Digital Communications and Networks 6 (2020) 157–166

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

The effect of propagation delay on the dynamic evolution of the Bitcoin blockchain

Moustapha BA *

MODAL'X (Laboratoire de Modélisation aléatoire) of Université Paris Nanterre (Paris X), MODAL'X is the Mathematics and Computer Sciences laboratory of Université Paris Nanterre (Paris X), ministerial label EA 3454, UFR SEGMI, département Mathématiques-Informatique, 200 avenue de la République, 92000, Nanterre, France

ARTICLE INFO

MSC:
60B05
60G55
90B15
90B18

Keywords:
Blockchain
Bitcoin
Mining
Selfish-mine attack
Propagation delay of information

ABSTRACT

This paper analyzes the selfish-mine strategy in the Bitcoin blockchain introduced in 2013 by I. Eyal and E. G. Sirer. This strategy could be used by a colluding pool of miners to earn more than their fair share of the mining revenue and in consequence to force other honest miners to join them to decrease the variance of their revenues and make their monthly revenues more predictable. It is a very dangerous dynamic that could allow the rogue pool of miners to go toward a majority by accumulating powers of news adherents and control the entire network. Considering that the propagation delay of information between any two miners in the network, which is not negligible and follows a normal distribution with mean proportional to the physical distance between the two miners, and a constant variance independent of others' delays, we prove that no guarantee can be given about the success or failure of the selfish-mine attack because of the variability of information propagation in the network.

1. 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 to 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 Bob and, if you can decrypt Bob's message using his public key, then you have confirmed that the message was encrypted by using Bob's private key, and therefore the message indisputably came from Bob. But how can you verify that Bob has sufficient bitcoins to pay you? The Bitcoin system solves this problem by verifying transactions in a coded form in a data structure called blockchain, which is maintained and secured by a community of participants, known as miners. It happens that different miners have different versions of the blockchain, which is caused in general by the propagation delay of information in the network, as proved by Decker and Wattenhofer in Ref. [2]. But as described by Satoshi in his original white paper, only one branch is considered by the community of miners as the honest branch containing the validated blocks, and the other blocks are considered as orphan blocks. The "official" chain is called a public chain or public branch, the one containing the validated blocks, also the longest chain. It can be seen simply by the longest path from any block to the genesis block.

In the Introduction, we briefly describe the mining dynamics in the Bitcoin network, which is our main focus of interest. In section 2, we recall the theory of the selfish-mine strategy widely detailed in Ref. [1], and circumstances that can occur when the propagation delay is taking place in the network. In section 4, we give the Poisson model that follows the pools in the network, then announce two main results and proofs. We conclude our work before discussing the subject and open problems at the end of the paper.

- *Recall: Blockchain and its mining dynamics*

First, we formalize a model that captures the essentials of Bitcoin mining behaviors and introduces a notation for relevant system parameters. The system is comprised of a set of miners or pool-miners $\boldsymbol{M} = \{M_1,$

* Corresponding author.
E-mail address: ba.moustapha@parisnanterre.fr.

https://doi.org/10.1016/j.dcan.2019.01.006
Received 14 February 2018; Received in revised form 21 January 2019; Accepted 22 January 2019
Available online 29 January 2019

$M_2,\ldots,M_n\}$. Each miner M_i has mining power α_i. Since the probability of solving the cryptographic puzzle is proportional to the hash power, we can assume that $\sum_{i=1}^{n}\alpha_i = 1$. Each miner chooses a chain head to mine, and finds a subsequent block for that head after a time interval that is exponentially distributed with mean $(\alpha_i)^{-1}$. Suppose miner M_i is on 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 compute a solution s_i to the cryptographic puzzle with nonce n_i. Miner M_i will add block B_i to blockchain C and broadcast (B_i, n_i, h_i, s_i) to the network. When another miner M_j, who is also working on blockchain C, receives the communication, it will compute:

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

If $s' = s_i$, then miner M_j will add block B_i to its blockchain C, abandon the block B_j that it has been working on, and start trying to add a block to the chain CB_i. Any transactions in B_j that are not in B_i and verified by M_j will be incorporated into this new block. Importantly, miners M_i and M_j now have identical versions of the blockchain.

The propagation delay in actual Bitcoin is defined as the difference between the time that a node announces the discovery of a new block or a transaction and the time that other nodes receive the information for a period of operation (cf. [2]). The existence of such a delay, which is not negligible, can upset the above process, because blocks can be discovered while communication and validation are in process. Decker and Wattenhofer in Ref. [2] observed that the median time for a node to receive a block was 6.5 s, the mean was 12.6 s and the 95th percentile of the distribution was around 40 s. Moreover, they showed that an exponential distribution provided 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 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 miner M_j has mined block B_j at time $t' \in [t, t+\delta_j]$. Miner M_j now knows about two versions CB_i and CB_j of the blockchain, which are of the same length. From the point of view of miner M_j, the blockchain has been 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 knows about first. However, miner M_j knows about CB_j first and then attempts to build on this version of the blockchain. Other miners will work on either CB_i or CB_j, depending on which version they hear 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^*. Then it is likely that block B_j will not be part of the long term blockchain, and it will become an orphan block. Any verified 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 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 s on average for the community to mine a block shows that we should expect that the probability that a new block is discovered while the communication and validation of a block discovery is taking place is of the order of 12.6/600, 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.

Based on what has been described above, some miners can modify their implementation in order to delay the transmission of the mined block and attack the network. This strategy is called a hidden block strategy or selfish-mine strategy.

2. Previous work: the selfish-mine strategy and its outcome

The Bitcoin network miners are rational; that is, they try to maximize their revenue and may deviate from the protocol to do so. A group of miners can form a *pool* that behaves as a single agent with a centralized coordinator. In this case, the mining power of the pool is the sum of the mining power of its members, and its revenue is divided among its members according to their relative mining power (see Ref. [3]). The expected relative revenue, or simply the revenue of a pool is the expected fraction of blocks that have been mined by that pool out of the total number of blocks on the longest chain. All members of a pool work together to mine each block and share their revenues when one of them successfully finds a block. It is known in actual Bitcoin that, a single home miner using a dedicated ASIC is unlikely to mine a block for years [4]. Then to maximize their revenues, individual miners join the great pools voluntarily. While joining a pool does not change a miner's expected revenue, it decreases the variance and makes the monthly revenues more predictable.

As introduced in Ref. [1], a mining strategy is called selfish-mine strategy (hidden blocks strategy) if, simply, a miner or pool miners find a block and hide it secretly without publishing it automatically in the network as indicated by the protocol, intentionally bifurcating the chain. As proved in Ref. [1] and later to be shown in Section 4, it allows a pool of sufficient size to obtain a revenue larger than its ratio of mining power. When all miners follow the Bitcoin protocol, a pool or single miner's share of the payoffs is equal to the fraction of computational power that it controls (out of the computational resources of the entire network). Then clearly, the selfish-mine strategy is a mining strategy that enables pool miners or a single miner to adopt it to earn revenues in excess of their mining power. Higher revenues can lead new miners to join a selfish miner pool, a dangerous dynamic that enables the selfish mining pool to grow into a majority (in terms of hash power).

The key insight behind the selfish mining strategy is to force the honest miners into performing wasted computations on the stale public branch. Specifically, selfish mining forces the honest miners to spend their cycles on blocks that are destined not to be part of the blockchain. Selfish miners achieve this goal by selectively revealing their mined blocks to invalidate the honest miners' work. Approximately speaking, the selfish mining pool keeps its mined blocks private, secretly bifurcating the blockchain and creating a private branch. Meanwhile, the honest miners continue mining on the shorter public branch. Because the selfish miners command a relatively small portion of the total mining power, their private branch will not remain ahead of the public branch indefinitely. Consequently, the selfish miners judiciously reveal blocks from the private branch to the public, such that some honest miners will switch to recently revealed blocks, abandoning the shorter public branch. This renders their previous effort spent on the shorter public branch wasted and enables the selfish pool to collect higher revenues by incorporating a higher fraction of its blocks into the blockchain.

