A Formal Analysis of Blockchain Consensus

Cosimo Laneve
Dept. of Computer Science and Engineering, University of Bologna – INRIA Focus
cosimo.laneve@unibo.it

Adele Veschetti
Dept. of Computer Science and Engineering, University of Bologna – INRIA Focus
adele.veschetti2@unibo.it

Abstract

Nakamoto's blockchain protocol implements a distributed ledger on peer-to-peer asynchronous networks. In this paper we study so-called forks that may devolve the ledger into inconsistent copies. We define the behaviour of blockchain's key participants – the miners – as stochastic pi calculus processes and describe the whole system as a parallel composition of miners. We therefore compute the probability that ledgers turn into a state with more severe inconsistencies, e.g. with longer forks. The instances of this probability with current rate-values give upper-bounds for Bitcoin and Ethereum. We also study how the presence of hostile nodes mining blocks in wrong positions impacts on the consistency of the ledgers.

2012 ACM Subject Classification Theory of Computation → Models of computations; Concurrency → Process Calculi

Keywords and phrases Distributed consensus, distributed ledgers, blockchain, stochastic pi calculus, forks.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Blockchain is an emerging technology that implements a distributed ledger on peer-to-peer asynchronous networks that are dynamic (nodes may either join or leave) [19]. It enables many applications, including cryptocurrencies – the Bitcoin is the most famous one –, decentralized applications – the Ethereum smart contracts are very popular nowadays –, voting systems and other application specific protocols.

When implementing a distributed ledger on a dynamic peer-to-peer asynchronous network one has to address the problem of inconsistent updates of the ledger performed by different nodes. This problem is actually a distributed consensus variant, which has been known to be unsolvable since 1985 [9]. To overcome this shortcoming, blockchain uses an ingenious breakthrough: it guarantees a so-called *eventual consistency* whereby the various replicas of the ledger may be temporarily inconsistent in at most the last m blocks.

The blockchain protocol is very complex and the current research is actively involved in understanding all the critical points because they might be used for designing possible attacks. For example, inconsistencies of the ledger replicas, which are called *forks*, may be used to rewrite the transaction histories and making the blockchain devolve to a wrong state. This might be due to adversaries that are powerful enough to create new blocks more frequently than others (e.g. *have a larger hashing power for mining blocks than other nodes*). Or it might be due to adversaries that broadcast their block updates more quickly than other nodes (e.g. the *delays of communications* is larger for a subset of senders than for others because, for instance, they are better connected to the network). Or because new nodes are entering in the protocol and may cluster their decisions. Or even to a wrong ratio between the mining rate and the number of (honest) nodes in the network. Forks of significative lengths T already occurred in blockchain-based systems: in March 2013 a 24-blocks fork

occurred in Bitcoin because some nodes upgraded their software to a version creating blocks that were not recognized by nodes running the older version. (This fork was resolved by human intervention, rejoining the chain of old-versioned block, with a loss of block rewards.) Two other forks, in summer and fall 2017, were driven by the Bitcoin Community and led to two different cryptocurrencies: Bitcoin Cash and Bitcoin Gold. Ethereum has experimented a more dramatic fork in the summer 2016, in correspondence of the TheDAO attack. In this case, no agreement was found (whether rolling-back or not) and Ethereum split in two incompatible branches (Ethereum and Ethereum Classic).

Blockchain is a concurrent system with asynchronous communications between nodes called *miners* (we are sticking to the key participants of the protocol). Henceforth, in order to model blockchain formally, we decided to describe it by means of a *process calculus* where the system is viewed as a *parallel composition* of processes whose basic actions are communications and internal moves. However, actions have a *duration* in blockchain, i.e. they require time to be completed. For example, this is the case for minting a block by a node or for broadcasting blocks in the network. To model this feature faithfully we were forced to use a stochastic variant of process calculi. In particular, since the durations of mining or broadcasting actions can be expressed by an *exponential distribution* with a so-called *rate parameter* [1, 26, 8, 3], it turned out that the correct mathematical framework of blockchain corresponds to a Continuous Time Markov Chain transition system. This is why the process calculus we have chosen is the stochastic pi calculus [24, 4] (actually an extension of it that includes ledgers). In Section 3 we define our calculus. In this calculus, all enabled actions in a state attempt to proceed, but fastest ones succeed with higher probabilities; e.g. if a state has n outgoing transitions with rates $r_1, \cdots, r_n$, the probability that the i-th transition is taken is $r_i/(\sum_{1\leq j\leq n} r_j)$.

The ledgers and their properties are described in Section 2, following the ledger description in [19]. The modelling of the blockchain protocol is given in Section 4. In this section we also compute the probability of devolving into a "larger inconsistency", e.g. transiting from a state with a fork of length m to a state with a fork $m+1$. This probability, which depends on the number of nodes, their hashing power and the latency of the network, has required a time-consuming analysis of the stochastic transition system due to the state explosion with respect to the number of nodes. According to our results, given the current rate-values, the above probability is less than $\mathtt{10}^{-3}$ in the Bitcoin system, while it is less than $\mathtt{10}^{-2}$ in the Ethereum system. We notice that these upper bounds simply follow by instantiating the formula we compute with the rate-values of the two systems.

In Section 5 we apply the same technique to analyze an attack to blockchain that has been already studied in [19]: the presence of hostile nodes mining new blocks in positions that are different from the correct one (the first block inserted at maximal depth). The probability that we compute depends on the hashing power of the attacker and the depth m of the fork created by the hostile node. For example, if `BTC.com`, which is a cluster currently retaining the 14,7% of the Bitcoin hashing power, decided to become hostile, then the probability to create an alternative attacker chain and achieving consensus from the other nodes is $\mathtt{6}^{-m}$.

We report our concluding remarks in Section 6. For space constraints, the proofs of the main statements are omitted; reviewers may find them in the Appendix.

Related work.

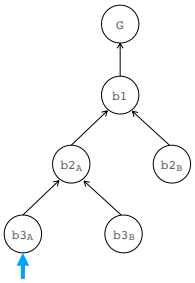
The blockchain protocol was introduced by Haber and Stornetta [13] and only in the last few years, because of Bitcoin, the problem of analyzing the consistency of the ledgers has caught the interest of several researchers. In [10], Garay *et al.* demonstrate the correctness of the

protocol when the network communications are synchronous, focusing on its two key security properties: *Common Prefix* and *Chain Quality*. The first property guarantees the existence of a common prefix of blocks among the chain of honest players; Chain Quality constrains the number of blocks mined by hostile players, when the honest players are in the majority and follow the protocol. The extension of this analysis to asynchronous networks with bounded delays of communications and with new nodes joining the network has been undertaken in [21]. In the above contributions, the properties are verified by using oracles that drive the behaviours of actors. Then, combining the probabilistic behaviours and assuming possible distributions, one computes expected values. In contrast with the above works, in [23], Pirlea and Sergey propose a formalization of blockchain consensus focusing on the notion of global system safety. They present an operational model that provides an executable semantics of the system where nondeterminism is managed by external schedules and demonstrate the correctness by means of a proof assistant. The main difference between these contributions and our work is that we formalize the blockchain protocol as a stochastic system (with exponential distribution of durations) and derive the properties by studying the model. In fact, the probabilities that we compute are, up-to our knowledge, original. As regards stochastic models and blockchain, few recent researches use them to select optimal strategies for maximizing profit of a player [1] and for formalizing interactions between miners as a game [5, 2].

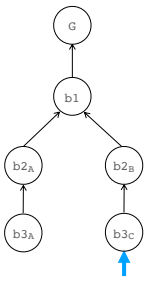
A number of researches address attacks to the blockchain protocol. The works [6, 25, 12] address the delays of communications and [25] also demonstrates that an attacker with more than 51% of the total hashing power could change the past transactions. A larger set of attacks is analyzed in [18, 10, 21], where it is also proved that the blockchain protocol is safe as long as honest miners are in the majority. In [20], Ozisik and Levine give a very detailed description of Nakamoto's double spending attack, gathering the mathematics for its modelling. The probability of a successful double spending attack in several scenarios (both fast and slow payments) is analyzed in [16]. Finally, a fully implemented attack against Ethereum blockchain, which covers both a network and a double spending attack, is delivered in [7]. In contrast with these contributions, our results are achieved by analyzing a stochastic transition system, rather than constraining miners' behaviour to adhere to a certain statistical model.

