RepuCoin: Your Reputation is Your Power

Jiangshan Yu *, David Kozhaya†, Jeremie Decouchant‡, and Paulo Esteves-Verissimo‡
* *Monash University, Australia.*
† *ABB Corporate Research, Switzerland.*
‡ *SnT, University of Luxembourg, Luxembourg.*

Abstract—**Existing proof-of-work cryptocurrencies cannot tolerate attackers controlling more than 50% of the network's computing power *at any time*, but assume that such a condition happening is "unlikely". However, recent attack sophistication, e.g., where attackers can rent mining capacity to obtain a majority of computing power temporarily, render this assumption unrealistic.**

This paper proposes RepuCoin, the first system to provide guarantees even when more than 50% of the system's computing power is temporarily dominated by an attacker. RepuCoin physically limits the rate of voting power growth of the entire system. In particular, RepuCoin defines a miner's power by its 'reputation', as a function of its work integrated over the time of the entire blockchain, rather than through instantaneous computing power, which can be obtained relatively quickly and/or temporarily. As an example, after a single year of operation, RepuCoin can tolerate attacks compromising 51% of the network's computing resources, even if such power stays maliciously seized for almost a whole year. Moreover, RepuCoin provides better resilience to known attacks, compared to existing proof-of-work systems, while achieving a high throughput of 10000 transactions per second (TPS).

1. Introduction

Bitcoin [40] is the most successful decentralized cryptocurrency to date. However, despite its enormous commercial success, many weaknesses have been associated with Bitcoin, including weak consistency, low transaction throughput and vulnerabilities to attacks, such as double spending attacks [27, 3], eclipse attacks [20, 23], selfish-mining attacks [18, 47], and flash attacks [11].

Several promising existing solutions [34, 45, 14, 48, 19] targeted the low throughput problem of Bitcoin. Nevertheless, these solutions either provide only probabilistic guarantees about transactions (weak consistency) [19] or can provide strong consistency but suffer from liveness problems even when an attacker has a relatively small computing power [30]. Moreover, the resilience of such solutions against attacks, such as selfish mining attacks where an attacker has more than 25% of the computing power [3, 18, 47, 42], remains unsatisfactory. In addition, all existing contemporary proof-of-work (PoW) based variants of Bitcoin (e.g. [2, 44, 19, 30]) rely on the assumption that an attacker cannot have more than 33% or 50% of computing power *at any time*. However, with the sophistication of attacks mounted on Bitcoin, e.g., flash attacks (a.k.a. bribery attacks), where an attacker can obtain a temporary majority (>50%) of computing power by renting mining capacity [11], all these systems would fail. In brief, existing solutions that address the weaknesses associated with Bitcoin still suffer from significant shortcomings.

This paper addresses these shortcomings —liveness of current high-throughput solutions, and vulnerability to attacks such as selfish mining and flash attacks. In particular, we propose RepuCoin, the first system that can prevent attacks against an attacker who may possess more than 50% computing power of the entire network temporarily (e.g., a few weeks or even months). Our proof-of-concept implementation shows that while providing better security guarantees than predecessor protocols, RepuCoin also ensures a very high throughput (10000 transactions per second). In practice, Visa confirms a transaction within seconds, and processes 1.7k TPS on average [51]. This shows that RepuCoin satisfies the required throughput of real world applications.

Design principle. Our system addresses the aforementioned challenges by defining a new design principle, called *proof-of-reputation. Proof-of-reputation* is based on proof-of-work, but with two fundamental improvements.

First, under proof-of-reputation a miner's decision power (i.e., the voting power for reaching consensus in the system) is given by its reputation. A miner's reputation is not measured by what we call the miner's 'instantaneous' power, i.e., the miner's computing power in a short time range, as in classic PoW. Instead, the reputation is computed based on both the *total amount of valid work* a miner has contributed to the system and the *regularity of that work*, *over the entire period of time* during which the system has been active. We call this the miner's 'integrated power'. So, when an attacker joins the system at time t, even if it has a very strong mining ability that is, high computational (i.e., instantaneous) power, it would have no integrated power at time t, or even shortly after, as it did not contribute to the system before t.

Second, when a miner deviates from the system specifications, RepuCoin lowers the miner's reputation, and hence its integrated power, in consequence of this negative contribution. This prevents a powerful malicious miner from attacking the system repeatedly without significant consequences. In contrast, classic PoW systems either do not support any feature for punishing miners that do not abide by system specifications, or they punish these miners by merely revoking their rewards — this does not prevent them from attacking the system again immediately after.

To improve the robustness and the liveness of the system, we further improve the elegant concept of decoupling leader election from transaction serialization. In its original form (as proposed in Bitcoin-NG [19] and adopted by ByzCoin [30]), a miner creates *keyblocks* by solving the Bitcoin mining puzzle and may become a leader for a period of time. A leader can then verify and include transactions into *microblocks* directly. While providing a good solution to the throughput problem of BitCoin, decoupling leader election from transaction serialization as suggested by [19, 30] also allows any miner that created a keyblock to disrupt the system's liveness temporarily, e.g., by becoming a leader and not issuing any microblocks. The success rate of such a miner is proportional to the miner's instantaneous power. In other words, a malicious miner with x% computing power can reduce the throughput to (100-x)% of the average value. We improve on this situation, by selecting leaders at random only from the top reputable miners. To further broaden the reputation reward for 'good' actions, RepuCoin splits the transaction fees carried in the microblocks into two parts. The keyblock creator will obtain one part of the transaction fees at an amount determined by its reputation, and the randomly selected leader gets the remainder.

RepuCoin provides deterministic guarantees on transactions by employing a *reputation-based weighted voting consensus*. Consensus is carried by a group formed of the top reputable miners. Every member of that group has a weight associated to its vote. The weight of a member's vote is the percentage of that member's reputation w.r.t. the entire group's reputation. Such weights ensure that one's voting power depends, not on the sheer instantaneous (computing) power — which is the enabler of flash attacks— but on the integrated power, which both takes time to build, and is built on miner's honesty and historical performance. In fact, quantifying a miner's voting power based on its performance in the entire blockchain highlights the self-stabilizing characteristics of our approach: qualitatively, to acquire power in RepuCoin a miner is urged to exhibit normal, honest behavior; quantitatively, the speed with which a miner can gain power is dictated by the regularity (rhythm) and amount of that miner's contributions in the entire blockchain.

To illustrate the robustness of the design choices underlying RepuCoin, we present our analysis of the security provided by the mechanisms used in RepuCoin. In particular, we show that achieving the safety and liveness correctness conditions of the reputation-based weighted voting consensus protocol is physically guaranteed by the growth rate of the proof-of-reputation function, and the rate of decision power growth of the entire system is bounded. In addition, we present experiments exemplifying concrete values for the decision power growth in several situations, showing that the network achieves very high stochastic robustness against attacks on its liveness or safety. For instance, we demonstrate that after a single year of operation, RepuCoin is resilient to all attacks that compromise 26%, 33%, and 51% of the network's computing resources, even if such power stays maliciously confiscated for almost 100 years, 2 years, and 1 year respectively. Also, in the same setting, even if an attacker can afford to seize a huge computational power for a specific period, due to the cost of such attacks (e.g., 90% for up to 3 months), it will not break RepuCoin. Moreover, we provide an analysis of the non-rationality of infiltration attacks, with a comparison of the cost of attacking different systems. Furthermore, we provide in detail how RepuCoin prevents known attacks.

In short, RepuCoin provides significant resilience to attacks that may instantaneously break all known PoW based systems [30, 38, 2, 22, 31].

2. Background and Related Work

Consensus is the key component and backbone of Blockchains, which use two main types of consensus mechanisms, namely proof-of-X based consensus, and Byzantine-fault tolerant (BFT) consensus. The former is generally permissionless, where anyone can join and leave the potentially large consensus group; and the latter is permissioned, where the set of participants running consensus is small and predefined.

Proof-of-X based consensus has gained much interest since the use of Proof-of-work (PoW) in Bitcoin, already described in the introduction. Proof-of-stake, which was first discussed in a Bitcoin community forum [46], has been proposed to use virtual voting to provide quicker transaction confirmation. Proof-of-space (a.k.a. proof-of-capacity) [37, 43] has been proposed to use physical storage resources to replace computing power in the proof-of-work mechanism. Proof-of-coin-age [29] shares a similar concept as proof-of-stake, as participants also perform virtual "mining" by demonstrating possession of a quantity of currency. Proof-of-activity [8] puts every coin owner into a mining lottery; periodically, winners are randomly determined by transactions. A winner is expected to respond with a signed message within a small time interval to claim its award. Proof-of-elapsed time [25], proposed by Intel and implemented as a Hyperledger project, uses Intel SGX enabled CPUs to do virtual voting through a random sleeping time to replace the proof-of-work mechanism. Proof-of-membership system in ByzCoin [30] creates a consensus group formed by recent PoW-based block creators, and this group runs a BFT protocol for reaching consensus. However, it is shown that ByzCoin still suffers from selfish mining attack [30], and that it can permanently lose liveness during reconfiguration, if too many miners disappear in a short time [26] or can repeatedly lose liveness temporarily [30]. Unfortunately, none of the permissionless consensus protocols described is able to provide a meaningful security guarantee when an attacker is able to control a majority of mining power.

Notable state of the art of BFT protocols include PBFT [13], Zyzzyva [32], MinBFT/MinZyzzyzva [52], and ReBFT [15].

PBFT [13] proposes a Byzantine fault-tolerant algorithm that has an acceptable performance in practice. The protocol is based on a primary replica that orders and propagates client requests using Byzantine quorums. To tolerate a primary replica being faulty, the protocol allows replicas to trigger view changes through which a new replica is selected as the primary of the view. PBFT is indulgent, hence guarantees safety properties regardless of the system's synchrony, and assumes F=n-1/3 Byzantine faulty replicas during the lifetime of the system, attackers cannot forge MACs. PBFT also allows faulty replicas to recover. This allows the system to tolerate any number of faulty nodes during the system's lifetime given that no more than n-1/3 are simultaneously faulty in a small vulnerably window.

Zyzzyva [32] is a speculation-based BFT protocol that reduces cryptographic over-heads and increases peak throughput for demanding workloads compared to traditional state machine replication. In Zyzzyva [32] replicas execute client requests speculatively, i.e., without running an agreement protocol on the order of execution of these requests. Clients on the other hand, observe the history information carried within replica replies to determine if the replies and history are stable and guaranteed to be eventually committed. Consequently, clients only use those replies with a stable history. Clients can speed the process of converging to stable histories by either supplying helpful information within the current view or by triggering a view change. However, a recent analysis [1] identified a safety violation in Zyzzyva.

MinBFT/MinZyzzyva [52] presents non-speculative and speculative BFT algorithms that are efficient in terms of three metrics: number of replicas, trusted service simplicity, and number of communication steps. In particular, MinBFT protocols require 2f+1 replicas in total rather than 3f+1. This reduction is achieved by relying on a simple trusted service - a trusted monotonic counter, used to associate sequence numbers to each operation. Besides being simple in design, an important aspect of this trusted entity that allows it to be implemented even on COTS trusted hardware is a clear simple interface for interaction. The interface provides operations only to increment the counter and to verify if other counter values are correctly authenticated. In nice executions, the proposed protocols run in the minimum known number of communication steps for non-speculative (4 steps) and speculative algorithms(3 steps).