With this description, we can fully specify the selfish mining strategy shown in Algorithm 1 in Ref. [1]. The strategy is driven by mining events by the selfish pool or by the others. Its decisions depend only on the relative length of the selfish pool's private branch versus the public branch. It is best to illustrate the operation of the selfish mining strategy by going through sample scenarios involving different public and private chain lengths.

When the public branch is longer than the private ones, the selfish mining pool is behind the public branch. Because of the power difference between the selfish miners and the others, chances for the selfish pools to mine on their own private branch and overtake the main branch are small. Consequently, the selfish miner pool simply adopts the main branch whenever its private branch falls behind. As others find new blocks and publish them, the pool updates and mines at the current public head. When the selfish miner pool finds a block, it is in an advantageous position with a single block leading on the public branch on which the honest miners operate. Instead of naively publishing this private block and notifying the rest of the miners of the newly discovered block, the selfish miners keep this block private to the pool. There are two possible outcomes at this point: either the honest miners discover a new

block on the public branch, nullifying the pool's lead, or else the pool mines a second block and extends its lead on the honest miners. In the first scenario where the honest nodes succeed in finding a block on the public branch and nullifying the selfish pool's lead, the selfish pool immediately publishes its private branch (of length one). This yields a toss-up where either branch may win. The selfish miners unanimously adopt and extend the previously private branch, while the honest miners will choose to mine on either branch, depending on the notifications. If the selfish pool manages to mine a subsequent block ahead of the honest miners that have not adopted the pool's recently revealed block, it publishes immediately to enjoy the revenue of both the first and second blocks of its branch. If the honest miners mine a block after the pool's revealed block, the pool enjoys the revenue of its block, while the others get the revenue from their block. Finally, if the honest miners mine a block after their own block, they enjoy the revenue of their two blocks while the pool gets nothing. In the second scenario, where the selfish pool succeeds in finding a second block, it develops a comfortable lead of two blocks that provides it with some cushion against discoveries by the honest miners. Once the pool reaches this point, it continues to mine at the head of its private branch. It publishes one block from its private branch for every block the others find. Since the selfish pool is a minority, its lead will, with high probability, eventually reduce to a single block. At this point, the pool publishes its private branch. Since the private branch is longer than the public branch by one block, it is adopted by all miners as the main branch, and the pool enjoys the revenue of all its blocks. This brings the system back to a state where there is just a single branch until the pool bifurcates it again.

In brief, selfish-mining works as follows: when a miner belonging to the pool 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. The honest miners are unaware of the blocks in the secret extension and continue to mine and publish their mined blocks and solutions according to the standard protocol. The risk to the pool, by keeping secret their blocks mined, is that, if it has established a lead of exactly one by mining a block B_p (pool's block), which it has kept secret, and then it is informed that the community has mined a block B_h (honest's block), the pool may end up not getting credit for the block B_p. In order to minimize this risk, the selfish-mine strategy dictates that the pool should publish the block B_p immediately it hears about B_h. In this state, there are two public branches of the same length. Miners in the selfish pool all mine on the pool's branch, because a subsequent block discovery on this branch will yield a reward for the pool. The honest miners, following the standard Bitcoin protocol implementation, mine on the branch they have heard at first. 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. We denote by γ the ratio of honest miners that choose to mine on the pool's block, and the other $1-\gamma$ of the non-pool miners mining on the other branch.

With this analysis, the authors of [1], I. Eyal and E. G. Sirer, have proved that a pool with hash power α can obtain revenue larger than its relative size, provided that

$$\frac{1-\gamma}{3-2\gamma} \leq \alpha \leq \frac{1}{2}, \tag{1}$$

without taking into account the block's propagation delay in the network. The authors of [1] have considered a model with only two groups of miners: a colluding pool with hash power α and an honest community obviously with a hash power $1-\alpha$. More precisely, without taking in account the propagation delay of information in the network, the selfish-mine strategy is profitable (in the sense that the pool earns more than its fair share) if its hash power α satisfies equation (1). The simple observation is: when communication delays are zero, that is, when one assumes that the communication time between any two miners (including colluding pools) is negligible, as they assume in Ref. [1], according to Eyal and Sirer's expression (see formula (1)), the minimum proportion of computing power required (the threshold) for profitable selfish mining ranges from $0 < \alpha \leq \frac{1}{2}$ (if $\gamma = 1$) to $\frac{1}{3} < \alpha \leq \frac{1}{2}$(if $\gamma = 0$). They also propose a simple backward-compatible progressive modification to the Bitcoin protocol that would raise the threshold from zero to $\frac{1}{4}$ (Section 6 of [1]). Remark that paper [1] has only dealt the revenues earned by the attacker adopting the selfish-mine strategy and proposes an alternative protocol in order to increase the computational hash rate's threshold required for selfish-mining to be profitable. Other authors [1] do not give any quantitative value of the probability of success or of failure to the selfish-mine attack. They focused on the revenue earned by the selfish-mine pool and did not take into account the effect of propagation delay of information in the network.

Their results are significant and give alerts to users of Bitcoin, compared with the Nakamoto's prevision about the security of Bitcoin's network in Ref. [5]: the hash power α must be at least $\frac{1}{2}$ of the total hash power of the network. In other words, the threshold proved by Nakamoto to make a successful attack in the network is $\frac{1}{2}$. In the selfish-mine strategy, the honest community has a head start in propagating B_h before the dishonest miners have heard about it and then there is a further propagation delay before B_p reaches other honest miners. The natural intuition is that γ is likely to be very low in the presence of propagation delays.

3. Main assumptions, motivation and contributions

3.1. Assumptions

In this paper, we consider the selfish-mining algorithm in [1] and propose some simple models, in which we explicitly take into account the propagation delay between any two miners unless they belong to the same pool as a random normal distribution, which we can use to compare the behavior of the Bitcoin network when all miners are observing the standard protocol with its behavior when there are several pools following the selfish-mine strategy. We call those pools in this paper *SM-pool*. We think that the hypotheses are realistic and the dishonest pools of miners are distributed all over the honest community and that there is delay communication between any two miners (unless they belong to the same pool). This model fits perfectly with the current Bitcoin protocol.

With this assumption, we will prove that γ defined in the last paragraph of Section 2 can be surprisingly high, depending on the density of pool miners in the Bitcoin network.

3.2. Motivation

What motivates this work, namely, to study the effect of propagation delays on selfish-mine attack in the Bitcoin network, is based first on the fact that the propagation delay is the first cause of blockchain fork, as conjectured in Ref. [2]. Secondly, we believe that a blockchain's fork increases chances for an SM-pool to succeed in a selfish-mine attack in view of the above description (see Section 2. Even if in Ref. [1] the minimum threshold ($\gamma = 0$) required for the selfish-mine strategy to be profitable is $\frac{1}{3}$ (eq. (1)) without the assumption of delay, we believe that the success or failure of such an attack depends on the variance of the propagation delay and the density of colluding pools in the network. In other words, we will prove that the proportion of honest miners that choose to mine on the private block released by the SM-pool can be surprisingly high, depending on the density of pool miners in the Bitcoin network and also on the variance of the propagation delay. Or if the proportion of honest miners who choose to mine in the private branch increases, the threshold for the success of selfish mining is significantly reduced (see formula (1).