2 The ledger datatype

A *ledger*, noted $\mathtt{L}$, $\mathtt{L}'$, $\cdots$, is a pair $(\mathtt{T},\mathtt{h})$ where $\mathtt{T}$ is a nonempty *tree of blocks* and h is the *handle*, e.g. a pointer to a leaf block at maximal depth. We note $\mathtt{T}$ with $\mathit{tree}(\mathtt{L})$ and $\mathtt{h}$ with $\mathit{handle}(\mathtt{L})$. The root of $\mathit{tree}(\mathtt{L})$ is called *genesis block*. Every block b in $\mathit{tree}(\mathtt{L})$ has a *pointer* to its parent that is addressed by $b.\mathtt{id}$; the set of blocks in $\mathtt{L}$ is addressed by $\mathtt{L.blocks}$. The following picture illustrates two ledgers – `L1` and `L2` where the handles are blue pointers.



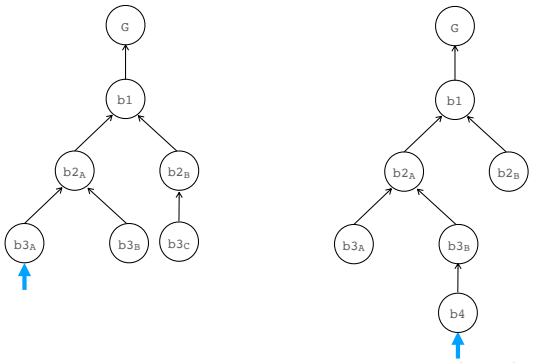
the ledger `L1` – `L1` $\uparrow= [b3_A, b2_A, b1, \mathtt{G}]$



the ledger `L2` – `L2` $\uparrow= [b3_C, b2_B, b1, \mathtt{G}]$

The *blockchain of* L, noted $\mathtt{L}\uparrow$, is the sequence $[b_0, b_1, b_2, \cdots]$ such that $b_0 = \mathit{handle}(\mathtt{L})$ and, for every i, b_{i+1} is the parent of b_i (therefore the last block of the sequence is the genesis block).

A key operation of ledger is $\mathit{addBlock}(\mathtt{L}, b)$ that returns a ledger where b is connected to the block pointed by $b.\mathtt{id}$. This operation may change the handle of the ledger. In particular, the handle of $\mathit{addBlock}(\mathtt{L}, b)$ is equal to the handle of L if the new block has not changed the maximal depth of the tree; it is a pointer to b if this block has a depth strictly greater than the maximal one of L. For example, considering the ledgers L1 and L2 in the foregoing picture, let $b3_C.\mathtt{id}$ be a pointer to $b2_B$ and $b4.\mathtt{id}$ be a pointer to $b3_B$. The ledgers $\mathit{addBlock}(\mathtt{L1}, b3_C)$ and $\mathit{addBlock}(\mathtt{L1}, b4)$ are



$\mathit{addBlock}(\mathtt{L1}, b3_C)$ $\mathit{addBlock}(\mathtt{L1}, b4)$

In particular, the handle of $\mathit{addBlock}(\mathtt{L1}, b3_c)$ is the same of L1, while L1 handle is different from the one of $\mathit{addBlock}(\mathtt{L1}, b4)$ because, in this case, the depth of the tree is changed.

3 The modelling language

Our modelling language is a stochastic pi calculus with lists and ledgers datatypes. We use three countable sets: *names*, ranged over by x, y, z, $\cdots$; *process names*, ranged over by A, B, $\cdots$; *variables* X, Y, Z, $\cdots$. As usual, we address tuples with $\overline{u}$. The stochastic pi calculus has three syntactic categories, *values* written u, *matches* written M, and *processes* written P. The grammar is detailed below.

$$
\begin{array}{lcll}
P & \stackrel{\text{def}}{=} & M \mid (\nu\ x@r)\ P \mid P \mid P \mid \mathsf{A}(\overline{e}) & \text{processes} \\
M & \stackrel{\text{def}}{=} & \mathsf{if}\ e\ \mathsf{then}\ M\ \mathsf{else}\ M \mid \Sigma & \text{conditional summands} \\
\Sigma & \stackrel{\text{def}}{=} & \mathtt{0} \mid \tau_r.P + \Sigma \mid x\mathtt{?}(\overline{X}).P + \Sigma \mid x\mathtt{!}\overline{e}.P + \Sigma & \text{summands} \\
e & \stackrel{\text{def}}{=} & x \mid X \mid b \mid \mathtt{1} \mid \mathtt{L} \mid \mathtt{op}(e, e) & \text{expressions}
\end{array}
$$

A process can be a conditional summand M, a restricted process of the form $(\nu\ x@r)\ P$ where x is a new (channel) name with rate r, with $r \in \mathbb{R}^+$, whose scope is P, a choice, a parallel composition, an if-then process, or a process name invocation $\mathsf{A}(\overline{e})$, in which case we ask for a unique equation $\mathsf{A}(\overline{X}) = P$ defining A. Additionally, in equations $\mathsf{A}(\overline{X}) = P$, we assume that recursive invocations of A are guarded by an input or output or an internal move.

A conditional summand M is a cascading nesting of if-then-else conditionals where basic elements are summands Σ. In turn, Σ can be a choice between processes that are either the inert process $\mathtt{0}$, a process $\tau_r.\ P$ performing an internal move at rate r and becoming P, an input $x\mathtt{?}(\overline{X}).P$, an output $x\mathtt{!}\overline{e}.P$. The number of summands in Σ is denoted by $|\Sigma|$.

Expressions are names, variables, (unspecified) blocks $b, \cdots$, lists $\mathtt{l}$ of blocks, ledgers $\mathtt{L}$, and operations on these elements, generically addressed by $\mathtt{op}(e, e')$. The empty list is denoted ε; a list containing the elements $b_1, \cdots, b_n$ is denoted $\mathtt{[}b_1, \cdots, b_n\mathtt{]}$; $\mathtt{l}^\frown b$ returns the list appending b to $\mathtt{l}$. We use the operation $\mathtt{l} = \varepsilon$, which is $\mathtt{true}$ if $\mathtt{l}$ is empty, $\mathtt{false}$ otherwise; the operations $\mathit{head}(\mathtt{l})$ and $\mathit{tail}(\mathtt{l})$ that, when $\mathtt{l}$ is nonempty, return the head and tail of $\mathtt{l}$, respectively. The ledger $\mathtt{G}$ represents the initial ledger containing the *genesis block* only (with an abuse of notation, the genesis block will be also addressed by $\mathtt{G}$).

Expressions are evaluated into *values*, which are names, blocks, lists and ledgers. We assume defined an evaluation function $[\![e]\!]$ that returns the value of an expression e that does not contain variables (it is undefined, otherwise).

Variables represent formal parameters of a process name definition or of an input operation and, sometimes, when the corresponding parameter is a name, we simply use a name rather than a variable. Restrictions bind names, that is $(\nu\ x@r)\ P$ binds the name x wherever it is free in P and likewise, input and agent definition bind variables, that is $x\mathtt{?}(\overline{X}).P$ and $\mathsf{A}(\overline{X}) = P$ bind the free occurrences of the variables $\overline{X}$ in P. Names and variables that are not bound are called *free* as usual and we write $\mathit{fn}(P)$ for the set of such names and variables in P.

▶ **Definition 1.** *The* equality*, noted* $\equiv$*, is the least equivalence on processes containing alpha-conversion of bound names, associativity of* $\mid$ *with identity* $\mathtt{0}$ *and containing*

$$\begin{array}{rcll} (\nu\ x@r)\ \mathtt{0} & \equiv & \mathtt{0} & \\ (\nu\ x@r)\ (\nu\ y@r')\ P & \equiv & (\nu\ y@r')\ (\nu\ x@r)\ P & \\ (\nu\ x@r)\ (P \mid Q) & \equiv & P \mid (\nu\ x@r)\ Q & \text{if } x \notin \mathit{fn}(P) \\ \mathsf{A}(\overline{u}) & \equiv & P\{\overline{u}/\overline{X}\} & \text{if } \mathsf{A}(\overline{X}) = P \end{array}$$

Because of the axioms of $\equiv$, we abbreviate $P_1 \mid \cdots \mid P_n$ into $\prod_{i\in 1..n} P_i$. A process P is in *canonical form* whenever P is equal to $(\nu\ \overline{x}@\overline{r})\ \prod_{i\in I} M_i$. It is easy to verify that, for every P, there is always a P' in canonical form such that $P \equiv P'$.

