Safety Analysis of Bitcoin Improvement Proposals

Emmanuelle Anceaume, Thibaut Lajoie-Mazenc
CNRS, UMR 6074 - IRISA
firstname.lastname@irisa.fr

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

Bruno Sericola
INRIA RBA
bruno.sericola@inria.fr

***Abstract*—Decentralized cryptocurrency systems offer a medium of exchange secured by cryptography, without the need of a centralized banking authority. Among others, Bitcoin is considered as the most mature one. Its popularity lies on the introduction of the concept of the blockchain, a public distributed ledger shared by all participants of the system. Double spending attacks and blockchain forks are two main issues in blockchain-based protocols. The first one refers to the ability of an adversary to use the very same bitcoin more than once, while blockchain forks cause transient inconsistencies in the blockchain. We show through probabilistic analysis that the reliability of recent solutions that exclusively rely on a particular type of Bitcoin actors, called miners, to guarantee the consistency of Bitcoin operations, drastically decreases with the size of the blockchain.**

***Keywords*—** Bitcoin; Peer-to-Peer Systems; Safety ; Analytical Performance Evaluation

I. Introduction

The goal of decentralized cryptocurrency systems is to offer a medium of exchange secured by cryptography, without the need of a centralized banking authority. An important design issue of such a platform is to prevent the occurrence of double-spending attacks, which consists in using the same resources (the same "coins") in more than one transaction. Classically, the centralized banking authority serves as a trusted third-party that verifies the validity of every single transaction which prevents this kind of attacks. In absence of such a trusted entity, an alternative mechanism must be implemented.

Satoshi Nakamoto proposed a solution in 2008 [20]: the *Bitcoin cryptocurrency system*, the first decentralized ecosystem providing users with a virtual currency to buy and sell services or goods. Bitcoin relies on a public distributed ledger, the so-called *blockchain*, that records all the valid transactions ever issued in the Bitcoin system. Technically, the blockchain is built by some of the participants of the ecosystem, the *miners*, through the creation of blocks. Each block contains the set of most recently issued transactions. The strength of the blockchain design is that it does not require participants to trust each other, each one maximizing its self-interest. Thus, it can be viewed as a way to create a global trusted third-party from a network populated by untrusted participants.

To prevent double-spending attacks, the blocks (and thus all the transactions) must be totally ordered so that every participant of the system can check their validity. However, concurrent blocks can be created due to propagation delays and this must be carefully handled to enforce the consistency of the blockchain. This is achieved by introducing some synchronization among the miners: to successfully create a block, a miner must first solve a resource consuming computation, the so-called *proof-of-work*. Miners are rewarded for each successfully created block, which introduces a competition among them to create the next block as fast as possible.

Such a competition may in its turn give rise to concurrent blocks, and thus to the creation of different branches in the blockchain. This phenomenon is called a *blockchain fork*. Even if Bitcoin eventually converges to a legal state (*i.e.* a unique branch), stabilization may take time. For instance, the March 2013 fork [7] was resolved after several hours. During a fork, an attacker may repeatedly perform double-spending attacks. Given the increasing popularity of Bitcoin, one may expect that the abuse of the weaknesses in its design will increase.

A large amount of work has been devoted to mitigating this salient issue and, among them, three propositions have recently emerged as the very first convincing solutions to solve the double-spending attack. These solutions, respectively called Bitcoin-NG [6], PeerCensus [5], and BizCoin [13], propose to give additional specific power to miners, by coordinating their view of the blockchain through either executions of Byzantine-tolerant consensus protocols or leadership of one of them.

a) Our contributions: The paper is devoted to a thorough analysis of the behavior of these three works with respect to their capacity to handle numerous transactions and their resilience to malicious miners. Prior to this analysis, we provide an extensive description of the Bitcoin ecosystem from which we derive a formalization of its properties in terms of validity, confirmation, safety and liveness. We then present the model that allows us to investigate the properties of the three above mentioned solutions. The outcome of this study contains a mixture of both favorable and negative results. Bitcoin-NG, by relying on a leader, neither improves upon Bitcoin safety, nor scales to a large number of transactions. Despite the support of Byzantine-tolerant consensus algorithms, PeerCensus does not tolerate the well-known threshold of $1/3$ malicious miners. On the other hand, by combining the design of both Bitcoin-NG and PeerCensus and by relying on the CoSi protocol [24] for collective signing, BizCoin shows good theoretical performance. Its resilience to malicious miners corroborates results of Byzantine tolerant distributed algorithms [15] for large enough signature groups: in the presence of less than one third of Byzantine miners, BizCoin is safe with high probability if the number of miners involved in the signature group exceeds $1,000$. However, BizCoin has not yet been implemented with more than 148 miner because

of the complexity of the underlying CoSi protocol [12].

b) Road map: To summarize, the remainder of the paper is organized as follows. Section II presents the necessary background to understand Bitcoin properties in terms of safety and liveness. Section III presents an analysis of the safety guaranteed by respectively Bitcoin-NG, PeerCensus and BizCoin. Section IV presents a brief survey of the crypto systems that have appeared since 2009. Finally, Section V concludes.

II. Background on The Bitcoin Network

The Bitcoin network [20] is a peer-to-peer payment network that relies on distributed algorithms and cryptographic tools to allow entities to pseudonymously buy goods or services with digital currencies called bitcoins. Its main ingredients are *(i) transactions* issued by buyers each time they wish to spend bitcoins, *(ii)* the *blockchain*, an ordered sequence of blocks, each one being a set of issued transactions, maintained distributively by the entities of the system, and *(iii)* a pool of pending transactions eligible for confirmation in the blockchain, locally maintained by each entity. Three types of entities participate in the Bitcoin ecosystem: *users*, that send and receive bitcoins, *peers* that propagate transactions in the network and maintain a local copy of the blockchain, and *miners*, that establish which transactions will appear in the blockchain, and the order in which they will appear. Of course, at any time, an entity may play any of the roles in the Bitcoin ecosystem.