ReBFT [15] is an approach that relies on a passive-replication paradigm to minimize the number of non-faulty replicas that participate in system operations in the absence of faults. This reduction of the resource footprint of BFT protocols does not jeopardize the system's liveness when faults actually occur. This is achieved by defining two operation modes: a normal mode (when no faults are detected) and fault-handling mode (when faults are detected). In the normal mode only a subset of replicas is active. Upon the suspicion of some fault the remaining replicas are activated. Besides reducing the resources used, ReBFT does not require a complete design of a BFT system. In fact, it could be introduced as a subprotocol in existing BFT protocols such as PBFT [13] and MinBFT [52].

3. System and Threat Model

3.1. System Context

We adopt the concept of separating the proof-of-work (keyblocks) from committing transactions (microblocks) [19]. However, unlike previous work we further refine this separation as follows. As in Bitcoin-NG, miners in RepuCoin solve cryptographic puzzles to create keyblocks. However, in RepuCoin the miner who creates a keyblock is not necessarily the one who commits transactions into microblocks. Instead, one of the most reputable miners is randomly elected to commit transactions. The keyblock creator obtains a share of the transaction fees according to its reputation, and the selected leader gets the remaining transaction fees as a reward for its work.

A miner can build its reputation by producing work at a high enough rhythm, and can also lose its reputation if it did not perform as expected (see §4.3 for details). In this case, the miner's reputation is affected negatively depending on the slowdown. This not only eliminates attacks where a malicious miner becomes a leader and makes the system lose liveness by not issuing any microblock, but also incentivises miners to increase their reputation to gain extra profit.

All participants of the system, no matter whether they are miners or mobile clients, learn about new transactions and blocks in the same way as in BitCoin and other blockchains, through a peer-to-peer protocol.

In RepuCoin, we verify and commit microblocks, as well as decide on the keyblocks to be added to the blockchain, using Byzantine fault tolerance (BFT) protocols with minor modifications, as presented in §4.4. Such form of agreement prevents a malicious leader from double spending a coin, and resolves potential forks resulting from simultaneously mined keyblocks. According to Byzantine quorums theory [36], in order to reach an agreement, classic BFT protocols require votes from at least $2f + 1$ nodes[1], to prevent an adversary controlling f nodes from breaking the protocol. *In practice, however, an open BFT-based system cannot guarantee that an attacker will never be able to control more than f nodes.* To enforce the assumption that no more than f Byzantine nodes are ever involved in a consensus, we introduce novel mechanisms to make it infeasible, in practice, for an attacker to seize f nodes within the consensus group, as detailed below.

3.2. System Model

RepuCoin is a system composed of a non-predetermined number of nodes, called *miners*. Each miner has a reputation score, which determines that miner's ability of obtaining rewards. A miner's reputation score is based on the correctness of its behavior and its regularity in adding blocks to the existing chain, hence correlated with the miner's computing

1. Hybrid BFT protocols with trusted hardware components, such as MinBFT [52], only require votes from $f + 1$ nodes.

power. RepuCoin considers a network that is untrustworthy and unreliable. In addition, we assume the network has partial synchrony [16].

To address the above-mentioned uncertainty (Section 3.1) in the definition of the consensus quorums and outcomes for open BFT-based protocols, RepuCoin resorts to two main techniques. First, we rely on the notion of having a *consensus group*, which is a subset of the miners denoted by $\mathbb{X}$. This group is capable of controlling the operations of RepuCoin, namely running the consensus protocol. The members of the consensus group are selected from the miners with the highest reputation scores. A miner's reputation score is based on the correctness of its behavior and its regularity when adding blocks to the existing chain, which is correlated with the miner's computing power. Second, the voting rules in the underlying consensus schemes are not nominal, but based on a novel *reputation-based weighted voting*. To this end, we refine the definition of quorums as follows: reaching agreement not only requires votes from $2f+1$ nodes, but also demands that these nodes collectively have more than $\frac{2}{3}$ of the cumulative reputation of the consensus group.

The consensus group size is defined as the minimum number of miners with enough decision power (i.e., cumulative reputation) to ensure safe and live control of the system, given our quorum definition. Therefore, the consensus group members are obviously the miners with the highest reputation scores. This stratagem has two virtuous effects. First, RepuCoin's safety is guaranteed by consensus, which can be viewed as a deterministic control orchestrated by a set of miners with overwhelming cumulative reputation. By definition, such reputation itself gives an expectation of the correct behavior of these miners. Second, openess and fairness of RepuCoin relies on $\mathbb{X}$ being parametric and agnostic of identity of network members, as we show in Section 4.2. At configuration time, the size of $\mathbb{X}$ is calculated by meeting a target percentage of the overall decision power, and so it can be large or small, depending on the mining pool composition, but not pre-determined. On the other hand, as miners gain or lose reputation, they can (by merit or demerit) enter and leave the consensus group.

3.3. Threat Model

We consider a malicious (a.k.a. Byzantine) adversary, who can arbitrarily delay, drop, re-order, insert, or modify messages. We also consider collusions of an arbitrary number of miners, to model a malicious real organization capable of deploying a significant number of virtual miners under its direct dependence. We assume the security of the used cryptographic primitives, including a secure hash function and a secure signature scheme.

Such adversary can potentially control as many miners as it wishes, and coordinate them in real time with no delay. In consequence, the consensus group can be infiltrated by adversaries. However, we assume that the adversary has the ability to control at most $f \leq \lfloor \frac{|\mathbb{X}|-1}{3} \rfloor$ group members whose collective reputation is less than $\frac{1}{3}$ of the cumulative reputation of the members of consensus group $\mathbb{X}$.

The coverage of this assumption, i.e., how to constrain the adversary's ability of infiltrating the consensus group, to meet the aforementioned assumption, is explained in Section 4.3. The dynamics of the reputation mechanism allow the consensus group controlling RepuCoin to be infiltrated with safety, making it practically infeasible for an attacker to break the system (as shown in Table 3 of Section 6).

4. RepuCoin

In this section, we present details describing the different concepts and modules underlying RepuCoin. In particular, Section 4.1 details the different types of blocks, the leader election mechanism, and the reward system proposed. Then, we present our reputation-based weighted voting mechanism in Section 4.2, and the reputation system in Section 4.3.

4.1. Block Mining and Reward System

As mentioned earlier, in order to support higher throughput rates, RepuCoin decouples leader election (keyblocks) from transaction serialization (microblocks).

Keyblock and Leader Election. Miners solve Bitcoin-like puzzles to create keyblocks, and receive rewards corresponding to keyblock creations. However, the keyblock creator is not necessarily the leader that commits transactions into microblocks. Rather, the leader is randomly elected from the reputable miners. The puzzle is defined as follows:

$$H(prev\_keyblock\_hash || Nonce || PK) < target$$

where $H(\cdot)$ is a cryptographically secure hash function, $prev\_keyblock\_hash$ is the hash value of the previous keyblock, PK is the miner's public key, which the miner's reputation score R is associated with, and $target$ is a target value defined by the system. (For simplicity, we use reputation score and reputation interchangeably in this paper.)

RepuCoin solves forked chains on the fly by dynamically forming the consensus group and agreeing on which chain to choose. The consensus group members are the top reputed miners in the mining network. The reputation score of miners can be calculated by using data from the blockchain, and is maintained locally by each miner. When different miners have the same reputation, a naive solution would be to order them according to their public key PK (a.k.a. address). However, this gives a miner with small PK an advantage. To avoid this, in RepuCoin, miners with the same reputation score are ordered by $H(PK, R)$, where reputation R (and therefore the hash value) is updated each time a new keyblock becomes part of the blockchain.

Each time a new keyblock is created, the creator proposes it to the consensus group. The group verifies the received keyblocks, and runs the underlying Byzantine agreement protocol to decide which keyblock to choose (if multiple conflicting keyblocks are proposed). We call a keyblock that is agreed upon and signed by the group a *pinned* keyblock. A pinned keyblock is final and canonical, it defines the unique global blockchain from the very first block (known as the genesis block) up to the pinned keyblock. All keyblocks that conflict with a pinned keyblock are considered invalid. New keyblocks are mined based on the hash value of the previous pinned keyblock. The format of a pinned keyblock is as follows:

$$\boxed{\overbrace{\text{prev\_keyblock\_hash} \mid \text{Nonce} \mid \text{PK} \mid \text{R} \mid \text{K\_sig}}^{\text{keyblock}} \mid \overbrace{\text{sig\_keyblock\_agreement}}^{\text{keyblock pin}}}$$

The (new) $keyblock\_hash$ is the hash of a pinned keyblock, i.e., all the material in the frame above, where K_sig is a signature on the hash value of ($prev\_keyblock\_hash$, $Nonce$, PK, R), and $sig\_keyblock\_agreement$ is the signed agreement from the consensus protocol on committing this keyblock. The first keyblock is called the genesis block (as in Bitcoin), which is defined as part of the system. Note that to verify a keyblock, consensus group members check the validity of K_sig, the solution to the mining puzzle, and the reputation R.

Each time a new keyblock is pinned, the next leader — which verifies transactions and commits them into microblocks — is selected as follows:

$$l_i := x_j \quad s.t.\ x_j \in \mathbb{X} \ \wedge \ j = H(K\_sig_i) \mod |\mathbb{X}|$$

where l_i is the i-th leader determined by the hash value $H(K\_sig_i)$ of the signature K_sig_i contained in the i-th pinned keyblock, and $\mathbb{X}$ is the set of miners constituting the consensus group. Since a cryptographically secure hash function is considered a random oracle, the leader is selected randomly in the consensus group with probability $\frac{1}{|\mathbb{X}|}$. However, one concern with this leader selection process is that consensus group members can determine the following leader before pinning a block. Thus, a consensus member interested in getting more rewards would only accept (decide to pin) a block that makes itself the new leader. To address this issue, a simple approach is to determine a leader by using $H(sig_i)$ instead of $H(K\_sig_i)$, where sig_i is a signature on the current length of the blockchain issued by the keyblock creator. Note that it is important to issue this signature only after the keyblock has been pinned by the consensus group. This way, each consensus member would accept a block with equal probability.

Remark 1. Note that in consensus schemes, a sufficient number of signed votes determines an agreement, and only one valid agreement can be reached. However, different combinations of (a sufficient number of) signed votes can be used to form this agreement. So, even though the agreement is unique, the collection (i.e., $sig\_keyblock\_agreement$) of signed votes may have different valid values. This results in different valid $prev\_keyblock\_hash$ on the same keyblock.

Hence, miners may solve their puzzles based on different valid $prev\_keyblock\_hash$. However, this will not form a fork in the system, as only one of the valid solutions to the puzzle will be validated by the consensus group, as detailed in §4.2. In addition, upon reaching the agreement on the next keyblock, the $prev\_keyblock\_hash$ is considered the only valid hash value of the previous pinned keyblock, and only this value will be recorded in the blockchain.

Moreover, the reason why miners in RepuCoin solve puzzles based on $prev\_keyblock\_hash$ rather than based on $H(K\_sig)$ is the following. Consensus group members learn $H(K\_sig)$ earlier than other miners in the system, and this gives these members extra mining advantage, if the puzzle is only based on $H(K\_sig)$. Using $prev\_keyblock\_hash$ in the puzzle prevents these members from getting such advantage.

Microblock. The current leader commits transactions into microblocks. To prevent double spending, each microblock is proposed to the consensus group before being accepted, and hence committing the transactions it encompasses. The group members verify the microblock, and initiate a consensus instance to agree on that microblock, which upon agreement is called a *pinned* microblock.