Table 1 collects the *intensional semantics* of the stochastic pi calculus. This semantics is described as a transition system on syntactic processes with transitions labelled by certain *terms*. According to Table 1, there are three types of transitions: (i) for conditional summands $M \overset{\mu,h}{\longmapsto} P$, where μ is either r or $x\mathtt{?}(\overline{Y})$ or $x\mathtt{!}\overline{u}$ and h represents the index of the addend in M that transits; (ii) $P \overset{\mu,\ell}{\longmapsto} P'$, where $\ell = k \cdot h$ indicates the index k of the conditional summand M in P that transits and the index h of the addend in M; (iii) $P \overset{\nu,\ell,\ell'}{\longmapsto} P'$, where ν is either r or x and ℓ, ℓ' are the indexes of the two subprocesses that move.

The intensional semantics of stochastic pi calculus performs an accurate estimation of positions of processes that actually move. In rules [TAU], [INP] and [OUT], the transition's label records both the move and the position of the process in the moving summand. In oder to determine k, in [PAR], we single out the parallel subprocess where the rightmost component moves. Then, according to [COM-L] and [COM-R], communications may happen between contiguous subprocesses only. In this case, the k-index of the subprocess to the right is updated according to the corresponding value of the process to the left. Once a communication has been singled-out, [PAR-P] admits liftings of transitions to contexts where the parallelism is to the right, consequently, the positions in the labels remain unchanged.

▶ **Definition 2.** *The* structural equivalence*, noted* $\equiv^+$*, is the least equivalence on processes containing equality* $\equiv$ *and containing commutativity and associativity of* $\mid$ *and* $+$ *with identity* $\mathtt{0}$.

$$\text{[TAU]}\ \frac{|\Sigma| = h}{\Sigma + \tau_r.P + \Sigma' \overset{r,h+1}{\longmapsto} P} \qquad \text{[INP]}\ \frac{|\Sigma| = h}{\Sigma + x?(\overline{Y}).P + \Sigma' \overset{x?(\overline{Y}),h+1}{\longmapsto} P} \qquad \text{[OUT]}\ \frac{[\![\overline{e}]\!] = \overline{u} \quad |\Sigma| = h}{\Sigma + x!\overline{e}.P + \Sigma' \overset{x!\overline{u},h+1}{\longmapsto} P}$$

$$\text{[IF-TRUE]}\ \frac{[\![e]\!] = \texttt{true} \quad M \overset{\mu,h}{\longmapsto} P}{\text{if } e \text{ then } M \text{ else } M' \overset{\mu,h}{\longmapsto} P} \qquad \text{[IF-FALSE]}\ \frac{[\![e]\!] = \texttt{false} \quad M' \overset{\mu,h}{\longmapsto} P}{\text{if } e \text{ then } M \text{ else } M' \overset{\mu,h}{\longmapsto} P}$$

$$\text{[PAR]}\ \frac{Q = \prod_{i\in 1..k-1} M_i \quad M \overset{\mu,h}{\longmapsto} P}{Q \mid M \overset{\mu,k\cdot h}{\longmapsto} Q \mid P} \qquad \text{[PAR-P]}\ \frac{P \overset{\nu,\ell,\ell'}{\longmapsto} P'}{P \mid Q \overset{\nu,\ell,\ell'}{\longmapsto} P' \mid Q}$$

$$\text{[COM-L]}\ \frac{P \overset{x!\overline{u},k\cdot h}{\longmapsto} P' \quad Q \overset{x?(\overline{Y}),k'\cdot h'}{\longmapsto} Q'}{P \mid Q \overset{x,k\cdot h,\ k+k'\cdot h'}{\longmapsto} P' \mid Q'\{\overline{u}/\overline{Y}\}} \qquad \text{[COM-R]}\ \frac{P \overset{x?(\overline{Y}),k\cdot h}{\longmapsto} P' \quad Q \overset{x!\overline{u},k'\cdot h'}{\longmapsto} Q'}{P \mid Q \overset{x,k\cdot h,k+k'\cdot h'}{\longmapsto} P'\{\overline{u}/\overline{Y}\} \mid Q'}$$

$$\text{[NEW]}\ \frac{P \overset{x,\ell,\ell'}{\longmapsto} Q}{(\nu\ x@r, \overline{z}@\overline{r'})\ P \overset{r,\ell,\ell'}{\longmapsto} (\nu\ x@r, \overline{z}@\overline{r'})\ Q} \qquad \text{[NEW-R]}\ \frac{P \overset{r,\ell,\ell'}{\longmapsto} Q}{(\nu\ \overline{z}@\overline{r'})\ P \overset{r,\ell,\ell'}{\longmapsto} (\nu\ \overline{z}@\overline{r'})\ Q}$$

Table 1 The intensional semantics of the stochastic pi calculus.

The following notations are relevant for the definition of the stochastic transition relation:

- $\texttt{next}(P) = \{((r,\ell,\ell'), Q) \mid P \overset{r,\ell,\ell'}{\longmapsto} Q\}$;
- let $\mathcal{P}$ be a set of pairs $((r,\ell,\ell'), Q)$, $[\mathcal{P}]_Q$ is the subset of $\mathcal{P}$ of those pairs $((r',\ell'',\ell'''), Q')$ such that $Q' \equiv^+ Q$;
- $\texttt{can}(\mathcal{P})$ is defined over sets of pairs $((r,\ell,\ell'), Q)$ such that the processes occurring as second element of the pairs are all structurally equivalent ($\equiv^+$). It returns a solution Q' such that there is an (r,ℓ,ℓ') with $((r,\ell,\ell'), Q) \in \mathcal{P}$ and Q is in canonical form.

▶ **Definition 3** (Stochastic transition relation)**.** *The pi calculus stochastic transition relation* $\overset{\lambda}{\longrightarrow}$*, where* $\lambda \in \mathbb{R}^+$*, is the least relation satisfying the following rule:*

- *if* $P \overset{r,\ell,\ell'}{\longmapsto} Q$ *then* $P \overset{\lambda}{\longrightarrow} \texttt{can}([\texttt{next}(P)]_Q)$*, where* $\lambda = \sum_{((r,\ell,\ell'),Q') \in [\texttt{next}(P)]_Q} r$ *.*

The stochastic transition relation of pi calculus corresponds to a *Continuous Time Markov Chain (CTMC) transition system* with only *silent interactive transitions* [14, 15]. In a markovian state with n outgoing markovian transitions labeled $\lambda_1, \cdots, \lambda_n$, the probability that the sojourn time is less than t is exponentially distributed with rate $\sum_i \lambda_i$, i.e. $\mathsf{Prob}(\mathit{delay} < t) = 1 - e^{-t\sum_i \lambda_i}$, and the probability that the j-th transition is taken is $\lambda_j / (\sum_i \lambda_i)$. Since CTMC are the standard models underlying traditional simulation algorithms [17, 11, 22], we may also use automatic analysis tools for experimenting *in silico* the dynamics of our specifications. (Well, these tools cannot be used in their current version: an extension with ledger values is necessary beforehand.)

4 The abstract modelling of Blockchain and its analysis

Blockchain realises a distributed ledger on a peer-to-peer network. The key participants of the protocols are the *miners* that create blocks of the ledger and broadcast them to the nodes of the network. Our modelling of the blockchain system details miners' behaviours as a stochastic pi calculus process. More precisely, a blockchain system is a parallel composition of n miners that communicate through channels $z_1, \cdots, z_n$ with rates $r_1, \cdots, r_n$, respectively,

$$(\nu z_1@r_1, \cdots, z_n@r_n)\Big(\prod_{z_i \in \{z_1,\cdots,z_n\}} \mathsf{Miner}_{\{z_1,\cdots,z_n\}\setminus z_i}(\mathtt{G}, \varnothing, z_i) \Big)$$

where $\mathtt{G}$ is the ledger with the genesis block only. We are assuming the presence of a finite number of process name definitions Miner_U – the miners –, where U is a finite set of channels [1]. Their definition is

$$\begin{array}{l}
\mathsf{Miner}_U(L, X, z) \;= (\nu\, w@r_w)\big(\\
\qquad \big(z?(b).\, \mathsf{Miner}_U(L, X^\frown b, z) \\
\qquad + w!\, newBlock(L) \\
\qquad + \mathsf{if}\ (X = \varepsilon)\ \mathsf{then}\ \tau_{r'}.\mathsf{Miner}_U(L, X, z) \\
\qquad\quad \mathsf{else\ if}\ (head(X).\mathtt{id} \in L.\mathtt{blocks})\ \mathsf{then} \\
\qquad\qquad\qquad \tau_{r'}.\mathsf{Miner}_U(addBlock(L, head(X)), tail(X), z) \\
\qquad\quad \mathsf{else}\ \tau_{r'}.\ \mathsf{Miner}_U(L, tail(X)^\frown head(X), z) \\
\qquad \big) \mid\ w?(b).(\mathsf{Miner}_U(addBlock(L, b), X, z) \mid \textstyle\prod_{z' \in U} z'!(b)) \\
\big)
\end{array}$$

Miners retain a local copy of the ledger – the argument L – and a set X of blocks that have been received from the network through z and that have not been inserted in L. This set is implemented as a list in our modelling. A miner behaves as follows:

- it may receive a block from the network – operation $z?(b)$. The block is stored in X because it is possible that b cannot be inserted in L since its parent block is not already in the ledger.
- it may create (e.g. mine) a new block – operation $w!\, newBlock(L)$. In our setting, mining a block amounts to transmitting it on a channel with a given rate – the channel w. This rate indicates the nodes' rate of generating new blocks. Therefore, it corresponds to the computational power of miners to solve the cryptopuzzles of the proof-of-work. [In this way we abstract away from the proof-of-work technique for mining blocks.] When a block is created by a miner, it is added to the local ledger and it is broadcasted to all the other miners of the network – the parallel subprocess starting with $w?(b)$.