A miner is considered to start a selfish-mine as soon as it finds a block and keeps it secretly. Then, to fix ideas, we consider that the pool miner

finds a block he keeps secretly. Because the SM-pool is the minority (in terms of relative hash power), it has little probability to solve the mathematical problem at first and to mine a subsequent block ahead of the honest miners. Its advantage is then to wait for the publication of honest miners, and respond to it by releasing automatically its block, hoping to reach other honest miners as soon as possible. What we call this in this paper is doing a *good fork*. This strategy is appropriate only if there exists a non-zero propagation delay of information in the network. Its algorithm shows that creating a *"good fork"* is the best way for a pool with a low proportion of hash power to succeed in defending the selfish-mine attack and to take control of the public chain. Paper [1] that is our main source of inspiration did not deal with the hypothesis of a random delay. Even if in paper [2], the authors have verified that the delay is exponentially distributed, we think that our hypothesis (normal distribution) is more realistic and does not contradict those of [2]. The normal distribution and the exponential distribution are strongly linked by their density functions. Another motivation for this study is to prevent double-spending. Indeed, we will show in Section 6 that, the selfish-mine attack is one of the most flexible strategies for a pool miner to achieve double-spending. As a consequence, evaluating the chances for an SM-pool in the Bitcoin network that has adopted the selfish-mine strategy to succeed in controlling the public branch remains for us a very interesting work about the security of the Bitcoin users.

3.3. Contributions

Our first contribution goes to the proportion of honest miners who can naively join the private branches after the SM-pools release their private blocks in response to the honest block published by the honest community. We have considered a complete network with several honest miners and several SM-pools distributed independently at random around the honest miners according to a Poisson point process with a constant intensity. We notice that this proportion increases in function of the density of SM-pools (Theorem 2). The second contribution is to evaluate, in the case of *"good fork"*, the probability that one hided branch succeeds to become a public one, which, naturally, depends on the total dishonest hash power α in the network, the density of SM-pool in the network, λ, and the variance of propagation delay of information in the network. One dangerous consequence of the success of the selfish-mine attack is the possibility of a double-spending as detailed in Ref. [3]. We will discuss this problem in section 6. If one SM-pool takes control of the public branch, it takes control of the entire network, since only the public branch contains the validated blocks. In this case, Bitcoin would not be a decentralized payment system. The system will depend on the SM-pool who will validate or not make transactions. Indeed, if a set of colluding miners come to command a majority of the mining power in the network, the currency stops being decentralized and becomes controlled by the colluding group. Such a group can, for example, prohibit certain transactions, or all of them. The consequence is, for example, the possibility of double-spending, which we will relate in Section 6.

4. Model, results and proofs of results

The SM-pool are distributed independently at random according to a spatial Poisson point process $\mathscr{D} = \{X_i\}$ with a constant intensity $\lambda > 0$ over the same region $\mathscr{D} \subseteq \mathbb{R}^2$ that contains all miners of the rest of the Bitcoin community, so that $\mathscr{D}$ can be considered as a random set of SM-pool locations $\{X_i\}$. This means that each SM-pool chooses location dx in $\mathscr{D}$ independent of others, with probability distribution $\lambda(dx) = \lambda dx$, where dx is the Lebesgue measure on $\mathbb{R}^2$. The Poisson point process can be seen as a countably-finite collection of points in the space $\mathbb{R}^2$, without accumulation points, a discrete subset of $\mathbb{R}^2$. By defining the measure $\mathscr{N}(A) = card\{X_i \in A\}$, the Poisson process $\mathscr{N}$ can be seen as a stochastic process $\{\mathscr{N}(A)\}_{A\in\mathscr{B}(\mathbb{R}^2)}$ with state-space $\{0, 1, 2, \ldots\} \ni \mathscr{D}(A)$, and where the index A runs over bounded Borel subsets of $\mathbb{R}^2$. Since $\mathscr{N}$ is a Poisson point process with intensity λ, $\{\mathscr{N}(A)\}_{A\in\mathscr{B}}$ is a Poisson point process with intensity $\lambda|A|$, where $|A|$ is the area of A. For all bounded Borel subset A, $\mathscr{N}(A)$ follows a Poisson distribution with intensity $\lambda|A|$. The Poisson process is widely used for stochastic models of communication networks. For the best understanding, we suggest readers to learn [6,7].

Since $\mathscr{N}(A)$ is a Poisson distribution with intensity $\lambda|A|$, the following remark is obvious.

Remark 1. The probability for any bounded subset of $\mathbb{R}^2$, A, to contain no SM-pool is:

$$\mathbb{P}(\mathscr{N}(A) = 0) = e^{-\lambda|A|} \tag{2}$$

where $|A|$ is the area of A. The set of honest miners is called $\mathscr{H} = \{M_1, M_2, \ldots, M_m\}$, $m < \infty$. We assume that M_1 has found a block B_h and automatically broadcasted this information to the whole Bitcoin community, either honest miners or dishonest miners called SM-pool, as dictated by Bitcoin's protocol. To study the dynamic evolution of the Bitcoin network when all miners are observing the standard protocol and its behavior when there are several pools following the selfish-mine strategy, we assume that there is a communication delay between any two miners or pools of miners, M_i and M_j, for all $i \neq j$ in the network, whether colluding pool or honest pool, and that lying a distance d_{ij} apart is normally distributed with a mean kd_{ij} proportional to this distance and a constant variance σ^2 independent of other transmission delays. This assumption does not contradict Decker and Wattenhofer in Ref. [2], who have modeled the unconditional communication delays with exponential random variables.

We assume that miner M_1 has found a block honestly. This means, M_1 has resolved its mathematical problem with the last public hash known by the majority and broadcasted its solution (as we have described in Section 1.1), and its result is verified by other miners as soon as they receive the solution. The aim of SM-pools is to beat direct communication between the honest miner called M_1 and other honest miners in the set $\mathscr{H}$. Automatically releasing their secret blocks $B_p^1, B_p^2, \ldots, B_p^l$ respectively as soon as they receive the information from M_1, each one of them hopes to beat the direct communication between M_1 and other honest miners. Since the communication delay in the network is not null in reality, we expect that luckily one proportion of honest miners will mine on these private blocks. Even if the authors of [1] have not evaluated this proportion and only have studied the reward earned by mining dishonestly, we keep the same notation as in Ref. [1]. The parameter γ in equation (1) above is the proportion of honest miners that choose to mine on the secret blocks. $\{B_p^i\}$ is released by the SM-pools. The first aim in this paper is to evaluate this proportion in the presence of propagation delay as a function of the distance between honest miners and the density of the colluding pools in the network.

We denote for all j,

$$\{D_i^j\} = \{\rho(X_i), X_i \in \mathscr{D}\} \tag{3}$$

the distance from miner M_1 to miner M_j via dishonest SM-pools $\{X_i\}$. This process forms a point process on the infinite interval $[d_{1j}; \infty[$, where $d_{1,j}$, for all j, is the deterministic direct distance between M_1 and M_j. More importantly, which SM-pool will get information from M_1 at first? One would think that the SM-pool who will have access to the information from M_1 at the earliest time is the one who is closer to M_1 in terms of distance. In other words, the SM-pool whose distance is minimal among all of the SM-pools will get information from M_1 at first. We denote this SM-pool by X_{i_0}. However, there might be other SM-pools who have a round-trip distance that is not much further than that via X_{i_0}. To do a complete analysis, we take into account the fact that one of them (others than X_{i_0}) can beat the direct communication between honest miners M_1 and M_j for all j.

