A Formal Analysis of Blockchain Consensus

Cosimo Laneve and Adele Veschetti

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

Keywords: Distributed consensus · Distributed ledgers · Blockchain · Stochastic pi calculus · forks.

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) [21]. 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 [11]. 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 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, 28, 9, 3], it turned out that the correct mathematical framework of blockchain corresponds to a Continuous Time Markov Chain. This is why the process calculus we have chosen is the stochastic pi calculus [26, 5, 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 [21]. 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 [21]: the presence of hostile nodes mining new blocks in positions that are different from the correct one (blocks are not 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 6^{-m}.

In Section 6, in order to highlight the expressive power of our modelling language, we discuss a blockchain system where new miners may dynamically join the system. We report our concluding remarks in Section 7. For space constraints, the proofs of our main statements are omitted; reviewers may find them in the full paper [1].

Related work. The blockchain protocol was introduced by Haber and Stornetta [15] 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 [12], 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 [23]. 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 [25], 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 [6, 2].

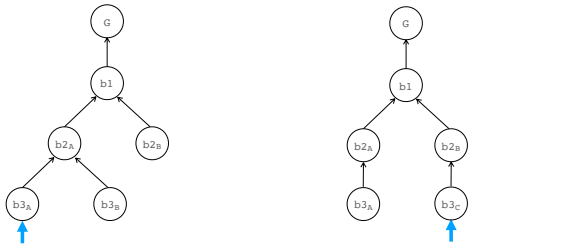
A number of researches address attacks to the blockchain protocol. The works [7, 27, 14] address the delays of communications and [27] also demonstrates

[1] `http://www.cs.unibo.it/~laneve/papers/AFABC.pdf`

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 [20, 12, 23], where it is also proved that the blockchain protocol is safe as long as honest miners are in the majority. In [22], 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 [18]. Finally, a fully implemented attack against Ethereum blockchain, which covers both a network and a double spending attack, is delivered in [8]. 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

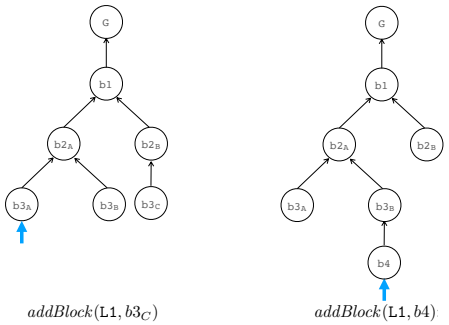
A *ledger*, noted $\mathtt{L}$, $\mathtt{L}'$, ..., is a pair $(\mathtt{T}, \mathtt{h})$ where $\mathtt{T}$ is a nonempty *tree of blocks* and h is the *handle*, e.g. a pointer to a leaf block at maximal depth. We note $\mathtt{T}$ with $\mathit{tree}(\mathtt{L})$ and $\mathtt{h}$ with $\mathit{handle}(\mathtt{L})$. 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* $\mathtt{L}$, noted $\mathtt{L}\uparrow$, is the sequence $[b_0, b_1, b_2, \ldots]$ 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 $\mathtt{L}$ 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 $\mathtt{L}$. 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



$addBlock(\texttt{L1}, b3_C)$ $addBlock(\texttt{L1}, b4)$

In particular, the handle of $addBlock(\texttt{L1}, b3_c)$ is the same of `L1`, while `L1` handle is different from the one of $addBlock(\texttt{L1}, b4)$ 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, ...; *process names*, ranged over by A, B, ...; *variables* X, Y, Z, ... As usual, we address tuples with $\overline{u}$. The stochastic pi calculus has four syntactic categories, *processes* written P, *conditional summands* written M, *summands* written Σ, and *expressions* written e. The grammar is detailed below.

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

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. In equations $\mathsf{A}(\overline{X}) = P$, P is called *body* of A and we assume that process name invocations in bodies, including possible recursive invocations, are always 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 $|\Sigma|$. Let also $|P|$ be the number of conditional summands in P defined as follows: $|M| = 1$, $|(\nu\ x@r)\ P| = |P|$, $|P \mid P'| = |P| + |P'|$ and $|\mathsf{A}(\overline{e})| = |P|$, where P is the body of A (this is well-defined because of the guardedness constraint of invocations in bodies).