[1] This formalization does not fully comply with the language defined in Section 3 because Miner is actually parametric with respect to sets of names. A more appropriate formalization would have been (1) to admit systems with a global finite set $\mathtt{Ch}$ of constant (channel) names and rates and (2) to extend Table 1 with the additional rule

$$[\textsc{glob}]\quad \frac{P \overset{x,\ell,\ell'}{\longmapsto} Q \quad (x,r) \in \mathtt{Ch}}{P \overset{r,\ell,\ell'}{\longmapsto} Q}$$

The process name Miner would then be indexed by a set of constant names. However, we have preferred the current presentation, even if not perfectly proper, because it is simpler and it avoids to deviate from the standard definition of stochastic pi calculus. What matters is that our results are in no way dependent on the presentation of the blockchain system.

- it may take a block that is stored in the local bag X. Since X is a list, the node extracts the first block of the list – $head(X)$ –, if any. There are two cases: either the block can be added (the parent is already in the local ledger) or not. In this last case the block is re-inserted in the bag in the tail position – operation $tail(X)^\frown head(X)$. This behaviour is modelled by the conditional subprocess.

It is worth to notice that, when a block b is mined locally, its pointer is the handle. In this case $addBlock(L, b)$ returns a ledger where b is at maximal depth and the handle is a pointer to b. On the contrary, when a block b is received from the network, $b.\mathtt{id}$ may be different from the handle and $addBlock(L, b)$ connects b to the block pointed by $b.\mathtt{id}$. In this case, the handle of $addBlock(L, b)$ is equal to the one of L if the new block has not changed the maximal depth of the ledger; it is a pointer to b if this block is at a strictly greater depth than the maximal one of L. These two cases are discussed and illustrated in Section 2.

Properties

In the remaining part of the section we compute the probability of a blockchain system to devolve into inconsistent states, e.g. into a state where at least two nodes have different ledgers. In order to ease our arguments, among the possible states of the stochastic transition system, we select those where the broadcast messages have all been delivered. This scenario is usual in blockchain because the rate of block delivery is much higher than the one of mining. For example, in Bitcoin, the nodes that have not yet received the last block after 40 seconds are less than 5%, whilst blocks are mined every 10 minutes [6].

▶ **Definition 4.** *A state of a blockchain system is called* completed *when it is structurally equivalent* $(\nu\ z_1@r_1, \cdots, z_n@r_n)\Big(\ \prod_{i\in 1..n} \mathsf{Miner}(\mathtt{L}_i, \varepsilon, z_i)\ \Big)$. *Namely, in a completed state, there is no block to deliver and the blocks in the local lists X_i have been already inserted in the corresponding ledgers.*

▶ **Proposition 5.** *Let P be a completed state of a blockchain system and let* L *and* L' *be two ledgers in different nodes. Then $tree(\mathtt{L}) = tree(\mathtt{L}')$. Therefore, if $\mathtt{L} \neq \mathtt{L}'$ then $handle(\mathtt{L}) \neq handle(\mathtt{L}')$.*

▶ **Definition 6.** *Let* L *and* L′ *be two ledgers and let*

- *m be the length of* L ↑,
- *n be the length of* L′ ↑,
- *h be the length of the maximal common suffix of* L ↑ *and* L′ ↑.

We say that L *and* L′ have a fork of length k, *where* $k = \max(m - h, n - h)$.

In the foregoing definition of miner, the channels $z_1, \cdots, z_n$ broadcast blocks with rates $r_1, \cdots, r_n$, respectively. These rates are actually parameters of an exponential distribution [6]. In the following theorems, for sake of simplicity, we identify all these parameters by taking the parameter of the exponential distribution mean, which we call r.

▶ **Theorem 7.** *Let P be a completed state of a blockchain system consisting of n miners with ledgers $\mathtt{L}_1, \ldots, \mathtt{L}_n$, respectively, such that $\mathtt{L}_1 = \cdots = \mathtt{L}_k$ and $\mathtt{L}_{k+1} = \cdots = \mathtt{L}_n$ and $\mathtt{L}_1 \neq \mathtt{L}_{k+1}$. Let $\mathtt{L}_1$ and $\mathtt{L}_{k+1}$ have fork of length m. Then the probability $\mathsf{Prob}(P_{\rightsquigarrow m+1})$ to reach a completed state with fork of length $m+1$ is smaller than ($R = \sum_{j=1}^{n} r_{w_j}$ and we assume that, for every i, j, $r = r_i = r_j$)*

$$\sum_{\substack{1 \le i \le n \\ H \subset \{1,\cdots,n\} \setminus i \\ i \le k \ \Rightarrow\ j \in \{k+1,\cdots,n\} \setminus H \\ i > k \ \Rightarrow\ j \in \{1,\cdots,k\} \setminus H}} \Theta(i, |H|, j)$$

$$\textit{where}\quad \Theta(i,\ell,j) = \frac{r_{w_i}\ r_{w_j}}{R\ (R+(n-1-\ell)r)}\ \prod_{1\leq h\leq \ell}\frac{h\ r}{R+(n-h)r}\ \prod_{1\leq a\leq 2n-2-\ell}\frac{a\ r}{R+a\ r}\ .$$

It is worth to notice that the probability $\mathsf{Prob}(P_{\rightsquigarrow m+1})$ of Theorem 7 depends on the number of nodes, their hashing power and the latency of the network. The proof has required a time-consuming analysis of the stochastic transition system due to the state explosion with respect to the number of nodes. Using a technique similar to Theorem 7 we may compute the probability that a blockchain system in a completed consistent state (the nodes have the same ledger) devolves into an inconsistent state. In this case, the proof is simpler than Theorem 7 because every node may mine after the first one.

▶ **Proposition 8.** *Let P be a completed state of a blockchain system consisting of n miners having ledger* `L`*. The probability $\mathsf{Prob}(P_{\rightsquigarrow 1})$ to reach a completed state with fork of length 1 is smaller than ($R = \sum_{j=1}^{n} r_{w_j}$ and we assume that, for every i, j, $r = r_i = r_j$)*

$$\sum_{\substack{1\ \leq\ i\ \leq\ n\\ H\ \subset\ \{1,\cdots,n\}\ \setminus\ i\\ j\ \in\ \{1,\cdots,n\}\ \setminus\ H}} \Theta(i,|H|,j)$$

$$\textit{where}\quad \Theta(i,\ell,j) = \frac{r_{w_i}\ r_{w_j}}{R\ (R+(n-1-\ell)r)}\ \prod_{1\leq h\leq \ell}\frac{h\ r}{R+(n-h)r}\ \prod_{1\leq a\leq 2n-2-\ell}\frac{a\ r}{R+a\ r}\ .$$

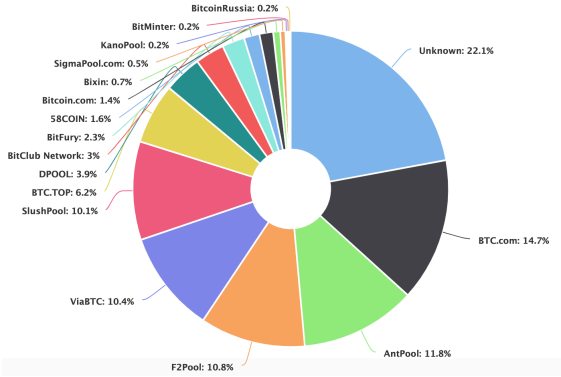


Figure 1 Hashrate distribution of Bitcoin mining pools on January 2019.
Source: https://www.blockchain.com/.

