Bitcoin Blockchain Dynamics: the Selfish-Mine Strategy in the Presence of Propagation Delay

J. Göbel*, H.P. Keeler†, A.E. Krzesinski‡ and P.G. Taylor†
*Department of Informatics, University of Hamburg, 22527 Hamburg, Germany
Email: goebel@informatik.uni-hamburg.de
†Department of Mathematics and Statistics, University of Melbourne, Vic 3010, Australia
Email: {keeler,taylorpg}@unimelb.edu.au
‡Department of Mathematical Sciences, Stellenbosch University, 7600 Stellenbosch, South Africa
Email: aek1@cs.sun.ac.za

***Abstract*—In the context of the 'selfish-mine' strategy proposed by Eyal and Sirer, we study the effect of propagation delay on the evolution of the Bitcoin blockchain. First, we use a simplified Markov model that tracks the contrasting states of belief about the blockchain of a small pool of miners and the 'rest of the community' to establish that the use of block-hiding strategies, such as selfish-mine, causes the rate of production of orphan blocks to increase. Then we use a spatial Poisson process model to study values of Eyal and Sirer's parameter γ, which denotes the proportion of the honest community that mine on a previously-secret block released by the pool in response to the mining of a block by the honest community. Finally, we use discrete-event simulation to study the behaviour of a network of Bitcoin miners, a proportion of which is colluding in using the selfish-mine strategy, under the assumption that there is a propagation delay in the communication of information between miners.**

Keywords*—Bitcoin, blockchain, block hiding strategies, honest mining, selfish-mine.

I. Introduction

Bitcoin is a peer to peer electronic payment system in which transactions are performed without the need for a central clearing agency to authorize transactions. Bitcoin users conduct transactions by transmitting electronic messages which identify who is to be debited, who is to be credited, and where the change (if any) is to be deposited.

Bitcoin payments use Public Key Encryption. The payers and payees are identified by the public keys of their Bitcoin wallet identities. Each Bitcoin transaction is encrypted and broadcast over the network. Suppose you receive a transaction from Mary. If you can decrypt Mary's message using her public key, then you have confirmed that the message was encrypted using Mary's private key and therefore the message indisputably came from Mary. But how can you verify that Mary has sufficient bitcoins to pay you?

The Bitcoin system solves this problem by verifying transactions in a coded form in a data structure called the *blockchain*, which is maintained by a community of participants, known as *miners*.

It can happen that different miners have different versions of the blockchain, something which occurs because of propagation delays, see Decker and Wattenhofer [1]. For Bitcoin to be able to function, it is essential that these inconsistencies are resolved within a short timescale. We are interested in how the inconsistencies arise and how they are resolved (1) when all participants are acting according to the Bitcoin protocol, and (2) when a pool of participants is using the 'selfish-mine' strategy proposed by Eyal and Sirer [2].

A. The blockchain

At the heart of the Bitcoin system is the computational process called *mining*, which involves the solution of a computationally-difficult cryptographic problem. Bitcoin miners receive copies of all transactions as they are generated. They examine the blockchain to investigate the history of the bitcoins involved in each transaction. If the proposed transaction has sufficient bitcoin credit, then it is accepted for incorporation into the block that the miner is currently working on.

Each transaction is identified with a double SHA-256 hash. Miners gather transactions together and use their hashes, together with the hash that is at the current head of the blockchain, as inputs to the cryptographic problem. If a miner succeeds in solving the problem, it is said to have *mined* a block that contains records of all the transactions that were part of the calculation. The miner receives a reward (currently 25 bitcoins) for accomplishing this, along with a small transaction fee gathered from each transaction in the block.

The process works as follows. A miner M computes a block hash h over a unique ordering of the hashes of all the transactions that it is intending to incorporate into its next block B. It also takes as input the block solution s_{i-1} at the head of its current version of the blockchain. Denoting concatenation of strings by the symbol $+$, the cryptographic problem that M has to solve is: compute a SHA-256 hash

$$s_i = hash(n + h + s_{i-1}), \tag{1}$$

such that s_i has at least a specified number x of leading zeros where $x \sim 64$. The string n is a random "nonce" value. If s_i

The work of Anthony Krzesinski is supported by the Research Foundation of South Africa (Grant specific unique reference number (UID) 83965) and Telkom SA Limited.
The work of Peter Taylor is supported by the Australian Research Council Laureate Fellowship FL130100039 and the ARC Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS).

does not have at least x leading zeros, then n is updated and s_i is recomputed until a solution is found with the required number of leading zeros.

Once mined, the new block is communicated to the members of the peer network and, subject to the fine detail of the rules that we shall discuss in the next section, the new block is added to the blockchain at each peer. The blockchain thus functions as a public ledger: it records every Bitcoin payment ever made.

The objective of the designers of the Bitcoin protocol was to keep the average rate at which blocks are added to the long-term blockchain at six blocks per hour. To this end, the value of x, which reflects the difficulty of the computational problem inherent in (1), is adjusted after the creation of each set of 2016 new blocks. If the previous 2016 blocks have been created at an average rate faster than six blocks per hour, then the problem is made more difficult, if they have been created at a slower average rate, then it is made less difficult. The effect is that the difficulty varies in response to the total amount of computational power that the community of miners is applying.

The test of whether a particular hash has the required number of leading zeros is a success/failure experiment whose outcome is independent of previous experiments. Therefore, the number of experiments required for the first success is geometrically distributed and, given that the individual success probabilities are very low and the time taken to perform an experiment is correspondingly very small, the time taken to achieve a success is very well-modelled by an exponential random variable. It is thus reasonable to model block creation instants as a Poisson process with a constant rate of six per hour.

The *difficulty* of a sequence of blocks is a measure of the amount of computing effort that was required to generate the sequence. This can be evaluated in terms of the numbers of leading zeros that were required when the blocks in the sequence were created. When Bitcoin was started, miners used PCs to solve the cryptographic puzzle and earn bitcoins. The difficulty of the puzzle was increased to limit the rate of producing bitcoins. Miners started using the parallel processing capabilities of Graphical Processing Units (GPUs) to solve the cryptographic puzzle. The difficulty of the puzzle was increased again. Miners started using General Programmable Field Arrays (GPFAs). The difficulty was increased yet again. Today miners use Application Specific Integrated Circuit (ASIC) computers.

Miners communicate by broadcasting newly-discovered blocks via a peer-to-peer network. Each miner maintains its own version of the blockchain based upon the communications that it receives and its own discoveries. The protocol is designed so that blockchains are locally updated in such a way that they are identical at each miner or, if they differ, then the differences will soon be resolved and the blockchains will become identical. The way that this process works is explained in the next subsection.

B. Blockchain rules

The material discussed here is obtained from [3]. The *main branch* of the blockchain is defined to be the branch with highest total difficulty.

- **Blocks.** There are three categories of blocks
  1) Blocks in the main branch: the transactions in these blocks are considered to be tentatively confirmed.
  2) Blocks in side branches off the main branch: these blocks have tentatively lost the race to be in the main branch.
  3) Blocks which do not link into the main branch, because of a missing predecessor or nth-level predecessor.

  Blocks in the first two categories form a tree rooted at the very first block, which is known as the *genesis block*, linked by the reference to the hash of the predecessor block that each block was built upon. The tree is almost linear with a few short branches off the main branch.
- **Updating the blockchain.** Consider the situation where a node learns of a new block. This block could either be mined locally or have been communicated after being mined at another node. The actions that the node takes are to:
  1) Reject the new block if a duplicate of the block is present in any of the three block categories mentioned above.
  2) Check if the predecessor block (that is, the block matching the previous hash) is in the main branch or a side branch. If it is in neither, query the peer that sent the new block to ask it to send the predecessor block.
  3) If the predecessor block is in the main branch or a side branch, add the new block to the blockchain. There are three cases.
     a) The new block extends the main branch: add the new block to the main branch. If the new block is mined locally, relay the block to the node's peers.
     b) The new block extends a side branch but does not add enough difficulty to cause it to become the new main branch: add the new block to the side branch.
     c) The new block extends a side branch which becomes the new main branch: add the new block to the side branch and
        i) find the fork block on the main branch from which this side branch forks off,
        ii) redefine the main branch to extend only to this fork block,
        iii) add each block on the side branch, from the child of the fork block to the leaf, to the main branch,
        iv) delete each block in the old main branch, from the child of the fork block to the leaf,
        v) relay the new block to the node's peers.
  4) Run all these steps (including this one) recursively, for each block for which the new block is its previous block.

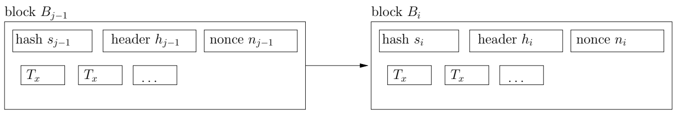

Fig. 1. Mining a block.