Expressions are names, variables, (unspecified) blocks b, ..., 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 $b_1, \ldots, b_n$ is denoted $[b_1, \ldots, b_n]$; $\mathtt{l}^\frown b$ returns the list appending b to $\mathtt{l}$. We use the operation $\mathtt{l} = \varepsilon$, which is `true` if $\mathtt{l}$ is empty, `false` otherwise; the operations $head(\mathtt{l})$ and $tail(\mathtt{l})$ 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 $fn(P)$ for the set of such names and variables in 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 $M \overset{\mu,1\cdot h}{\longmapsto} P$, where μ is either r or $x\mathtt{?}(\overline{Y})$ or $(\overline{z}@\overline{r})x\mathtt{!}\,\overline{u}$, with $\overline{z} \subseteq \overline{u}$, 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 change state. Let ψ ranges over labels μ, ℓ and ν, ℓ, ℓ'. Let also $nm(r) = \varnothing$ and $nm(x\mathtt{?}(\overline{Y})) = nm(x) = \{x\}$ and $nm((\overline{z}@\overline{r})x\mathtt{!}\,\overline{u}) = \{x\} \cup (\overline{u} \setminus \overline{z})$. Finally, let $\psi + k$ defined as follows: $(\mu, k' \cdot h) + k = \mu, (k + k') \cdot h$ and $(\nu, k' \cdot h', k'' \cdot h'') + k = \nu, (k + k') \cdot h', (k + k'') \cdot h''$.

The intensional semantics of stochastic pi calculus performs an accurate estimation of positions of processes that actually move. In rules [TAU], [INP], [OUT], [SUM-L], and [SUM-R], the transition's label records both the move and the position of the process in the moving summand – in this case, the k-index is 1. In general, in oder to determine k, in [PAR-L], we single out the parallel subprocess to the left of the moving one, say Q, compute the number of conditional summands therein, namely $|Q|$, and add the result to the k-index. We use the same technique for communications in [COM-L] and [COM-R]. In this case, the k-index of the subprocess to the right is updated according to the number of conditional summands in the process to the left. [PAR-R] admits liftings of transitions to contexts where the parallelism is to the right, consequently, the positions in the labels remain unchanged.

The transition relation of Table 1 is unsuitable for modelling the stochastic semantics of a concurrent system because it is excessively intensional. Consider

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

$$\text{[SUM-L]}\quad \frac{|\Sigma'| = h' \quad \Sigma \overset{\mu,k\cdot h}{\longmapsto} P}{\Sigma' + \Sigma \overset{\mu,k\cdot(h+h')}{\longmapsto} P} \qquad \text{[SUM-R]}\quad \frac{\Sigma \overset{\mu,k\cdot h}{\longmapsto} P}{\Sigma + \Sigma' \overset{\mu,k\cdot h}{\longmapsto} P}$$

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

$$\text{[NEW]}\quad \frac{P \overset{x,\ell,\ell'}{\longmapsto} Q}{(\nu\ x@r)\ P \overset{r,\ell,\ell'}{\longmapsto} (\nu\ x@r)\ Q} \qquad \text{[NEW-BO]}\quad \frac{P \overset{(\overline{z}@\overline{r})x!\,\overline{u},\ell}{\longmapsto} Q \quad z' \in \overline{u} \quad z' \neq x}{(\nu\ z'@r')\ P \overset{(\overline{z}z'@\overline{r}r')x!\,\overline{u},\ell}{\longmapsto} Q} \qquad \text{[NEW-NO]}\quad \frac{P \overset{\psi}{\longmapsto} Q \quad z \notin nm(\psi)}{(\nu\ z@r)\ P \overset{\psi}{\longmapsto} (\nu\ z@r)\ Q}$$

$$\text{[IDE]}\quad \frac{\mathsf{A}(\overline{X}) = P \quad P\{\overline{u}/\overline{X}\} \overset{\mu,\ell}{\longmapsto} Q}{\mathsf{A}(\overline{u}) \overset{\mu,\ell}{\longmapsto} Q} \qquad \text{[PAR-L]}\quad \frac{|Q| = k \quad P \overset{\psi}{\longmapsto} P'}{Q \mid P \overset{\psi+k}{\longmapsto} Q \mid P'} \qquad \text{[PAR-R]}\quad \frac{P \overset{\psi}{\longmapsto} P'}{P \mid Q \overset{\psi}{\longmapsto} P' \mid Q}$$

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

Table 1. The intensional semantics of the stochastic pi calculus.