In order to bear some numerical results, we instantiate our probability with realistic channel rates. In [19], the time a miner takes to create a block is exponential with parameter θ, which represents the probability that the miner solves the cryptopuzzle problem in a given time-slot [1]. It follows that $\theta = h/D$, where h is miner's hashing power and D is the cryptopuzzle difficulty set by the protocol in order to set constant to 10 minutes the average duration between two blocks. In our encoding, θ is represented by r_{w_i}, therefore $r_{w_i} = h_i/D$ and, taking the current hashing power distribution of the Bitcoin system illustrated in Figure 1, and letting $D = 600$, we obtain $r_{w_{\texttt{BTC.com}}} = 0.000245$, $r_{w_{\texttt{ANTpool}}} = 0.000196$, $r_{w_{\texttt{F2pool}}} = 0.00018$, etc. As regards the broadcast of messages, in the blockchain protocol, it is a combination of transmission time and the local verification of the block. From [6] we know that in a Bitcoin environment, the broadcast can be approximated as an exponential distribution with mean time 12.6 seconds. Therefore we may assume that every r_i is 1/12.6.

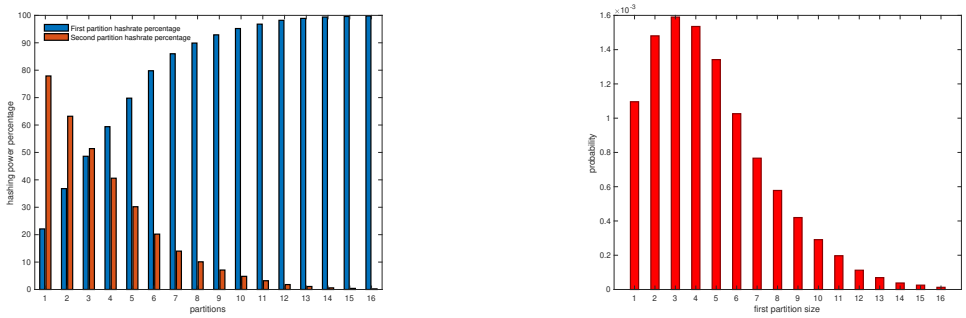
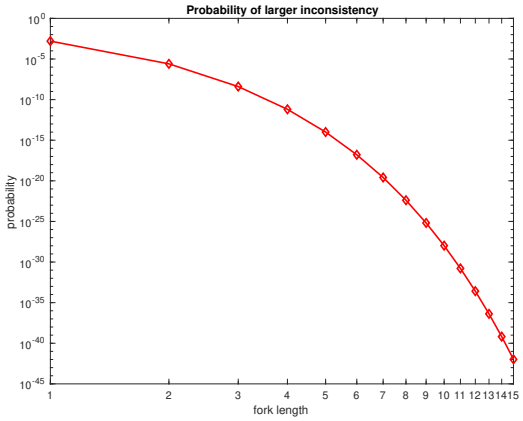


Figure 2 Probability trend with the first partition composed by the pools with largest hashrate percentage.

In Figure 2 we illustrate our results. The probability that ledgers turn into a state with longer forks. In the left picture there are 16 possible initial partitions (the sum of blue and red columns is always 100); in the right picture there are the corresponding 16 probabilities to increase the inconsistency computed according to our formula. The reader can observe that the highest probability to increase inconsistency corresponds to the case where the two partitions have equivalent hash rates.

In the following figure we highlight the probability decay of creating larger and larger



inconsistencies in the case where the two initial partition have equivalent hash rates (which is the case where the probability is higher). For example, it is around 10^{-30} for forks of length 10.

▶ Remark 9. Our analysis addresses the case of a fixed set of miners where communications always succeed. In particular, it does not cover those (blockchain) systems where nodes may either leave or join the network (dynamic networks) or networks where nodes may fail or broadcasts manifest delay or loss of information during the communication. We remark that our technique can be also used for analyzing these general situations because they may be modelled in stochastic pi calculus (actually, this calculus has been used because we had this extension in mind). The analysis of these cases is left to future work because computing the probabilities by hand is extremely time-consuming (since the processes become more complex). In this respect, using an automatic tool for the stochastic pi calculus with ledger datatype will save us a lot of time. [The extension of an analyzer, such as [17, 11, 22], with this feature is under scrutiny.]

5 Analysis of a possible attack

In this section we analyze the behaviour of the Blockchain system in presence of hostile miners. The attack we model is the one described in [19], namely a hostile miner tries to create an alternate chain faster than the honest one. This scenario admits that a merchant can be convinced that a transaction has been accepted and then create a new branch of the chain, longer than the valid one, with some other transaction spending the same money (double spending attack). Technically, the difference with Miner_U is that the dishonest miner, called $\mathsf{Miner}^{\mathsf{D}}{}_U$, mines on a block d that is not the correct one (e.g. the first block added at maximal depth). We use the operation $newBlock^{\mathsf{D}}(\mathtt{L}, d)$ that takes a ledger $\mathtt{L}$ and a block $d \in \mathtt{L.blocks}$ and returns a new block whose pointer is d (therefore it will be connected to d). The definition of $\mathsf{Miner}^{\mathsf{D}}{}_U$ is

$$
\begin{array}{l}
\mathsf{Miner}^{\mathsf{D}}{}_U(L, X, z, d) \;=\\
\quad (\nu\; w@r)\big(\quad \big(\; z?(b).\; \mathsf{Miner}^{\mathsf{D}}{}_U(L, X^\frown b, z, d)\\
\qquad\qquad + \; w!\, newBlock^{\mathsf{D}}(L, d)\\
\qquad\qquad + \; \mathsf{if}\ (X = \varepsilon)\ \mathsf{then}\ \tau_{r'}.\ \mathsf{Miner}^{\mathsf{D}}{}_U(L, X, z, d)\\
\qquad\qquad\quad \mathsf{else\ if}\ (head(X).\mathtt{id} \in L.\mathtt{blocks})\ \mathsf{then}\\
\qquad\qquad\qquad\qquad\qquad \tau_{r'}.\mathsf{Miner}^{\mathsf{D}}{}_U(addBlock(L, head(X)), tail(X), z, d)\\
\qquad\qquad\quad \mathsf{else}\ \tau_{r'}.\ \mathsf{Miner}^{\mathsf{D}}{}_U(L, tail(X)^\frown head(X), z, d)\\
\qquad\qquad \big)\ \mid\ w?(b).(\mathsf{Miner}^{\mathsf{D}}{}_U(addBlock(L, b), X, z, b)\ \mid \prod_{z' \in U} z'!(b))\\
\quad \big)
\end{array}
$$

The hostile miner has an additional argument with respect to honest ones, the block d, which is the block on which he wants to mine. Following the same pattern of Section 4

▶ **Theorem 10.** *Let P be a completed state of a blockchain system of n miners with exactly one that is hostile and let r_{w_d} its mining rate. The probability $\mathsf{Prob}(P_m)$ to reach a completed state where the hostile miner has created an alternate chain longer than the honest one from $m, m \geq 1$, blocks behind is smaller than ($R = \sum_{j=1}^{n} r_{w_j}$ and we assume that, for every i, j, $r = r_i = r_j$)*

$$\sum_{k \geq 1} \Big[\Phi(r_{w_d}, r, R)^k \Big(\sum_{1 \leq j \leq n-1} \Phi(r_{w_j}, r, R) \Big)^{k-1} \Big]^m$$

where $\Phi(r_w, r, R) = \dfrac{r_w}{R} \prod_{1 \leq a \leq n-1} \dfrac{a\ r}{R + (n-a)r}$.

As for Theorem 7, the technique used for demonstrating the above statement consists of analyzing the stochastic transition system. The technique is therefore different from the one in [19], where it is assumed *a priori* that the ratio between blocks mined by the attacker and those mined by the honest miners is the expected value of a Poisson distribution. In fact, this distribution expresses the probability of a given number of events occurring in a fixed interval of time with a known constant rate and independently of the time since the last event. Thus, Nakamoto models the attack counting the number of successes, i.e. the number of blocks caught up from the attacker, in a series of intervals measured in times asusming that the probability for success does not change during the experiment.

Let us discuss the probability trend of a successful attack in Bitcoin. We consider the current main pools of Bitcoin miners (see Figure 1) and assume that one of them decides to become hostile. Our results, for BTC.com, AntPool, and F2Pool are reported in Table 2 (these pools have a decreasing hashrate). We derive that the probability a hostile miner

Pool	$m = 1$	$m = 5$	$m = 10$	Approximation
BTC.com	0.168	0.00013	0.000000017	6^{-m}
AntPool	0.131	0.000039	0.0000000015	7^{-m}
F2Pool	0.119	0.000024	0.00000000059	8^{-m}

Table 2 Probability of a successful attack in the actual Bitcoin setting.