C. Blockchain dynamics

Suppose miner M_i is mining block B_i with hash h_i on its version C of the blockchain which has s_{i-1} as its previous hash, and computes a solution s_i to the cryptographic puzzle with nonce n_i. Miner M_i will add B_i to C and broadcast (B_i, n_i, h_i, s_i) to the network. When another miner M_j, who is also working on the blockchain C, receives the communication, it will compute

$$s' = hash(n_i + h_i + s_{j-1}).$$

With reference to Figure 1, if $s' = s_i$ then miner M_j will add block B_i to its blockchain C, abandon the block B_j that it is working on and commence trying to add a block to the chain CB_i. Any transactions in B_j that are not in B_i will be incorporated into in this new block. Importantly, miners M_i and M_j now have identical versions of the blockchain.

The existence of propagation delays can upset the above process, because blocks can be discovered while communication and validation is in process. Decker and Wattenhofer [1] measured the difference between the time that a node announced the discovery of a new block or a transaction and the time that it was received by other nodes for a period of operation in the actual Bitcoin network. They observed that the median time until a node receives a block was 6.5 seconds, the mean was 12.6 seconds and the 95th percentile of the distribution was around 40 seconds. Moreover, they showed that an exponential distribution provides a reasonable fit to the propagation delay distribution.

Suppose all miners are working on the same version C of the blockchain and miner M_i mines block B_i at time t. It will then add B_i to the blockchain C and broadcast block B_i to all its peers. Suppose that this communication reaches miner M_j at time $t + \delta_j$ and that M_j has mined a block B_j at time $t' \in [t, t + \delta_j)$.

Miner M_j now knows about two versions CB_i and C_Bj of the blockchain, which are of the same length. From the point of view of Miner M_j, the blockchain has split, and we can think of the node as being in a 'race' to see which version of the blockchain survives.

Miner M_i will build on CB_i because this is the version of the blockchain that it knew about first. However miner, M_j knew about CB_j first, and will attempt to build on this version of the blockchain. Other miners will work on either CB_i or CB_j depending on which version they heard about first. The 'race' situation is resolved when the next block B^* is mined, say on CB_i, and communicated via the peer network. Then CB_iB^* will be longer than CB_j and all miners will eventually start building on CB_iB^*. It is then likely that the block B_j will not be part of the longterm blockchain and it will become an *orphan block*. Any transactions that are in B_j, but not in B_i or B^*, will be incorporated into a future block.'

The above situation can get more complicated if yet more blocks are mined while communication is taking place, although this would require the conjunction of two or more low-probability events.

A rough calculation based upon the fact that it takes 600 seconds on average for the community to mine a block shows that we should expect that the probability that a new block is discovered while communication and validation of a block discovery is taking place is of the order of $12.6/600 \approx 1/50$, which is small but not negligible. Given that, on average, 144 blocks are mined each day, we should expect this circumstance to occur two to three times each day, which accords with the observed rate of orphan blocks [4].

D. Transaction integrity

In his seminal paper proposing the Bitcoin system [5], Nakamoto dealt with the issue of transaction integrity. He proposed that a vendor should wait until his/her payment transaction has been included in a block, and then z further blocks have been added to the blockchain, before dispatching the purchased goods. The rule-of-thumb that has been adopted is to take $z = 6$, which roughly corresponds to waiting for an hour before dispatching the goods. Assuming that the community can generate blocks at rate λ_2, Nakamoto presented a calculation of the probability P_A that an attacker with enough computing power to generate blocks at rate $\lambda_1 < \lambda_2$ could rewrite the history of the payment transaction by creating an alternate version of the blockchain that is longer than the community's version. Unfortunately, Nakamoto's calculation is incorrect, a fact that was observed by Rosenfeld in [6].

Let the random variable K be the number of blocks created by the attacker in the time that it takes the community to create z blocks. Then, we can get the correct expression for the probability that the attack is successful by noting that $z+K$ is the number of Bernoulli trials required to achieve z successes, with the success probability of an individual trial given by $p \equiv \lambda_2/(\lambda_1 + \lambda_2)$. It is thus a *negative binomial random variable* with parameters p and z.

Now, using Nakamoto's observation that, conditional on the attacker having created K blocks when the vendor dispatches the goods, the probability of the attacker ever being able to build a blockchain longer than the community blockchain is $(\lambda_1/\lambda_2)^{z-K}$ if $K < z$, and one otherwise, we arrive at the expression in equation (1) of [6],

$$P_A = 1 - \sum_{k=0}^{z} \binom{z+k-1}{z-1} \left(p^z(1-p)^k - p^k(1-p)^z\right). \quad (2)$$

E. Selfish-mine

It follows from an analysis similar to that in Section I-D that, if a group of miners control more than half of the total computer power, they can collude to rewrite the history of the transactions. There might, however, be ways for a group to gain an advantage even if it does not control a majority of the computational power.

In [2], Eyal and Sirer proposed a strategy, called 'selfish-mine', and claimed that, using this strategy, a pool of colluding ‘dishonest’ miners, with a proportion $\alpha < 1/2$ of the total computational power, can earn a proportion greater than α of the mining revenue. In this sense, a pool of miners collaborating in using the selfish-mine strategy can earn more than its fair share of the total revenue.

In brief, selfish-mine works as follows. When a pool miner mines a block, it informs its colluding pool of miners, but not the whole community of miners. Effectively, the mining pool creates a *secret* extension of its blockchain, which it continues to work on. The honest miners are unaware of the blocks in the secret extension and continue to mine and to publish their mined blocks and solutions according to the standard protocol.

The computational power available to the honest miners is greater than that available to the mining pool. So, with probability one, the public branch will eventually become as long as the pool’s secret extension. However it is possible that the secret extension will remain longer than the public branch in the short term. The mining pool is giving up the almost certain revenue that it would receive if it published its recently-mined block in return for a bet that its secret branch will become long enough for it to take short-term control of the mining process.

Specifically, if the lead happens to become two or more, then the pool can publish a single block every time that the honest community mines a block, and publish two blocks when its lead is eventually reduced to one. In this way the pool works on its version of the blockchain while allowing the honest community to be engaged in a fruitless search for blocks that have no chance of being included in the long-term blockchain.

The risk to the pool is that, if it has established a lead of exactly one by mining a block B_p, which it has kept secret, and then it is informed that the community has mined a block B_h, the pool may end up not getting credit for the block B_p. To minimise this risk, the selfish-mine strategy dictates that the pool should publish the block B_p immediately it hears about B_h. The pool continues working on B_p itself, and it hopes that at least some of the honest community will also work on B_p, so that the pool will get the credit for B_p if an honest miner manages to extend it.

When Eyal and Sirer [2] modelled the selfish-mine strategy, they included a parameter γ to denote the proportion of the honest community that work on B_p after it has been published according to the scenario described above. They deduced that the pool can obtain revenue larger than its relative size provided that

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2}. \tag{3}$$

Eyal and Sirer’s analysis did not, however, take propagation delay into account. Since the honest community has a head start in propagating B_h before the dishonest miners have even heard about it and then there is a further propagation delay before B_p reaches other honest miners, our first intuition was that γ is likely to be very low in the presence of propagation delays.

In a survey of subversive mining strategies [7], Courtois and Bahack state (provisionally) that the claims made for efficacy of the selfish-mine strategy [2], which is one of the block discarding attacks studied in [8], are exaggerated. However, the conclusions presented in [7] concerning the selfish-mine attack are not based on experimental or modelling analysis.

The purpose of the rest of this paper is to propose some simple models that explicitly take propagation delay into account, which we can use to compare the behaviour of the Bitcoin network when all miners are observing the standard protocol with its behaviour if there is a pool following the selfish-mine strategy.

In next section, we shall introduce and analyse a simple continuous-time Markov chain model that tracks the contrasting states of belief of a ‘pool’ and the ‘rest of the community’ under the assumption that the pool and the community are physically-separated so that communication between the pool and the community takes longer than communication within the pool and within the community. Effectively, we assume that there is no communication delay within the pool and within the community. We conclude that the rate of production of orphan blocks is likely to be much higher when the pool is keeping its newly-discovered blocks secret.

In the following Section III, we study the value of Eyal and Sirer’s parameter γ in a model in which pool miners are distributed according to Poisson processes in the plane and the propagation delay between two miners is normally distributed with a mean that depends on the distance between them.

Finally, in Section IV, we shall report results from a simulation of a network of 1,000 miners, of which a fraction form a dishonest pool, again with propagation delays between all miners that depend on their spatial separation. Some conclusions and further observations are given in Section V.

II. A SIMPLE MARKOV CHAIN MODEL

In this section we shall describe and analyse a simple Markovian model that takes into account the separate states of belief of a ‘pool of Bitcoin miners’ and the ‘rest of the community’ about the blockchain. We assume that communication within the pool and within the community always happens faster than communication between the pool and the community, effectively taking the propagation delay for the former type of communication to be zero.

Such a dichotomy between immediate communication within both the pool and community and delayed communication from pool to community and vice-versa is unlikely to be realistic. However, the model is useful because it illustrates the effect that block-withholding strategies have on the rate of blockchain splits. In the following Sections III and IV, we shall analyse models with more realistic assumptions about communication delay.

If the pool and the rest of the community agree about the blockchain, then we denote the state by $(0,0)$. On the other hand, if the pool has built k blocks onto the last ‘fork block’ where it agreed with the community, and the community has built ℓ blocks beyond the fork block, then we denote the state by (k,ℓ). Given the mechanisms that are in place to resolve inconsistencies, we would expect that states (k,ℓ) for k and ℓ greater than one or two would have a very low probability of occurrence.