the process

$$x!\,1.P + y!\,2.P' \mid x?(u).Q \mid x?(u).Q \mid y?(v).R$$

where x has rate r and y has rate r'. The above process may transit with either $\overset{r,1\cdot 1,2\cdot 1}{\longmapsto}$ or $\overset{r,1\cdot 1,3\cdot 1}{\longmapsto}$ or $\overset{r',1\cdot 2,4\cdot 1}{\longmapsto}$ where the labels ℓ and ℓ' distinguish the two pairs of interacting processes, and this is not reasonable in practice because the processes may be stored in some part of the memory and may be relocated, or may be in different nodes of a distributed system and may migrate. More reasonably, the transition relation should represent collectively all the possible positions of processes. For instance, in the above case, abstracting out the order of the processes, we obtain a unique transition whose rate is twice the rate r. It turns out that the right abstraction follows by quotienting processes with so-called structural congruence.

Definition 1. *The* structural congruence*, noted* $\equiv$*, is the least equivalence on processes containing alpha-conversion of bound names, associativity and commutativity of* $\mid$ *and* $+$ *with identity* $\mathsf{0}$ *and containing*

$$\begin{aligned} (\nu\ x@r)\ \mathsf{0} &\equiv \mathsf{0} \\ (\nu\ x@r)\ (\nu\ y@r')\ P &\equiv (\nu\ y@r')\ (\nu\ x@r)\ P \\ (\nu\ x@r)\ (P \mid Q) &\equiv P \mid (\nu\ x@r)\ Q \qquad \text{if } x \notin fn(P)\,. \end{aligned}$$

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 $P \equiv P'$.

Proposition 1. *If $P \equiv Q$ and $P \overset{\nu,\ell,\ell'}{\longmapsto} P'$ then there are Q', ℓ'' and ℓ''' such that $Q \overset{\nu,\ell'',\ell'''}{\longmapsto} Q'$ and $P' \equiv Q'$. Similarly for $P \overset{\mu,h}{\longmapsto} P'$ and $P \overset{\mu,\ell}{\longmapsto} P'$.*

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

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

The stochastic transition relation of pi calculus induces to a *Continuous Time Markov Chain (CTMC)* with only *silent interactive transitions* [16, 17]. In a markovian state with n outgoing markovian transitions labeled $\lambda_1, \ldots, \lambda_n$, 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 [19, 13, 24], 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 forward 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 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 [2]. Their definition is

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

In this first phase we suppose that there are not hostile miners. 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 $w\,!\,\mathit{newBlock}(L)$. 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 forwarded to all the other miners of the network – the parallel subprocess starting with $w?(b)$.
- 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 – $\mathit{head}(X)$ –, 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

[2] 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 Ch of constant (channel) names and rates and (2) to extend Table 1 with the additional rule

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

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.

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 obtained from the model, we select those where the forwarding 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 [7].

Definition 3. *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 2. *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 4. *Let* L *and* L' *be two ledgers and let*

- *m be the length of* $\mathtt{L}\uparrow$,
- *n be the length of* $\mathtt{L}'\uparrow$,
- *h be the length of the maximal common suffix of* $\mathtt{L}\uparrow$ *and* $\mathtt{L}'\uparrow$.

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 $z_1, \cdots, z_n$ send blocks with rates $r_1, \cdots, r_n$, respectively. These rates are actually parameters of an exponential distribution [7]. 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 1. *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, $r = r_i = r_j$)

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

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

It is worth to notice that the probability $\mathsf{Prob}(P_{\rightsquigarrow m+1})$ of Theorem 1 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 1 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 1 because every node may mine after the first one.

Proposition 3. *Let* P *be a completed state of a blockchain system consisting of* n *miners having ledger* $\mathtt{L}$. *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, $r = r_i = r_j$)

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

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

In order to bear some numerical results, we instantiate our probability with realistic channel rates. In [21], 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 $r_{w_{\mathtt{BTC.com}}} = 0.000245$, $r_{w_{\mathtt{ANTpool}}} = 0.000196$, $r_{w_{\mathtt{F2pool}}} = 0.00018$, 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 [7] 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.

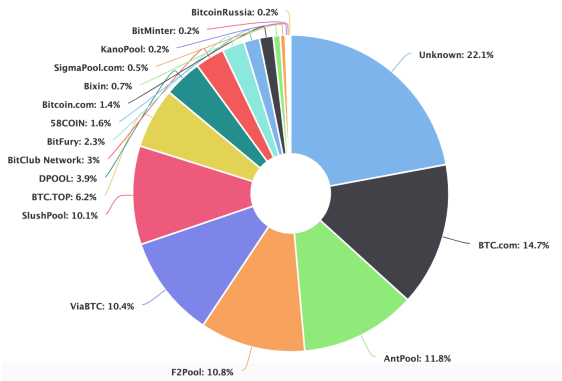


Fig. 1. Hashrate distribution of Bitcoin mining pools on January 2019.
Source: https://www.blockchain.com/.