So there exists a random time of information propagation between M_1

and SM-pool X_i and a random time of information propagation between X_i and M_j. Let us call by T_i^j the sum of these two random times. This time is the random propagation delay between M_1 and M_j through X_i, where X_i is the supposed SM-pool who has intercepted B_h^1 at first before it reaches M_j. Indeed, as dictated by the selfish-mine protocol, as soon as an SM-pool miner X_i receives B_h^1 from M_1, it diffuses immediately B_p^i (private block) and then hopes that M_j receives B_p^i before receiving B_h^1. The miner M_j will mine with the hash as dictated by the Bitcoin's protocol. This is correct because when M_j hears the message from X_i at first, it mines with the hash of the block B_p^i and ignores all the following messages from the others, either honest or dishonest miners. The information diffused by miner M_1 can be intercepted by several SM-pools, but the one who has the best chance of beating direct communication is the one which minimizes the round trip time. More precisely, instead of calculating the probability that the communication time via the pool node X_i, which minimizes the round-trip distance to be less than the direct transmission time, we calculate the probability that the minimum of the communication time via all the dishonest SM-pools is less than the direct transmission time because there might be other SM-pools who have a round-trip distance that is not much further than via the straight line between honest miners M_1 and M_j, and that would be able to transmit the information to M_j faster than others. Based upon our assumptions that the dishonest nodes are distributed as a spatial Poisson process, and that transmission delays are normally-distributed, the Eyal and Sirer [1] parameter we have defined above, γ, is exactly the probability for the time when there exists one honest miner M_j such that the minimum random round-trip delay $\min_i T_i^j$ is less than the direct transmission between M_1 and M_j, called T_{1j}. In other words, we get

$$\gamma = \mathbb{P}\Big(\bigcup_{j\in\mathscr{J}}\{\min_i T_i^j < T_{1j}\}\Big) \tag{4}$$

In order to maximize its chance to reach many honest miners, each SM-pool miner is to maximize the number of nodes that will release the secret block in response to the block being mined by the honest community. That is why we naturally assume that all nodes belonging to the SM-pool release at the same time the secrete block in the sense that, as soon as the pool's coordinator receives the honest block published in the network, and it is assumed that the communication between miners belonging to the same pool is negligible.

Now, let us evaluate the quantity in equation (4) in the following theorem.

Theorem 2. The proportion (γ) of honest miners that choose to mine naively with the previous hash released by SM-pools in response to the block found by the community of honest miners is:

$$\gamma = 1 - \frac{1}{\sigma\sqrt{2\pi}}\sum_{j\in\mathscr{J}}\int_{-\infty}^{+\infty} e^{\left(\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2} - \Lambda_{T^j}(u)\right)} du \tag{5}$$

Proof. First we remark that the events

$$\{\min_i T_i^j < T_{1j}\}_{j\in\mathscr{J}}$$

are independent. And then, we can rewrite formula (4) as

$$\sum_{j\in\mathscr{J}} P\{\min_i T_i^j < T_{1j}\}$$

It suffices now to evaluate each term. For convenience, we suggest the readers to see Figure (1) below.

The round-trip time T_i^j is a normal random variable with mean proportional to the random round-trip distance $k.D_i^j$. If the SM-pool X_i satisfies the round-trip time, then D_i^j is the distance between M_1 and X_i added to the distance between X_i and M_j ($\rho(X_i)$). It is easy to see that for all j, $D_i^j > x$ if and only if X_i is not located on the Ellipse with focus M_1 and M_j denoted by E_{1j}. In view of Remark (2), the random variable D_i^j satisfies:

$$\begin{aligned} F_{D_i^j}(x) &= \mathbb{P}(D_i^j > x) \\ &= \mathbb{P}(X_i \notin E_{1j}) \\ &= 1 - \mathbb{P}(X_i \in E_{1j}) \\ &= 1 - \mathbb{P}\big(\#\{X_i : X_i \in E_{1j}\} > 0\big) \\ &= 1 - \mathbb{P}\big(Poisson(\lambda A_j(x)) > 0\big) \\ &= \mathbb{P}\big(Poisson(\lambda A_j(x)) \le 0\big) \\ &= \mathbb{P}\big(Poisson(\lambda A_j(x)) = 0\big) \\ &= e^{-\lambda A_j(x)} \quad \forall x \ge d_{1j} \end{aligned} \tag{6}$$

where $A_j(x)$ is the area of the Ellipse with focus M_1 and M_j called here E_{ij}. Recall that

$$A_j(x) = \frac{\pi x}{4}\left(x^2 - d_{1j}^2\right)^{\frac{1}{2}} \tag{7}$$

By using the mapping theorem ([8], p.26) for all fixed j, $\{D_i^j\}_i$ is an inhomogeneous point process with intensity or (mean) measure

$$\Lambda_{D^j}(x) := \Lambda_{D^j}\big([d_{1j}, x]\big) = \lambda A_j(x) \quad x \ge d_{1j} \tag{8}$$

where λ is the constant intensity of the pool's Poisson point process $\{X_i\}$. For all fixed j, $\{T_i^j\}_i$ is also an inhomogeneous point process with intensity measure

$$\begin{aligned} &\Lambda_{T^j}(u) := \Lambda_{T^j}(]-\infty, u]) \\ &= \int_{d_{1j}}^{+\infty} \lambda A'_j(x) F_{N(0,1)}\left(\frac{u - kx}{\sqrt{2}\sigma}\right) dx \\ &= \int_0^{+\infty} \lambda F_{N(0,1)}\left(\frac{u - kA_j^{-1}(\omega)}{\sqrt{2}\sigma}\right) d\omega \end{aligned} \tag{9}$$

where

$$A_j^{-1}(\omega) = \frac{1}{\sqrt{2}}\left(\left[\left(\frac{8\omega}{\pi}\right)^2 + d_{1j}^4\right]^{\frac{1}{2}} + d_{1j}^2\right)^{\frac{1}{2}} \quad \omega \ge 0$$

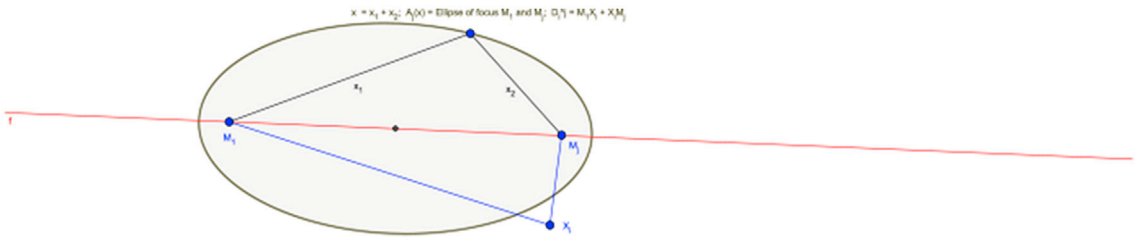

Fig. 1. $\mathbb{P}(D_i^j > x)$ *is the probability that no SM-pool miner located in the ellipse of focus* M_1 *and* M_j*, with* $x = x_1 + x_2$

Table 1
Values of as function of d12(1st col) and small values of _(1st line).

NaN	0.0050	0.0100	0.0150	0.0200	0.0250	0.0300
1.0000	0.3129	0.5303	0.6794	0. 7815	0.8512	0.8987
2.0000	0.3111	0.5277	0.6768	0. 7791	0.8492	0.8971
3.0000	0.3087	0.5245	0.6735	0. 7761	0.8465	0.8949
4.0000	0.3062	0.5211	0.6700	0. 7728	0.8438	0.8926
5.0000	0.3014	0.5177	0.6665	0. 7697	0.8410	0.8904
6.0000	0.3016	0.5146	0.6633	0. 7667	0.8385	0.8882

Table 2
Values of as function of d12(1st col) and large values of _(1st line).