A. The pool mines honestly

We assume that the pool discovers new blocks at rate λ_1, while the rest of the community does so at rate λ_2, with $\lambda_2 > \lambda_1$. Without paying attention to node locations, Decker and Wattenhofer [1] observed that it is reasonable to model communication delays with exponential random variables. Since an exponential assumption also helps with analytic tractability, we make such an assumption in this first model. Specifically, we assume that the time that it takes to communicate a discovery of a block from the pool to the community and vice-versa is exponentially-distributed with parameter $\mu \gg \lambda_2$.

If the system is in a state (k, ℓ) with $k \neq \ell$, then it returns to state $(0, 0)$ once communication has occurred, because then the pool and the community will agree about the new state of the blockchain. However, if $k = \ell \geq 1$, then the pool and the community have different, but equal length, versions of the blockchain and will continue mining on the blockchain as they see it. The system therefore remains in state (k, k) until a new block is discovered.

The Markov model has transition rates

$$\begin{aligned} q((k,\ell),(k+1,\ell)) &= \lambda_1, \quad k \geq 0, \ell \geq 0 & (4) \\ q((k,\ell),(k,\ell+1)) &= \lambda_2, \quad k \geq 0, \ell \geq 0 & (5) \\ q((k,\ell),(0,0)) &= \mu, \quad k \neq \ell & (6) \\ q((k,\ell),(k',\ell')) &= 0, \quad \text{otherwise.} & (7) \end{aligned}$$

The first two types of transition, reflected in (4) and (5), occur when the pool (respectively the community) mine a block, while the third, in (6), occurs once communication has occurred when the chain is in a state (k, ℓ) with $k \neq \ell$. This latter rate is a simplification of what could have been assumed: if $|k - \ell| \geq 2$, there are multiple communication tasks in progress, reporting the last $|k - \ell|$ block discoveries in the longest branch and it is only when the communication reporting the discovery of the final block on the longest branch arrives that the state of the system returns to $(0, 0)$. For the sake of tractability in this simple first model, this is the only transition that we have taken into account. As we observed above, states with $|k - \ell| \geq 2$ have a very low probability of occurrence and we can expect that this modification will not have a great effect on the stationary distribution.

The equations for the stationary distribution are

$$\pi(0,0)\,(\lambda_1 + \lambda_2) = \sum_{k=0}^{\infty}\sum_{\ell=0}^{\infty} \pi(k,\ell)\mu I(k \neq \ell), \qquad (8)$$

for $k \neq \ell$,

$$\begin{aligned} \pi(k,\ell)\,(\lambda_1 + \lambda_2 + \mu) &= \pi(k-1,\ell)\lambda_1 I(k > 0) \\ &+ \pi(k,\ell-1)\lambda_2 I(\ell > 0) \qquad (9) \end{aligned}$$

and, for $k = \ell$,

$$\begin{aligned} \pi((k,\ell))\,(\lambda_1 + \lambda_2) &= \pi(k-1,\ell)\lambda_1 I(k > 0) \\ &+ \pi(k,\ell-1)\lambda_2 I(\ell > 0). \qquad (10) \end{aligned}$$

To express the solution of these equations, we need to define a function $n(k, \ell; i)$ which denotes the number of paths that start at the origin and finish at (k, ℓ), take steps on the integer lattice in the directions $(1, 0)$ (north) and $(0, 1)$ (east) and which contain exactly i points (j, j) for $j > 0$.

As an example, we can see that $n(3, 2; 2) = 4$ because there are four paths

$$\begin{aligned} &[(0,0),(1,0),(1,1),(2,1),(2,2),(3,2)], \\ &[(0,0),(1,0),(1,1),(1,2),(2,2),(3,2)], \\ &[(0,0),(0,1),(1,1),(2,1),(2,2),(3,2)], \text{ and} \\ &[(0,0),(0,1),(1,1),(1,2),(2,2),(3,2)] \end{aligned}$$

that link the origin to $(3, 2)$, containing two points of the form (j, j) for $j > 0$.

With $n(k, \ell; i) = 0$ for $i > \min(k, \ell)$, $n(k, 0; 0) = n(0, \ell; 0) = 1$ for all $k, \ell > 0$, the $n(k, \ell; i)$ for $k\ell \neq 0$ are given by the recursion

$$\begin{aligned} &n(k,\ell;i) \\ &= I(k = \ell)\,[n(k-1,\ell;i-1) + n(k,\ell-1;i-1)] \\ &+ I(k \neq \ell)\,[n(k-1,\ell;i) + n(k,\ell-1;i)]\,. \qquad (11) \end{aligned}$$

For $1 \leq i \leq k$, the numbers $T(k, i) = n(k, k; i)$ are known in the literature. They give the number of *Grand Dyck paths* from $(0, 0)$ to $(2k, 0)$ that meet the x-axis i times, which is a simple transformation of our definition. An expression for these numbers [9, Equation 6.22] is

$$T(k,i) = \frac{i2^i\binom{2k-i}{k}}{2k-i}. \qquad (12)$$

For $k \neq \ell$, the numbers $n(k, \ell; i)$ do not appear in the Encyclopedia of Integer Sequences [10], and we are not aware of a previous instance where they have been used. However, in a private communication, Trevor Welsh [11], produced an expression for $n(k, \ell; i)$ with $k \neq \ell$. He showed that, for $k > \ell$,

$$n(k,\ell;i) = n(\ell,k;i) = \frac{(k-\ell+i)2^i\binom{k+\ell-i}{k}}{k+\ell-i}, \qquad (13)$$

which generalises (12) in an elegant way.

With the numbers $n(k, \ell; i)$ in hand, we are in a position to write down the stationary distribution of the Markov chain.

Theorem 2.1: The stationary distribution of the Markov chain defined above has the form

$$\begin{aligned} &\pi(k,\ell) = \pi(0,0)\lambda_1^k\lambda_2^\ell \\ &\sum_{i=0}^{\min(k,\ell)} \frac{(|k-\ell|+i)2^i\binom{k+\ell-i}{k}}{(k+\ell-i)(\lambda_1+\lambda_2)^i(\lambda_1+\lambda_2+\mu)^{k+\ell-i}}, \qquad (14) \end{aligned}$$

where $\pi(0, 0)$ is determined by normalisation.

Proof: The result is established by using (11) to verify that (14) satisfies (8), (9) and (10). □

For the case where, $\lambda_1 = 0.6/hr$, $\lambda_2 = 5.4/hr$ (which corresponds to the pool having 10% of the processing power) and $\mu = 285/hr$, corresponding to Decker and Wattenhofer's [1] observed average communication delay of 12.6 seconds, the values of $\pi(k, \ell)$ for $k, \ell = 0, \ldots, 3$ are given in Table I.

We see that the pool and the community agree about the blockchain 97.5% of the time, the community has a block that the pool is yet to hear about for about 1.8% of the time, the

TABLE I. THE STATIONARY PROBABILITIES $\pi(k,\ell)$ FOR $k,\ell = 0,\ldots,3$, WHEN THE POOL MINES HONESTLY.

(k,ℓ)	0	1	2	3
0	0.9757	0.0181	0.0003	0.0000
1	0.0020	0.0037	0.0001	0.0000
2	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000

pool has a block that the community hasn't heard about for 0.2% of the time, while the pool and the community have versions of the blockchain with a single different final block about 0.4% of the time. All other possibilities have a stationary probability less than 10^{-3}, which supports the intuition that splits in the blockchain with branches of length greater than one occur with low probability.

Each time that the blockchain is in a state $(1,1)$ and a new block is mined, approximately one orphan block will be created. This is because the new state will become $(1,2)$ or $(2,1)$ and, with high-probability, no other state change will occur before the successful communication returns the state to $(0,0)$. The block on the shorter branch will then become an orphan block. With these parameter values, the rate of creation of orphan blocks is approximately $\pi(1,1)(\lambda_1+\lambda_2) = 0.022$ per hour, which translates to an average of about 0.53 per day.

Readers will note that this value is much less than the average number of orphan blocks that are observed each day in the real Bitcoin network, which lies between two and three. The discrepancy is explained by the fact that, in this simple model, we have assumed instantaneous communication within the pool and within the community. We have not counted orphan blocks caused by communication delays within the pool and within the community, which occur in the real network. However, we believe that the model still has interest because, as we shall see in Section II-B, it can be used to demonstrate that the rate of production of orphan blocks becomes much higher if the pool is using a block-hiding strategy such as selfish-mine.

B. The pool uses the selfish-mine strategy

Now we assume that the pool is using the selfish mine strategy described by Eyal and Sirer in [2]. As in the model of Section II-A, we assume that the pool discovers blocks at rate λ_1 and the community discovers blocks at rate λ_2, with $\lambda_1 < \lambda_2$, independently of the state.

Under the selfish-mine strategy, the pool does not necessarily publish blocks immediately it discovers them. Rather, it keeps them secret until it finds out that the community has discovered a block, and then publishes one or more of its blocks in response to this news. Most commonly, this will occur when the pool has a single block B_p that it has kept secret from the community and then it is notified that the community has discovered a block B_h. The pool's response to this news is immediately to publish B_p, hoping that some of the community will mine on it. Whether or not this happens, the pool will keep mining on its own version of the blockchain. The situation resolves itself when the next block is discovered, and the state becomes either $(2,1)$ or $(1,2)$, in which case, with high probability, the state will revert to $(0,0)$ once communication has taken place.