The format of a pinned microblock is shown below:

$$\boxed{\overbrace{\text{H(K\_sig)} \mid \text{prev\_microblock\_hash} \mid \text{TXs} \mid \text{M\_sig}}^{\text{microblock}} \mid \overbrace{\text{sig\_microblock\_agreement}}^{\text{microblock pin}}}$$

where $H(K\_sig)$ is the hash value of the K_sig contained in the current pinned keyblock, $prev\_microblock\_hash$ is the hash value of the previous pinned microblock; TXs is a set of transactions organized as a Merkle tree, M_sig is a signature on the hash value of $(keyblock\_hash, prev\_microblock\_hash, TXs)$, and $sig\_microblock\_agreement$ is the signed agreement from the consensus protocol on the microblock. In this sense, in order to verify a microblock, consensus group members check the validity of M_sig, verify the hash values of the keyblock and the previous microblock, and verify the set of transactions TXs. If invalid transactions are detected, then the leader is punished, as presented in §4.3. The (new) $microblock\_hash$ is the hash of the microblock (without a pin). In other words, the 'microblock pin' presented in the above frame is not part of the hash function's input. In this way, a leader can issue microblocks without waiting for the agreement, which optimizes the throughput of RepuCoin. The blockchain structure containing both keyblocks and microblocks is presented in Figure 1.

Reward system. In RepuCoin there are two types of rewards, namely mining rewards and transaction fees. Upon successfully mining a keyblock, a miner is entitled to get a reward, precisely if that miner's keyblock gets pinned. This mining reward is of a pre-set amount.

Every transaction within the microblock carries a transaction fee. The randomly elected leader shares the transaction fees with the miner of the pinned keyblock according to Algorithm 1. Roughly speaking, the miner's reputation determines the number of microblocks from which it can obtain transaction fees; the leader gets the rest. However, a leader that can determine the microblocks from which it gets

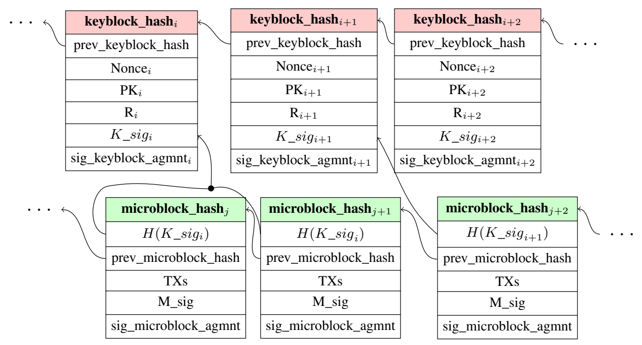


Figure 1: A figure presentation of the blockchain structure. This example contains three pinned keyblocks (i, $i+1$, and $i+2$), and three pinned microblocks ($j, j+1$ and $j+2$). Microblock j and $j+1$ were created by the leader determined by keyblock i, and microblock $j+2$ was created by the leader determined by keyblock $i+1$.

transaction fees, may optimize its income by putting transactions with higher transaction fees into these microblocks. To avoid this unfair game, RepuCoin uses the hash value of the next pinned keyblock, i.e., the keyblock pinned at the end of the new epoch, to decide which pinned microblocks are allocated to the miner and which go to the leader. Since the hash value of the next pinned keyblock cannot be predicted, RepuCoin eliminates the above situation.

More precisely, let $\mathbb{M} = \{m_0, m_1, \ldots, m_{n-1}\}$ be the sequence of n microblocks that are pinned by the consensus group. Let $R \in [0, 1]$ be the reputation score of the miner which creates the $(i-1)$-th pinned keyblock. The transaction fees contained in the set $\mathbb{M}'$ and $\mathbb{M}''$ of microblocks are shared between the miner of the $(i-1)$-th pinned keyblock and the leader, respectively, as shown in Algorithm 1.

Algorithm 1 Reward sharing algorithm

Input: The sequence $\mathbb{M} = \{m_0, m_1, \ldots, m_{n-1}\}$ of microblocks pinned in the $(i-1)$-th epoch, the signature K_sig_i contained in the i-th pinned keyblock, and the reputation R of the miner who created the $(i-1)$-th keyblock.
Output: Two subsets $\mathbb{M}', \mathbb{M}'' \subseteq \mathbb{M}$ of microblocks, where transaction fees contained in $\mathbb{M}'$ (resp. $\mathbb{M}''$) are allocated to the miner (resp. the leader) as reward.

```
1: $i' = H(K\_sig_i) \mod n$
2: $k = 0$
3: $\mathbb{M}' = \emptyset$
4: while $k < R \cdot n$ do
5:     $j = i' + k \mod n$
6:     $\mathbb{M}' = \mathbb{M}' \cup \{m_j\}$
7:     $k = k + 1$
8: end while
9: $\mathbb{M}'' = \mathbb{M} \setminus \mathbb{M}'$
```

The way transaction fees are shared between keyblock creators and leaders motivates miners to increase their reputation. First, keyblock creators with higher reputation gain higher shares of rewards. Second, highly reputed miners constitute the consensus group; hence they can become leaders and get shares of transaction fees.

Similar to the Bitcoin system, to spend a reward, the miner simply makes transactions by using the SK which is associated with the PK contained in the keyblock, and provides the hash of the keyblock or microblock as an input of the transaction.

In Bitcoin, a miner needs to wait a maturity period of 100 blocks to avoid non-mergeable transactions from forks. In RepuCoin, each pinned keyblock and its underlying pinned microblocks are canonical, so leaders do not need to wait for this period to avoid non-mergeable transactions.

Remark 2. With RepuCoin, miners, even the newly joined ones with initial reputation, will get much more reward than what they can get in Bitcoin.

There are two types of rewards, namely mining reward and transaction fees. For the former, every successful miner, even a new joiner, gets full reward, like in Bitcoin. For the latter, the miner shares the transaction fees with the leader of the next epoch. However, even though the transaction fees are shared between the miner and the leader, the miner gains more reward than what it can gain in the Bitcoin system. The reason for this is that, with RepuCoin, more microblocks (so transactions) are committed in an epoch, which is the time period between any two successive keyblocks. Taking the existing parameter, an epoch is 10 minutes in average. Our analysis in §5.3 shows that the throughput is about 1000 TPS in RepuCoin, and 7 TPS in Bitcoin. This means that, in total, the transaction fees of 600k transactions will be shared between a miner and a leader in RepuCoin, according to our Algorithm 1. In contrast, a miner gets all transaction fees of only 4200 transactions in average in Bitcoin. So, even though the miner in Bitcoin gets all transaction fees, it only earns the fees of 4k transactions, whereas in RepuCoin, a miner shares the transaction fees of 600k transactions with a leader. Taking the parameter used in our performance evaluation (§5), even a new joiner would get 40% shares of the transaction fees. Thus, with RepuCoin, a new joiner will gain a full mining reward and transaction fees of roughly 240k transactions, whereas with Bitcoin, a miner only gets the transaction fee of roughly 4k transactions. Therefore, even for a newly joined miner, with RepuCoin, it gains the same mining reward as with Bitcoin, and gains transaction fees that are at least 60 times as high as with Bitcoin.

4.2. Block Pinning

In RepuCoin, we use consensus to pin both keyblocks and microblocks. Transactions belonging to pinned microblocks cannot be unrolled at a later point in time. In this section, we describe in more detail the underlying consensus mechanism we use to pin such blocks.

Byzantine fault-tolerant consensus algorithms typically rely on processes voting. A process needs to collect a quorum of votes on a given value/action for that value/action to be considered legal by the system. The size of the quorum is selected in a way that guarantees (i) safety of decisions, e.g., avoiding conflicting decisions, and (ii) liveness of the system, i.e., miners should be able to hear eventually a number of votes on some value/action from a quorum of miners. It has been shown that in systems, as soon as $\frac{1}{3}$ or more of the miners are compromised, an attacker can make the system inconsistent — an attacker can make different parts of the system decide differently [36]. In order to make our system robust to such attacks, we propose to modify the traditional nominal voting mechanism, i.e., hearing from a sufficient number (quorum) of miners, by requiring as well to hear from a sufficient number of miners such that their added reputation is above a defined threshold. Such a modification prevents an attacker from breaking the correctness of the system directly upon compromising any $\frac{1}{3}$ of the miners: it should compromise as well enough miners that their added reputation is at least $\frac{1}{3}$ of the total reputation of the consensus group. Details on how to adapt a chosen BFT protocol to RepuCoin are presented in §4.4.

Consensus in RepuCoin employs a novel *reputation-based weighted voting mechanism*, i.e., rather than treating each vote from consensus group members equally (e.g., as in classic Byzantine protocols), the weight of each vote becomes its reputation over the total reputation of the group. More precisely, let $\{x_1, \ldots, x_{|\mathbb{X}|}\}$ be the consensus group, and each member x_i of the group has its reputation score R_i. The weight of x_i's vote is $\frac{R_i}{\sum_{i=1}^{|\mathbb{X}|} R_i}$, for all possible x_i.

Instead of only waiting to hear from at least $2f + 1$ nominal members to validate a value or an action, it is also necessary that the collective reputation of those members is more than $\frac{2}{3}$ of the total reputation of the consensus group.

Remark 3. Weighted-voting [21] is a classic and well known concept. The novelty of our weighted-voting system comes from the way that this weight is defined. More precisely, the weight of a miner's vote is given by this miner's reputation, i.e., its 'integrated power'. In particular, it considers the quantity and regularity of contributions over the entire blockchain, and provides a model to punish misbehaved miners (see section 4.3). In other words, we constrain the evolution of reputation over time. Thus, unlike in traditional systems, it becomes significantly difficult for an attacker to re-obtain enough voting power after a deviation, or to flash-build it like in flash attacks.

The effectiveness of our weighted-voting scheme is confirmed in our analysis in section 6. For example, it shows that a 3-month-late-joining attacker needs 90% computing power of the entire network to successfully attack the system in one month; this attack becomes infeasible for 6-month-late-joiners. However, in classic systems an attacker with 90% computing power can join at any time and successfully attack the system quickly.

Consensus Group. As mentioned in Section 3.2, the size $|\mathbb{X}|$ of the consensus group is not pre-determined, but rather calculated by meeting a target percentage[2]of the overall decision power. We select the members of the consensus group based on our reputation system; namely, the $|\mathbb{X}|$ miners with the top reputations constitute the members.

We define an epoch as the period between any two successive keyblocks that become part of the blockchain. Every epoch possesses a leader, which is the miner that should issue a maximum pre-specified amount of microblocks. In every epoch, the reputation of only one miner, the creator of that pinned keyblock, may gain an extra increase; the reputation of all other miners would only have a very minor change according to Algorithm 2, or drops to "0" if they lie (see Section 4.3). Accordingly, given that $f \leq \lfloor \frac{|\mathbb{X}|-1}{3} \rfloor$ can be malicious, the members of the consensus group in any two consecutive epochs can differ by at most f members. This stability in the members of the consensus group of consecutive epochs ensures the safety of consensus decisions. Namely, at the beginning of a new epoch correct consensus group members are aware of all committed transactions and hence do not accept/validate any conflicting transactions proposed by the new leader.

