# Relying on Consensus does not Make Bitcoin Safer

Emmanuelle Anceaume, Romaric Ludinard, Bruno Sericola

# Relying on Consensus does not Make Bitcoin Safer

Emmanuelle Anceaume
CNRS, UMR 6074 - IRISA
emmanuelle.anceaume@irisa.fr

Romaric Ludinard
ENSAI, UMR 9194 - CREST
romaric.ludinard@ensai.fr

Bruno Sericola
INRIA Rennes - Bretagne Atlantique
bruno.sericola@inria.fr

## I. Introduction

Several distributed cryptocurrencies systems have emerged, and among them, Bitcoin [1] is often designated as the pioneer this kind of systems. As such, Bitcoin shows some vulnerabilities in presence of malicious entities, and some recent works have proposed to improve upon Bitcoin weaknesses. This brief abstract is devoted to the analysis of one of these recent works, and shows through an analytical performance evaluation that new Bitcoin improvements are still needed.

## II. The Bitcoin Network

The Bitcoin network [1] is a peer-to-peer payment network that relies on distributed algorithms and cryptographic tools to allow entities to anonymously buy items or services with bitcoins (*i.e.,* Bitcoin currencies). The main ingredients of this network are *transactions* issued by buyers each time they wish to spend bitcoins and the *blockchain*, a public transaction ledger which eventually contains an ordered sequence of all the issued transactions (more precisely, an ordered sequence of transactions blocks, each one being a set of issued transactions). Three types of entities participate to the Bitcoin ecosystem: *users*, that spend and receive bitcoins, *peers* that propagate transactions in the Bitcoin network and maintain a local copy of the blockchain, and *miners*, that establish the order in which transactions are committed in the blockchain. Of course at any time, an entity may play any role in the Bitcoin ecosystem. Specifically, suppose that Alice wishes to buy some item from Bob. Alice creates a transaction $T$, in which she indicates the price $b$ of that item, a set of *outputs* which represents the recipient accounts of the $b$ bitcoins, *i.e.*, Bob's one, and a set of *inputs* which provides a digest of transactions $\{T_1, T_2, \ldots, T_\ell\}$ Alice's account was recipient of (*i.e.*, these transactions contribute to Alice earnings). The total number of bitcoins $s$ received by Alice in $\{T_1, T_2, \ldots, T_\ell\}$ must be at least greater than $b$. An output from a previous transaction can be referenced at most once in the input of a subsequent transaction, otherwise it would mean that the same bitcoins could be spent several times. Thus, if $s > b$ Alice may add herself as one of the outputs of her own transaction to get change. Note that Alice may voluntarily pay a small transaction fee which will be kept by the miner that contributes to the commitment of transaction $T$. Finally, Alice digitally signs transaction $T$ and submits her transaction $T$ to any peer of Bitcoin for validation purpose. When peer Carol receives $T$ for validity check, she scans the transactions recorded in her local copy of the blockchain. Validity check is achieved by verifying that none of these transactions $\{T_1, T_2, \ldots, T_\ell\}$ already appear in the blockchain. Once this is positively checked, Carol informs Bob that $T$ is valid (Bob can provide his item to Alice), and disseminates $T$ to the Bitcoin network so that eventually all the peers will locally be aware of $T$.

The validation process is not sufficient to guarantee that Alice is not trying to spend the very same bitcoins to David. To handle this *double spending* issue without introducing a trusted central authority, all the transactions must be publicly announced, and the order in which they are committed must be unique. The implementation proposed in Bitcoin consists in incentivizing participants of the network, the so-called *miners*, to spend lot of their CPU to satisfy a given proof-of-work. The effort to build a proof-of-work is large enough to dissuade miners from changing a block in the blockchain as it requires to redo the work for that block and for all the subsequent ones to ensure the consistency of the blocks. Once the proof-of-work has been generated, it forms, together with the set of locally pending transactions, a numbered block that the miner includes in its local copy of the blockchain, and disseminates it to all the entities of the Bitcoin network so that each one will append it to its local copy of the blockchain. Bitcoin miners are incentivized by receiving a reward which is a function of the number of transactions recorded in the blocks they have successfully generated, and the possible transactions fees that appear in those transactions. Note that those rewards are the way bitcoins are created.