Since we have assumed that communication is instantaneous within the pool and community, but takes time from one to the other, Eyal and Sirer's parameter γ, the proportion of the honest community that mines on the pool's recently-released block when the state is $(1,1)$, is equal to zero. Thus, when the state is $(1,1)$, a new block will be created on the pool's leaf at rate λ_1 and on the community's leaf at rate λ_2.

If the pool has a lead that is greater than or equal to three (a rare occurrence), it does nothing until it is notified of the discovery of a block by the community. It then publishes its first block. However, since the pool and the community will still keep working on the blocks at the ends of their respective branches, this does not affect the state of the system, and therefore we put $q((k,\ell),(0,0)) = 0$ when $\ell \leq k-2$.

If the pool has a lead of exactly two and it is notified of the discovery of a block by the community, the system moves to state $(2,1)$ (or, indeed, the very unlikely states $(3,2)$, $(4,3)$ etc.), and then the pool will publish all its blocks. Once the communication of the final block has occurred, the rest of the community will start working on the longer pool branch, thus returning the state of the system to $(0,0)$. When it publishes blocks in this situation, the pool is 'cashing-in' on the lead that it has built up, rendering useless the work that the community has been doing on its branch. This behaviour is reflected in our Markov model by putting $q((k,k-1),(0,0)) = \mu$ when $k \geq 2$, where the time taken to communicate a block from the pool to the community and vice-versa is exponentially-distributed with parameter $\mu \gg \lambda_2$.

Finally, we have $q((k,\ell),(0,0)) = \mu$ when $k < \ell$, because the honest miners always publish blocks that they discover, and the pool has no choice but to build on the community's version of the blockchain if it is longer. As above and in Section II-A, we are taking into account only the communication that reports the discovery of the final block in the community's chain in assigning this transition rate.

Our model of the state of the blockchain when the pool is using the selfish-mine strategy has transition rates

$$q((k,\ell),(k+1,\ell)) = \lambda_1, \quad k \geq 0, \ell \geq 0, \tag{15}$$
$$q((k,\ell),(k,\ell+1)) = \lambda_2, \quad k \geq 0, \ell \geq 0, \tag{16}$$
$$q((k,\ell),(0,0)) = \mu, \quad k < \ell, \tag{17}$$
$$q((k,k-1),(0,0)) = \mu, \quad k \geq 2, \tag{18}$$
$$q((k,\ell),(k',\ell')) = 0, \quad \text{otherwise.} \tag{19}$$

The equations for the stationary distribution are

$$\pi(0,0)(\lambda_1+\lambda_2) = \sum_{k=0}^{\infty}\sum_{\ell=k+1}^{\infty} \pi(k,\ell)\mu + \sum_{k=2}^{\infty} \pi(k,k-1)\mu, \tag{20}$$

for $\ell > k$,

$$\pi(k,\ell)(\lambda_1+\lambda_2+\mu) = \pi(k-1,\ell)\lambda_1 I(k>0) + \pi(k,\ell-1)\lambda_2 I(\ell>0), \tag{21}$$

for $\ell = k$,

$$\pi(k,\ell)(\lambda_1+\lambda_2) = \pi(k-1,\ell)\lambda_1 I(k>0) + \pi(k,\ell-1)\lambda_2 I(\ell>0), \tag{22}$$

TABLE II. THE STATIONARY PROBABILITIES $\pi(k, \ell)$ FOR $k, \ell = 0, \ldots, 3$, WHEN THE POOL MINES SELFISHLY.

(k, ℓ)	0	1	2	3
0	0.8177	0.0121	0.0002	0.0000
1	0.0818	0.0749	0.0011	0.0000
2	0.0082	0.0002	0.0003	0.0000
3	0.0008	0.0008	0.0000	0.0000

for $\ell = k - 1$,

$$\begin{aligned} \pi(k, \ell)\,(\lambda_1 + \lambda_2 + \mu) &= \pi(k-1, \ell)\lambda_1 I(k > 0) \\ &+ \pi(k, \ell - 1)\lambda_2 I(\ell > 0) \quad (23) \end{aligned}$$

and, for $\ell < k$ otherwise,

$$\begin{aligned} \pi(k, \ell)\,(\lambda_1 + \lambda_2) &= \pi(k-1, \ell)\lambda_1 \\ &+ \pi(k, \ell - 1)\lambda_2 I(\ell > 0). \quad (24) \end{aligned}$$

Like the Markov chain in Section II-A, this Markov chain has countably-many states but, unlike the former chain, it does not appear to be possible to write down a simple closed-form expression similar to (14) for its stationary distribution. However, as we observed in respect of the model of Section II-A, the stationary probabilities decay very quickly to zero as k and ℓ increase, and we can get a good approximation by truncating the state space and augmenting the transition rates in a physically reasonable way so that the Markov chain remains irreducible. To get the results that we report below, we truncated the state space so that only states with $k + \ell \leq 6$ were considered and solved the resulting linear equations in Matlab. For the same parameters that we used in the model above, Table II contains the stationary probabilities for the subset of these states where $k, \ell \leq 3$.

We see now that the blockchain is in a state where the pool and the community agree for only 82% of the time. For about 8% of the time, the pool is working on a block that it has kept secret and, for another 7.5% of the time the pool and the community have separate branches of length one. As we observed in Section II-A, each time that the blockchain is in state $(1, 1)$ and a new block is mined, an orphan block will eventually be created. Also, each time the pool publishes a block in response to the community finding a block, a further orphan block is created. The conditions for the latter event occur with a probability of the order of 10^{-4}, and we therefore see that the rate of creation of orphan blocks if the pool is playing the selfish mine strategy is approximately $\pi(1, 1)(\lambda_1 + \lambda_2) = 0.4494$ per hour, which is about 10.8 per day.

Comparing with the similar calculation in Section II-A in which the same parameters λ_1, λ_2 and μ led to a rate of creation of orphan blocks of 0.5 per day, this illustrates that the increased rate of orphan block creation has the potential to be used as a diagnostic tool as to whether there is a pool of miners that have adopted the selfish-mine strategy. Specifically, the community can monitor whether a significant proportion of the miners is using any type of block-hiding strategy by looking for increases in the rate of production of orphan blocks. In particular, it would be possible to detect the presence of a pool of miners implementing the selfish-mine strategy in this way.

III. EYAL AND SIRER'S PARAMETER γ

In the model of Section II, we assumed that the pool and the community were remote from each other, so that communication within the pool and within the community could effectively be considered to be instantaneous, while communication between the pool and community incurred a delay. This is clearly unrealistic. Indeed, it is likely that the miners of the pool are distributed throughout the honest community and that there is delay in communication between any two miners, whether they are both in the pool or not.

To illustrate the type of approach that can be taken to model this situation, we shall make some assumptions about the spatial relationships and the communication delays between pool miners and miners in the honest community, and derive some insights about the behaviour of the blockchain. While the assumptions would need to be varied to reflect the characteristics of a mining pool in the actual Bitcoin network, we believe that the insights hold in general.

Specifically, we assume that the pool miners are distributed according to a spatial Poisson point process $\Psi = \{X_i\}$ with constant intensity $\nu > 0$ over the same region $\mathbb{R}^2$ that contains the honest miners, so Ψ can be considered a random set of pool miner locations $\{X_i\}$. The Poisson process is widely used for stochastic models of communication networks, for example, the positioning of transmitters [12]. Although we restrict ourselves to Euclidean space $\mathbb{R}^2$ for illustration purposes, Baccelli, Norros and Fabien [13] introduced a general framework using Poisson processes to study peer-to-peer networks, which was then later used by Baccelli *et al.* [14] to study the scalability of these networks. It has been remarked [13], [14] that the Poisson process in this model can be defined on other spaces more suitable for studying networks such as hyperbolic space [15], which offers a possible avenue for further research.

Furthermore, we assume that the communication delay between two Miners M_i and M_j, whether pool or honest, that lie a distance d_{ij} apart is normally distributed with a mean kd_{ij} proportional to this distance and a constant variance σ^2, independently of other transmission delays. This assumption does not contradict Decker and Wattenhofer [1] who modelled the *unconditional* communication delays with exponential random variables.

The quantity that we are interested in is Eyal and Sirer's [2] proportion γ of the honest community that mines on a block released by the selfish-mine pool in response to the honest community publishing a block. With reference to Figure 2, we are interested in analysing the communication between two honest Miners M_1 and M_2 that lie a distance d_{12} from each other. Miner M_3 is the pool miner for which the length of the path between M_1 and M_2 via M_3 is minimised. Denote the (random) distances between M_1 and M_3 and M_3 and M_2 by D_{13} and D_{32} respectively.

Consider the situation where the pool has discovered a block B_p that it has kept secret from the honest community and then honest Miner M_1 subsequently discovers and publishes a block B_h. The selfish-mine strategy dictates that Miner M_3 should release B_p immediately it receives B_h from M_1. We are interested in the probability that the other honest Miner M_2 will receive B_p before B_h because, with equal length branches, it will then mine on the branch that it heard about first.