A. The Bitcoin protocol

To illustrate the description of the Bitcoin ecosystem, we will take the example of individual users called Alice, Bob and Carol. Alice owns bitcoins, and she wishes to send them to Bob and Carol for the goods they provide to her. Bitcoins are entirely virtual. They are accessible via Bitcoin accounts. An account, which refers to the elementary functional entity of the Bitcoin ecosystem, is described by a key, derived from the public key of the public/private key generated by Bitcoin users. Keys are used to prove the ownership of bitcoins. A bitcoin account is locked by its owner, and spending bitcoins amounts to unlocking that account and transferring its value to that of the recipient of the transaction who will be credited once the transaction is confirmed in the blockchain (more details will be given in the sequel). Note that to hide their profile, it is recommended that users generate a new public/private key for each transaction they are recipient of. An important aspect of Bitcoin accounts is their indivisibly, meaning that once an account has been created by a user, it will be credited by a single transaction and will be debited by a single subsequent transaction. Note that Alice may voluntarily pay a small *transaction fee* which will be kept by the miner that will succeed in confirming Alice's transaction in the blockchain. In this case, the total amount of bitcoins in the input accounts is greater than the amount of bitcoins transferred to the output accounts. The successful miner creates an account that will be credited with the fees from all the transactions of the block, along with a block reward (whose amount is currently set to 12.5 bitcoins; it is halved every 210,000 blocks, which last occurred in July 2016) through a *coinbase* transaction, the way Bitcoin creates money. Specifically, to send bitcoins to Bob and Carol, Alice creates a transaction $T = (\{a_{i_1}, \ldots, a_{i_m}\}, \{b_j, c_\ell\})$, where $\{a_{i_1}, \ldots, a_{i_m}\}$, $m \geq 1$, refers to Alice's credited accounts that she wishes to spend. Set $\{a_{i_1}, \ldots, a_{i_m}\}$ is called the input set of transaction T, and is denoted by I. The inputs of a transaction are actually the hash of the transactions that credited Alice's accounts. We refer directly to the accounts for a sake of clarity. Accounts b_j and c_ℓ have respectively been created by Bob and Carol to receive bitcoins from Alice's accounts. Set $\{b_j, c_\ell\}$ is called the output set of T, and is denoted by O. The amount of bitcoins of account u_i is denoted by $v(u_i)$. T also includes Alice's digital signature on the input and output accounts; thus, any user can verify its integrity by checking the chain of signatures.

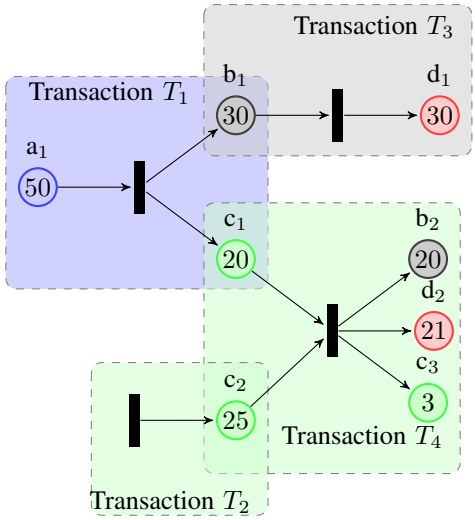


Fig. 1. Modelling the evolution of users' accounts

To describe the evolution of user accounts, we have adopted a place/transition model as depicted in Figure 1. User accounts are represented by places (circles) and transactions by transitions (vertical bars). The place from which an arc runs to a transition is an input place of the transition, and the place to which an arc runs to is an output place of the transition. The number of bitcoins in a user account represents the tokens of the place. A transition may fire if there are sufficiently many tokens in its input places, and it consumes all of them upon firing. Places and transitions are dynamically created.

In Figure 1, Alice creates transaction T_1 to transfer the 50 bitcoins of her account a_1 to Bob and Carol's accounts: 30 bitcoins to b_1 and 20 to c_1. Transaction T_4 contains a transaction fee equal to $(25+20)-(20+21+3) = 1$ bitcoin. T_2 is a coinbase transaction.

We say that a transaction $T = (I, O)$ is *well-formed* if the transition can be fired, *i.e* $\sum_{i \in I} v(i) \geq \sum_{o \in O} v(o)$, and the creator of T is the owner of the input accounts of T. In the following we consider that all the transactions are well-formed.

When Bob and Carol receive the digitally signed transaction T, they submit it to any peer p of Bitcoin for a validity check.

Informally, a transaction $T = (I, O)$ is valid if p has received all the transactions that have credited all the accounts in I and not received any transaction already using any of those same accounts. To formally define the validity property we introduce the notion of a local view. The local view of p is the pool of pending transactions at p together with the content of p's blockchain $\mathcal{B}^{(p)}$. If we respectively denote by $\mathcal{V}_k^{(p)}$ and $\mathcal{P}_k^{(p)}$ the local view of p and p's pool of pending transactions when p receives its k-th transaction, then we have

$$\mathcal{V}_k^{(p)} = \mathcal{P}_k^{(p)} \cup \mathcal{B}^{(p)}.$$

The current view and the current pool of pending transactions of peer p are simply denoted by $\mathcal{V}^{(p)}$ and $\mathcal{P}^{(p)}$, respectively. We suppose that p has initialised $\mathcal{V}_1^{(p)}$ with the first block broadcast by Satoshi, containing a single coinbase transaction. When p receives a new transaction T, p declares it valid according to the following definition.