NaN	0.4000	0.8000	1.2000	1.6000	2.0000	2.4000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

where $F_{N(0,1)}(.)$ is a distribution function of a centered and reduced normal random variable, and $d_{1,j}$ is the deterministic direct distance between M_1 and M_j for all fixed j. The Proof of formula (9) is given by formula (8), the theory of the marked point process, and the two-dimensional process, all combined to the mapping theorem (see [8], page 17). We deduce easily by formula (9) that, for all fixed j, $\forall u \geq 0$,

$$P\left(\left\{\min_i T_i^j < T_{1j|T_{1j}=u}\right\}\right) = 1 - e^{-\Lambda_{T^j}(u)} \tag{10}$$

By integrating this quantity between $-\infty$ and $+\infty$ with respect to the density of the random variable T_{1j} (the density of the normal distribution), we get

$$\gamma = 1 - \frac{1}{\sigma\sqrt{2\pi}} \sum_{j \in \mathcal{J}} \int_{-\infty}^{+\infty} e^{\left(\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2} - \Lambda_{T^j}(u)\right)} du \tag{11}$$

Interpretation of the result: In the following tables, we take in the preceding formula, $m = 2$ (meaning $\mathcal{J} = \{2\}$) and then a model of two honest miners and one SM-pool. We simply take the constant $k = 1$, and we give some values of γ with respect to some arbitrary values of d_{12}, λ, σ. Under the simple hypotheses, the elements which allow us to compute the following tables are :

$$\left\{\Lambda_{T^2}(u) = \int_0^{+\infty} \lambda F_{N(0,1)}\left(\frac{u - kA_2^{-1}(\omega)}{\sqrt{2}\sigma}\right) d\omega, \right\}$$

$$\left\{\gamma = 1 - \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{\left(\frac{-\left(u-kd_{12}\right)^2}{2\sigma^2} - \Lambda_{T^2}(u)\right)} du\right\}$$

Let's see the tables generated by MATLAB in the next pages.

We observe first that: for all fixed distance d_{12}, γ increases with respect to the intensity of the SM-pool's process, says λ; and for all fixed distance σ, γ increases with respect to the intensity of the SM-pool's process, says λ. This makes sense because when the density of the process described by SM-pools is high, luckily one of them should beat the direct communication (Tables 1, 2 and 5, 6).

Another observation is that, the values of γ increase with respect to the variance (Tables 3–5, 6). This can be explained by the fact that the smaller the variance is, the smaller the difference between two propagation delays is, and then the less the SM-pool is likely to reach other miners before M_1. And thus, the less the proportion (γ) is.

Table 3
Values of as function of d12(1st col) and small values of _(1st line).

NaN	0.0025	0.0075	0.0125	0.0175	0.0225	0.0275
1.0000	0.6675	0.7646	0.8026	0.8234	0.8420	0.8502
2.0000	0.6451	0.7549	0.7912	0.8130	0.8325	0.8411
3.0000	0.6276	0.7376	0.7791	0.8019	0.8223	0.8314
4.0000	0.6132	0.7256	0.7683	0.7918	0.8109	0.8224
5.0000	0.6074	0.7198	0.7594	0.7835	0.8029	0.8147
6.0000	0.5948	0.7124	0.7525	0.7768	0.7986	0.8913

Table 4
Values of as function of d12(1st col) and large values of _(1st line).

NaN	0.2500	0.5000	0. 7500	1.0000	1.2500	1.5000
1.0000	0.9948	0.9982	0.9992	0.9996	0.9997	0.9998
2.0000	0.9944	0.9980	0.9991	0.9995	0.9997	0.9998
3.0000	0.9940	0.9979	0.9991	0.9995	0.9996	0.9997
4.0000	0.9936	0.9977	0.9990	0.9995	0.9996	0.9997
5.0000	0.9932	0.9976	0.9989	0.9994	0.9996	0.9997
6.0000	0.9928	0.9974	0.9989	0.9994	0.9996	0.9997

Table 5
Values of as function of _(1st col) and small values of _(1st line).

NaN	0.0025	0.0075	0.0125	0.0175	0.0225	0.02 75
0.1000	0.6675	0. 7646	0.8026	0.8234	0.8420	0.8502
0.2000	0.8907	0.9457	0.9618	0.9695	0.9754	0.9932
0.3000	0.9642	0.9880	0.9926	0.9948	0.9962	0.9995
0.4000	0.9883	0.9973	0.9986	0.9991	0.9994	1.0000
0.5000	0.9962	0.9994	0.9997	0.9998	0.9999	1.0000
0.6000	0.9988	0.9999	0.9999	1.0000	1.0000	1.0000

Table 6
Values of as function of _(1st col) and large values of _(1st line).

NaN	0.2500	0.5000	0. 7500	1.0000	1.2500	1.5000
0.1000	0.9948	0.9982	0.9992	0.9996	0.9997	0.9998
0.2000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.3000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.4000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.5000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.6000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

The first conclusion we make about the study is that the given threshold in equation (1) for selfish-mine strategy to be profitable can be less than $\frac{1}{3}$, depending on the variance of the propagation delay of information and the presence of colluding pool (the density of colluding pools).

The result above shows how the values of γ can be studied in function to the density of SM-pools λ, the delay's variance σ, and the distance between honest miners.

In the following section, we consider a scenario where several hided blocks created by several SM-pools located in different zones in the world are waiting for information from the network. As soon as the diffusion of information from honest miners in the network takes place and SM-pools receive it, the selfish-mine strategy dictates that the SM-pools automatically release their hided blocks, and it is known that the probability of occuring a fork is not negligible. If the proportion γ calculated in the last theorem is equal to 1, then the private branch newly released becomes public and the SM-pool enjoys the revenues of this mining block. By definition, a fork appears when a block is found in the public branch while another block is being propagated. Note that, obviously, only the uninformed nodes may produce a conflicting block and the long propagation delay in the network is the primary cause of blockchain forks as verified in paper [2]. The authors of [2] also derive that, by using the probability of finding a block in the network and the ratio of nodes that are uninformed, the probability of a blockchain fork is about 1.78% under the hypothesis that the delay is exponentially distributed. To better

understand the blockchain's fork, we suggest readers to learn [2]. When a fork appears, all SM-pools build on their respective blockchain heads. Eventually, one branch will be longer than the other branches, and the partitions that have not adopted this branch as theirs will switch over to this branch. An advantage of the good SM-pool is: all miners belonging to the pool unanimously adopt and extend its previously private branch, while the honest miners will choose to mine on either branch, depending on the propagation of the notifications. By consequence, the moment when the dishonest miner has a branch of length, one constitutes a risk for the network. If the SM-pool manages to mine a subsequent block ahead of the honest miners who have not adopted the pool's recently revealed block, it publishes immediately to enjoy the revenue of both the first and the second blocks of its branch. If the honest miners mine a block after the pool's revealed block, the pool enjoys the revenue of its block, while the others get the revenue from their block. Finally, if the honest miners mine a block after their own block, they enjoy the revenue of their two blocks while the pool gets nothing. It may happen that the pool gains nothing. The blockchain fork is resolved and the ledger replicas are consistent up to the blockchain head. The blocks discarded by the blockchain resolution are referred to as orphan blocks because Bitcoin's protocol requires that the public blockchain is defined as the longest path from the last block to the genesis block.

We assume that there exist several SM-pools that have created several hided blocks with the penultimate block of the public branch. The following definition will be used in the rest of this paper.

Definition 3. A private branch is called a good private branch if it is linked with the penultimate block of the public branch. And we call by good SM-pools those who get good private branches.

In order to evaluate the chance that the public branch is overthrown by one dishonest private branch, we consider l good SM-pools in the network, $l < \infty$ obviously, and with the sum of hash power equal to α. We denote by M_1 the miner who has diffused the head of the public blockchain. Let $\mathcal{J} = \{2, 3, \ldots, m\}$ be the set indexing the honest miners waiting for the information from miner M_1. The public branch is reversed by one private branch, in the sense that one private block is extended if and only if more than half of the total mining power mines on this block, meaning half of the total mining power receives the hash released by the colluding miner at first. In other words, if and only if there exists one good SM-pool, X_i such that :

$$\sum_{j\in\mathcal{J}:\, T_i^j<T_{1j}} \alpha_j > \frac{1-\alpha}{2}$$

where T_i^j is the round-trip time between M_1 and M_j via X_i, and T_{1j} is the direct transmission time between M_1 and M_j. If such an event exists, then the last block released by the SM-pool X_i is verified by the majority of honest miners in the network. In the following theorem, we evaluate the probability for this event to occur. In other words, we will look for what would happen as soon as the SM-pool diffuses its hided branch.