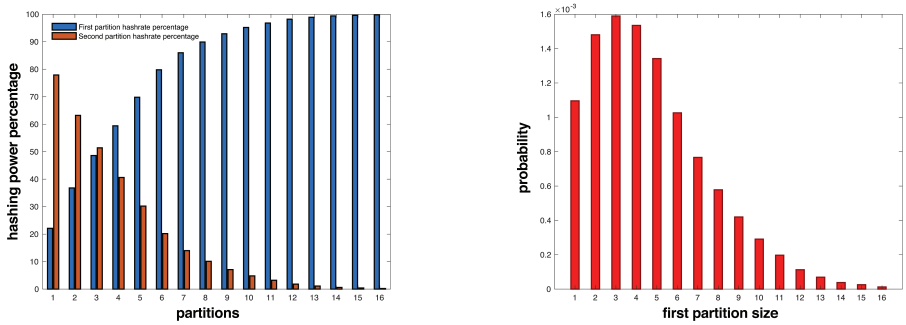


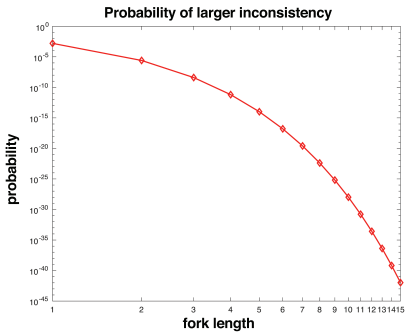
Fig. 2. Probability of inconsistency trend in relation of the hashrate percentage of the two partitions.

In Figure 2 we illustrate our results obtained via simulation. We compute the probability of a fork putting the realistic channel rates in the formula of Theorem 1, in order to simulate the real hashrate percentage of the Bitcoin mining pools.

We analyse hashrate distribution of Bitcoin network described in Figure 1 in which there are 17 main mining pools. We suppose that the network is in an inconsistent state: there are two disjoint subsets of the miners with two different handles.

In the left picture there are the 16 possible initial network 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 hashrates.

In the following figure we highlight the probability decay of creating larger and larger inconsistencies in the case where the two initial partition have equiv-



alent hashrates (which is the case where the probability is higher). For example, it is around 10^{-30} for forks of length 10.

Remark 1. 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 – see Section 6). 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 [19, 13, 24], 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 [21], 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 `L` and a block $d \in$ `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

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

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 2. *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 1, the technique used for demonstrating the above statement consists of analyzing the stochastic transition system. The technique is therefore different from the one in [21], 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 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.

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.

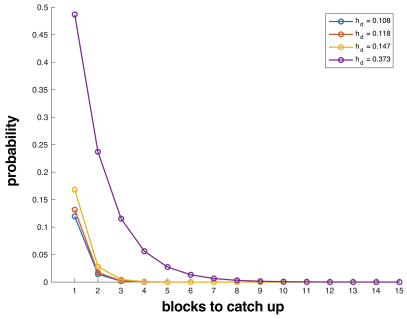


Fig. 3. Probability of a successful attack depending on the hashing power of n_d.

6 Extensions

Our abstract modelling of blockchain in Section 4 defines the interactions between a fixed set of miners. It turns out that the stochastic pi calculus is too expressive for defining such systems and a less expressive calculus, such as CCS plus values [10] (extended with ledgers), would have been sufficient. However, this choice would not have allowed the analysis of more complex systems, such as those in which new miners may dynamically join the system. In this section we discuss how such kind of systems may be described in the modelling language of Section 3. Its analysis is deferred to the full paper.