Definition 1 (Validity Property)**.** *Given a peer p of the Bitcoin network, p considers its k-th transaction $T = (I, O)$ as locally valid if and only if the following three properties hold:*

$$\forall a \in I, \exists T' = (I', O') \in \mathcal{V}_{k-1}^{(p)}, \quad a \in O' \tag{1}$$

$$\forall T' = (I', O') \in \mathcal{V}_{k-1}^{(p)}, \quad I \cap I' = \emptyset \tag{2}$$

$$\forall a \in O, \forall T' = (I', O') \in \mathcal{V}_{k-1}^{(p)}, \quad a \notin O'. \tag{3}$$

In the following, in accordance with Bitcoin requirements, we suppose that a user creates a new account for each transaction she receives. That is Relation (3) is always satisfied.

As soon as $T = (I, O)$ is considered locally valid, p inserts it in its transaction pool, that is $\mathcal{P}_k^{(p)} \leftarrow \mathcal{P}_{k-1}^{(p)} \cup \{T\}$, then positively acknowledges Bob and Carol, and finally disseminates T in the Bitcoin network. On the contrary, if $T = (I, O)$ is not locally valid, two cases must be considered: either Relation (1) or Relation (2) does not hold. In the former case, p inserts T in its local pool (*i.e.* $\mathcal{P}_k^{(p)} \leftarrow \mathcal{P}_{k-1}^{(p)} \cup \{T\}$) and disseminates it in the Bitcoin network. Transaction T becomes locally valid when p receives the transactions crediting the missing accounts. In the latter case, p has already received a locally valid transaction $T' = (I', O')$ such that $a = I \cap I'$: account a is in a double-spending situation. Formally,

Definition 2 (Double-spending situation)**.** *Given a Bitcoin account a_i, account a_i is in a double-spending situation if there exist two transactions $T_1 = (I_1, O_1)$ and $T_2 = (I_2, O_2)$ such that :*

$$T_1, T_2 \in \bigcup_p \mathcal{V}^{(p)} \wedge a_i \in I_1 \cap I_2.$$

A transaction T is conflict-free if none of the inputs of $T = (I, O)$ is involved in a double-spending situation and all of the transactions that credited T's inputs are conflict-free:

Definition 3 (Conflict-free transaction)**.** *A transaction $T = (I, O)$ is conflict-free if $\forall a \in I$, a is not involved in a double-spending situation and the transaction $T' = (I', O') \in \mathcal{V}^{(p)}$ with $a \in O'$ is conflict-free.*

By construction, the induction is finite because Bitcoin creates money only through coinbase transactions, which are by definition conflict-free. Each T' exists by Relation (1).

Preventing a double-spending situation from transforming into a successful double-spending attack (i.e. Alice succeeds in converting the content of one of her accounts into goods twice) is the key challenge of many virtual crypto-systems.

The solution adopted in Bitcoin to mitigate double-spending attacks, without relying on a central trusted authority, consists in gathering transactions into blocks and totally ordering them in a publicly accessible and distributively managed ledger. This is the role of *miners*. Briefly, the construction of a well-formed block uses the hashcash proof-of-work function which consists in computing $\mathfrak{h}(x)/2^m(m-k)$, where $\mathfrak{h}$ is the double SHA-256 hash function, m is the size of the hash output, *i.e.*, $m = 256$, and k is the work factor. The value of x is obtained by combining, among others, the sequence number of the last block in the blockchain, the set of locally pending transactions and a counter c incremented by the miner until the first k bits of the hash output are 0. The work factor k is adjusted every 2016 blocks to provide an average block creation time of 10 minutes. Once the proof-of-work has been generated by some miner q, it forms, together with the set of locally pending transactions $\mathcal{P}^{(q)}$, a numbered block b_ℓ that q appends to $\mathcal{B}^{(q)}$. Miner q disseminates this block in the Bitcoin network so that each peer appends it to its local copy of the blockchain. The status of a transaction changes from *pending* to *locally confirmed* whenever it is included in a block.

Definition 4 (Local confirmation)**.** *Given a peer q of the Bitcoin network, and a locally valid transaction T,*

$$T \text{ is locally confirmed} \iff \exists! B \in \mathcal{B}^{(q)}, T \in B.$$

The *local confirmation level* of transaction T at peer q is equal to the depth of block B, which corresponds to the number of blocks appended in $\mathcal{B}^{(q)}$ after B, including B.

Bitcoin miners are incentivized by receiving, when they successfully generated a block, a reward in the form of the coinbase transaction, defined above. Blocks, being generated at a regular and very slow rate, provide a fully distributed synchronization of the network. Since bitcoins are only created through block rewards, it also ensures their rarity, leading to their high financial value and hence to the high incentive to create blocks.

B. Blockchain forks

Rewarding the creation of blocks introduces a competition among miners. This competition may lead to concurrent blocks (*i.e.* equally numbered blocks) and thus to a blockchain with a tree structure. This phenomenon is called a *blockchain fork*. A blockchain fork is resolved as soon as a miner generates a proof-of-work and disseminates the corresponding block b_ℓ quickly enough so that no concurrent miner has found a valid proof-of-work before it receives it. In that case, the branch of the local blockchain that contains b_ℓ is longer than any other concurrent branches, which are pruned from the

tree, leading to a blockchain with a unique branch. Note that "pruned" transactions that do not already belong to the unique branch are added again in the local transaction pools for a possible confirmation in subsequent blocks. Blockchain forks must be quickly resolved for two main reasons. Firstly, malicious miners can take advantage of this phenomena to trigger double-spend attacks. Such an attack is successful if the branch that remains after the resolution of the fork contains the illegitimate transaction issued by the attacker. Nakamoto's analysis, as well as subsequent ones [8], [11], [19] focus on the race between malicious miners and honest ones to generate the longer branch of the blockchain. Suppose that Bob is the recipient of a transaction issued by the malicious sender Alice, and that Alice manipulates a proportion μ of the miners of the system. Nakamoto has shown that with probability less than 0.1%, Bob's transaction will be rejected if its level of confirmation z in the local blockchain of some peer is less than 5 when $\mu = 10\%$. The level of confirmation must increase to $z = 8$ when μ increases to $\mu = 15\%$, and to $z = 15$ when 25% of the miners are corrupted. In the following, a transaction is said *deeply confirmed* once it reaches such a confirmation level. The second reason is that, in the presence of several branches, the global computing power of the miners is spread over the branches. This leads to an increase of the average block generation time, and accordingly to the augmentation of the time needed by transactions to become deeply confirmed.