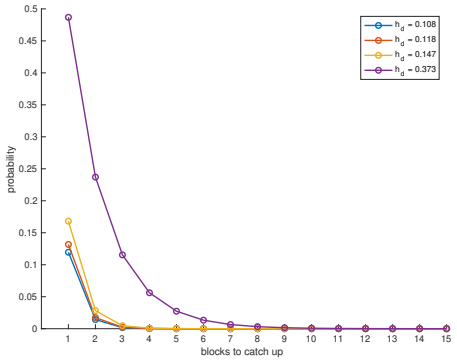


Figure 3 Probability depending on the hashing power of n_d.

catches up from 1 block behind increases with the percentage of the hashing power and drops exponentially with the number of blocks to catch up. These results are also graphically illustrated in Figure 3, with the additional hypothetical case (the purple line) that the three largest pools decide to join and become hostile. This coalition would possess more than the 30% of the total hashing power and it is clear that its probability to create an alternate chain is very high.

6 Conclusions

We have studied the probability that the blockchain protocol may devolve the ledger into inconsistent copies because of forks. Two cases have been analyzed: when the system consists of honest miners and when the system has one hostile miner that mines blocks in wrong positions. These results are gathered by means of an original modelling of the blockchain system by means of a stochastic process calculus – the stochastic pi calculus.

Our results can be applied to analyze other well-known attacks to the blockchain protocol, such as failures either of communications or of miners, the inception of new miners that may be hostile (actually, pi calculus has been used because we had this extension in mind), etc. In this paper we restricted to the case of static sets of nodes because computing the probabilities by hand is extremely time-consuming. The analysis of the above cases is left to future work.

As a matter of fact, stochastic calculi come with several automatic analysis techniques for experimenting *in silico* the dynamics of specifications, such as stochastic timed logics, stochastic temporal logics, and stochastic model checking [17, 11, 22]. However, it has not been possible to reuse these tools right-away because, in their current version, they miss ledger values. The extension of a stochastic analyzer with the ledger datatype is also left to future research.

References

1. Bruno Biais, Christophe Bisiere, Matthieu Bouvard, and Catherine Casamatta. The blockchain folk theorem, 2018.
2. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *In Proceedings of SP 2015*, pages 104–121. IEEE Computer Society, 2015.
3. R. Bowden, H. Paul Keeler, Anthony E. Krzesinski, and Peter G. Taylor. Block arrivals in the bitcoin blockchain. *CoRR*, abs/1801.07447, 2018.
4. Luca Cardelli and Radu Mardare. Stochastic pi-calculus revisited. In *Theoretical Aspects of Computing – ICTAC 2013*, pages 1–21. Springer, 2013.
5. Miles Carlsten, Harry Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 154–167, New York, NY, USA, 2016. ACM. URL: `http://doi.acm.org/10.1145/2976749.2978408`, `doi:10.1145/2976749.2978408`.
6. C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sep. 2013. `doi:10.1109/P2P.2013.6688704`.
7. Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. Double-spending risk quantification in private, consortium and public ethereum blockchains. *CoRR*, abs/1805.05004, 2018. URL: `http://arxiv.org/abs/1805.05004`, `arXiv:1805.05004`.
8. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, June 2018. URL: `http://doi.acm.org/10.1145/3212998`, `doi:10.1145/3212998`.
9. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
10. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Proceedings of EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.
11. Stephen Gilmore and Jane Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In *in Proc. 7th Computer Performance Evaluation, Modeling Techniques and Tools*, volume 794 of *Lecture Notes in Computer Science*, pages 353–368. Springer, 1994.
12. Johannes Göbel, Holger Paul Keeler, Anthony E. Krzesinski, and Peter G. Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Perform. Eval.*, 104:23–41, 2016.
13. Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, Jan 1991. URL: `https://doi.org/10.1007/BF00196791`, `doi:10.1007/BF00196791`.
14. Holger Hermanns. *Interactive Markov Chains: And the Quest for Quantified Quality*. Springer-Verlag, Berlin, Heidelberg, 2002.
15. Jane Hillston. *A Compositional Approach to Performance Modelling (Distinguished Dissertations in Computer Science)*. Cambridge University Press, New York, NY, USA, 2005.
16. Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proc. of CCS'12*, pages 906–917. ACM, 2012.
17. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *In Proc. TACAS 2002*, volume 2280 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2002.
18. Andrew Miller and Joseph J LaViola Jr. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. *Available on line: http://nakamotoinstitute.org/research/anonymous-byzantine-consensus*, 2014.
19. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: `http://www.bitcoin.org/bitcoin.pdf`.

20 A. Pinar Ozisik and Brian Neil Levine. An explanation of nakamoto's analysis of double-spend attacks. *CoRR*, abs/1701.03977, 2017. URL: `http://arxiv.org/abs/1701.03977`.

21 Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Proceedings of EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 643–673. Springer, 2017.

22 Andrew Phillips and Luca Cardelli. Efficient, correct simulation of biological processes in the stochastic pi-calculus. In *In Proc. CMSB 2007*, volume 4695 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 2007.

23 George Pîrlea and Ilya Sergey. Mechanising blockchain consensus. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 78–90. ACM, 2018.

24 Corrado Priami. Stochastic Pi-Calculus. *The Computer Journal*, 38(7):578–589, 1995.

25 Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Proc. of Financial Cryptography and Data Security 2015*, volume 8975 of *Lecture Notes in Computer Science*, pages 507–527. Springer, 2015.

26 Alexei Zamyatin, Nicholas Stifter, Philipp Schindler, Edgar R. Weippl, and William J. Knottenbelt. Flux: Revisiting near blocks for proof-of-work blockchains. *IACR Cryptology ePrint Archive*, 2018:415, 2018.

A Technical details

This appendix collects the demonstration of the statements in the paper. For readability sake we also report the statements.

Theorem 7. *Let P be a completed state of a blockchain system consisting of n miners with ledgers $\mathtt{L}_1,\ldots,\mathtt{L}_n$, respectively, such that $\mathtt{L}_1 = \cdots = \mathtt{L}_k$ and $\mathtt{L}_{k+1} = \cdots = \mathtt{L}_n$ and $\mathtt{L}_1 \neq \mathtt{L}_{k+1}$. Let $\mathtt{L}_1$ and $\mathtt{L}_{k+1}$ have fork of length m. Then the probability $\mathsf{Prob}(P_{\rightsquigarrow m+1})$ to reach a completed state with fork of length $m+1$ is smaller than ($R = \sum_{j=1}^{n} r_{w_j}$ and we assume that, for every i,j, $r = r_i = r_j$)*

$$\sum_{\substack{1 \leq i \leq n \\ H \subset \{1,\cdots,n\} \setminus i \\ i \leq k \;\Rightarrow\; j \in \{k+1,\cdots,n\} \setminus H \\ i > k \;\Rightarrow\; j \in \{1,\cdots,k\} \setminus H}} \Theta(i,|H|,j) \tag{1}$$

where $\Theta(i,\ell,j) = \dfrac{r_{w_i}\ r_{w_j}}{R\ (R+(n-1-\ell)r)}\ \prod_{1\leq h\leq \ell} \dfrac{h\ r}{R+(n-h)r} \prod_{1\leq a\leq 2n-2-\ell} \dfrac{a\ r}{R+a\ r}$.

Proof. The proof is split in two parts. In the first part we demonstrate that the above formula is smaller than 1; in the second part we demonstrate that it is an upper bound for $\mathsf{Prob}(P_{\rightsquigarrow m+1})$.