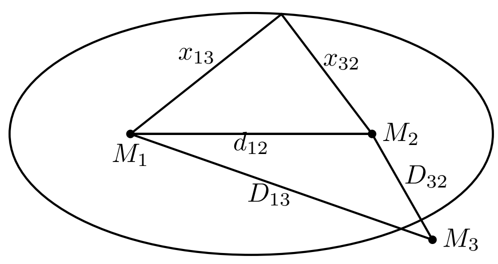

Fig. 2. $P(D > x)$ is the probability that no pool miner is located in the ellipse with $x_{13} + x_{32} = x$.

Making the further assumption that Miner M_3 requires no time to process the information that a block has arrived from M_1 and release B_p (which could be varied), γ is effectively the probability that communication from M_1 to M_3 and then M_3 to M_2 occurs faster than direct communication from M_1 to M_2.

Again with reference to Figure 2, Miner M_3 is chosen so that the distance $D = D_{13} + D_{32}$ is minimal amongst all of the pool miners. This means that, for any $x < D$, there is no pool miner in the ellipse whose foci are the locations of honest Miners M_1 and M_2 (taken to be at $(-d_{12}/2, 0)$ and $(d_{12}/2, 0)$ respectively) and semi-axes

$$a = \frac{x}{2}, \qquad b = \frac{1}{2}(x^2 - d_{12}^2)^{1/2}. \tag{25}$$

Hence

$$P(D > x) = e^{-\nu A(x)}, \quad x \geq d_{12}, \tag{26}$$

where

$$A(x) = \frac{\pi x}{4}(x^2 - d_{12}^2)^{1/2} \tag{27}$$

is the area of the ellipse (25). It follows that

$$F_D(x) \equiv P(D \leq x) = 1 - e^{-\nu A(x)}, \quad x \geq d_{12}. \tag{28}$$

Conditional on the random distances D_{13} and D_{32}, the transmission times T_{13} and T_{32} are independent and normally distributed with means kD_{13} and kD_{32} respectively and common variance σ^2, and therefore the difference $\Delta \equiv T_{13}+T_{32}-T_{12}$ is a normally distributed random variable with mean $k(D-d_{12})$ and variance $3\sigma^2$. Since the triangle inequality ensures that the mean of Δ, $k(D - d_{12})$ is nonnegative, we immediately see that $\widetilde{\gamma} = P(T < 0)$ is less than or equal to 0.5. Furthermore,

$$P(\Delta < 0|D = x) = \Phi\left(\frac{k(d_{12} - x)}{\sqrt{3}\sigma}\right), \tag{29}$$

where Φ is the distribution function of a standard normal random variable. Integrating with respect to the probability density of D derived from (28), we see that the probability that the honest Miner M_2 receives B_p before B_h is given by

$$\widetilde{\gamma} = \nu \int_{d_{12}}^{\infty} A'(x) e^{-\nu A(x)} \Phi\left(\frac{k(d_{12} - x)}{\sqrt{3}\sigma}\right) dx. \tag{30}$$

A change of variable $w = A(x)$ results in a numerically tractable Laplace transform

$$\widetilde{\gamma} = \nu \int_0^{\infty} e^{-\nu w} \Phi\left(\frac{k(d_{12} - A^{-1}(w))}{\sqrt{3}\sigma}\right) dw, \tag{31}$$

TABLE III. VALUES OF $\widetilde{\gamma}$ FOR DIFFERENT VALUES OF d_{12} AND ν.

(d_{12}, ν)	0.4	0.8	1.2	1.6
1	0.0341	0.0654	0.0942	0.1207
4	0.2034	0.3144	0.3779	0.4160
8	0.3687	0.4505	0.4758	0.4860
12	0.4430	0.4835	0.4925	0.4958

where

$$A^{-1}(w) = \frac{1}{\sqrt{2}}([(8w/\pi)^2 + d_{12}^4]^{1/2} + d_{12}^2)^{1/2}. \tag{32}$$

It is clear that $\widetilde{\gamma}$ depends on k and σ only through the ratio k/σ. Taking this ratio to be equal to 50, Table III presents some values of $\widetilde{\gamma}$ as the distance d_{12} between M_1 and M_2 and the density ν of pool miners are varied. We see that, as d_{12} increases, the value of $\widetilde{\gamma}$ approaches its theoretical limit of 0.5. The rate of convergence is faster if ν is larger, but the parameter $\widetilde{\gamma}$ is more sensitive to the distance d_{12} between honest Miners M_1 and M_2 than it is to the intensity of the Poisson process of pool miner locations. The intuition behind this is that, when d_{12} is large, there is a high probability that there will be a pool miner close to the straight line between Miners M_1 and M_2 even if the value of ν is only moderate.

This effect is illustrated in Figure 3, which presents an example where $d_{12} = 12$ and $\nu = 0.4$. Honest Miners M_1 $\odot$ and M_2 $\otimes$ are located at the points $(-6, 0)$ and $(6, 0)$ respectively. The round circles $\circ$ are the locations of pool miners, and the marked pool miner $\bullet$ is the pool Miner M_3, that minimises the distance $D_{13} + D_{32}$. Note that, even though the pool miners are not densely packed, M_3 lies very close to the straight line between M_1 and M_2.

Under the assumptions of the model, the above analysis calculates the probability that the pool miner M_3 closest to the straight line between honest Miners M_1 and M_2 succeeds in transmitting B_p to M_2 before M_2 directly receives B_h. Miner M_3 is the pool miner with the highest probability of succeeding in this transmission. However, there might be other pool miners that have a round-trip distance that is not much further than that via M_3, and a complete analysis should take into account the possibility that one of these miners succeeds

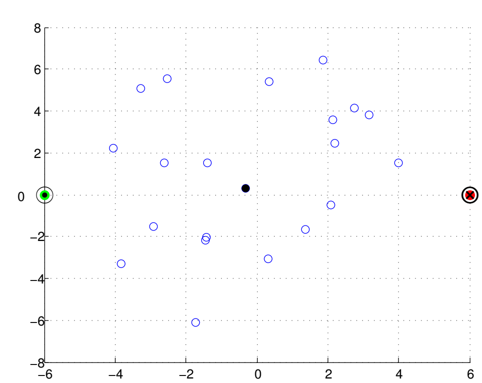

Fig. 3. An example simulation of the Poisson model with $d_{12} = 12$ and $\nu = 0.4$.

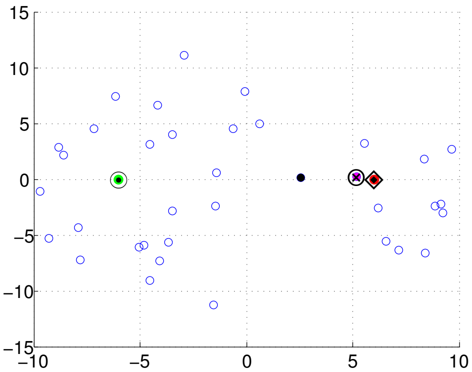

Fig. 4. A simulation where the pool node $\otimes$ with the shortest round trip time is not the pool node $\bullet$ that lies closest to the straight line between M_1 $\odot$ and M_2 $\diamond$; $d_{12} = 12$ and $\nu = 0.2$.

when M_3 does not. Such a situation is illustrated in Figure 4, where the Miner M_3 closest to the straight line between Miners M_1 and M_2 is not the miner that had the smallest value of the round-trip propagation delay.

More precisely, instead of calculating the probability that the communication time via the pool node M_3 that minimises the round-trip distance is less than the direct transmission time, we should calculate the probability that the minimum of the communication times via all the dishonest nodes is less than the direct transmission time. Based upon our assumptions that the dishonest nodes are distributed as a spatial Poisson process and that transmission delays are normally-distributed, the following result helps with this calculation.

Let $\rho(y)$ denote the distance from honest miner M_1 to miner M_2 via an intermediate node located at $y \in \mathbb{R}^2$. Then the distances

$$\{D_i\} = \{\rho(X_i) : X_i \in \Psi\}$$

from Miner M_1 to Miner M_2 via dishonest users $\{X_i\} = \Psi$ form a point process on the infinite interval $[d_{12}, \infty)$ and the following lemma is a consequence of the Mapping Theorem, see, for example, Kingman [16, page 17].

Lemma 3.1: The point process $\{D_i\}$ is an inhomogeneous Poisson point process with intensity (or mean) measure given by

$$\Lambda_D(x) := \Lambda_D([d_{12}, x]) = \nu A(x), \quad x \geq d_{12} \tag{33}$$

where $A(x)$ is given by (27).

We can make further use of the Mapping Theorem to obtain a lemma about the Poisson nature of the round trip times.

Lemma 3.2: The point process $\{T_i\}$ is an inhomogeneous Poisson point process on $(-\infty, \infty)$ with intensity measure given by

$$\Lambda_T(y) := \Lambda_T((-\infty, y]) = \nu \int_{d_{12}}^{\infty} A'(x) \Phi\left(\frac{y - kx}{\sqrt{2}\sigma}\right) dx, \tag{34}$$

where $A(x)$ is given by (27).