C. Bitcoin properties

We can now state Bitcoin's fundamental properties:

Property 1 (Bitcoin Liveness)**.** *A conflict-free transaction will eventually be deeply confirmed in the blockchain of an honest peer.*

Property 2 (Bitcoin Safety)**.** *A conflict-free transaction deeply confirmed by some honest peer will eventually be deeply confirmed by all honest peers with the same confirmation level.*

Two important remarks are in order.

Remark 1. *Properties 1 and 2 guarantee that the view of all honest peers have the same prefix.*

Remark 2. *Properties 1 and 2 each apply to transactions issued by honest users, but honest recipients of conflictual transactions have no guarantee of receiving the corresponding coins.*

To summarize, to prevent money counterfeiting, Bitcoin opens the door to double-spending attacks against users that optimistically assume that valid or even locally confirmed transactions will eventually be deeply confirmed.

Given the increasing popularity of Bitcoin, any user may legitimately expect a stronger liveness property than the one implemented by Bitcoin. Indeed, in its current implementation, a user cannot detect that a transaction it is recipient of is conflictual and, thus, has no guarantee to be paid for the service provided before said transaction becomes deeply confirmed, which takes one hour in average.

In the sequel of the paper, we present three recent works that aim at providing stronger guarantees to honest users through linearizable operations on Bitcoin accounts. We show that none of these works provide sufficient guarantees in presence of malicious miners.

III. Relying on miners as a trusted third party

Three recent works, Bitcoin-NG [6], PeerCensus [5], and BizCoin [13], have proposed to rely exclusively on miners to take in charge the full process of validation and confirmation to guarantee that all the operations triggered on the transactions are atomically consistent. Atomic consistency guarantees that all the updates on shared objects are perceived in the same order by all entities of the system. In all these protocols, time is divided into epochs. An epoch ends when a miner successfully generates a new block. This miner becomes the leader of the subsequent epoch. Each of these solutions rely on a dedicated set $\mathcal{E}_\ell$, with $\ell \in \{1, w, \infty\}$. This set is built along consecutive epochs as follows. At epoch k, if $|\mathcal{E}_\ell| < \ell$, the new leader is added to $\mathcal{E}_\ell$. Otherwise, the leader at epoch $k + 1 - \ell$ is removed from $\mathcal{E}_\ell$ and the new leader is added. Once set $\mathcal{E}_\ell$ reaches size ℓ, it remains at constant size ℓ.

Strong consistency is implemented in these protocols by different means. In Bitcoin-NG, it is achieved by delegating the validation process to $\mathcal{E}_1$, *i.e.* the leader of the current epoch. In PeerCensus it is implemented by relying on Byzantine Fault Tolerant consensus protocols (*e.g.* [4], [9], [14]) run by $\mathcal{E}_\infty$ (recall that it contains all the miners that successfully generated a block). Finally, BizCoin leverages both ideas by using the leader and a consensus run by $\mathcal{E}_w$. In all these protocols, members of $\mathcal{E}_\ell$, with $\ell \in \{1, w, \infty\}$, are entitled to validate and confirm issued transactions and blocks and to disseminate them so that each peer integrates them in its local blockchain. In the remainder of the section we show that, surprisingly enough, relying on miners to confirm transactions does not prevent malicious users from successfully double-spending bitcoins. Prior to doing that, we present the model we use to analyze the safety of these protocols.

A. Model

We assume the presence of an adversary controlling a proportion $\mu \in (0, 1)$ of the whole set of miners. This adversary aims at exploiting the protocol under consideration in order to perform double spending attacks. Miners that are controlled by the adversary and the blocks they generate are called Byzantine or malicious. On the contrary, miners that are not controlled by the adversary and their blocks are considered honest (*i.e.* they follow the prescribed protocol) and represent a proportion $(1 - \mu)$ of the whole set of miners. We assume that each miner (honest or not) has the same computational power. Finally, we assume a constant block generation time.

Let $B_k = (h, m)$ denote the state of the blockchain at epoch k, where h and m represent the number of honest (respectively malicious) blocks. We assume that Nakamoto, the Bitcoin system creator, is honest and thus we have $B_0 = (1, 0)$. Process $B = \{B_k \mid k \geq 0\}$ represents the evolution of the

blockchain composition over epochs. From state $B_k = (h, m)$ two transitions are possible: the next block can either be generated by a honest miner, and the blockchain goes to state $B_{k+1} = (h + 1, m)$ with probability $1 - \mu$, or generated by a malicious miner, and $B_{k+1} = (h, m + 1)$, which happens with probability μ. Process B is an homogeneous discrete time Markov chain over the discrete state space $\mathbb{N}^* \times \mathbb{N}$. Non null probability transitions are given for all $(h, m) \in \mathbb{N}^* \times \mathbb{N}$ by

$$\mathbb{P}\{B_{k+1} = (h + 1, m) \mid B_k = (h, m)\} = 1 - \mu, \quad (4)$$

$$\mathbb{P}\{B_{k+1} = (h, m + 1) \mid B_k = (h, m)\} = \mu. \quad (5)$$

B. Analysis of Bitcoin-NG Safety

As previously described, in Bitcoin-NG each epoch is led by a single miner entitled to validate the set of transactions it receives. Upon reception of a new one, the leader has to check if it is locally valid, *i.e.* if it satisfies Definition 1. If so, the leader cryptographically signs it and disseminates it. Only signed transactions may be confirmed, *i.e.* inserted in a block.