Committing Microblocks. The leader of the current epoch, issues transactions in the form of microblocks. After generating a microblock, the leader initiates a consensus instance for this microblock proposing an accept to commit that microblock. Other consensus members will propose to either accept or decline (by not accepting) this microblock, depending on the transactions contained within and of course their validity. In other words, if transactions within the micro-block are invalid then members should decline committing that microblock. The leader continues to issue microblocks that are proposed to the consensus group for validation and commitment, until a new leader is elected.

Committing Keyblocks. Upon successfully mining a keyblock, the miner of that block sends that keyblock to all members of the consensus group.

Upon receiving a keyblock, this group member initiates a consensus instance proposing the received keyblock (first received keyblock in the case when many such keyblocks are received). As a result, the members of the consensus group decide on a single keyblock to be part of the blockchain. The miner of that block is termed as the "winner". The hash of the new keyblock output by the consensus decides which member of the consensus group becomes the leader of the current epoch as previously mentioned.

A member of the consensus group that successfully decides on the identity of the new leader stops validating microblocks relative to the previous leader. Afterwards, that member initiates a consensus instance to agree on the total set of committed microblocks. Consensus group members need to agree on the total set of microblocks, since a leader is selected from this group. A leader that does not know the total set of committed microblocks might propose microblocks that are in conflict with committed ones and

2. Similar to the parameters of other systems, such as the block size of Bitcoin or the window size of ByzCoin, the target percentage of the overall decision power is a system parameter that can be reconfigured if necessary.

accordingly lose its reputation, not out of maliciousness but simply out of lack of knowledge. To avoid this situation, every member after reaching a decision on the identity of the new leader submits to consensus the largest sequence number of microblocks that have been committed along with a verifiable proof of this claim. Namely we assume that all consensus algorithms we use are implemented using digital signatures. As such, having a sufficient number N of signatures on a decision constitute a proof of its validity. We say the number N of signatures is sufficient if $N \geq 2f+1$ and if the total reputation of the miners issuing these N signatures is more than $\frac{2}{3}$ of the total reputation of the group.

Upon reaching a decision on the new leader and on the global set of committed microblocks, each consensus group member also sends a message to notify the winner, the current leader, and the newly elected leader of this result. A consensus group member waits to either hear from consensus or from a sufficient number N' of other group members about the identity of the new leader and the global set of committed microblocks, before it adopts that member as leader. We say N' is sufficient if the total reputation of the issuers of the N' signatures is more than $\frac{1}{3}$ of the total reputation of the group and if $N' \geq f+1$. In that case, the current leader simply stops issuing microblocks and the new leader takes over proposing microblocks.

Optimizing Agreement on Committed Microblocks. In order to have agreement on the set of committed microblocks without resorting to a consensus instance, we propose the following optimization. Due to the PoW, it is known that mining a keyblock successfully takes a certain time depending on the mining difficulty (e.g. on average 10 minutes in Bitcoin). If we assume that a leader issues microblocks to be committed at a pre-specified rate, then we can assume that on average a leader commits m microblocks per epoch. Accordingly, all consensus group members do not validate or commit more than m microblocks for any given leader. Upon reaching a decision on the identity of a new leader (as a result of having a new keyblock mined) a consensus group member only initiates consensus on the set of committed microblocks if it has not seen m committed microblocks from the previous leader.

The benefits of fixing the number of microblocks (that a leader can commit) to m microblocks per epoch extends beyond having a fast and efficient agreement on the set of committed microblocks. It can also be used to incentivise leaders not to hinder throughput, e.g., a malicious leader in the worst case might decide not to submit any microblocks to intentionally stall the throughput. However now, since a leader is expected to commit m microblocks per epoch, leaders which cannot meet that constraint can be punished for example by decreasing their reputation and hence decreasing their chances of staying part of the consensus group and becoming leaders again.

4.3. Reputation System

This section describes our reputation system and the proof-of-reputation. We first highlight the shortcomings of previous systems. For example, a proof-of-work based system requires a miner to show that it has done some work in order to include its set of proposed transactions, and hence extend the chain. Thus, a miner that has a high computing power can join the system at any time and can play attacks. Similarly, in the proof-of-membership system [30], a miner has to show that it has created enough blocks recently to demonstrate its computing power, then it can issue microblocks and can gain power in the consensus protocol. Again, an attacker with higher computing power can join the system at any time and can break the system. However, with proof-of-reputation, in addition to creating enough recent keyblocks, a miner has to show that it has behaved honestly and created keyblocks regularly for a period of time before being able to launch any attacks on the system.

TABLE 1: The notations.

Notation	Explanation
L	the length of the current blockchain;
c	the size of a block chunk, i.e., the number of keyblocks contained in a chunk, pre-defined by the system;
t	$t = \lceil \frac{L}{c} \rceil$ is the number of block chunks contained in a blockchain with length L;
Ext	the optional external source of reputation for the miner;
H	a binary presenting whether the miner is honest ("1") or not ("0");
k_i	the number of keyblocks created by the miner in chunk i;
N_l	the number of times that the miner is elected as a leader;
m_j	the number of valid microblocks created by the miner at the j-th time it is the leader;
m	the maximum number of microblocks that a leader is allowed to create, as defined by the system.
mean$_i$	the mean value of keyblocks (if $i = k$) or microblocks (if $i = m$) created by a miner or a leader across all epochs in the blockchain, respectively.
s_i	the standard deviation corresponding to mean$_i$, for $i \in \{k, m\}$.
(a, λ)	reputation system parameters

Given the blockchain, the reputation of any miner can be calculated at any point in time. Accordingly, each miner maintains its own copy of the reputation score of all miners, based on the globally agreed blockchain. We denote by R the reputation of a miner, which can take values in $[0, 1]$. R is calculated according to Algorithm 2. The notations are defined in Table 1.

In particular, $Ext \in [0, 1]$ is the (optional) external source reputation of the miner. For example, when Citybank joins RepuCoin, it may have a starting reputation that is higher than a random individual joiner. In RepuCoin, this is encoded by using Ext[3]. $H \in \{0, 1\}$ is the honesty of the miner, which is set to "1" for each new joiner, and is set to "0" if a miner has misbehaved[4]. A miner is said to misbehave if:

3. Note that this is only used to optimize the reputation system. However, to study the worst case, our analysis in §6.1 also shows the security guarantee without having this external source of reputation.

4. Once the honesty H of a miner has been set to "0", the Ext of the corresponding entity will also be set to "0" as this entity is not trustworthy any more.

- it presents conflicting signed messages to other consensus group members; or
- it commits microblocks with conflicting transactions when the miner is elected as leader.

Algorithm 2 Reputation algorithm

Input: L, $\{k_i\}_{i=1}^{t}$, $\{m_j\}_{j=1}^{N_l}$, m, c, a, and λ.
Output: Reputation $R \in [0, 1]$ of the corresponding miner.

```
 1: $\text{mean}_k = \frac{\sum_{i=1}^{t} k_i}{L}$
 2: $\text{mean}_m = \frac{1}{N_l} \cdot \sum_{j=1}^{N_l} \frac{m_j}{m}$
 3: $s_k = \sqrt{\frac{1}{t} \cdot \sum_{i=1}^{t} (\frac{k_i}{c} - \frac{\sum_{i=1}^{t} k_i}{L})^2}$
 4: $s_m = \sqrt{\frac{1}{N_l} \cdot \sum_{j=1}^{N_l} (\frac{m_j}{m} - \frac{1}{N_l} \cdot \sum_{j=1}^{N_l} \frac{m_j}{m})^2}$
 5: $y_1 = \frac{\text{mean}_k}{1+s_k}$
 6: if $N_l \geq 1$ then
        $y_2 = \frac{\text{mean}_m}{1+s_m}$
 7: else
 8:     $y_2 = 1$
 9: end if
10: $x = y_1 \cdot y_2 \cdot L$
11: $f(x) = \frac{1}{2}(1 + \frac{x-a}{\lambda+|x-a|})$
12: $R = min(1, H \cdot (Ext + f(x)))$
```

Upon their occurrence, an evidence of such misbehavior is included in the blockchain as a special transaction, similar to past work [4, 19]. Non-Byzantine miners are incentivised to place such a proof of fraud into the blockchain, to make malicious acts visible to everyone, hence preserving the health of the system. If a cryptocurrency system is not healthy, then its users will lose their confidence in the system. This may result in the plummeting of its currency exchange rate, and all miners will have a loss. So, miners are incentivised to keep the health of the system for their own profit.

The reputation function. We intended to define the social objectives of reputation in RepuCoin in a precise and parameterizable way. Those objectives are: (i) careful start, through an initial slow increase; (ii) potential for quick reward of mature participants, through fast increase in mid-life; (iii) prevention of over-control, by slow increase near the top.

The formula defining the progression (resp. regression) of reputation, $f(x)$ above, is a sigmoid function. It ensures that miners, at the start, can only increase their reputation slowly, even if having a strong computing power. A miner needs to stay in the system and behave honestly for a long enough period, to progressively increase its reputation up to the turning point, where it is trusted enough to be incentivized to make it grow more quickly, to more interesting levels. And finally, the curve inflects again, so that the reputation does not grow forever, but asymptotically reaches a plateau that promotes a balance of power amongst miners. The reputation function is also parameterized, to allow to mark these points precisely, namely the parameters (a, λ) can be tuned to adopt changes on when and how fast/slow miners can increase their reputation. The slope of $f(x)$ is directly correlated with the value of λ. The inflection point of $f(x)$ occurs at $x = a$, and is the point where a miner's reputation growth rate starts to decline.

We denote by a block chunk (or just 'chunk' for simplicity) a sequence of successive keyblocks in the blockchain. Blocks chunks satisfy the following: (i) all block chunks are of the same size, and (ii) any keyblock is included in exactly one block chunk.

y_1, defined at line 5 of Algorithm 2, captures the miner's "regularity" of generating keyblocks in each block chunk. In other words, y_1 *shows how regularly the miner contributes its computing power to the system.*

In particular, the numerator mean_k of y_1 is the percentage of pinned keyblocks generated by the miner, represents the fraction of valid work that a miner has contributed to the whole system. In the denominator, s_k is the standard deviation of the pinned keyblocks generated by the miner, indicates the regularity with which a miner contributes to every chunk. Together, they guarantee that a miner's reputation is computed based on the miner's *integrated power*. Hence, a miner's integrated power is given by the total amount of valid work a miner has done over the period of time it has been active and the regularity of that work in the entire blockchain, rather than the miner's mining ability at a given time (or instantaneous power) as in classic proof-of-work. As such, when the system has been operated for some time, even a miner with strong computing power cannot build-up its reputation quickly: it needs to contribute honestly and regularly to the system to gain reputation. We present a more detailed analysis in §6.1.

Similarly, y_2 represents the "regularity" with which a leader commits the defined number of microblocks when it is selected. This incentivises leaders to optimize the throughput of RepuCoin.

Remark 4. The solutions based on proof-of-work, leveraging (instantaneous) computing power, also require an attacker to do some work to successfully attack the system. For example, in Bitcoin, a miner needs to create a relatively small number (e.g. a few dozens) of blocks before being able to double spend. In ByzCoin, a miner needs to make enough contributions within a window of fixed size, e.g., the last hundred or thousand blocks, to successfully attack the system.