As previously described, miners are rewarded for generating blocks, which introduces a competition among them to find the next proof-of-work. This competition may lead to the generation of concurrent blocks leading to the creation of branches in the blockchain, and thus jeopardizing the existence of a unique history of validated transactions. Hopefully, this inconsistency should be transient. Indeed, Bitcoin relies on the assumption that eventually, some miner will be able to generate a proof-of-work quicker than any of the other miners which will make the branch of the blockchain his block depends on longer than any of the other concurrent ones. By construction, Bitcoin declares the longer branch as the legal one. Thus eventually, each peer will update its own blockchain replica with that branch and will remove the concurrent branches together with their transactions. Hence, Bitcoin system will eventually *stabilize* in a state where each peer will share a consistent version of the Blockchain. Anyway, even if Bitcoin eventually converges to a legal state, stabilization may take time, *i.e.,* up to several hours [2]. During this period of time, an

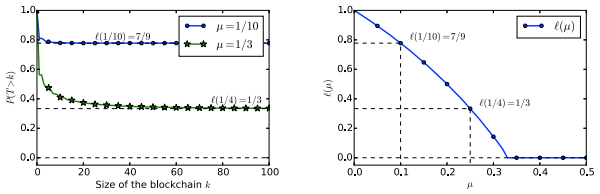
attacker may repeatedly perform double-spend attacks either by using its own CPU resources or by simply taking advantage of the presence of a fork. PeerCensus [3] proposes to solve this issue by providing strong consistency guarantees to Bitcoin.

## III. PeerCensus

PeerCensus [3] proposes to limit the occurrence of both forks and double-spend attacks by guaranteeing that the order in which transactions are committed follows the order in which they have been submitted to the network. This strong consistency schema is implemented by relying on Byzantine Fault Tolerant consensus protocols [4] run among a subset $\mathcal{E}$ of miners, namely, all the miners that successfully generated a block since the genesis of Bitcoin. From a scalability point of view, the PeerCensus approach highly depends on Bitcoin popularity. Indeed, the number of blocks in the blockchain has recently exceeded $k = 400,000$ blocks, meaning that if each block has been generated by a different miner, at least $k$ miners will be involved in the execution of each forthcoming consensus, leading to a message complexity in $O(k^3)$. Beyond this aspect, making $\mathcal{E}$ membership at the $k$-th execution of consensus depend on the decision obtained at the $(k-1)$-th consensus execution leads with high probability to the permanent pollution of $\mathcal{E}$. By pollution we mean the presence of more than one third of byzantine miners in $\mathcal{E}$, even if from a global point of view, Bitcoin network contains less than one third of Byzantine entities. The following analysis proves our assertion.

We denote by $\mu \in ]0,1[$ the proportion of Byzantine miners in Bitcoin, and by $1-\mu$ the proportion of correct ones. We assume that the delay that elapses between two consecutive blocks generations is constant (which reflects the real behavior of Bitcoin). Let $B_k = (h, m)$ denote the state of the blockchain at time $k$, where $h$ (resp. $m$) represents the number of blocks generated by a correct miner (resp. by a byzantine miner). We assume that Nakamoto, the Bitcoin system creator, is honest and thus we have $B_0 = (1, 0)$. Process $B = \{B_k | k \geq 0\}$, with $B_k \in \mathbb{N}^* \times \mathbb{N}$ is an homogeneous discrete time Markov chain that represents the evolution of the blockchain composition over time. From state $B_k = (h, m)$, two transitions are possible : either a new block is generated by a correct miner, and the blockchain goes to state $B_{k+1} = (h+1, m)$ with probability $1-\mu$, or the new block is generated by a byzantine miner and the blockchain goes to the state $B_{k+1} = (h, m+1)$ with probability $\mu$. State $B_k = (h, m)$ of the blockchain at time $k$ is said *polluted* if the number $m$ of Byzantine miners belonging to $\mathcal{E}$ is larger than or equal to $(k-1)/3$ [5]. Conversely, a state that is not polluted is said *safe*. We partition the space state $\mathbb{N}^* \times \mathbb{N}$ into two sub-spaces $\mathcal{S}$ and $\mathcal{T}$ corresponding respectively to the set of safe and polluted states. Blockchain composition evolution can be seen as a random walk over $\mathbb{N}^* \times \mathbb{N}$. Given $k \geq 0$, $h \geq 1$ and $m \geq 0$, and for $(1, 0)$ as initial state, by using the central limit theorem, we get