TABLE IV. VALUES OF γ FOR DIFFERENT VALUES OF d_{12} AND ν.

(d_{12}, ν)	0.4	0.8	1.2	1.6
1	0.0347	0.0678	0.0992	0.1292
4	0.2298	0.3914	0.5081	0.5946
8	0.4891	0.6937	0.7955	0.8530
12	0.6695	0.8372	0.9018	0.9336

Proof: We can write $T_i = kD_i + E_i$ where the sequence $\{E_i\}$ consists of i.i.d. $N(0, 2\sigma^2)$ random variables, independent of the sequence $\{D_i\}$, where, in the theory of marked point process, each E_i is referred to as a random mark. By the Marking Theorem [16, page 55], the two-dimensional process (D_i, E_i) is also a Poisson process, with intensity measure on rectangles of the form $(a, b] \times (-\infty, y]$ given by

$$\Lambda_{D,E}(a, b, y) = \nu \Phi\left(\frac{y}{\sqrt{2}\sigma}\right) \int_a^b A'(x) dx,$$

and the Poisson nature of the process $\{T_i\}$ follows again from the Mapping Theorem [16, page 17]. To get the expression (34) for the intensity measure of $\{T_i\}$, we condition on the possible value of D_i that leads to a given value of T_i. □

The probability γ that the pool block released by M_3 will reach M_2 before the block published by M_1 is the probability that there exists a point of the Poisson process $\{T_i\}$ less than the direct transmission time T_{12}. This latter time is normally distributed with mean kd_{12} and variance σ^2. If such a point exists, then there will be at least one pool node where the round-trip time is shorter than the direct time.

Conditional on $T_{12} = t_{12}$, we can use Lemma 3.2 to write the probability of the above event as

$$P(\min T_i \leq T_{12} | T_{12} = t_{12}) = 1 - \exp(-\Lambda_T(t_{12})), \tag{35}$$

where Λ_T is given by (34). This expression can also be derived by considering $\min T_i$ as extremal shot-noise, see, for example, Baccelli and Błaszczyszyn [12, Proposition 2.13].

Now, integrating with respect to the density of T_{12}, the unconditional probability that there is a point of the round trip process which is less than the direct transmission time is

$$\begin{aligned} \gamma &= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} (1 - \exp(-\Lambda_T(u))) \exp\left(\frac{-(u - kd_{12})^2}{2\sigma^2}\right) du \\ &= 1 - \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} \exp\left(\frac{-(u - kd_{12})^2}{2\sigma^2} - \Lambda_T(u)\right) du. \end{aligned} \tag{36}$$

For the same values of d_{12} and ν that were used in Table III, again with $k/\sigma = 50$, Table IV gives the values of γ calculated via (36).

We notice first that the values of γ in Table IV are all higher than than the values of $\widetilde{\gamma}$ depicted in Table III, reflecting the fact that pool nodes other than the pool node that is closest to the straight line between M_1 and M_2 might lie on the path that minimises the round-trip delay. Furthermore, we see that the values of γ are more sensitive to the density ν of the pool nodes than the values of $\widetilde{\gamma}$. This makes sense because, when the density of pool nodes is high, there are likely to be more pool nodes, other than the one that minimises the round-trip distance between M_1 and M_2, that have short round-trip times. Finally, we note that when distance d_{12} between nodes M_1 and M_2 is high, and the density ν of pool nodes is also high, the probability γ can be arbitrarily high, for example exceeding

0.9 when $d_{12} = 12$ and $\nu = 1.6$, even though the probability $\widetilde{\gamma}$ cannot be greater than 0.5.

The overall lesson from the analysis in this section is that, with randomly-varying communication delays, it is advantageous for the pool to maximise the number of nodes that release a secret block in response to a block being mined by the honest community. This maximises the probability of at least one of them succeeding in transmitting its released block to the other honest nodes before they receive the direct communication from M_1.

In fact, a similar observation can also be applied to the honest community itself. Rather than relying on the direct communication between M_1 and M_2 to occur faster than round-trip communication via the pool nodes, the honest community could also employ intermediate nodes as relays and there would be a good chance that faster communication would be achieved via one of these. Analysing such a situation using the techniques of this section is an interesting question for future research.

IV. Blockchain simulation experiments

We developed two blockchain simulators, one in C++ and one in Java, the latter based on the DESMO-J simulation framework [17]. We used the former to simulate a network of 1,000 nodes. The simulation worked as follows.

- The positions of the nodes were selected uniformly at random on the set $[0, 1000] \times [0, 1000]$.
- Blocks were mined at randomly-selected nodes at the instants of a Poisson process. On average, one block was mined every 10 minutes.
- Each node maintained a local copy of the blockchain.
- The communication delay between two nodes was a random variable sampled from a normal distribution whose mean was proportional to the Euclidean distance between the two nodes and whose coefficient of variation CV was kept constant. Note that this differed from the delay model described in Section III, where we assumed that the normally distributed communication delay had a constant variance σ^2. In the model discussed in this section, the variance increases with the distance between the nodes.
- A total of 10,000 blocks were mined. This represents 70 days of mining.
- Each simulation experiment was replicated 12 times and 95% confidence intervals for all performance measures that we shall discuss below were computed.
- The simulation results are generally presented below in the form of plots. The plotted points are sample means. Confidence interval half widths are shown if they are distinguishable, otherwise they are omitted. The plotted points are connected by continuous curves constructed from segments of cubic polynomials whose coefficients are found by weighting the data points.

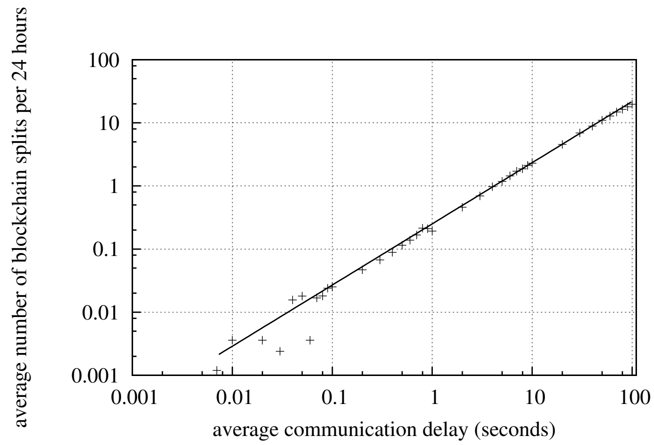

Fig. 5. The average number of blockchain splits per 24 hours.

A. Honest mining

Figure 5 shows the average number $b(t)$ of blockchain splits per 24 hours as a function of the communication delay t, averaged over all the nodes in the network. The delay was varied from 1 msec to 100 seconds. Both axes are logarithmic. Fitting a straight line to the log-log plot yields $b(t) = 0.2508t^{0.9695}$ so that the average split rate was almost linearly proportional to the average communication delay.

The simulation experiments showed that when the expected communication delay was 10 seconds, on average 2.34 splits were observed per 24 hours. This is roughly in agreement with the observations made by Decker and Wattenhofer [1] that an average communication delay of 12.6 seconds results in an average split rate of 2.4 per 24 hours in the actual Bitcoin network.

Suppose there is a (hypothetical) mechanism that is invoked when a block is attached to a blockchain. The mechanism can simultaneously inspect the blockchains at all the nodes and report if each blockchain has a single leaf and if all the blockchains are identical: if this condition occurs then the blockchains are said to be *synchronised*.

Consider an instant of time t_0 when the mechanism reports that the blockchains are synchronised. Let $t > t_0$ denote the first time instant after t_0 when the mechanism reports that

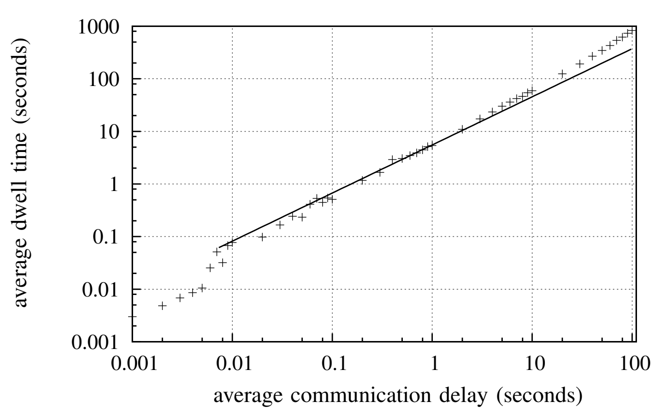

Fig. 6. The average dwell time.

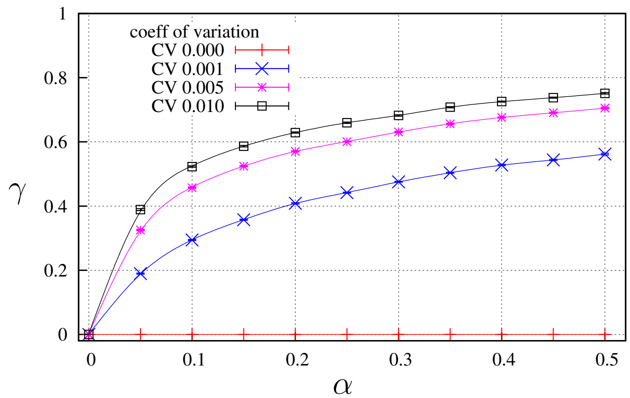

Fig. 7. The ratio γ.