Part 1. the formula (1) is smaller than 1. The formula (1) can be rewritten

$$\sum_{\substack{1 \leq i \leq k \\ H \subset \{1,\cdots,n\} \setminus i \\ j \in \{k+1,\cdots,n\} \setminus H}} \Theta(i,|H|,j) + \sum_{\substack{k+1 \leq j \leq n \\ H \subset \{1,\cdots,n\} \setminus j \\ i \in \{1,\cdots,k\} \setminus H}} \Theta(j,|H|,i) \tag{2}$$

We analyze the first addend and we demonstrate that it is smaller than $1/2$. Since H is every possible subset of $n-2$ indices, then, letting $\ell = |H|$, there are $\binom{n-2}{\ell}$ different possible sets H. Picking $j \notin H$, we can rewrite the first addend of (2) as:

$$\sum_{1\leq i\leq k} \frac{r_{w_i}}{R} \sum_{0\leq \ell\leq n-2} \binom{n-2}{\ell} \prod_{1\leq h\leq \ell} \frac{h\ r}{R+(n-h)r} \sum_{k+1\leq j\leq n,\, j\notin H} \frac{r_{w_j}}{R+(n-1-\ell)r} \prod_{1\leq a\leq 2n-2-\ell} \frac{a\ r}{R+a\ r} \tag{3}$$

First observe that $\frac{a\ r}{R+a\ r} \leq 1$; therefore $\prod_{1\leq a\leq 2n-2-\ell} \frac{a\ r}{R+a\ r} \leq 1$, as well. Henceforth (3) can be over-approximated as

$$\sum_{1\leq i\leq k} \frac{r_{w_i}}{R} \sum_{0\leq \ell\leq n-2} \binom{n-2}{\ell} \prod_{1\leq h\leq \ell} \frac{h\ r}{R+(n-h)r} \sum_{k+1\leq j\leq n,\, j\notin H} \frac{r_{w_j}}{R+(n-1-\ell)r} \tag{4}$$

Moreover, since $\frac{1}{R+(n-h)r} \leq \frac{1}{(n-h)r}$, then

$$\prod_{1\leq h\leq \ell} \frac{h\ r}{R+(n-h)r} \;\leq\; \prod_{1\leq h\leq \ell} \frac{h\ r}{(n-h)r} \;\leq\; \frac{\ell!\ r^{\ell}}{(n-1)\ldots(n-\ell)r^{\ell}}\ .$$

Thus, we obtain that (4) is less than or equal to:

$$\sum_{1\le i\le k}\frac{r_{w_i}}{R}\sum_{0\le \ell\le n-2}\binom{n-2}{\ell}\frac{\ell!}{(n-1)\dots(n-\ell)}\sum_{k+1\le j\le n,\, j\notin H}\frac{r_{w_j}}{R+(n-1-\ell)r}$$

$$\le \sum_{1\le i\le k}\frac{r_{w_i}}{R}\sum_{0\le \ell\le n-2}\frac{(n-2)!}{\ell!(n-2-l)!}\frac{\ell!}{(n-1)\dots(n-\ell)}\sum_{k+1\le j\le n,\, j\notin H}\frac{r_{w_j}}{R+(n-1-\ell)r}$$

$$\le \sum_{1\le i\le k}\frac{r_{w_i}}{R}\sum_{0\le \ell\le n-2}\frac{(n-2)\dots(n-1-\ell)}{(n-1)\dots(n-\ell)}\sum_{k+1\le j\le n,\, j\notin H}\frac{r_{w_j}}{R+(n-1-\ell)r}$$

$$\le \sum_{1\le i\le k}\frac{r_{w_i}}{R}\sum_{0\le \ell\le n-2}\frac{(n-2)\dots(n-1-\ell)}{(n-1)}\sum_{k+1\le j\le n,\, j\notin H}\frac{r_{w_j}}{R+(n-1-\ell)r} \qquad (5)$$

The sum of r_{w_j} terms with $j \notin H$ can be over-approximated by the sum of all the terms of the second partition, i.e.

$$\sum_{k+1\le j\le n,\, j\notin H} r_{w_j} \le \sum_{k+1\le j\le n} r_{w_j}\ .$$

As a consequence, (5) can be over-approximated as follows:

$$\sum_{1\le i\le k}\frac{r_{w_i}}{R}\sum_{0\le \ell\le n-2}\frac{n-1-\ell}{(n-1)(R+(n-1-\ell)r)}\sum_{k+1\le j\le n} r_{w_j}$$

$$\le \frac{1}{R}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j}\sum_{0\le \ell\le n-2}\frac{n-1-\ell}{(n-1)(R+(n-1-\ell)r)}$$

$$\le \frac{1}{R}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j}\sum_{0\le \ell\le n-2}\frac{n-1-\ell}{(n-1)(R+(n-1-\ell)r)}$$

$$\le \frac{1}{R}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j}\sum_{0\le \ell\le n-2}\frac{n-1-\ell}{(n-1)((n-1-\ell)r)}$$

$$\le \frac{1}{R\,r}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j}\sum_{0\le \ell\le n-2}\frac{n-1-\ell}{(n-1)(n-1-\ell)}$$

$$\le \frac{1}{R\,r}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j}\sum_{0\le \ell\le n-2}\frac{1}{n-1}$$

$$\le \frac{1}{R\,r}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j}$$

Since

$$\sum_{0\le \ell\le n-2}\frac{1}{n-1} \le \frac{n-1}{n-1} = 1$$

and taking $R \le r \le \frac{1}{2}$ it follows that

$$\frac{1}{R\,r}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j} \le \frac{R}{r} \le \frac{1}{2}\ .$$

Thus, for every $n > 2$:

$$\sum_{\substack{1\,\le\, i\,\le\, k\\ H\,\subset\,\{1,\cdots,n\}\setminus i\\ j\,\in\,\{k+1,\cdots,n\}\setminus H}} \Theta(i,|H|,j) \le \frac{1}{R\,r}\sum_{1\le i\le k} r_{w_i}\sum_{k+1\le j\le n} r_{w_j} \le \frac{1}{2}$$

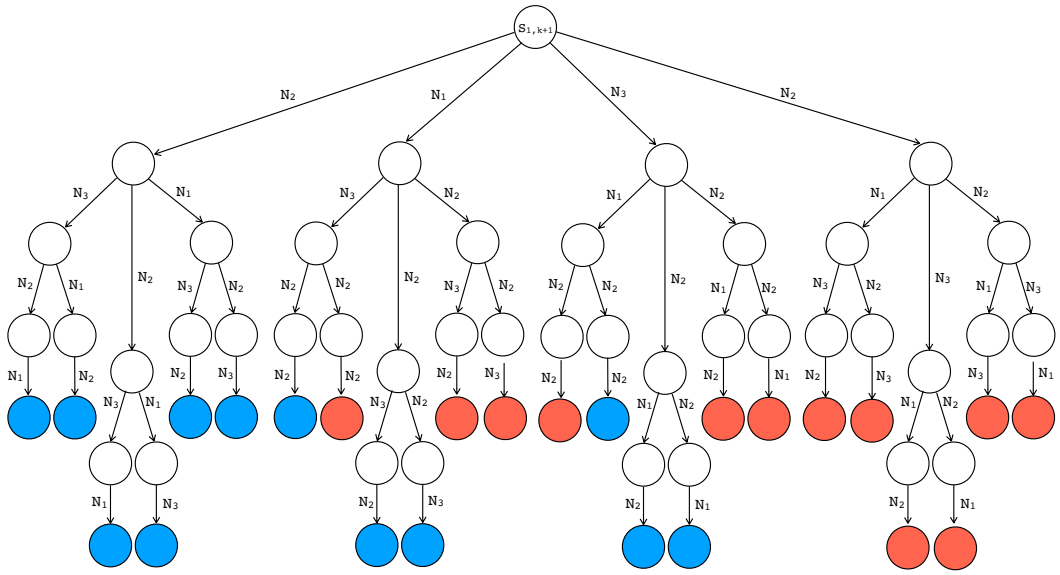


Figure 4 The computations to a completed state from $s_{1,3}$ of a systems consisting of miners $\mathtt{N}_1, \mathtt{N}_2, \mathtt{N}_3$ where $\mathtt{N}_1$ and $\mathtt{N}_2$ have the same ledger and $\mathtt{N}_3$ a different one. The final states with red bullet represent a completed state where $\mathtt{N}_1, \mathtt{N}_2$ have a same ledger and $\mathtt{N}_3$ a different one; the blue bullets those states where $\mathtt{N}_2, \mathtt{N}_3$ have the same ledger and $\mathtt{N}_1$ a different one. Transitions are labelled with the miners that move.

Therefore, it follows that:

$$\sum_{\substack{1 \leq i \leq n \\ H \subset \{1,\cdots,n\} \setminus i \\ i \leq k \;\Rightarrow\; j \in \{k+1,\cdots,n\} \setminus H \\ i > k \;\Rightarrow\; j \in \{1,\cdots,k\} \setminus H}} \Theta(i,|H|,j) \;\leq\; \frac{2}{R\; r} \sum_{1 \leq i \leq k} r_{w_i} \sum_{k+1 \leq j \leq n} r_{w_j} \leq 1 \tag{6}$$

Part 2. the formula (1) **is** $\mathsf{Prob}(P_{\rightsquigarrow m+1})$**.** The probabilities is computed by analyzing the states of the transition system in detail. To illustrate the technique we first illustrate the simple case that nodes 1 and $k+1$ mine before any broadcast. So assume to be in a state $s_{1,k+1}$ where such minings have already occurred. From $s_{1,k+1}$ it is possible to reach a completed state of fork $m+1$ with computations of length $2n-2$ (because there are two blocks to be delivered to $n-1$ nodes). The probability that one of these computations is chosen is

$$\prod_{1 \leq a \leq 2n-2} \frac{r}{R + a\; r}\;.$$

Next, we notice that there are $(2n-2)!$ different computations that reach a completed state of fork $m+1$. [The different completed states are actually 2^{n-2} because the partition of nodes with same ledgers may change, but this is not relevant in the following discussion. In Figure 4 we show the case of three nodes.]