In fact, the necessary actions to make the system fail are identical in classic approaches and in our approach. The robustness of our approach comes from the fact that, in classic systems, the 'decision' power to achieve them is dictated by instantaneous power (proportional to computing power), which can be harnessed in "no time", on demand (see e.g., the flash attack), whereas in our approach it is given by integrated power (proportional to reputation), which needs a significant amount of time to harness. So, whilst it is not impossible to break RepuCoin, it takes a very heavy toll on the attacker — indeed significantly higher than the effort to attack classic approaches — in a way that makes it infeasible. We show this fact

TABLE 2: Reputation distribution of miners. It presents the changes of the percentage of miners in each range of reputation score over time.

Time	[0, 0.2)	[0.2, 0.4)	[0.4, 0.6)	[0.6, 0.8)	[0.8, 1]
1 month	100%	-	-	-	-
6 months	64.7%	35.3%	-	-	-
1 year	21.8%	78.2%	-	-	-
2 years	9.6%	31.7%	38.1%	15.2%	-
3 years	2.7%	21.6%	19.5%	38.1%	15.2%
4 years	2.7%	19.1%	-	25%	53.2%
4 years	2.7%	15.1%	4%	17.9%	60.3%
20 years	0.4%	2.3%	-	3%	94.3%

explicitly in our results in §6.1.

Reputation distribution. We use the top 24 mining pools in the Bitcoin mining network[5] to simulate the computing power distribution in RepuCoin, since the total computing power of the top 24 mining pools are about 99.6%. We chose the following values for the parameters of the reputation function: $a = 10000$ and $\lambda = 50000$. Values for a and λ were chosen based on the time that would be necessary for powerful miners to reach a high reputation. Indeed, to provide security against bribery attacks the most powerful miners in the current computing power distribution need more than 2 years to reach a reputation higher than 0.8. Based on this computing power distribution, Table 2 shows the reputation distribution of miners over time.

4.4. Adapting Existing BFT Protocols

RepuCoin uses existing leader-based BFT protocols supporting digital signatures, to pin blocks. Apart from modifying the weight of the votes, RepuCoin also requires the following changes to adapt the existing BFT protocols.

Secure bootstrapping. With the potentially unbalanced amount of mining power in different blockchains, how to do secure bootstrapping is an open challenge in all blockchain systems. This, however, is not a problem with the classical BFT protocols, where the set of participants are predefined and fixed. Thus, to adapt existing BFT protocols, we need to provide a mechanism to establish a secure way to initialise consensus group, when no keyblock is created.

In RepuCoin, we assume the existence of a social community where participants vote to make decisions on several aspects of the system, such as the security parameters and external reputation factors. Such community exists for almost all permisionless blockchains. For example, with BitCoin this is the community who votes for proposals such as changing the maximum block size.

This community in RepuCoin votes a set of parties with external reputation to ensure a controlled bootstrapping, and record the result in the genesis block. The parties with initial higher external reputation will form the consensus group. During the secure bootstrapping phase, other miners with small or zero external reputation will gradually gain reputation and enter the consensus group.

View change. Classical BFT protocols provide view change — leader election and membership update — as a housekeeping function in the course of failures or recoveries in the (static) system participants roster. In addition to these technical functions, these protocols can be used in non-standard ways in blockchain consensus. That was the case for example of ByzCoin, where a new leader is elected every time a new keyblock is created, by invoking the PBFT view-change protocol [30]. Similarly, with RepuCoin a view change is enforced by the pinning of a new keyblock, which ends an epoch, and thus establishes a consistent cut where the systems flushes (achieving consensus on the blockchain closing the epoch). Thus, several operations can be safely performed at this clean (re-)starting point: (i) a new consensus leader is elected; (ii) and the consensus membership is updated.

To adapt existing classical PBFT-like systems, the first difference is that, for (i), the leader election is deliberately provoked in this case (not on account of e.g., a failure) and the criterion changes to random selection, as presented in §4.1. For (ii), what happens is, again, a non-standard re-definition of the membership: the (ending epoch) consensus group re-evaluates the rule for consensus group formation (a quorum of the top reputed miners, see §4.2). Note that this can directly and deterministically be derived from the data in the blockchain, so consensus is safely achieved on the new roster of $\mathbb{X}$, which is installed for the new epoch. We recall that these operations occur through stable and safe states of the system, as mentioned before.

Crash/leave detection. As this is a permissionless environment, any miner can join, leave or crash at any time. If a consensus group member left the system, then RepuCoin will eventually detect that this has happened, by checking whether this member has been involved in the last instances of the consensus. If it is the epoch leader, view change ensues in the usual manner in BFT consensus protocols.

Message size. Most existing BFT protocols have been designed for state machine replication, and even optimized for short messages/commands. Performance shown in existing publications mostly concerns tests with relatively "small" block sizes. As shown in [49], the impact of largely increased block sizes on blockchain consensus performance should not be neglected either when choosing existing protocols, or when designing blockchain-specific BFT protocols.

5. Performance Evaluation

This section presents an evaluation of the performance of RepuCoin. We focus specifically on evaluating the consensus latency and transaction throughput under different system settings.

5. The sequence of computing power of the top 24 pools is (15.1, 10.1, 10.0, 9.5, 8.3, 7.1, 6.4, 5.9, 5.5, 4.0, 2.9, 2.8, 2.4, 2.2, 1.7, 1.5, 1.5, 0.7, 0.5, 0.5, 0.3, 0.3, 0.2, 0.2), respectively. https://bitcoinchain.com/pools (as of April 2017)

5.1. Implementation

Library. In our deployment of RepuCoin, we adapt BFT-SMaRt [9] — a state-of-the-art Byzantine-resilient consensus protocol that is widely used, e.g., in Hyperledger Fabric [24] — to incorporate our reputation-based weighted voting mechanism for pinning blocks (see §4.2). We use a low keyblock generation frequency, similar to the current Bitcoin system, i.e., generating a keyblock takes 10 minutes in average. Differently, the miner that creates microblocks sends them at the highest rate it can maintain.

Mining power distribution. To better simulate the system in the real world scenario, *we make use of the mining power distribution from the Bitcoin mining network*[6]. Unsurprisingly, due to the oligopoly of mining pools, Bitcoin has a high centralisation of mining power distribution — the top 24 mining pools collectively have a computing power of 98.1%. We simulate the performance based on this power distribution. Assuming honest behavior, we compute the reputation distribution of miners. Given the computed reputations, we determine the $|\mathbb{X}|$ members of the consensus group, namely the miners with the highest reputation. In particular, we consider consensus groups that initially control from about 50% to 98.1% of computing power. With the current Bitcoin mining power distribution, the corresponding consensus group sizes would range from 4 to 19.

In fact, we show, in §6.1, that security is hampered for consensus groups that control more than 90% computing power, given Bitcoin's computing power distribution. This is due to the fact that a small number of miners individually account for a very high proportion of the mining power (5%-16%), whereas the majority of miners are much weaker (less than 1%). Thus, it would be easier for an attacker to become a consensus member in a larger consensus group, and therefore to attack the system. However, in a situation where the mining power would be uniformly distributed, then a larger group would provide a better security guarantee.

Setup. We deploy each member of the consensus group on a different machine, each having the following specifications: Dell FC430, Intel Xeon E5-2680 v3 @2.5GHz, 48GB RAM. We only use a single core to deploy within every machine, and limit each miner's RAM usage to a maximum of 4GB (even though this limit is far from being reached). To simulate wide-area network conditions in our experiments, we impose a round-trip network latency of 200 ms between any two machines. We also set the maximum communication bandwidth between any pair of machines to 35 Mbps. Similar settings were used to simulate wide-area network conditions in previous works (e.g., [30]).

5.2. Consensus Latency

In this section, we measure the latency of our consensus implementation , and compare it with the latency of the original BFT-SMaRt. Such a comparison illustrates the timing overhead that is incurred relative to using our reputation-based weighted voting mechanism. We recall that in order to reach consensus, BFT-SMaRt requires at least $2f+1$ members to agree on a value, while our reputation-based weighted voting variant requires in addition that these members (which are at least $2f+1$) have collectively more than $\frac{2}{3}$ of the reputation of the entire consensus group.

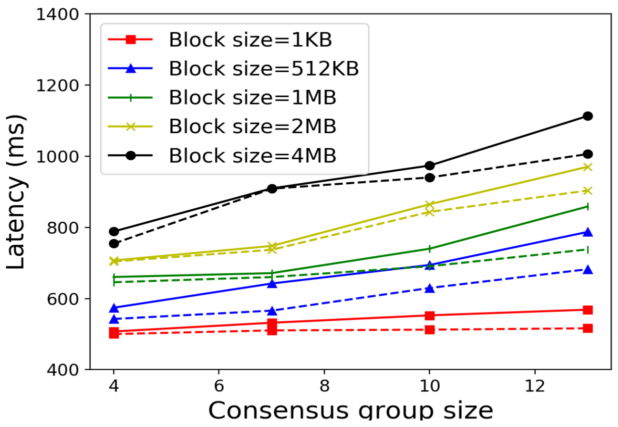


Figure 2: Consensus latency comparison. The dashed lines and straight lines are used to represent the consensus latency of BFT-SMaRt and RepuCoin, respectively.

We run experiments using keyblocks of size 1KB and microblocks of sizes 512KB, 1MB, 2MB, and 4MB. Unlike microblocks, keyblocks are typically small in size as they do not contain any transactions. We report our results in Figure 2, which shows that RepuCoin and BFT-SMaRt have a similar consensus latency values and patterns. For example, in both RepuCoin's consensus and BFT-SMaRt, consensus latency increases dramatically with the block size. The reason behind this trend can be explained by two things. First, it takes longer for the leader to propose microblocks to the consensus group, and for the group members to transmit a batch of the PROPOSE message which contains un-hashed microblock. Second, computing the hash value of a larger block and verifying the transactions it contains consume more time.

Moreover, in both RepuCoin's consensus and BFT-SMaRt, when the group size increases from 4 (which controls 44.7% computing power of the network) to 13 (which controls 90% computing power of the network), the consensus latency increases by more than 50%. However, despite this increase in latency, consensus can be reached in about 0.5-1.2 second, even when considering the blocks of size 4MB.

5.3. Throughput

We now analyze the maximum throughput that RepuCoin can achieve in terms of the number of processed transactions per second (TPS).

6. The sequence of computing power of the top 24 pools is (15.1, 10.1, 10.0, 9.5, 8.3, 7.1, 6.4, 5.9, 5.5, 4.0, 2.9, 2.8, 2.4, 2.2, 1.7, 1.5, 1.5, 0.7, 0.5, 0.5, 0.3, 0.3, 0.2, 0.2), respectively. https://bitcoinchain.com/pools (as of April 2017)

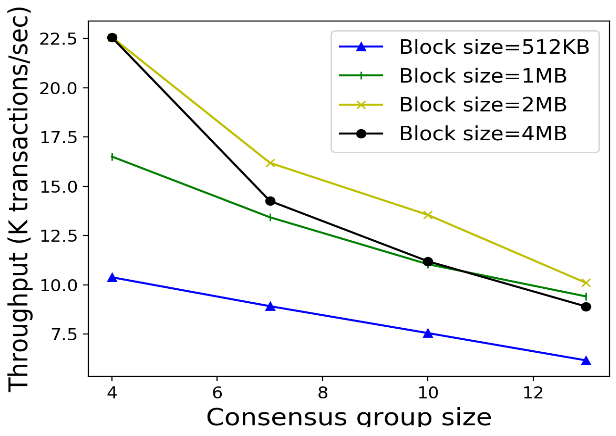


Figure 3: Throughput of RepuCoin.