In the following stochastic pi calculus process, a blockchain system is a parallel composition of miners and a process modelling the network. The initial state, assuming to start with n miners, is

$$(\nu\ net@r_{net}, newn@r_{newn}, \nu z_1@r_1, \cdots, z_n@r_n)\Big(\\ \mathsf{Net}(\{z_1, \cdots, z_n\}) \mid \textstyle\prod_{z_i \in \{z_1, \cdots, z_n\}} \mathsf{Miner}(\mathtt{G}, \varnothing, z_i) \Big)$$

where *net* and *newn* are two channels for sending new blocks to the miners and creating new miners (see below). In order to define Net we extend the model of Section 3 with the MAP operation on lists of pairwise different channels. $\text{MAP}(\overline{z}, b)$ is defined by the following equivalence relations

$$\begin{aligned} &\text{MAP}(\varepsilon, b) \equiv \mathsf{0} \\ &\text{MAP}(\overline{z}, b) \equiv \text{MAP}(\overline{z'}, b) \qquad \overline{z'} \text{ is a permutation of channels in } \overline{z} \end{aligned}$$

and has transition relation

$$\frac{i \in 1..n}{\textsc{map}(z_1 \cdots z_n, b) \overset{z_i\,!\,b,i\cdot 1}{\longmapsto} \textsc{map}((z_1 \cdots z_n) \setminus z_i, b)}\ [\textsc{map}]$$

where $(z_1 \cdots z_n) \setminus z_i$ removes z_i from the list $z_1 \cdots z_n$. Additionally, we let $|\textsc{map}(\overline{z}, b)| = |\overline{z}|$. The equation of Net is:

$$\begin{aligned}\mathsf{Net}(\overline{z}) \;=\; & \mathit{net}?(z', b).\ (\ \mathsf{Net}(\overline{z})\ \mid\ \textsc{map}(\overline{z} \setminus z', b)\ ) \\ & +\ \mathit{newn}?(L, X).\ (\nu\ z'@r_{z'})\ \mathsf{Net}(\overline{z}^\frown z')\ \mid\ \mathsf{Miner}(L, X, z')\end{aligned}$$

The argument $\overline{z}$ is the list of miner's channels. Net performs two operations: either (i) it receives on the channel *net* a new block from a miner, together with the corresponding channel name or (ii) receives on *newn* the request for creating a new miner. In the first case, the new block is forwarded to the miners by mean of the operation MAP. In the second case, the input $\mathit{newn}?(L, X)$ carries a ledger L and a list X of blocks still to be inserted. These data are used for triggering a new miner, once a new communication channel (z') has been created for it. The values of L and X are transmitted by an existing miner – see the last but one line in the code below –, e.g. a new miner dynamically arise as a sibling copy of an existing miner, which however will evolve independently.

The definition of Miner is similar to that of Section 4, except for the last two lines:

$$\begin{aligned}\mathsf{Miner}(L, X, z) \;=\; & (\nu\ w@r)\big(\\ & \big(\ z?(b).\ \mathsf{Miner}(L, X^\frown b, z) \\ & +\ w\,!\,\mathit{newBlock}(L) \\ & +\ \mathsf{if}\ (X = \varepsilon)\ \mathsf{then}\ \tau_{r'}.\mathsf{Miner}(L, X, z) \\ & \quad \mathsf{else\ if}\ (\mathit{head}(X).\mathtt{id} \in L.\mathtt{blocks})\ \mathsf{then} \\ & \qquad\qquad \tau_{r'}.\mathsf{Miner}(\mathit{addBlock}(L, \mathit{head}(X)), \mathit{tail}(X), z) \\ & \quad \mathsf{else}\ \tau_{r'}.\ \mathsf{Miner}(L, \mathit{tail}(X)^\frown \mathit{head}(X), z) \\ & +\ \mathit{newn}\,!\,(L, X).\ \mathsf{Miner}(L, X, z) \\ & \big)\ \mid\ w?(b).(\mathsf{Miner}(\mathit{addBlock}(L, b), X, z)\ \mid\ \mathit{net}\,!\,(z, b))\ \big)\end{aligned}$$

We discuss the last line, because the last-but-one line has been already spelled out. In this case, when a new block is created, the forward is delegated to the Net process by communicating its genesis through the channel *net*.

We conclude by observing that operations such as communicating channel names or carrying list of channels as arguments are proper of pi-calculus and cannot be modelled in CCS-like languages.

7 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:

the first one, for sake of simplicity, represents when the system consists of honest miners and the second reflects a system in which exists one hostile miner that mines blocks in wrong positions. The adversary model used in this paper is not the best one an adversary can implement, but the analysis of further strategies is left to future work. 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.

The main contribution of our paper is a process calculus for modeling the behaviour of miners, based on the stochastic pi calculus. This approach is, as far as we know, original and 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, etc. In this paper we restricted to the case of static sets of nodes because computing the probabilities by hand is extremely time-consuming, In the future research we plan to model the more realistic setting of a dynamic network, actually this is the reason why we decided to use the pi calculus.