Note that if the leader is malicious, it may easily create double-spending transactions and sign them with no consideration for the other transactions whose recipients are honest. By assumption a proportion $\mu \in (0, 1)$ of miners are controlled by the considered adversary. Thus in expectation, a proportion μ of blocks are malicious as well. The evolution of the blockchain 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 with $B_0 = (1, 0)$ as initial state, we easily derive :

$$\mathbb{P}\{B_k = (h, m)\} = \binom{k}{h-1}(1-\mu)^{h-1}\mu^m 1_{\{k=h+m-1\}} \quad (6)$$

The probability that at epoch k the blockchain does not contain any malicious block is equal to $\mathbb{P}\{B_k = (h, 0)\} = (1-\mu)^{k-1}$ if $h = k-1$ and 0 otherwise. Currently, the Bitcoin blockchain counts more than 420 000 blocks, making this probability close to 0. Furthermore, one can note the scalability issue in this protocol: in the current setting, a leader has to sign on average 1500 transactions per epoch, this volume being steadily growing. Consequently, Bitcoin-NG approach cannot cope with an adversarial environment, and hardly scales to a high number of transactions.

C. Analysis of PeerCensus Safety

Contrarily to Bitcoin-NG, PeerCensus [5] proposes to involve the whole set $\mathcal{E}_\infty$ of successful miners in a Byzantine Fault Tolerant consensus protocol like PBFT [4]. Prior to focusing on PeerCensus safety, one may easily notice that the scalability of this solution highly depends on the blockchain size. Indeed, by involving in the k-th execution of the Byzantine tolerant consensus algorithm the $k-1$ previously successful miners, this would lead, today, to a consensus run by $k \geq 420,000$ miners. The message complexity of Byzantine tolerant consensus is classically in $\mathcal{O}(k^3)$, leading these algorithms to barely scale beyond 10 participants which clearly weakens the feasibility of this approach.

Beyond this aspect, making $\mathcal{E}_\infty$ 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 $\mathcal{E}_\infty$. By pollution we mean the presence of more than one third of byzantine miners in $\mathcal{E}_\infty$, even if from a global point of view, the Bitcoin network contains less than one third of byzantine entities (*i.e.* $\mu < 1/3$). The following analysis proves our assertion.

According to [15], a consensus cannot be reached among n participants if more than $(n-1)/3$ participants are byzantine. We say that the state $B_k = (h, m)$ of $\mathcal{E}_\infty$ at epoch k is *polluted* if the number m of byzantine miners belonging to $\mathcal{E}_\infty$ is larger than or equal to $(k-1)/3$. Conversely, a state that is not polluted is said to be *safe*. We partition the space state $\mathbb{N}^* \times \mathbb{N}$ into two sub-spaces $\mathcal{S}_\infty$ and $\mathcal{P}_\infty$ corresponding respectively to the set of safe and polluted states. We have

$$\mathcal{S}_\infty = \{(h, m) \in \mathbb{N}^* \times \mathbb{N} \mid h \geq 2m + 1\}$$
$$\text{and } \mathcal{P}_\infty = \{(h, m) \in \mathbb{N}^* \times \mathbb{N} \mid h \leq 2m\}.$$

Thus, using Relation (6), the probability that $\mathcal{E}_\infty$ is in a safe state at epoch k is given by

$$\begin{aligned}\mathbb{P}\{B_k \in \mathcal{S}_\infty\} &= \sum_{h=1,\ 3h\geq 2k+3}^{k+1} \binom{k}{h-1}(1-\mu)^{h-1}\mu^{k-h+1} \\ &= \sum_{h=\lceil 2k/3 \rceil}^{k} \binom{k}{h}(1-\mu)^h\mu^{k-h}.\end{aligned}$$

Using the central limit theorem, we get

$$\lim_{k\longrightarrow\infty} \mathbb{P}\{B_k \in \mathcal{S}_\infty\} = \begin{cases} 0 & \text{if} \quad \mu > 1/3 \\ 1/2 & \text{if} \quad \mu = 1/3 \\ 1 & \text{if} \quad \mu < 1/3. \end{cases} \quad (7)$$

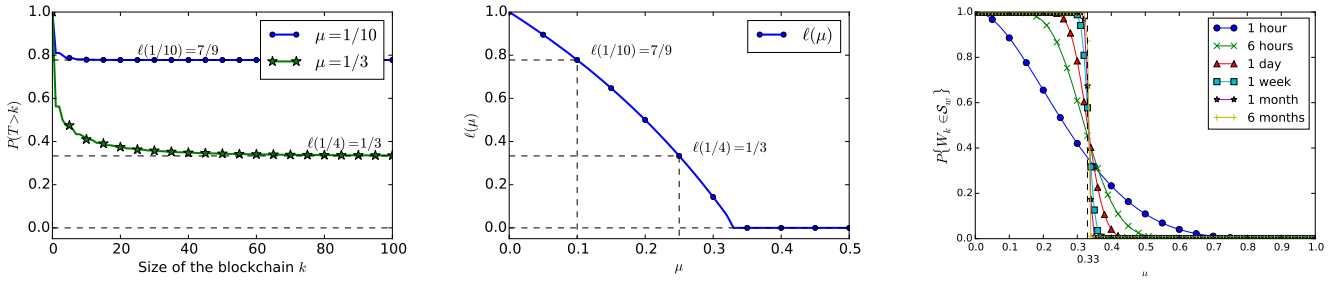
Relation (7), while in accordance with [5], does not allow one to claim that the execution that led to state B_k was safe, *i.e.*, $\forall k' \leq k, B_{k'} \in \mathcal{S}_\infty$. This argument is of prime importance, as once $\mathcal{E}_\infty$ is polluted, the adversary will be able to impose its decision at each forthcoming consensus, either on the transactions to be confirmed or on the blocks to be included in the blockchain.