$$\lim_{k \longrightarrow \infty} \mathbb{P}\{B_k \in \mathcal{S}\} = \begin{cases} 0 & \text{if} \quad \mu > 1/3 \\ 1/2 & \text{if} \quad \mu = 1/3 \\ 1 & \text{if} \quad \mu < 1/3. \end{cases} \qquad (1)$$




(a) $\mathbb{P}\{T > k\}$ as a function of $\mu$ and the blockchain size $k$ (b) Asymptotic behavior of $\mathbb{P}\{T > k\}$ as a function of $\mu$

Relation 1, while in accordance with [3], does not allow us to claim that all the executions that lead to state $B_k$ are safe, *i.e.*, $\forall 0 \geq k' \geq k, B_{k'} \in \mathcal{S}$. This argument is of prime importance, as once $\mathcal{E}$ is polluted, the adversary will be able to impose its decision at each forthcoming consensus, either on the transactions to be committed or on the blocks to be included in the blockchain. Let us now derive the probability of $k$ consecutive safe executions of consensus. Let $T$ be the time spend in states of $\mathcal{S}$ before reaching for the first time a state of $P$. Formally, the random variable $T$ is defined by $T = \min\{k \geq 0 | B_k \in \mathcal{P}\}$, and we have $\mathbb{P}\{T > k\} = \mathbb{P}\{B_0 \in \mathcal{S}, B_1 \in \mathcal{S}, \ldots, B_k \in \mathcal{S}\}$. Theorem 1 gives the law of the first instant of pollution of the blockchain, as well as its asymptotic behavior.

**Theorem 1.** *Given $0 < \mu < 1$, for all $k \geq 0$, we have*

$$\mathbb{P}\{T > k\} = \frac{1}{1-\mu} \sum_{h=\lceil 2k/3 \rceil + 1}^{k+1} \binom{k+1}{h} (1-\mu)^h \mu^{k+1-h} - \frac{3\mu}{1-\mu} \sum_{h=\lceil 2k/3 \rceil + 1}^{k} \binom{k}{h} (1-\mu)^h \mu^{k-h}.$$

*Let $\ell(\mu) = \lim_{k \longrightarrow \infty} \mathbb{P}\{T > k\}$. We have $\ell(\mu) = 0$ if $\mu > 1/3$ and $\ell(\mu) = 1 - 2\mu/(1-\mu)$ otherwise.*

We observe in Figure 1(a) the fast convergence of $T$ to its limit $\ell(\mu)$, while Figure 1(b) shows that when $0 < \mu \leq 1/3$, the probability to have a series of safe consensus executions is strictly less than 1. For exemple, for $\mu = 1/4 < 1/3$, we have $\ell(\mu) = 1/3$ meaning that among all the trajectories of $k$ consensus executions, only $1/3$ of them are safe. This result clearly shows the limitations of the PeerCensus approach to solve Bitcoin weaknesses.

## References


[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008.

[2] "Bitcoin Unconfirmed Transactions," https://blockchain.info/unconfirmed-transactions.

[3] C. Decker, J. Seidel, and R. Wattenhofer, "Bitcoin Meets Strong Consistency," in *17th International Conference on Distributed Computing and Networking (ICDCN)*, 2016.

[4] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, 1999.

[5] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, 1982.