Theorem 4. Let us consider the Bitcoin network where there exist a finite number of honest miners that follow the protocol with total hash power $1-\alpha$, and several good SM-pools in the network as defined in Definition 3 with total hash power α, each one hiding a good private branch (doing a good fork). Then the probability that one private branch succeeds in becoming the public one is the following quantity denoted by $\delta(\alpha)$:

$$\delta(\alpha) = P\left(\bigcup_{i=1}^{l}\left\{\sum_{j\in\mathcal{J}:\, T_i^j<T_{1j}} \alpha_j > \frac{1-\alpha}{2}\right\}\right)$$

$$= l.\left(1 - F_{N(0,1)}\left(\left(\frac{1-\alpha}{2} - E\right)\left(\frac{1}{\sqrt{V}}\right)\right)\right)$$

where $F_{N(0,1)}(.)$ is the standard normal distribution function and the quantities E and V are exactly :

$$E = \sum_{j\in\mathcal{J}} \alpha_j p_j$$

$$V = \sum_{j\in\mathcal{J}} \alpha_j^2 p_j (1-p_j)$$

Where

$$p_j = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \lambda e^{-\lambda\omega} F_{A_j^{-1}(\omega)}(u) d\omega e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} du$$

$$A_j^{-1}(\omega) = \frac{1}{\sqrt{2}} \left(\left[\left(\frac{8\omega}{\pi}\right)^2 + d_{1j}^4 \right]^{\frac{1}{2}} + d_{1j}^2 \right)^{\frac{1}{2}}$$

given by setting $A(x) = \omega$ in equation (7). $F_{A_j^{-1}(\omega)}(.)$ is the normal distribution function with mean $kA_j^{-1}(\omega)$ and variance σ^2.

The Proof of this theorem is given in the following pages.

Proof. For all Borel set A, $1_{\{A\}}$ means the indicator function of A. Recall that T_i^j means the round-trip time between M_1 and M_j over the SM-pool X_i. Then, $\{T_i^j\}_{j\in\mathcal{J}}$ are independent for all fixed i, the random Bernoulli variables $\{\alpha_j 1_{\{T_i^j \le T_{1j}\}}\}_{j\in\mathcal{J}}$ are also independent with parameters $\{\alpha_j p_i^j\}_{j\in\mathcal{J}}$. Each p_i^j can be given by

$$p_i^j = \mathbb{P}\left(T_i^j < T_{1j}\right)$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} \mathbb{P}\left(T_i^j < T_{1j}|_{T_{1j}=u}\right) e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} du$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} F_{T_i^j}(u) e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} du$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \frac{1}{2\sigma\sqrt{\pi}} \int_{-\infty}^{+\infty} \int_{-\infty}^{u} e^{\frac{-\left(y-kD_i^j\right)^2}{4\sigma^2}} e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} dy du$$

$$p_i^j = p^j = \frac{1}{2\sqrt{2}\sigma^2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{u} \int_{d_{1j}}^{+\infty} \lambda A'_j(x) e^{-\lambda A_j(x)} e^{\frac{-(y-kx)^2}{4\sigma^2}} e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} dx dy du$$

$$= \frac{1}{2\sqrt{2}\sigma^2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{u} \int_{0}^{+\infty} \lambda e^{-\lambda\omega} e^{\frac{-\left(y-kA_j^{-1}(\omega)\right)^2}{4\sigma^2}} e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} d\omega\, dy\, du$$

$$= \frac{1}{2\sqrt{2}\sigma^2\pi} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \lambda e^{-\lambda\omega} \int_{-\infty}^{u} e^{\frac{-\left(y-kA_j^{-1}(\omega)\right)^2}{4\sigma^2}} dy d\omega e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} du$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \lambda e^{-\lambda\omega} F_{A_j^{-1}(\omega)}(u) d\omega e^{\frac{-\left(u-kd_{1j}\right)^2}{2\sigma^2}} du$$

where $F_{A_j^{-1}(\omega)}(.)$ is the distribution function of the standard normal random variable with mean $kA_j^{-1}(\omega)$ and variance $2\sigma^2$. The expectation and the variance of the random variable $Y_i = \sum_{j\in\mathcal{J}} \alpha_j 1_{\{T_i^j \le T_{1j}\}}$ are: $E = \sum_{j\in\mathcal{J}} \alpha_j p^j$ and $V = \sum_{j\in\mathcal{J}} \alpha_j^2 p^j (1-p^j)$ for all i. Remark that $\{Y_i\}$ are

independent and identically distributed. Since the random normal variables $\{T_{1j}\}_{j\in\mathcal{J}}$ are also independent, then the quantity $\delta(\alpha)$ of Theorem 4 is :

$$\mathbb{P}\left(\sum_{j\in\mathcal{J}:\ T_i^j<T_{1j}}\alpha_j>\frac{1-\alpha}{2}\right)=$$

$$\mathbb{P}\left(\sum_{j\in\mathcal{J}}\alpha_j 1_{\{T_i^j<T_{1j}\}}>\frac{1-\alpha}{2}\right)$$

$$=\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty}\mathbb{P}\left(\sum_{j\in\mathcal{J}}\alpha_j 1_{\{T_i^j<T_{1j}|T_{1j}=u_j\}}>\frac{1-\alpha}{2}\right)$$

$$f_{(T_{12},\ldots,T_{1m})}(u_2,\ldots,u_m)du_2\ldots du_m$$

$$=\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty}\mathbb{P}\left(\sum_{j\in\mathcal{J}}\alpha_j 1_{\{T_i^j<T_{1j}|T_{1j}=u_j\}}>\frac{1-\alpha}{2}\right)$$

$$\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{m-1}e^{\left(-\sum_{j=2}^{m}\frac{(u_j-k.d_{1j})^2}{2\sigma^2}\right)}du_2\ldots du_m$$

$$=\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty}\mathbb{P}\left(N(0,1)>\left(\frac{1-\alpha}{2}-E\right)\left(\frac{1}{\sqrt{V}}\right)\right)$$

$$\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{m-1}e^{\left(-\sum_{j=2}^{m}\frac{(u_j-k.d_{1j})^2}{2\sigma^2}\right)}du_2\ldots du_m$$

$$=1-\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{m-1}\int_{-\infty}^{+\infty}.\int_{-\infty}^{+\infty}\mathbb{P}\left(N(0,1)\le\left(\frac{1-\alpha}{2}-E\right)\left(\frac{1}{\sqrt{V}}\right)\right)$$

$$e^{\left(-\sum_{j=2}^{m}\frac{(u_j-k.d_{1j})^2}{2\sigma^2}\right)}du_2\ldots du_m$$

$$=1-\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{m-1}\int_{-\infty}^{+\infty}..\int_{-\infty}^{+\infty}F_{N(0,1)}\left(\left(\frac{1-\alpha}{2}-E\right)\left(\frac{1}{\sqrt{V}}\right)\right)$$

$$e^{\left(-\sum_{j=2}^{m}\frac{(u_j-k.d_{1j})^2}{2\sigma^2}\right)}du_2\ldots du_m$$

$$=1-F_{N(0,1)}\left(\left(\frac{1-\alpha}{2}-E\right)\left(\frac{1}{\sqrt{V}}\right)\right)$$

By summing i from 1 to l, we get $\delta(\alpha)$.