We now derive the probability of k consecutive safe executions of the consensus. Let T be the number of epochs spent in states of $\mathcal{S}_\infty$ before reaching for the first time a state of $\mathcal{P}_\infty$. Formally, the random variable T is defined by $T = \min\{k \geq 0 \mid B_k \in \mathcal{P}_\infty\}$, and we have $\mathbb{P}\{T > k\} = \mathbb{P}\{B_0 \in \mathcal{S}_\infty, B_1 \in \mathcal{S}_\infty, \ldots, B_k \in \mathcal{S}_\infty\}$. Theorem 1 provides a way to compute the probability of being in a given state $B_k = (h, m) \in \mathcal{S}_\infty$ before the first corruption.

Theorem 1. *For all* $(h, m) \in \mathcal{S}_\infty$ (i.e. $h \geq 1$, $m \geq 0$ *et* $h \geq 2m + 1$*) and* $k = m + h - 1$*, we have*

$$\begin{aligned}&\mathbb{P}\{T > k, B_k = (h, m)\} \\ &= \left[\binom{k+1}{h} - 3\binom{k}{h}\right](1-\mu)^{h-1}\mu^m 1_{\{k=m+h-1\}}. \quad (8)\end{aligned}$$

Proof. We define $f(h, m) = \mathbb{P}\{T > k, B_k = (h, m)\}$ for $(h, m) \in \mathcal{S}_\infty$ (with $k = m+h-1$) and $f(h, m) = 0$ otherwise.



(a) $\mathbb{P}\{T > k\}$ as a function of μ and the blockchain size k

(b) Asymptotic behavior of $\mathbb{P}\{T > k\}$ as a function of μ

(c) $\mathbb{P}\{W_k \in \mathcal{S}_w\}$ as a function of μ and w

Fig. 2. Analysis of the safety of PeerCensus and BizCoin algorithms

The initial state being $(1, 0)$, we have $f(1, 0) = 1$. Using the Markov property, we have

$$f(h, m) = (1 - \mu)f(h - 1, m)1_{\{h \geq 2m+2\}} + \mu f(h, m - 1).$$

The relation is true for $m = 0$. Indeed the previous relation gives, for $m = 0$,

$$f(h, 0) = (1 - \mu)f(h - 1, 0)1_{\{h \geq 2\}},$$

that is $f(h, 0) = (1 - \mu)^{h-1}$, for every $h \geq 1$. Moreover, Relation (8) gives the same result for $m = 0$. We now use a recurrence on two levels. Suppose that Relation (8) is true for integer $m - 1$. For $h = 2m + 1$, we have

$$\begin{aligned} f(2m + 1, m) &= \mu f(2m + 1, m - 1) \\ &= \left[\binom{3m}{2m + 1} - 3\binom{3m - 1}{2m + 1}\right](1 - \mu)^{2m}\mu^m \\ &= \frac{(3m)!}{(2m + 1)!m!}(1 - \mu)^{2m}\mu^m, \end{aligned}$$

which is the result given by Relation (8).

Suppose that the relation is true for integers $h - 1$ and m, with $h \geq 2m + 2$. The recurrence hypothesis gives

$$\begin{aligned} f(h, m) &= (1 - \mu)f(h - 1, m) + \mu f(h, m - 1) \\ &= \left[\binom{m + h - 1}{h - 1} - 3\binom{m + h - 2}{h - 1}\right. \\ &\quad \left. + \binom{m + h - 1}{h} - 3\binom{m + h - 2}{h}\right](1 - \mu)^{h-1}\mu^m. \end{aligned}$$

Grouping the first and the third term, and the second and the fourth leads, since $k = m - h + 1$, to

$$f(h, m) = \left[\binom{k + 1}{h} - 3\binom{k}{h}\right](1 - \mu)^{h-1}\mu^m,$$

which completes the proof. □

Theorem 2 gives the distribution of the first instant T of pollution of $\mathcal{E}_\infty$, as well as its asymptotic behavior.

Theorem 2. *For all $\mu \in (0, 1)$ and $k \geq 0$, we have*

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

The limit $\ell(\mu) = \lim_{k\longrightarrow\infty} \mathbb{P}\{T > k\}$ is then given by

$$\ell(\mu) = \begin{cases} 0 & \text{if} \quad \mu > 1/3 \\ \\ 1 - \dfrac{2\mu}{1 - \mu} & \text{if} \quad \mu \leq 1/3. \end{cases} \tag{9}$$

Proof. Theorem 1, gives for all $k \geq 0$,

$$\begin{aligned} \mathbb{P}\{T > k\} &= \sum_{(h,m)\in\mathcal{S}}\left[\binom{k + 1}{h} - 3\binom{k}{h}\right] \\ &\quad \times (1 - \mu)^{h-1}\mu^m 1_{\{k=m+h-1\}} \\ &= \sum_{h=1,\ 3h\geq 2k+3}^{k+1}\binom{k + 1}{h}(1 - \mu)^{h-1}\mu^{k-h+1} \\ &\quad - \sum_{h=1,\ 3h\geq 2k+3}^{k}\binom{k}{h}(1 - \mu)^{h-1}\mu^{k-h+1} \\ &= \frac{1}{1 - \mu}\sum_{h=\lceil 2k/3\rceil+1}^{k+1}\binom{k + 1}{h}(1 - \mu)^h\mu^{k+1-h} \\ &\quad - \frac{3\mu}{1 - \mu}\sum_{h=\lceil 2k/3\rceil+1}^{k}\binom{k}{h}(1 - \mu)^h\mu^{k-h}. \end{aligned}$$

The second result is derived from the central limit theorem. □

We observe in Figure 2(a) the fast convergence of T to its limit $\ell(\mu)$, while Figure 2(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 instance, 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.