Figure 3 presents the throughput of RepuCoin. First, as expected, our results in Figure 3 show that the smaller the consensus group the higher the throughput. For example, using 2MB microblocks, the throughput increases from slightly more than 10000 TPS with a consensus group of size 13 (controlling 90% computing power), to 22500 TPS with a consensus group of size 4 (controlling 44.7% computing power). Second, for all group sizes, one can see, as expected, that the throughput tends to increase as blocks become larger, and this is what we observe up to 2MB. For example, when the consensus controls 90% computing power of the entire network (group size of 13), the throughput for blocks of 512KB, 1MB, and 2MB, is respectively equal to 6200, 9400, and 10000 TPS. We observe that using larger block sizes (e.g., 4MB), decreases the throughput. However, this outlier is an artefact of the underlying protocol we use, i.e., the BFT-SMaRt library, whose sheer performance, as a regular BFT protocol, is seemingly affected for very large block sizes, as discussed in Section 4.4.

When RepuCoin provides the best security guarantee (when the consensus group controls 90% computing power, as shown in §6.1), RepuCoin achieves a throughput of 10000 TPS when using 2MB blocks. This means that RepuCoin can handle the average transaction rates of Paypal and VISA as measured in real-life, which are 115 TPS and 1700 TPS respectively.

According to a survey [5], our throughput, i.e., 10K TPS, is outstanding among the analysed systems, as the reported peak figure for permissionless ledgers is also 10k TPS. We refer readers to the survey for more details.

6. Security Analysis

In this section, we present our analysis of the security provided by the mechanisms used in RepuCoin, namely reputation-based weighted voting consensus and proof-of-reputation function. We begin by discussing the safety and liveness correctness conditions of the reputation-based weighted voting consensus protocol in Section 6.1, with predefined bounds on the relative reputation scores of participants. Then, we prove in Section 6.2 that the achievement of those scores, which give decision power, is physically bounded by the conditions imposed on the growth rate of the proof-of-reputation function. Next, in Section 6.3, we present experiments exemplifying concrete values for the decision power growth vs. time in several situations, showing that RepuCoin indeed achieves very high stochastic robustness against attacks on its liveness or safety. Moreover, in Section 6.4 we provide an analysis of the non-rationality of infiltration attacks, with a comparison on the cost of attacking different systems. Finally, we describe in detail how RepuCoin prevents known attacks in Section 6.5.

6.1. Reputation-based Consensus Safety and Liveness

Unlike proof-of-work based mechanisms, when using proof-of-reputation, an attacker cannot break the system by merely relying on its mining ability, i.e., its computing power. An attacker rather needs to gain reputation and hence contribute to the blockchain, by yielding pinned keyblocks. We recall that the reputation of a miner with correct behavior, in RepuCoin, builds essentially on its continued and regular contribution to the entire blockchain in addition to its external source of reputation Ext.

In case of a network where there are trustworthy miners (e.g. certified operators), an attacker may even never be in the top $|\mathbb{X}|$ reputable miners. For example, this could happen if the number of those trustworthy miners is large enough ($|\mathbb{X}|$ or more) and if they each have a reputation score of "1" (which can be achieved only if $Ext \neq 0$). Hence an outside attacker, which has not established an agreed upon trust, i.e., its $Ext = 0$, may happen to never become part of the consensus group. In that case, the attacker cannot affect the system: safety and liveness of the system are always guaranteed.

However, in order to study the worst case, we do not consider any miner to have established an agreed upon trust, i.e. $Ext = 0$ for all miners. For presentation simplicity we assume that every miner behaves honestly for some period of time that allows the attacker to have a sufficient reputation when intending to attack the system.

Let $\mathbb{X} = \{x_1, \ldots, x_{|\mathbb{X}|}\}$ be the consensus group. We note R_i the reputation score of miner x_i, which gives it decision power. name can rely on any underlying secure consensus algorithm, that can be adapted according to Section 4.4, to guarantee safety and liveness.

RepuCoin guarantees consensus *safety*, if: (i) the attacker controls no more than f miners in the consensus group; or (ii) the consensus group members compromised by the attacker have a total reputation R_A such that

$$R_A < \frac{\sum_{i=1}^{|\mathbb{X}|} R_i}{3}$$

where R_i is calculated according to the proof-of-reputation Algorithm 2. In other words, an attacker cannot break the safety unless both (i) and (ii) do not hold.

In addition, if any of the above two conditions does not hold, then an attacker can break the *liveness* of the system, i.e., an agreement may not be made on any block.

6.2. Bounded Proof-of-Reputation Function Growth

We hook the stochastic equations governing the evolution of our system to 'physics': it remains impossible to gain power faster than some upper bound derived from the need to perform a number of continued honest contributions to the network — what we called 'integrated power', to differentiate from 'instantaneous power', haunting all previous works by leading to flash attacks.

Let $|\mathbb{X}|$ be the size of the consensus group in RepuCoin, $mean_{k,i}$ the mean value of keyblocks created by a miner i across all epochs in the blockchain. We prove in our Theorem 1 that the proof-of-reputation function growth rate is bounded. In consequence, the rate of change of the decision power mentioned in the previous section is, at any time, limited.

Theorem 1. In RepuCoin, if the mining power of each participant remains the same, then at any time of the system, the rate of reputation increase of any node is bounded by $\frac{1}{2\lambda}$, and the corresponding increase of the decision power of any consensus group member i is bounded by $\frac{1}{2\lambda}\Delta mean_{k,i}\Delta L$.

Proof: Let Pd be the *decision power* of a node for consensus in RepuCoin. In RepuCoin, the reputation, which changes over time t, links directly to the decision power, provided that the reputation is among the top ones. A non top reputed node has no decision power. Since the reputation is also affected by the ratio of the miner's computing power, the reputation change is also affected by the computing power that the newly joined nodes brought in. For the ease of presentation, we assume that every node's computing power is fixed in the system. Then, we have that the current decision power changes over both the number of newly joined nodes and the time increase, as follows:

$$\frac{dPd}{dN \cdot dt} = \frac{\partial Pd}{\partial N}dN + \frac{\partial Pd}{\partial t}dt$$

where dPd is the change of the decision power Pd with dN and dt, the former being the change of the total number of nodes, and the latter the advancement of time.

Since a newly joined node only has a default reputation, and cannot be a part of the top $|\mathbb{X}|$ reputed nodes, we have that

$$\frac{\partial Pd}{\partial N}dN = 0$$

So, no matter how many nodes with whatever computing power have joined the system, they will not change the decision power of the current consensus group at the time of joining. In contrast, with e.g. Bitcoin, the newly joined nodes with computing power p would have p/P decision power, where P is the total amount of the computing power in the entire network.

For any node who is already in the consensus group, we need to consider its decision power change over time, which is directly linked to $f'(x)$, the derivative of reputation function $f(x) = \frac{1}{2}(1 + \frac{x-a}{\lambda+|x-a|})$, where a and λ are system parameters and they are constant, $x = y_1 \cdot y_2 \cdot L$, such that $y_1 = \frac{\text{mean}_{k,i}}{1+s_k}$, where $\text{mean}_{k,i}$ is the mean value of keyblocks created by a miner i in the blockchain, and s_k is the standard deviation corresponding to $\text{mean}_{k,i}$. So, y_1 is $\frac{S_i \cdot t^i}{\sum_{i=1}^{N} S_i \cdot t^i}$. y_2 is the miner's performance on producing microblocks, and the upper bound is 1. L is the length of the blockchain, which growth according to the time (1 per 10 minutes, as suggested in the paper). Thus, we have,

$$\frac{\partial Pd}{\partial t}dt = f'(x) = \frac{1}{2}\frac{\lambda}{(\lambda + |x-a|)^2} \leq \frac{1}{2\lambda}$$

That is, the power growth rate is constrained by the system parameter λ. The actual power growth is bounded by

$$\frac{1}{2\lambda}\Delta x \leq \frac{1}{2\lambda}\Delta mean_{k,i}\Delta L, \textit{ if } \Delta mean_{k,i} > 0$$

where ΔL is the speed of blockchain growth (e.g. 1 per 10 minutes), $\Delta mean_{k,i}$ is the change of the mean value of the number of keyblocks created by a node i w.r.t. the total number of keyblocks created by all nodes. □

The decision power of RepuCoin is given by the collective reputation of the consensus group members. Since in RepuCoin, only the top ranked nodes can join the consensus group, their decision power is proportional to their share in the collective reputation of the group. Thus, the growth of the system's decision power is the sum of the reputation growth of all group members, when the members of the group member don't change. If the members change, then the growth of the system's decision power is the sum of the reputation growth of all unchanged group members, plus the difference brought in by the new members. Since the upper bound of reputation growth is $\frac{\Delta x}{2\lambda}$, the decision power growth in the consensus group from a new group member is also bounded by the same formula. In consequence, regardless whether the group member is changed or not, the decision power growth of the system is bounded by $\frac{\sum_1^{|\mathbb{X}|}\Delta x}{2\lambda}$. Thus, with RepuCoin, the rate of increase of decision power in the entire system is limited regardless of the newly joined computing power. This makes RepuCoin secure against flash attacks launched by a late joiner, even when the attacker has a large amount of computing power (e.g. more than 50% computing power of the entire network).

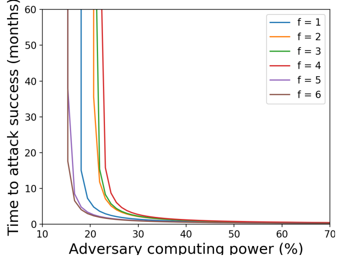
So far, we have shown analytically that: RepuCoin is safe and live (Section 6.1) as long as decision power of attackers is below a defined threshold; then, we showed (Section 6.3) that it takes a known and bounded effort for attackers to reach that threshold and to control the network. Remains to be seen if stochastically, the network shows realistic robustness, that is, how much would the attack effort be to reach the above-mentioned control, and how do those limits for the network to give in to attacks compare to previous works. We show that next, through some experiments.

6.3. Proof-of-Reputation Attack Resilience

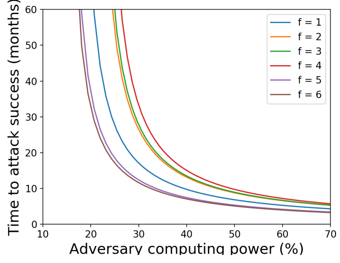
Figure 4 and Table 3 show the requirements both in terms of the computing power and the time that should be spent doing honest work in the system, in order for an attacker to successfully launch any attack. In other words, they present the minimum effort to attack the liveness of RepuCoin.

Figure 4 indicates that the system is most secure when the consensus group $\mathbb{X}$ controls 90% computing power (with 13 nodes, i.e., $f = 4$), and is most vulnerable when it controls 98.1% computing power. In fact, we can observe that the system (when $\mathbb{X}$ controls less than 90% computing power) becomes more secure as the consensus group controls more computing power of the network. After that increase the group size begins to depreciate the system security. These results can be explained by the fact that when the consensus group size grows beyond some point, the distribution of the computing power and reputation in the enlarged group could highly vary. For example, when $\mathbb{X}$ controls about 100% computing power of the system, more miners with relatively low reputation might become part of the consensus group; hence an attacker needs less time (and reputation) to infiltrate the consensus group and launch attacks. Our results, in Figure 4, show that if the consensus group controls 90% initial computing power, then an attacker joining after 3 months of system operation with 26%, 34%, and 51% computing power of the entire network would need to work honestly for 22 months, 6 months, and 2.4 months respectively, to break the liveness of the system. If an attacker joins after 1 year, then it is infeasible for this attacker to break the system's liveness (and thus the system's safety) with a computing power $\leq 26\%$; and the attacker would need 2 years (resp. 10 months) when possessing 34% (resp. 51%) of the system's computing power. We say that it is infeasible for an attacker with $\leq 26\%$ to successfully launch attacks, as our analysis shows that the attacker would have to contribute to the system honestly for 108 years before being able to do so. It is worth noting that an attacker with computing power p_a joining a system whose computing power is p_s would have $\frac{p_a}{p_a+p_s} \times 100\%$ of the system's computing power.