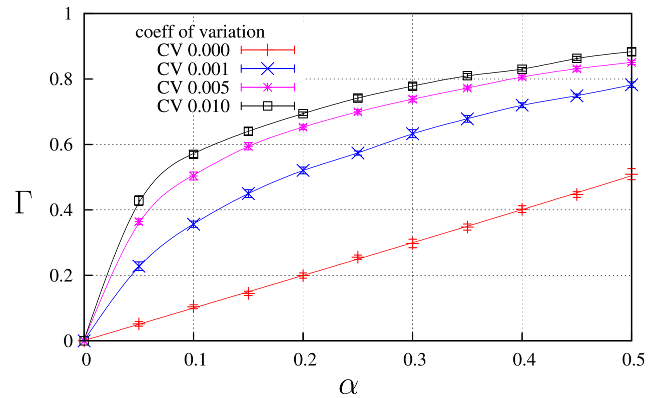

Fig. 8. The ratio Γ.

the blockchains are not synchronised. Let $t' > t$ denote the first time instant after t when the mechanism reports that the blockchains are again synchronised. We shall refer to the interval $t' - t$ as the *dwell time*.

Figure 6 plots the average dwell time as a function of the average communication delay. Again, the axes are logarithmic. The figure shows that the dwell time was also almost linearly proportional to the average communication delay.

As the average communication delay increased, the number of splits increased and the time until the splits were resolved and the blockchains were synchronised increased. The average dwell time exceeded 10 minutes (the average time between mining events) when the average communication delay was of the order of 100 seconds.

B. Dishonest mining

In remainder of this section we shall report the application of our simulator to the situation where a pool of miners used Eyal and Sirer's selfish-mine approach [2]. The details of our implementation of the selfish-mine algorithm are given in the Appendix.

As in Section I-E, we use α to denote the fraction of the total computing capacity of the network that is controlled by the dishonest pool, and γ to denote the probability that an honest miner will mine on the block B_p, rather than B_h.

When the communication delays are zero, according to Eyal and Sirer's expression (3), the minimum proportion of computing power required for profitable selfish mining ranges from $\alpha > 0$ (if $\gamma = 1$) to $\alpha > 1/3$ (if $\gamma = 0$).

We simulated the communication delays between miners in the network as independent normal random variables whose mean was proportional to distance between the miners and whose coefficient of variation CV was kept constant. If $CV = 0$ then, by the triangle inequality, we would have expected B_h to reach M_2 before B_p reaches M_2, unless the three nodes M_1, M_2 and M_3 are collinear, which is an event of probability zero. This expectation was confirmed in the simulation.

However, if $CV > 0$ then B_p can arrive at M_2 before B_h, and so γ will be positive. We used our simulator to investigate the value of γ as the number of pool miners was varied from from 0 to 500, and thus the proportion α of pool computing power was varied from 0 to 0.5. Figure 7 presents the observed proportion $\widehat{\gamma}$ as a function of α for several values of the coefficient of variation CV. The figure confirms our expectation that, when $CV > 0$ and there are dishonest miners present, then $\widehat{\gamma}$ ought to be positive.

Furthermore, the value of $\widehat{\gamma}$ increased quickly as a function of α even when CV was taken to be quite small. This reinforces the insight that we gained in Section III that, because there were many possibilities for the intermediate pool node, the probability of a communication path via one of them beating the direct communication was unexpectedly high.

The fact that an honest miner M_i is mining on a block B_p revealed by the dishonest pool does not guarantee that the next block to be attached to the blockchain C_i at node M_i will be linked to block B_p. If C_i has two leaves B_p and B_h and a block B_{new} arrives from another node, then B_{new} can attach to B_p or to B_h.

Let Γ denote the probability that the next block attached to the blockchain at an honest node was linked to block B_p. Figure 8 shows that the sample means of Γ indicated by the points $(+, \times, *, \square)$ corresponded closely with the theoretical value $\Gamma = \alpha + (1 - \alpha)\gamma$ given by the continuous curves.

C. The relative pool revenue

Let N_h denote the total number of blocks mined by the honest miners that were included in the blockchain at the end of the experiment. The revenue earned from these blocks has been credited to the honest miners. Let N_p denote the total number of blocks mined by the pool that were finally included in the blockchain. Define the relative pool revenue $R = N_p/(N_h + N_p)$.

Figure 9 presents a map of the relative pool revenue R as a function of the total number of miners (varied from 100 to 1,000) and the pool size as a fraction α of the total number, with the average communication delay fixed at 10 seconds. The figure demonstrates that the relative pool revenue was roughly constant with respect to the number of nodes, and increased with increasing values of α. Significantly, R became greater than 0.5 when α reached 0.4, which indicates that the pool was earning more than its fair share of revenue in this region.

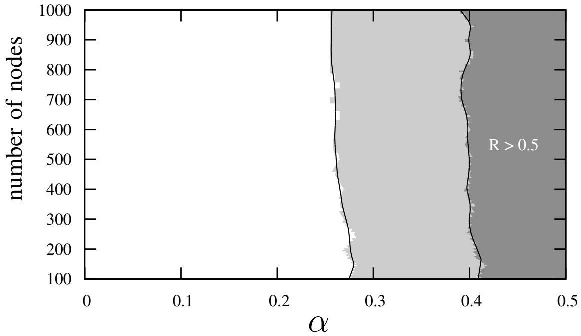

Fig. 9. The relative pool revenue R in networks of increasing size.

D. Detecting the presence of dishonest miners

In this section and the following sections, we follow the theme of Section II, and discuss how the honest miners can detect the presence of a pool of miners implementing the selfish-mine strategy. Consider a network of 1,000 miners, with an average communication delay of 10 seconds and a coefficient of variation $CV = 0.001$.

Figure 10 presents the average number of blockchain splits per 24 hours as a function of the relative size α of the dishonest pool. As the size of the dishonest pool increased, the average number of splits per unit time increased by an order of magnitude. Thus the simulation has confirmed the conclusion of the model that we investigated in Section II that an increase in the split rate can provide a means for the honest miners to detect the presence of the dishonest miners.

In a network of 1,000 nodes, assuming that all miners have the same computational power, each miner expects to earn an average of $25 \times 6/1000 = 0.15$ bitcoins per hour. Figure 11 shows that as the number of dishonest miners increased, the honest miners earned less than the expected average of 0.15 bitcoins per hour. This may also afford a means for the honest miners to detect the presence of a pool of miners implementing the selfish-mine strategy.

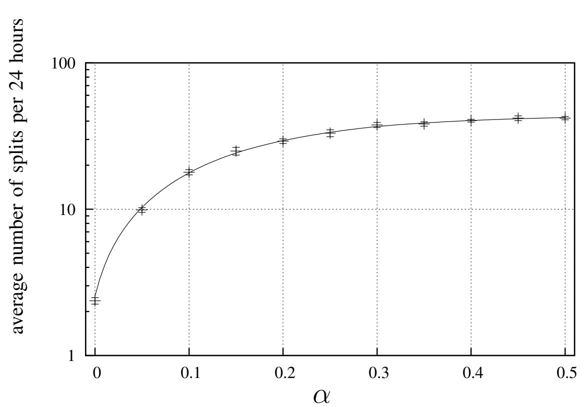

Fig. 10. The average number of blockchain splits per 24 hours.

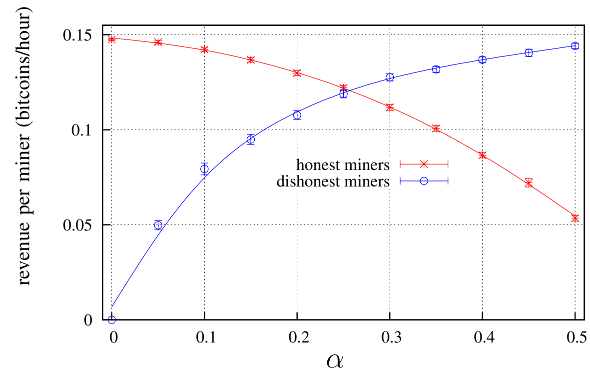

Fig. 11. The average confirmed revenue earned per hour per miner.

E. Dishonest mining is not profitable

Figure 12 presents the relative pool revenue R as a function of the relative size α of the dishonest pool. The figure shows that for $\alpha > 0.25$ dishonest mining outperformed honest mining. However, this does not imply that the pool incorporated more blocks into the main branch than it would have if the dishonest miners had followed the bitcoin rules.

Figure 13 illustrates this by exhibiting the performance of both the dishonest pool and the honest miners in terms of the numbers of blocks they mined that end up in the main branch. It presents the average number of blocks mined per hour by the pool, by the honest miners and in total, that were included in the long-term blockchain as a function of the relative size α of the dishonest pool. The average block mining rate was held constant at 6 blocks per hour.

The figure demonstrates that, when there is a pool implementing selfish-mine, both the pool and the honest miners were worse off than they would have been if no dishonest mining was present. The total number of blocks that the pool and honest nodes incorporated into the main branch when dishonest mining was present was always less than the number that would have been incorporated if dishonest mining were not present.