D. BizCoin

BizCoin [13] combines some of the ideas proposed in PeerCensus and Bitcoin-NG: BizCoin uses the last successful miner as the leader of the current epoch but the confirmation process is handled by tolerant Byzantine consensus executions, implemented through a cryptographic collecting signing scheme [24]. Furthermore, differently from PeerCensus, which relies on $\mathcal{E}_\infty$, BizCoin restricts the consensus membership to $\mathcal{E}_w$, containing the current leader and the $w-1$ previous ones.

We proceed as above to analyze BizCoin safety. Let us consider the random variable M correponding to the type of the current leader. According to Relations (4) and (5), at epoch k, the leader is honest, *i.e.* $M = 0$, with probability $\mathbb{P}\{M = 0\} = 1 - \mu$ and byzantine, *i.e.* $M = 1$, with probability $\mathbb{P}\{M = 1\} = \mu$. We denote by $M_{0,k}, \ldots, M_{w-1,k}$ the type of the last w leaders at epoch k. The vector $W_k = (M_{0,k}, \ldots, M_{w-1,k}) \in \{0,1\}^w$ represents the state of $\mathcal{E}_w$. The process $W = \{W_k,\ k \geq 0\}$ evolves as follows:

$$\forall k \geq 1, \forall 1 \leq i \leq w-1, M_{i,k} = M_{i-1,k-1} \tag{10}$$

where $(M_{0,k})_{k\geq 1}$ is a sequence of independent and identically distributed Bernoulli random variable with $\mathbb{P}\{M_{0,k} = 0\} = 1-\mu$ and $\mathbb{P}\{M_{0,k} = 1\} = \mu$. The process $W = \{W_k,\ k \geq 0\}$ is thus a homogeneous discrete-time Markov chain over the state space $\{0,1\}^w$, representing the evolution of the composition of $\mathcal{E}_w$ over epochs.

Similarly to Section III-C, a state W_k is *polluted* if the number of Byzantine miners belonging to $\mathcal{E}_w$ is larger than $(w-1)/3$. Conversely, a state that is not polluted is *safe*. We are interested in the number of Byzantine miners in $\mathcal{E}_w$ at epoch k. We denote this random variable by N_k which is given by $N_k = \sum_{i=0}^{w-1} M_{i,k}$. We partition the space state $\{0,1\}^w$ into two subsets $\mathcal{S}_w$ and $\mathcal{P}_w$ corresponding respectively to the set of safe and polluted states. We then have

$$\mathcal{S}_w = \{(m_0, \ldots, m_{w-1}) \in \{0,1\}^w \mid \sum_{i=0}^{w-1} m_i \leq (w-1)/3\},$$

$$\mathcal{P}_w = \{(m_0, \ldots, m_{w-1}) \in \{0,1\}^w \mid \sum_{i=0}^{w-1} m_i > (w-1)/3\}.$$

Theorem 3 derives the probability of having $\mathcal{E}_w$ in a safe state in steady state, and shows that the steady state is reached at epoch $k = w$.

Theorem 3. *For all $k \geq w$, we have*

$$\mathbb{P}\{W_k \in \mathcal{S}_w\} = \sum_{\ell=0}^{(w-1)/3} \binom{w}{\ell} \mu^\ell (1-\mu)^{w-\ell}.$$

Proof. From Relation (10), we easily get for any $k \geq 1$, $N_k = N_{k-1} + M_{0,k} - M_{w-1,k-1}$. Expanding this relation gives

$$N_k = N_0 + \sum_{\ell=1}^{k} M_{0,\ell} - \sum_{\ell=0}^{k-1} M_{w-1,\ell}. \tag{11}$$

Observing that

$$\sum_{\ell=0}^{k-1} M_{w-1,\ell} = \begin{cases} \sum_{\ell=0}^{k-1} M_{w-1-\ell,0} \text{ if } k \leq w \\ \sum_{\ell=0}^{w-1} M_{w-1-\ell,0} + \sum_{\ell=w}^{k-1} M_{0,\ell-w+1} \text{ if } k > w \end{cases}$$

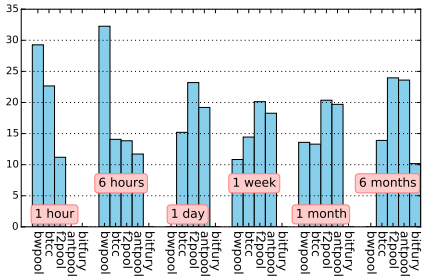


Fig. 3. Proportion of blocks mined by the most represented mining pools according to the epoch length w

and putting this last relation into Relation (11) provides the following result:

$$N_k = \begin{cases} \sum_{\ell=1}^{k} M_{0,\ell} + \sum_{\ell=0}^{w-1-k} M_{\ell,0} \text{ if } k \leq w-1 \\ \sum_{\ell=k-w+1}^{k} M_{0,\ell} \text{ if } k \geq w. \end{cases} \tag{12}$$

Thus, for $k \geq w$, N_k is the sum of w i.i.d. Bernoulli random variables with the parameter μ, so we have

$$\begin{aligned} \mathbb{P}\{W_k \in \mathcal{S}_w\} &= \mathbb{P}\{N_k \leq \lfloor (w-1)/3 \rfloor\} \\ &= \sum_{\ell=0}^{(w-1)/3} \binom{w}{\ell} \mu^\ell (1-\mu)^{w-\ell}. \end{aligned}$$

Note that this expression does not depend on k, for $k \geq w$, meaning that the stationary regime is reached at epoch w. □

Note that the result provided by Theorem 3 is consistent with Relation (7) when w tends to infinity.

