan, N.: Towards fairness of cryptocurrency payments. arXiv preprint arXiv:1609.07256 (2016)
11. Merkle, R.C.: A Digital Signature Based on a Conventional Encryption Function, pp. 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg (1988)
12. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of bft protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 31–42. CCS '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2976749.2978399
13. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
14. Pappalardo, G., di Matteo, T., Caldarelli, G., Aste, T.: Blockchain inefficiency in the bitcoin peers network. CoRR abs/1704.01414 (2017), http://arxiv.org/abs/1704.01414
15. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. IACR Cryptology ePrint Archive 2016, 454 (2016)
16. Poon, J., Dryja, T.: The bitcoin lightning network. i pp. 1–22 (2015)
17. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education (2010)
18. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 515–532. Springer (2016)