In Table 3, we provide a different view on the attackers ability of successfully attacking the system's liveness. Breaking RepuCoin's safety is even harder, at best as difficult as breaking liveness. It shows that an attacker who wants to break the system within one month after joining, i.e., by making the system lose liveness, would need to control at least 90% of the system's computing power, if that attacker joins after 3 months of the system being in operation. An attacker joining the system at a later time, e.g. 1 year (resp. 1.5 years) after the system operation, would never succeed in breaking the system's liveness nor safety within a period of 3 months after joining, and would require at least 68% (resp. 91%) of the system's computing power to launch at attack within 6 months after joining.



(a) Attack after 3 months



(b) Attack after 12 months

Figure 4: Minimum effort to break the liveness of RepuCoin. We consider cases where an attacker joins after the system has operated for 3 months and a year, and where the consensus group controls mining power ranging from 44.7% (i.e., f=1,$|\mathbb{X}| = 4$) to 98.1% (i.e., f=6,$|\mathbb{X}| = 19$.). The x-axis shows the needed computing power and the y-axis shows the required time to attack the system.

TABLE 3: The minimum computing power required to break the liveness of RepuCoin within a targeted time period, when an attacker joins the system at different times.

Joining time\ Target	1 week	1 month	3 months	6 months
1 month	infeasible	45%	30%	27%
3 months	infeasible	90%	45%	33%
6 months	infeasible	infeasible	68%	45%
9 months	infeasible	infeasible	90%	54%
12 months	infeasible	infeasible	infeasible	68%
18 months	infeasible	infeasible	infeasible	91%
20 months	infeasible	infeasible	infeasible	infeasible

6.4. Non-Rationality of Infiltration Attacks

We hook heuristics to well-founded 'rationality': there is basically no rational economic model in RepuCoin that makes it worth to attack the network, i.e. it always costs much more than what would be gained, due to the following three main reasons.

First, attacks can only be successful after gaining enough reputation, by means of a lot of past investment and time spent (unlike all previous works, based on instantaneous power, a.k.a. computing power, which can be harnessed in several expeditious ways). Second, reputation goes to zero after the first detected attack (unlike all previous works, which essentially don't have memory and allow repeated attacks). Last but not least, the bribery attack by buying reputation is also made ineffective. More precisely, an adaptive adversary may try to acquire one or more nodes with high reputation, in order to trigger a 'flash reputation' attack. However, buying that reputation based power should cost at least as much as the investment previously made by the sellers, and the gain upon first use (and last, since reputation goes to zero), would never match the expense. So, it always costs much more than what would be gained.

TABLE 4: The minimum cost of successfully attacking RepuCoin. $sys : *N$ means that the cost of attacking RepuCoin is at least N times as high as the cost of attacking sys, where sys is either Bitcoin (BTC) or ByzCoin (BYZ).

Joining time\ Target	1 week	1 month	3 months	6 months
1 month	infeasible	BTC: *635; BYZ: *6	BTC: *1271; BYZ: *11	BTC: *2287; BYZ: *20
3 months	infeasible	BTC: *1270; BYZ: *11	BTC: *1906; BYZ: *17	BTC: *2795; BYZ: *25
6 months	infeasible	infeasible	BTC: *2880; BYZ: *26	BTC: *3812; BYZ: *34
9 months	infeasible	infeasible	BTC: *3812; BYZ: *34	BTC: *4574; BYZ: *41
12 months	infeasible	infeasible	infeasible	BTC: *5760; BYZ: *51
18 months	infeasible	infeasible	infeasible	BTC: *7708; BYZ: *69
20 months	infeasible	infeasible	infeasible	infeasible

To better illustrate the cost in different systems, we show the cost of successful attacks in Bitcoin, ByzCoin, and RepuCoin. For the analysis, we make use of the Bitcoin real-world mining power distribution, as presented in Section 5.1.

To successfully attack Bitcoin, in the best case (not considering the selfish mining attack), an attacker needs to have 51% of the computing power, and is required to maintain this computing power only for about an hour if 6 confirmations are required, to mine its own private chain on the side. Let α be the computing power (in unit) of the entire network, then the cost for a successful attack on Bitcoin is about $0.51\alpha r$, where r is the price of maintaining 1 unit of computing power per hour. The cost of repeating this attack is the same.

With ByzCoin, in the best case (not considering the selfish mining attack), an attacker needs to have 34% computing power, and to maintain this power for the entire window (i.e. 1008 blocks), which is about a week. Thus, the cost is about $168 \cdot 0.34\alpha r$, which is $57.12\alpha r$. The cost of repeating this attack is the same, i.e., $57.12\alpha r$.

With RepuCoin, *in the worst case*, where an attacker joins at the beginning of the system, RepuCoin does no better than Bitcoin and ByzCoin upon the first attack. However, to repeat the same attack, the attack would cost much more, as the reputation of the attacker would go to zero, and the attacker would be considered a late joiner.

For a later joined attacker, the minimum cost can be calculated by using Table 3, as shown in Table 4. For example, for a 6-month late joined attacker, to successfully attack the system within 3 months, the cost of attack is about $2160 \cdot 0.68\alpha r$, which is about $1469\alpha r$. That is, the cost is 26 times as high as the cost of attacking ByzCoin, and 2880 times as high as the cost of attacking Bitcoin. Taking another example scenario, for a 1-year late joined attacker, it is infeasible to successfully attack the system within 3 months, even with the computing power of the entire network. However, the attacker is able to attack the system within 6 months with a cost of $4320 \cdot 0.68\alpha r$, which is about $2938\alpha r$. In this case, the cost is 52 times as high as the cost of attacking ByzCoin, and 5760 times of the cost of attacking Bitcoin.

TABLE 5: Summary comparison of attack resilience

Attacks	Bitcoin	Bitcoin-NG	ByzCoin	RepuCoin
Flash attack	×	×	×	√
Selfish mining attack	×	×	×	√
Attack on consistency	×	×	√	√
Attack on liveness	√	√	×	√
Double spending attacks	×	×	√	√
Eclipse attacks	×	×	√[1]	√[1]

√ – The system is secure against this attack.
× – The system is vulnerable to this attack.

[1] The system is secure against eclipse attacks for double spending purpose, however, if an attacker is able to partition the network, then it can temporally delay the consensus process and reduce the throughput.

6.5. Defense Against Specific Attacks

This section discusses defences of existing protocols against known attacks. Table 5 summarizes a comparison between Bitcoin, Bitcoin-NG, ByzCoin, and RepuCoin.

Flash attacks. In flash attacks [11], an attacker is able to obtain a temporary majority of computing power by renting enough mining capacity. This would break the security assumption of classic proof-of-work based systems.

RepuCoin, however, is resilient to flash attacks. Even an attacker with high computing power, depending on when that attacker joins, might require a very long period of time before being able to gain enough reputation to harm the system. For example, as shown in §6.1 an attacker that joins after 1.5 years of system operation would need to have 91% of the system's computing power for 6 months to successfully attack the system.

Selfish mining attack. In a selfish mining attack [18], an attacker (with more than 25% computing power) keeps its mined blocks private, and only works on its own local private version of the chain, which is different from the chain that the rest of the network works on. The attacker publishes its blocks according to some strategy that would allow the attacker to claim all rewards. So the attacker gets advantage of gaining rewards by playing this unfair game. We refer readers to [18, 47, 42] for more details on the selfish mining attack, its generalization and its optimization.

RepuCoin pins each created keyblock, and new keyblocks can only be created based on the pinned keyblock. Given that RepuCoin relies on a reputation-based consensus and a secure signature scheme, no attacker can predict the hash value of a pinned block without controlling at least $2f+1$ consensus group members that collectively have more than $\frac{2}{3}$ of the reputation of the entire consensus group. So, a selfish miner cannot gain any advantage in RepuCoin by hiding its created blocks.

Blockchain consistency and system liveness. Although Bitcoin-NG provides a high transaction throughput, it does not solve or address the consistency issues of Bitcoin. Namely, all transactions are only probabilistically valid. ByzCoin addresses these consistency issues providing deterministic transaction guarantees while achieving a high throughput. However, ByzCoin can permanently lose liveness during reconfiguration [2, 26], and a malicious miner can repeatedly make ByzCoin lose liveness temporarily in the following two ways [30]: (i) if the losing miner can propagate its block to more than 33% of the consensus members before they hear the winner's block, and (ii) if a malicious miner that becomes a leader decides not to generate any microblocks. Notice that both attacks can be repeatedly launched on the system. In addition, when an attacker controls more than 33% of the computing power for a short period, e.g., through a flash attack, then the attacker can break the liveness easily by not participating in the consensus protocol.

RepuCoin provides strong transaction consistency similar to that of ByzCoin, however, with better liveness guarantees, as RepuCoin relies on a reputation-based Byzantine fault-tolerant consensus. Specifically, RepuCoin eliminates case (i) found in ByzCoin by using a consensus protocol to choose the "winning" keyblocks. RepuCoin also limits the liveness issue pointed by case (ii) as follows: a current leader which is not generating enough valid microblocks gets punished by decreasing its reputation. In that case, that leader may be outcast from the consensus group (due to its decreased reputation); hence it cannot become a leader again and hinder system throughput.

Moreover, as mentioned previously, if an attacker with more than 33% computing power joins the system when it initiates, then we do no better than ByzCoin. However, if the system has been operated for some time, for example, if the attacker joins after 3 months (resp. after a year), then an attacker with 34% of computing power would need to work honestly and regularly for at least 6 months (resp. 2 years) to compromise the system's liveness. This makes a flash attack very expensive to launch.

Double spending attacks. There are two types of double spending attacks, i.e., 0-confirmation double spending attacks and N-confirmation double spending attacks.

Cryptocurrencies are expected to support fast ($<$ 30 seconds) payment scenarios, where the exchange time between the currency and goods is short. However, with Bitcoin, users need to wait at least 10 minutes to get a confirmation that the transaction is included in a block, and an hour (i.e., 6 blocks) to confirm that the block is very likely to be a part of the longest chain. If users accept a transaction without waiting for a confirmation, then attacks [27] on the 0-confirmation transactions can happen. These are referred to as 0-confirmation double spending attacks.

Nonetheless, even if a Bitcoin user has waited for N confirmations, if an attacker controls a sufficient percentage of computing power, the attacker can still build a fork of the chain that will be the longest chain and contains a double spending transaction, with non-negligible probability [10]. As result, an attacker is able to reverse its previous transactions (for double spending coins) even if the victim has waited for N-confirmations. Such attack is referred to as N-confirmation double spending attacks.

RepuCoin addresses these problem by speeding up the confirmation process to less than a second, even when a block size is as large as 4MB. In addition, RepuCoin provides a deterministic consistency guarantee rather than a probabilistic one. Determinism is achieved by pinning microblocks and keyblocks through a consensus scheme using a reputation-based voting mechanism. Pinned microblocks and keyblocks are non-reversible, a guarantee provided by our use of consensus.