Interpretation of result: In the following figures, we represent values of $\delta(\alpha)$ as the function of α. To simplify computation, we consider the case of two honest miners M_1 and M_2 (i.e $m = 2$) and one SM-pool miner in the network (i.e $l = 1$). We fix the distance between M_1 and M_2 equal to 1 (meaning $k.d_{12} = 1$). We also recall the choice of constants λ, while σ remains free and does not constitute the real values. The point of thinking in this paper is to demonstrate how the value of γ could be studied as a consequence of these assumptions: the propagation delay of information between two miners is normally distributed with mean proportional to the physical distance ($k.d_{12}$) and a constant variance independent of other transmissions (σ^2). Let us recapitulate the quantities which allow us to compute the following Figures with MATLAB :

$$\left\{\begin{array}{c} A^{-1}(\omega)=\frac{1}{\sqrt{2}}\left(\left[\left(\frac{8\omega}{\pi}\right)^2+d_{12}^4\right]^{\frac{1}{2}}+d_{12}^2\right)^{\frac{1}{2}}. \\ p^2=\frac{\lambda}{\sigma}.\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty}\int_{0}^{+\infty}e^{-\lambda\omega} \\ F_{A_2^{-1}(\omega)}(u)d\omega e^{\frac{-(u-kd_{12})^2}{2\sigma^2}}du. \\ E=\alpha_2 p^2,\quad V=\alpha_2^2p^2(1-p^2) \\ \delta(\alpha,\lambda,\sigma)=1- \\ F_{N(0,1)}\left(\left(\frac{1-\alpha}{2}-E\right)\left(\frac{1}{\sqrt{V}}\right)\right). \end{array}\right\}$$

- In the first case (Fig. 2 and Fig. 3), we observe that the smaller the variance of the delay between M_1 and M_2, the smaller the probability of reversing. More precisely, for a fixed power α, the curves are increasingly superimposed with respect to the delay's variance. This can be explained by the fact that the smaller the variance, the smaller the difference between two propagation delays, the less the SM-pool is lucky enough to inform M_2 before M_1.
- In the second case, in particular in Fig. 3, we observe approximately that, the reversing probability is increasing with respect to the intensity. That is also in line with the first theorem of this paper. This part is to deal with the proportion of honest miners that choose to mine on the private branch.

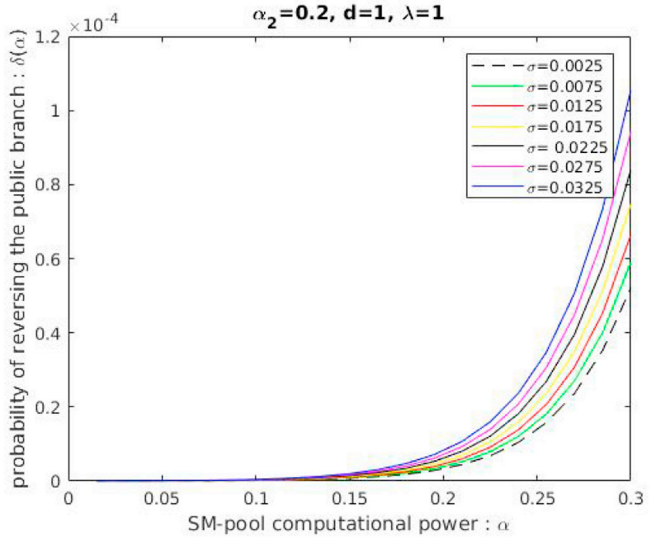

Fig. 2. Curves representing the probability of reversing the public chain in function of the hash power, *for different low values of the variance.*

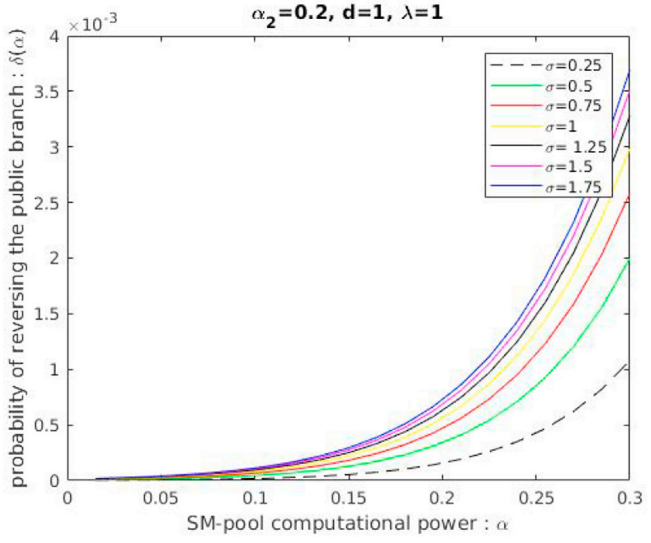

Fig. 3. Curves representing the probability of reversing the public chain in function of the hash power, *for different high values of the variance.*

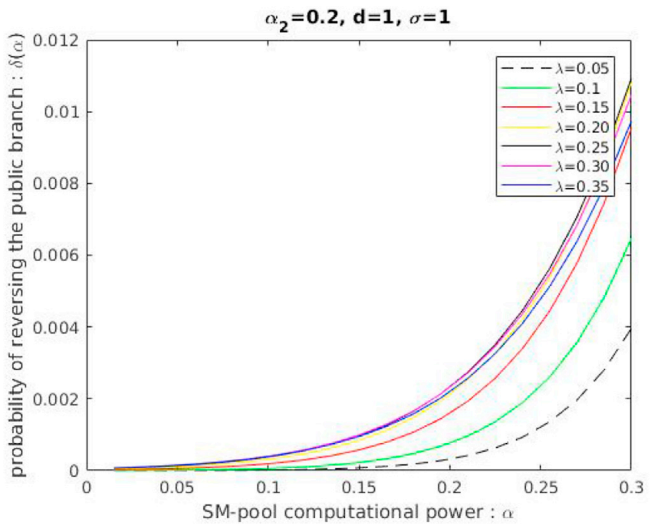

Fig. 4. Curves representing the probability of reversing the public chain in function.

In both cases of representation, the obvious remark is: $\delta(\alpha)$ is increasing with respect to α. This is correct and could be predicted after a qualitative analysis of the selfish-mining attack. Indeed, if α is large enough, $1-\alpha$ is smaller. And by consequence, it is less difficult to broadcast a mined block and reach other honest miners such that the computational power working on this block becomes more than half of $1-\alpha$. Our work gives more clarification on the fact that the more powerful α is, the more obviously the probability of succeding to reverse the public branch increases. The second remark is, even if the values of $\delta(\alpha)$ are very low, they remain positive and contradict the fact that below $\frac{1}{3}$, the probability is zero, just as announced in [1]. In other words, the authors of [1] show that the selfish-mine strategy will never be profitable if the power threshold of attack is less than $\frac{1}{3}$.

Do not forget that in [1] the authors do not take into account the propagation delay of information in the network. Our contribution is: there is no guarantee that beyond $\frac{1}{3}$, the attack will not succeed because of the propagation delay of information in the network. However, we recall that we don't know the exact values of constants λ and σ in the actual Bitcoin network. The constants token here to represent in the figures and tables above are arbitrary and constitute a free choice. These choices show that the threshold values of γ and $\delta(\alpha)$ can be changed, contradictory to those announced in the preceding work of [1]. Despite different values of these constants, we remark that the probability that one private branch becomes a public branch is very low in general but not zero.

Impacts of a high value of δ : We recall that the probability that one private branch newly revealed by the SM-pool after a competing block of the honest community, will be extended by the network (becomes public) is $\delta(\alpha)$. The interest of the study, the quantity $\delta(\alpha)$, is to keep the user from a dangerous dynamic that can follow the network in case of a successful attack. Indeed, after succeeding to reverse the public branch, there are two possible outcomes for the SM-pool: either it extends its branch, or the honest community extends it. In the first case, the SM-pool is in a comfortable situation with respect to the honest community and then, with a high probability, will continue to extend this public branch. Because at this time, the members of this pool already have a head start in the competition. Whenever they finish mining a block, they can hide it again for a small time interval before broadcasting it totally in the network. Since honest miners are rational, then, they will preferentially join the SM-pool to reap their revenues, which are higher compared to those of other pools. And this is easy because a miner's joining a pool is a voluntary act and requires no cost. Moreover, the pool's members will want to accept new members, as this would increase their own revenue. The SM-pool would therefore increase in size (hash power), unopposed by any mechanism until it becomes a majority. Once the SM-pool becomes the majority, it controls the entire blockchain. At this time, the SM-pool would not need to use the selfish-mine strategy. It becomes unnecessary since the others are no longer faster than the pool. Instead, a majority pool can collect all the system's revenue by following the prescribed Bitcoin protocol, and ignore blocks generated outside the pool. It also has no motivation to accept new members. At this point, the currency is not a decentralized currency as originally envisioned.