Figure 2(c) depicts the proportion of safe execution of BizCoin as a function of w and μ. At the time of writing this paper, the Bitcoin blockchain contains $422,579$ blocks. We derive for $k \geq w$ values of $\mathbb{P}\{W_k \in \mathcal{S}_w\}$ for a size of 1 hour ($w = 6$), 6 hours ($w = 36$), 1 day ($w = 144$), 1 week ($w = 1008$), 1 month ($w = 4320$), 6 months ($w = 25920$). There are two trends depending on μ: against a weak adversary ($\mu \leq 1/3$), the system is safer with a large window, and conversely against a strong adversary ($\mu > 1/3$). The larger the window, the lower the variance (*i.e.* the deviation from the expected Byzantine fraction μ of $\mathcal{E}_w$); a lower variance prevents a weak adversary from randomly gaining power while a higher variance helps honest nodes to "steal" safe runs from strong adversaries.

Considering the effective power of an adversary, we investigate the Bitcoin blockchain. Over the last year, almost all blocks were generated through mining pools, which refer to groups of miners gathering their computational resources so as to increase their probability to successfully mine a block. If the block is effectively appended in the blockchain, its reward is shared among mining pool participants. Mining pools may embed a text data in blocks, allowing them to later identify all the blocks they generated. At the time of writing this paper, around 95.8% of the blocks generated over the last year contain such a text signature.

Figure 3 depicts the proportion of blocks generated by the most important mining pools, namely BWPool, BTCC, F2Pool, AntPool and BitFury, over different sizes w of set $\mathcal{E}_w$. These proportions are derived from two different blockchain trackers [1], [2]. We can note that for all sizes w that we considered for set $\mathcal{E}_w$, no mining pool has generated more than $w/3$ blocks, *i.e.* if we consider that an adversary controls the totality of a mining pool, we have $\mu < 1/3$. In this case, tuning the size of $\mathcal{E}_w$ to 1 week provides a good tradeoff between the probability of safe executions of BizCoin and the algorithmic complexity of these executions. Should these mining pools be colluding, *i.e.* under the control of a unique adversary, they would control around 60% of the miners, which clearly jeopardizes the reliability of BizCoin.

To summarize, we have shown that none of the studied solutions enhances Bitcoin's behavior. Beyond the complexity introduced by the consensus executions, the main issue comes from the fact that all important decisions of Bitcoin are solely under the responsibility of (a quorum of) miners, and the membership of the quorum is decided by the quorum members. This magnifies the power of malicious miners.

IV. Related Work

Bitcoin [20] is considered as the pioneer cryptocurrency systems. Since its inception, several altcoins [3] have emerged. Most of their differences lie in practical details like use of a database [17], block generation time [25], used hashing algorithm [16] or an unlimited number of coins [23]. The GHOST protocol [22] proposes a different rule to solve blockchain forks, based on the number of blocks contained in each blockchain subtree (in case of consecutive forks). Meanwhile, CoinJoin [18] and CoinShuffle [21] propose to mix transactions to avoid user linkability. Recent works have focused on Bitcoin modeling and evaluation. Authors of [19] prove that the Bitcoin protocol achieves consensus with high probability, while [8] show that peers participating in the Bitcoin network agree on a common prefix for the transaction history, both in failure-free environments. In contrast, authors of [10], [11] focused on adversarial environments. These works study the feasibility of double spending attacks and their detection. Finally, as analyzed in this paper, different approaches [6], [5], [13] have been proposed to enforce Bitcoin safety.

V. Conclusion

In this paper, we have formally exhibited the key concepts ruling the Bitcoin protocol. These concepts are used to derive fundamental properties of Bitcoin. To the best of our knowledge, this is the first time that they are highlighted. We then study three recent propositions aiming at enforcing strong consistency in Bitcoin. These propositions exclusively rely on miners. We have shown that *i)* none of these propositions is safe in an adversarial environment, *ii)* worse, these solutions amplify the ability of malicious users to exploit Bitcoin flaws. We are currently working on the protocol vulnerabilities related to double spending, and implementing our solution to demonstrate its feasibility and evaluate its performance in a real setting.

References

[1] Bitcoin Network Hashrate - Bitcoinity.org. https://data.bitcoinity.org/bitcoin/hashrate/, 2016.

[2] BlockTrail — Bitcoin API and Block Explorer. https://www.blocktrail.com/BTC, 2016.

[3] S. Ahamad, M. Nair, and B. Varghese. A survey on crypto currencies. In *Proceedings of the International Conference on Advances in Computer Science (AETACS)*, 2013.

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

[5] C. Decker, J. Seidel, and R. Wattenhofer. Bitcoin Meets Strong Consistency. In *Proceedings of the International Conference on Distributed Computing and Networking (ICDCN)*, 2016.

[6] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse. Bitcoin-ng: A scalable blockchain protocol. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.

[7] N. Fincham. https://mineforeman.com/2013/03/14/what-the-fork-was-that-a-forking-post-mortem/.

[8] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques - Advances in Cryptology (EUROCRYPT)*, 2015.

[9] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić. The Next 700 BFT Protocols. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, 2010.

[10] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS)*, 2012.

[11] G. O. Karame, E. Androulaki, M. Rzeschlin, A. Gervais, and S. Capkun. Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Transactions on Information and System Security*, 2015.

[12] E. Kokoris-Kogias, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Poster: Bitcoin meets collective signing. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2016.

[13] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2016.

[14] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: Speculative byzantine fault tolerance. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, 2007.

[15] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 1982.

[16] Litecoin. Global Decentralized currency based on blockchain technology. https://litecoin.org, 2011.

[17] A. Loibl. Namecoin. http://namecoin.info/, 2014.

[18] G. Maxwell. CoinJoin: Bitcoin privacy for the real world. https://en.wikipedia.org/wiki/CoinJoin, 2013.

[19] A. Miller and J. LaViola Jr. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. Available on line: http://nakamotoinstitute.org/research/anonymous-byzantine-consensus/, 2014.

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

[21] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, 2014.

[22] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013:881, 2013.

[23] S. N. Sunny King. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012.

[24] E. Syta, I. Tamas, D. Visher, D. Wolinsky, L. Gasser, N. Gailly, and B. Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2016.

[25] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. http://gavwood.com/Paper.pdf.