We caution that the above observation is made under the assumption that the difficulty of the cryptographic problem described in Section I-A was held constant. In the real Bitcoin

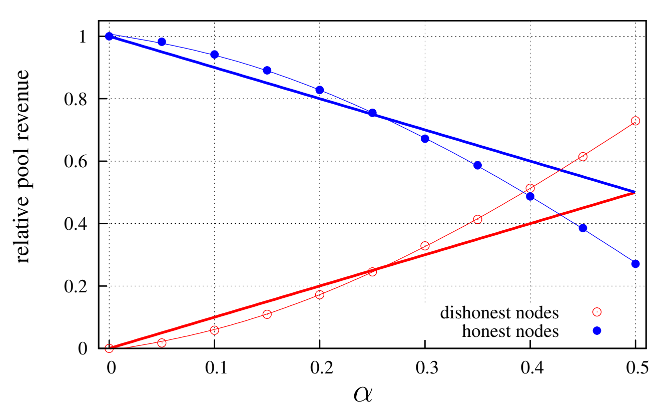

Fig. 12. The relative pool revenue R and $\widehat{R}$.

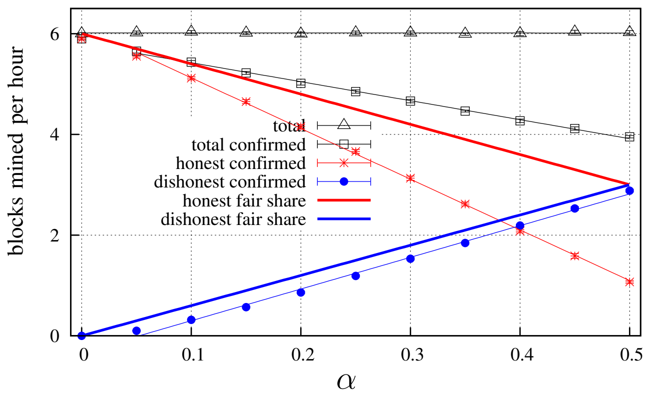

Fig. 13. The average block mining rate.

blockchain, the network would respond to an overall decrease in the rate of blocks being successfully mined by reducing the difficulty of the cryptographic problem. This decreased value of the difficulty may itself afford a means for the honest miners to detect the presence of the dishonest miners.

F. Adoption threshold

Figure 12 shows that in the range $0 < \alpha \leq 0.25$ there is no incentive for a solo miner to adopt the selfish-mine strategy, since by doing so a miner will be become part of a pool that has a lower relative pool revenue than it would have if all the members of the pool were honest. Moreover, in the range $0 < \alpha \leq 0.25$, solo honest miners benefit (in terms of their relative pool revenue $\widehat{R} = 1 - R$) from the activities of the dishonest pool. A larger participant may already possess more than 25% of the network mining capacity and may be able to attract miners with a promise of an enhanced pool revenue. However, as shown in Section IV-E, these miners will earn less than they would have earned had they remained honest, and if they perceive this they may withdraw from the dishonest pool.

V. Conclusions

In this paper we have studied the dynamics of the Bitcoin blockchain when propagation delays are taken into account, with specific reference to how the blockchain behaves when there is a pool of miners using the selfish-mine strategy proposed by Eyal and Sirer [2]. Our approach has been to construct simple models that provide insight into system behaviour, without attempting to reflect the detailed structure of the Bitcoin network.

In Section II we used a simple Markov chain model to demonstrate that it is possible for the whole mining community to detect block-hiding behaviour, such as that used in selfish-mine, by monitoring the rate of production of orphan blocks.

In Section III, our attention turned to Eyal and Sirer's parameter γ, which is the proportion of the honest community that mine on a previously-secret block released by the selfish-mine pool in response to the honest community mining a block. When there is no variability in the propagation delay, it follows from the triangle equality (at least within the Poisson network model that we assumed) that the value of γ is zero. However, the value of γ can increase surprisingly quickly with increasing variability of propagation delay. A key observation is that if all pool nodes release the secret block as soon as they are notified of the discovery of the public block, the chances of one of them beating the direct communication can be very high. We did not study the counter-balancing effect that would occur if all honest miners relayed the honest miner's block in the same way. A study of this would be an interesting topic for future research.

Finally, in Section IV, we used simulation to verify the observations that we made in Sections II and III, under slightly different assumptions. We also were able to study the long-term rate of block production, and hence revenue generation under both honest mining and selfish-mine strategies and make some observations about when selfish-mine is profitable. An important observation is that, in the absence of a relaxation of the difficulty of the mining cryptographic problem, the long-term rate of block production will decrease if a pool of miners is implementing the selfish-mine strategy. It can thus happen that, even if the selfish pool is earning a greater proportion of the total revenue than is indicated by its share of the total computational power, it is, in fact, earning revenue at a lesser rate than would be the case if it simply followed the protocol. This observation makes intuitive sense, since the whole point of selfish-mine is to put other miners in a position where they are wasting resources on mining blocks that have no chance of being included in the long-term blockchain.

We emphasize that our models, both analytic and simulation, are idealised. It would be an interesting line for future research to use network tomography techniques to discover the topology of the actual Bitcoin network and then employ the analytical and simulation techniques that we have discussed in this paper to study the effect of propagation delay on the dynamic evolution of the blockchain.

Acknowledgment

The authors would like to thank Maria Remerova for some valuable comments that led to an improvement of the paper.

References

[1] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *13th IEEE Conference on Peer-to-Peer Computing*, 2013, pp. 1–10.

[2] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer Berlin, Heidelberg, 2013, pp. 436–454.

[3] [Online]. Available: https://en.bitcoin.it/wiki/Protocol_rules

[4] [Online]. Available: https://blockchain.info/charts/n-orphaned-blocks

[5] S. Nakamoto. (2009) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: http://bitcoins.info/bitcoin-a-peer-to-peer-electroniccash-system-satoshi-nakamoto/

[6] M. Rosenfeld. (2014) Analysis of hashrate-based double-spending. [Online]. Available: http://arxiv.org/abs/1402.2009

[7] N. Courtois and L. Bahack. (2014) On subversive miner strategies and block withholding attack in bitcoin digital currency. [Online]. Available: arXiv:1402.1718v5

[8] L. Bahack. (2013) Theoretical bitcoin attacks with less than half of the computational power. [Online]. Available: eprint.iacr.org/2013/868

[9] E. Deutsch, "Dyck path enumeration," *Discrete Mathematics*, vol. 204, pp. 167–202, 1999.

[10] The on-line encyclopedia of integer sequences. [Online]. Available: https://oeis.org/

[11] T. Welsh, private communication.

[12] F. Baccelli and B. Błaszczyszyn, *Stochastic geometry and wireless networks: Volume 1: Theory*. Now Publishers Inc, 2009, vol. vol. 1.

[13] F. Baccelli, I. Norros, and F. Mathieu, "Performance of p2p networks with spatial interactions of peers." [Online]. Available: http://hal.inria.fr/inria-00615523v2

[14] F. Baccelli, F. Mathieu, I. Norros, and R. Varloot, "Can p2p networks be super-scalable?" in *INFOCOM*. IEEE, 2013, pp. 1753–1761.

[15] M. Boguná, F. Papadopoulos, and D. Krioukov, "Sustaining the internet with hyperbolic mapping," *Nature Communications*, vol. 1, p. 62, 2010.

[16] J. Kingman, *Poisson Processes*. Oxford University Press, 1993.

[17] B. Page and W. Kreutzer, *The Java Simulation Handbook – Simulating Discrete Event Systems with UML and Java*. Aachen: Shaker, 2005.

APPENDIX

The pseudo-code presented in Algorithm 1 summarises the actions of a dishonest node.

Algorithm 1 Selfish-mine algorithm at a dishonest node i.

```
// Initialise the blockchain at dishonest node i.
   function INITIALISE
       blockchain := publicly known blocks
       secretExtension := empty; race := FALSE
       mine on the last block in the blockchain
   end function

// Dishonest node i attaches a secret Block to its secretExtension.
   function SECRETMINE(block Block)
       append Block to secretExtension; n_s := n_s + 1
       if race then
           publish Block; race := FALSE; secretExtension := empty
       else if |secretExtension| > 5 then        // prevent runaway
           publish the first unpublished block of secretExtension
       end if
       mine on Block
   end function

// Dishonest node i attaches a public/published Block to its blockchain.
// The last block on blockchain has serial number n_p.
// The last block on secretExtension has serial number n_s.
   function PUBLICMINE(block Block)
       append Block to blockchain; n_p := n_p + 1
       Δ := n_s − n_p        // compute the lead
       if Δ = −1 then
           if race then
               race := FALSE; secretExtension := empty
           end if
           mine on Block
       else if Δ = 0 then
           race := TRUE
           publish the last (and only) block of secretExtension
           secretExtension := empty; mine on block n_s
       else if Δ = 1 then
           if |secretExtension| = 2 then
               publish secretExtension; secretExtension := empty
               mine on block n_s
           end if
       else       // Δ > 1
           publish the first unpublished block of secretExtension
           mine on block n_s
       end if
   end function
```