5. Conclusion

In this paper, we have revisited the fundamental question of selfish-mine attack in Bitcoin network by taking into account a very important hypothesis: the propagation delay of information in the network. This hypothesis on the delay has always been neglected in most of the scientific papers dealing with these types of attacks. The propagation delay assumption on the selfish-mine attack in Bitcoin gives surprising results about the probabilities of reversing the public branch by attackers, in function of its computational power. In view of the given results in function of the delay's variance, the presence of dishonest miners (i.e., the density), and the geographical distance between honest miners in the network (see Figures), no guarantee can be given that the selfish-mine attack will succeed if the computational power is less than $\frac{1}{3}$, contrary to what was announced in Ref. [1]. This has been observed firstly in Theorem 2 since the proportion of honest miners who choose to mine *naively* on the private branch released by dishonest miners also varies with respect to the intensity and the variance. This fact is confirmed in Theorem 4. Even if no guarantee of success or failure of this attack in the presence of delay's assumption is established because of the unknown real values of parameters (σ, λ), this work gives perspectives on future research topics as we have detailed in the last section.

6. Open problem

For the reader's convenience, we start by recalling how a successful double-spending process in Bitcoin network. The attacker:

- Broadcast to the network a transaction in which the attacked merchant is paid.
- Secretly mine a branch which is built on the latest block at the time (before the transaction made it into a block), which includes instead a conflicting transaction that pays the attacker.
- Wait until the transaction to the merchant receives enough confirmation and the merchant, confident in his payment, and sends the product.
- If necessary, continue extending the secret branch (which contradicts the transaction) until it is longer than the public branch (which includes the first transaction), then broadcast it. Because the new branch is longer than the one currently known by the network, it will be considered valid, and the payment to the merchant will be replaced by the payment to the attacker.

The well-known results about high double-spending attack in Bitcoin network, proved in Ref. [5] and clarified in Ref. [9] are:

> *As long as the attacker holds less than* 50% *of the total computational power in the network and all honest miners can communicate quickly (compared with the expected time of block creation), the probability for a double-spending attack to succeed decreases exponentially with respect to the number of confirmations it has received. Therefore, this probability is always 1, if the computer hash power is stronger than* 50%*, and for all numbers of confirmations(*Fig. 4 *of* [9]*).*

At the time we write this paper, in view of scientific papers, success in double-spending is very difficult for an attacker without 50% of the total computational hash power. We will show in this section that selfish-

mining can be an alternative strategy to succeed in double spending without great computational hash power.

How is it possible? How can the selfish-mine attack replace the double-spending attack?

The SM-pool works as follows (cf. Section 2 of this paper): when an SM-miner finds a block with the last public hash, he hides it secretly, waiting for information from the network before revealing it, and creating a private branch. Meanwhile, the honest miners continue mining on the first hash that they hear. The SM-pool continues to extend its secret branch and hope to get the longest chain. And by consequence, this one becomes the global public branch. If the attacker has already diffused a transaction in the last public block in the network before secretly starting to mine a block containing another transaction with another recipient address, with the last public hash, it is in the context of double-spending attack. Thus, the SM-pool continues mining its block secretly and will diffuse it as soon as information broadcasting takes place. The situation "the attacker finishes to mine its secrete block and diffuses it after one public block" is possible and it is defined here as doing a good fork. And we have looked for in this paper what is the probability that its private branch becomes public, in function to the hash power of the attacker α, denoted by $\delta(\alpha)$, by taking into account the effect of the propagation delay of information in the network. If the SM-pool has not finished mining its block containing its second transaction before the first information of the network, it continues to work secretely to extend its private branch by counting every confirmation from the honest network, then we are exactly in the standard double-spending context. This analysis shows clearly that the selfish-mine strategy can be an alternative strategy to succeed in a double spending. Since the values of $\delta(\alpha)$ are not zero in general, our result keeps close to the result of [10]. That means, there are no guarantee that the double-spending attack will fail. In Ref. [10], the authors revisit the question of double-spending and conclude that if the attacker can choose the time he prefers to transmit the transaction, no such guarantee can be given that this attack will fail.

We strongly believe that, studying the impact of propagation delay in the selfish-mine attack is a good starting point to see more in details the resulting probabilities from double-spending in presence of propagation delay. Authors of this paper suggest for future research to explore this problem: taking into account the Rosenfield's model in Ref. [3], it will be very interesting to look for the probability of succeeding in double spending in case of various hash power less than 50%, with the delay assumptions we consider here.

The simple selfish-mine strategy, as it is studied in Ref. [1], gives a result about the reward earned by the colluding pool. The results are given without taking into account the presence of propagation delay in the Bitcoin network. A natural question is: in presence of propagation delay, is selfish-mining always profitable? Otherwise, what is the computational power threshold required for a selfish-miner to earn more than its honest revenue by taking into account the hypothesis of delay?

It looks like that the variability of the delay is an advantage of the attacker in the selfish-mine attack. Then, solving these problems will need a comparative study on the revenues of dishonest miners and constitute a real advance for the crypto-currency's community. We emphasize that our models, both analytic and simulation, are idealized. 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 analytic and simulation techniques we have discussed in this paper.

Acknowledgements

Author of this article, M. BA, would like to thank the laboratory MODAL'X of Université Paris Nanterre to support this work. He would like to thank the members of the Laboratory MODAL'X for discussions he had with them, and the suggestions, remarks and lectures they have made on this paper. He would like also to thank the many thoughtful individuals from the Bitcoin & Ethereum Forum, whose ideas helped us to form the basis of this work, as well as the community of bitcoin for their encouragement and enthusiasm.

References

[1] I. Eyal, E.G. Sirer, Majority is not enough: bitcoin mining is vulnerable, financial cryptography and data security, in: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers, 2014, pp. 436–454.
[2] C. Decker, R. Wattenhofer, Information propagation in the bitcoin network, in: 13th IEEE International Conference on Peer-to-Peer Computing (P2P), Trento, Italy, 2013.
[3] M. Rosenfeld, Analysis of bitcoin pooled mining reward systems, CoRR abs/1112.4980. URL http://arxiv.org/abs/1112.4980.
[4] E. Swanson, Bitcoin's Mining Calculator, 2013. URL, http://www.alloscomp.com/bitcoin/calculator.
[5] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Bitcoin.org. URL http://www.bitcoin.org/bitcoin.pdf.
[6] I.N.F. Bacceli, F. Mathieu, R. Varloot, Can P2P Network Be Super-scalable ? INFOCOM, IEEE, 2016, pp. 1753–1761.
[7] I. N. F. Bacceli, F. Mathieu, Performance of p2p network with special interactions of peers Online. URL http://hal.inria.fr/inria-00615523v2.
[8] J.F.C. Kingman, Poisson Process, Oxford University Press, 1993.
[9] M. Rosenfeld, Analysis of Hashrate-Based Double Spending, CoRR Abs/1402, 2009. URL, http://arxiv.org/abs/1402.2009.
[10] Y. Sompolinsky, A. Zohar, Bitcoin's security model revisited, CoRR abs/1605.09193. URL http://arxiv.org/abs/1605.09193.