Therefore the whole probability is

$$\prod_{1 \leq a \leq 2n-2} \frac{a\; r}{R + a\; r}\;.$$

In general, we may write this probability as a function

$$\Psi(u,v) = \prod_{1 \le a \le 2u-2-v} \frac{a\ r}{R + a\ r}$$

where u is the number of nodes of the system and v is the number of nodes that have received the first block ($v \le n-2$ because otherwise we do not have forks). In the above simple case, $u = n$ and $v = 0$.

Since the probability to reach $s_{1,k+1}$ from the initial state is

$$\frac{r_{w_1}}{R}\ \frac{r_{w_{k+1}}}{R + (n-1)r}$$

then the probability to reach a completed state of fork $m+1$ starting from the initial state and traversing $s_{1,k+1}$ is

$$\frac{r_{w_1}}{R}\ \frac{r_{w_{k+1}}}{R + (n-1)r} \Psi(n,0)\ .$$

We notice that there is a symmetric state in the system, where node $k+1$ mines before node 1. The composite probability of these two states is therefore

$$2 \times \frac{r_{w_1}}{R}\ \frac{r_{w_{k+1}}}{R + (n-1)r} \Psi(n,0)\ .$$

We are in place to compute the probability that the two nodes that mine are i and j, where $1 \le i \le k$ and $k+1 \le j \le n$. This probability is

$$2 \times \sum_{1 \le i \le k, k+1 \le j \le n} \frac{r_{w_i}}{R}\ \frac{r_{w_j}}{R + (n-1)r} \Psi(n,0)\ .$$

The general case is when

1. a node $i \in \{1, \cdots, k\}$ mines,
2. the new block is communicated to a set of nodes H and
3. a node in $\{k+1, \cdots, n\} \setminus H$ mines.

The probability of this case is $\Theta(i, |H|, j)$

$$\frac{r_{w_i}\ r_{w_j}}{R\ (R + (n-1-|H|)r)} \left(\prod_{1 \le h \le |H|} \frac{h\ r}{R + (n-h)r} \right) \Psi(n, |H|)$$

where the factor $\left(\prod_{1 \le h \le |H|} \frac{h\ r}{R + (n-h)r} \right)$ is the probability of nodes in H to receive the first block mined. Henceforth, the probability to reach a completed state with fork $m+1$ from the initial state is

$$\sum_{\substack{1 \le i \le k \\ H \subset \{1, \cdots, n\} \setminus i \\ j \in \{k+1, \cdots, n\} \setminus H}} \Theta(i, |H|, j) + \sum_{\substack{k+1 \le j \le n \\ H \subset \{1, \cdots, n\} \setminus j \\ i \in \{1, \cdots, k\} \setminus H}} \Theta(j, |H|, i)$$

which is exactly what stated in the theorem. ◀ ◀

Theorem 10. *Let P be a completed state of a blockchain system of n miners with exactly one that is hostile and let r_{w_d} its mining rate. The probability* $\mathsf{Prob}(P_m)$ *to reach a completed state where the hostile miner has created an alternate chain longer than the honest one from $m, m \geq 1$, blocks behind is smaller than ($R = \sum_{j=1}^{n} r_{w_j}$ and we assume that, for every i, j, $r = r_i = r_j$)*

$$\sum_{k\geq 1}\Big[\Phi(r_{w_d}, r, R)^k \Big(\sum_{1\leq j\leq n-1} \Phi(r_{w_j}, r, R)\Big)^{k-1}\Big]^m$$

where $\Phi(r_w, r, R) = \dfrac{r_w}{R}\prod_{1\leq a\leq n-1}\dfrac{a\ r}{R+(n-a)r}$.

Proof. Assume to be in a completed state in which every node has the same ledger, i.e. $B_i = B_j$, $\forall i, j \in \{1, \ldots, n\}$. We want to compute the probability to reach a completed state in which the dishonest node has created an alternate chain from m blocks behind. We start by computing the probability that the dishonest node n_d has caught up by one block. This kind of attack succeeds if n_d mines one block and all the other nodes receive the block, i.e. the probability is:

$$\frac{r_{w_d}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}$$

Otherwise, it succeeds also in the case the honest nodes mine k blocks and n_d mines $k+1$ blocks in the same amount of time. Thus, we obtain the formula

$$\sum_{k\geq 1}\Big(\frac{r_{w_d}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}\Big)^k\Big(\sum_{1\leq j\leq n-1}\frac{r_{w_j}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}\Big)^{k-1}$$

Now, is trivial to prove that the probability that n_d create an alternative chain one block longer than the original chain from m blocks behind is

$$\Big[\sum_{k\geq 1}\Big(\frac{r_{w_d}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}\Big)^k\Big(\sum_{1\leq j\leq n-1}\frac{r_{w_j}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}\Big)^{k-1}\Big]^m$$

Finally, note that $\mathsf{Prob}(P_m)$ is less than one. In fact, by observing that $\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r} \leq 1$,

$$\begin{aligned}
\mathsf{Prob}(P_1) &= \textstyle\sum_{k\geq 1}\Big(\frac{r_{w_d}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}\Big)^k\Big(\sum_{1\leq j\leq n-1}\frac{r_{w_j}}{R}\prod_{1\leq a\leq n-1}\frac{a\ r}{R+(n-a)r}\Big)^{k-1}\\
&\leq \textstyle\sum_{k\geq 1}\Big(\frac{r_{w_d}}{R}\Big)^k\Big(\sum_{1\leq j\leq n-1}\frac{r_{w_j}}{R}\Big)^{k-1}\\
&= \textstyle\sum_{k\geq 1}\Big(\frac{r_{w_d}}{R}\Big)^k\Big(1-\frac{r_{w_d}}{R}\Big)^{k-1}\\
&= \textstyle\sum_{k\geq 1}\frac{\Big(\frac{r_{w_d}}{R}\Big)^k\Big(1-\frac{r_{w_d}}{R}\Big)^k}{1-\frac{r_{w_d}}{R}}\\
&= \textstyle\sum_{k\geq 1}\frac{\Big(\frac{r_{w_d}}{R}-\frac{r_{w_d}^2}{R^2}\Big)^k}{1-\frac{r_{w_d}}{R}}\\
&= \frac{R}{R-r_{w_d}}\textstyle\sum_{k\geq 1}\Big(\frac{r_{w_d}}{R}-\frac{r_{w_d}^2}{R^2}\Big)^k
\end{aligned}$$

This series is a geometric one with common ratio $\rho = \frac{r_{w_d}}{R} - \frac{r_{w_d}^2}{R^2} \in [0,1]$ because by hypothesis $r_{w_d} \leq R \leq 1$. Thus, we have

$$\begin{aligned}
\frac{R}{R - r_{w_d}} \sum_{k \geq 1} \Big(\frac{r_{w_d}}{R} - \frac{r_{w_d}^2}{R^2} \Big)^k &= \frac{R}{R - r_{w_d}} \Big(\frac{1}{1 - \frac{r_{w_d}}{R} + \frac{r_{w_d}^2}{R^2}} - 1 \Big) \\
&= \frac{R}{R - r_{w_d}} \, \frac{\frac{r_{w_d}}{R}(1 - \frac{r_{w_d}}{R})}{1 - \frac{r_{w_d}}{R} + \frac{r_{w_d}^2}{R^2}} \\
&= \frac{\frac{r_{w_d}}{R}}{1 - \frac{r_{w_d}}{R} + \frac{r_{w_d}^2}{R^2}} \\
&= \frac{r_{w_d} \; R}{R^2 - r_{w_d} \; R + r_{w_d}^2} \\
&\leq 1
\end{aligned}$$

Since $\mathsf{Prob}(P_1) \leq \frac{r_{w_d} \; R}{R^2 - r_{w_d} \; R + r_{w_d}^2} \leq 1$, it immediately follows that

$$\mathsf{Prob}(P_m) \leq \Big(\frac{r_{w_d} \; R}{R^2 - r_{w_d} \; R + r_{w_d}^2} \Big)^m \leq 1 \; .$$

◀ ◀