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 [19, 13, 24]. 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. Biais, B., Bisiere, C., Bouvard, M., Casamatta, C.: The blockchain folk theorem (2018)
2. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In: Proc. of SP 2015. pp. 104–121. IEEE Computer Society (2015)
3. Bowden, R., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Block arrivals in the bitcoin blockchain. CoRR **abs/1801.07447** (2018)
4. Bravetti, M.: Reduction semantics in markovian process algebra. J. Log. Algebr. Meth. Program. **96**, 41–64 (2018)
5. Cardelli, L., Mardare, R.: Stochastic pi-calculus revisited. In: Proc. 10th Theoretical Aspects of Computing. Lecture Notes in Computer Science, vol. 8049, pp. 1–21. Springer (2013)
6. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proc. Computer and Communications Security. pp. 154–167. CCS '16, ACM (2016)
7. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: Proc. 13th IEEE P2P. pp. 1–10. IEEE (2013)
8. Ekparinya, P., Gramoli, V., Jourjon, G.: Double-spending risk quantification in private, consortium and public ethereum blockchains. CoRR **abs/1805.05004** (2018), http://arxiv.org/abs/1805.05004
9. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (Jun 2018). https://doi.org/10.1145/3212998, http://doi.acm.org/10.1145/3212998

10. Fiore, M.P., Turi, D.: Semantics of name and value passing. In: Proc. of Logic in Computer Science 2001. pp. 93–104. IEEE Computer Society (2001)
11. Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
12. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Proc. of EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9057, pp. 281–310. Springer (2015)
13. Gilmore, S., Hillston, J.: The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In: Proc. 7th Computer Performance Evaluation, Modeling Techniques and Tools. Lecture Notes in Computer Science, vol. 794, pp. 353–368. Springer (1994)
14. Göbel, J., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. Perform. Eval. **104**, 23–41 (2016)
15. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. Journal of Cryptology **3**(2), 99–111 (Jan 1991). https://doi.org/10.1007/BF00196791, https://doi.org/10.1007/BF00196791
16. Hermanns, H.: Interactive Markov Chains: And the Quest for Quantified Quality. Springer-Verlag, Berlin, Heidelberg (2002)
17. Hillston, J.: A Compositional Approach to Performance Modelling (Distinguished Dissertations in Computer Science). Cambridge University Press, New York, NY, USA (2005)
18. Karame, G., Androulaki, E., Capkun, S.: Double-spending fast payments in bitcoin. In: Proc. of CCS'12. pp. 906–917. ACM (2012)
19. Kwiatkowska, M.Z., Norman, G., Parker, D.: Probabilistic symbolic model checking with PRISM: A hybrid approach. In: Proc. TACAS 2002. Lecture Notes in Computer Science, vol. 2280, pp. 52–66. Springer (2002)
20. Miller, A., LaViola Jr, J.J.: Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. Available on line: http://nakamotoinstitute.org/research/anonymous-byzantine-consensus (2014)
21. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), http://www.bitcoin.org/bitcoin.pdf
22. Ozisik, A.P., Levine, B.N.: An explanation of nakamoto's analysis of double-spend attacks. CoRR **abs/1701.03977** (2017), http://arxiv.org/abs/1701.03977
23. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Proc. of EUROCRYPT 2017. Lecture Notes in Computer Science, vol. 10210, pp. 643–673. Springer (2017)
24. Phillips, A., Cardelli, L.: Efficient, correct simulation of biological processes in the stochastic pi-calculus. In: Proc. CMSB 2007. Lecture Notes in Computer Science, vol. 4695, pp. 184–199. Springer (2007)
25. Pîrlea, G., Sergey, I.: Mechanising blockchain consensus. In: Proc. 7th Certified Programs and Proofs, CPP. pp. 78–90. ACM (2018)
26. Priami, C.: Stochastic Pi-Calculus. The Computer Journal **38**(7), 578–589 (1995)
27. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. In: Proc. of Financial Cryptography and Data Security 2015. Lecture Notes in Computer Science, vol. 8975, pp. 507–527. Springer (2015)
28. Zamyatin, A., Stifter, N., Schindler, P., Weippl, E.R., Knottenbelt, W.J.: Flux: Revisiting near blocks for proof-of-work blockchains. IACR Cryptology ePrint Archive **2018**, 415 (2018)