Eclipse attacks and isolated leaders. In partial (or full) eclipse attacks [20, 23], an attacker capable of delaying information that a victim expects to receive is able to launch double spending attacks and selfish-mining attacks.

RepuCoin does not prevent an attacker from fully isolating a victim or delaying messages from a victim. However, given that blocks are pinned, the attacker cannot successfully launch double spending attacks, as previously explained. In the extreme case, some group members may be isolated temporally due to attacks on the network, e.g. those creating partitions. Such network attacks may delay the block pinning process and prevent RepuCoin from making further progress. However, RepuCoin would recover as soon as the messages are delivered, and the attacker can neither create a fork of the blockchain nor double spend any coin.

An attacker may reduce the reputation of a leader and the throughput of our system by isolating the randomly selected leader. On one side, this can be solved by using the leader selection mechanism introduced in Algorand [22]: when a new block is pinned, miners issue a signature on the block, and the miner with smallest hash value of this signature will be the leader. The new leader does not reveal this information until it prepares all microblocks, and sends all of them out with the proof that it is the leader. This way, an attacker cannot predict who will be the leader before it has sent all the microblocks it issued, and therefore cannot block the leader's communication. On the other side, it is also up to the miners to ensure that their communication can not be isolated easily and completely. This is a part of the competition for gaining more reputation. If a miner can be isolated easily by an attacker, then the miner's reputation is decreased, if not enough microblocks are proposed. Such miners will have lower chances of becoming leaders in the future (due to their decreased reputation). Thus, in a long run the system would be robust.

7. Limitations and Future Work

Formal security analysis. Formal security analysis is vital and is desired by all cryptosystems. However, providing a formal analysis on cryptocurrencies is very challenging, and only a limited number of cryptocurrencies [7, 28] has been formally studied. This paper does not address the formal security analysis of RepuCoin, but recognizes it as an important future work.

Energy waste. Proof-of-work has many inherent drawbacks one of which is wasted energy. Promising alternatives are available in the literature. *Proof-of-stake (PoS)* [17, 28, 22] is a leading alternative where the probability to create a block and receive the associated reward is proportional to a user's ownership stake rather than its computing power in the system. Another alternative is called *proof-of-space* (a.k.a. proof-of-capacity) [37, 43], which uses physical storage resources to replace computing power in the proof-of-work mechanism. *Proof-of-coin-age* [29] shares a similar concept as PoS, as participants also perform "mining" by demonstrating possession of a quantity of currency. However, unlike PoS, each quantity of coins in the proof-of-coin-age is weighted by its coin age. Participants in *proof-of-lock (a.k.a. proof-of-deposit)* [33] mine coins by depositing existing coins in a time-locked bond account, during which they cannot be moved. *Proof-of-activity* [8] puts every coin owner into a mining lottery; winners are periodically determined randomly by transactions. A winner is expected to respond with a signed message within a small time interval to claim its award. *Proof-of-luck* [39] implements a similar concept by using Trusted Execution Environment (TEE) such as Intel SGX enabled CPUs. *Proof-of-burn* is another alternative, which mines coins by destroying existing coins, i.e., sending them to an unspendable burn address that no one owns.

These concepts are particularly interesting to provide an energy-friendly consensus. Intuitively, the concept of proof-of-reputation may also be applied to these alternatives, however, we regard this as a more generic study to be part of future work.

8. Conclusion

RepuCoin provides proof-of-reputation as an alternative way to provide a strong deterministic consensus, and be robust against attacks, in a permission-less distributed blockchain system. All BFT-based blockchain systems (e.g. [30, 38, 2, 22, 31]) are bound to the coverage of the assumption on the maximum number of faulty players, f, or their decision power quota thereof. RepuCoin, although belonging to that generation of systems, is the first to deploy effective mitigation measures that reduce brittleness in the face of overwhelming adversary power, where other systems give in. Namely, it provides security guarantees against an attacker who can control a majority of the overall computing power for a duration that increases with the joining time of the attacker.

Based on the strong deterministic guarantee derived from reputation-based weighted voting, the robustness of RepuCoin grows with legitimate operation time: the later the attacker joins, the more secure the system is. For example, an attacker that joins the system after it has been operating for a year, would need at least 51% of the overall computing power and would need to behave correctly in the system for 10 months before being able to successfully make RepuCoin lose liveness. Breaking RepuCoin's safety is at least as difficult as breaking its liveness.

Acknowledgment

This work is partially supported by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128.

References

[1] I. Abraham et al. "Revisiting Fast Practical Byzantine Fault Tolerance: Thelma, Velma, and Zelma". In: *CoRR* abs/1801.10022 (2018).

[2] I. Abraham et al. "Solidus: An Incentive-compatible Cryptocurrency Based on Permissionless Byzantine Consensus". In: *CoRR* abs/1612.02916 (2016).

[3] M. Apostolaki, A. Zohar, and L. Vanbever. "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies". In: *IEEE S&P*. 2017, pp. 375–392.

[4] A. Back et al. *Enabling blockchain innovations with pegged sidechains*. 2014. URL: tinyurl.com/mj656p7.

[5] S. Bano et al. "Consensus in the age of blockchains". In: *arXiv preprint arXiv:1711.03936* (2017).

[6] I. Bentov, A. Gabizon, and A. Mizrahi. "Cryptocurrencies without proof of work". In: *FC*. 2016.

[7] I. Bentov, R. Pass, and E. Shi. "Snow White: Provably Secure Proofs of Stake". In: *IACR Cryptology ePrint Archive* (2016).

[8] I. Bentov et al. "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake". In: *ACM SIGMETRICS Performance Evaluation Review* 42.3 (2014), pp. 34–37.

[9] A. N. Bessani, J. Sousa, and E. A. P. Alchieri. "State Machine Replication for the Masses with BFT-SMART". In: *DSN*. 2014.

[10] *Bitcoin Wiki: Bitcoin weaknesses*. 2017. URL: tinyurl.com/3hto9zz.

[11] J. Bonneau. "Why Buy When You Can Rent? - Bribery Attacks on Bitcoin-Style Consensus". In: *FC*. 2016.

[12] J. Bonneau et al. "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies". In: *S&P*. 2015.

[13] M. Castro and B. Liskov. "Practical byzantine fault tolerance and proactive recovery". In: *ACM Trans. Comput. Syst.* 20.4 (2002), pp. 398–461.

[14] K. Croman et al. "On scaling decentralized blockchains". In: *FC*. 2016.

[15] T. Distler, C. Cachin, and R. Kapitza. "Resource-Efficient Byzantine Fault Tolerance". In: *IEEE Transactions on Computers* 65.9 (2016), pp. 2807–2819.

[16] C. Dwork, N. Lynch, and L. Stockmeyer. "Consensus in the presence of partial synchrony". In: *J.ACM*. 1985.

[17] *Ethereum*. https://www.ethereum.org/. 2017.

[18] I. Eyal and E. G. Sirer. "Majority Is Not Enough: Bitcoin Mining Is Vulnerable". In: *FC*. 2014.

[19] I. Eyal et al. "Bitcoin-NG: A Scalable Blockchain Protocol". In: *NSDI*. 2016.

[20] A. Gervais et al. "Tampering with the Delivery of Blocks and Transactions in Bitcoin". In: *CCS*. 2015.

[21] D. K. Gifford. "Weighted Voting for Replicated Data". In: *SOSP*. 1979.

[22] Y. Gilad et al. "Algorand: Scaling Byzantine Agreements for Cryptocurrencies". In: *SOSP 2017*. 2017, pp. 51–68.

[23] E. Heilman et al. "Eclipse Attacks on Bitcoin's Peer-to-Peer Network". In: *USENIX Security*. 2015.

[24] *IBM Blockchain: Hyperledger Fabric*. 2017. URL: tinyurl.com/j966lrv.

[25] *Intel. Proof of elapsed time (PoET)*. 2016.

[26] P. Jovanovic. *ByzCoin: Securely Scaling Blockchains*. 2016.

[27] G. O. Karame, E. Androulaki, and S. Capkun. "Double-spending Fast Payments in Bitcoin". In: *CCS*. 2012.

[28] A. Kiayias et al. "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol". In: *CRYPTO 2017*. 2017, pp. 357–388.

[29] S. King and S. Nadal. "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake". In: *self-published paper, August* 19 (2012).

[30] E. Kokoris-Kogias et al. "Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing". In: *USENIX Security*. 2016.

[31] E. Kokoris-Kogias et al. "OmniLedger: A Secure, Scale-Out, Decentralized Ledger". In: *IEEE Symposium on Security and Privacy* (2018).

[32] R. Kotla et al. "Zyzzyva: Speculative Byzantine Fault Tolerance". In: *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*. SOSP '07. 2007, pp. 45–58.

[33] J. Kwon. “Tendermint: Consensus without mining”. In: *tinyurl.com/y7fm8r2u* (2014).
[34] *Lightning Network*. 2017. URL: https://lightning.network/.
[35] J. Liu et al. “Scalable Byzantine Consensus via Hardware-assisted Secret Sharing”. In: *CoRR* abs/1612.04997 (2016).
[36] D. Malkhi and M. K. Reiter. “Byzantine Quorum Systems”. In: *Theory of Computing*. 1997.
[37] A. Miller et al. “Permacoin: Repurposing Bitcoin Work for Data Preservation”. In: *S&P*. 2014.
[38] A. Miller et al. “The Honey Badger of BFT Protocols”. In: *CCS*. 2016.
[39] M. Milutinovic et al. “Proof of Luck: an Efficient Blockchain Consensus Protocol”. In: *SysTEX*. 2016.
[40] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2009.
[41] C. Natoli and V. Gramoli. “The Balance Attack Against Proof-Of-Work Blockchains: The R3 Testbed as an Example”. In: *arXiv:1612.09426* (2016).
[42] K. Nayak et al. “Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack”. In: *IEEE S&P*. 2016.
[43] S. Park et al. “Spacemint: A Cryptocurrency Based on Proofs of Space”. In: *IACR Cryptology ePrint Archive* (2015).
[44] R. Pass and E. Shi. “Hybrid Consensus: Efficient Consensus in the Permissionless Model”. In: *DISC*. 2017, 39:1–39:16.
[45] J. Poon and T. Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2015. URL: tinyurl.com/y9xoa42u.
[46] QuantumMechanic. *Bitcoin Forum - Proof of Stake instead of Proof of Work*. 2011. URL: https://bitcointalk.org/index.php?topic=27787.0.
[47] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. “Optimal selfish mining strategies in Bitcoin”. In: *FC*. 2016.
[48] Y. Sompolinsky and A. Zohar. “Secure high-rate transaction processing in bitcoin”. In: *FC*. 2015.
[49] J. Sousa, A. Bessani, and M. Vukolic. “A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform”. In: *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*. 2018, pp. 51–58.
[50] I. Stewart. *Proof of burn. bitcoin. it*. 2012.
[51] J. Vermeulen. *VisaNet – handling 100,000 transactions per minute*. MyBroadband. https://mybroadband.co.za/news/security/190348-visanet-handling-100000-transactions-per-minute.html. 2016.
[52] G. S. Veronese et al. “Efficient Byzantine Fault-Tolerance”. In: *IEEE Trans. Computers* 62.1 (2013), pp. 16–30.
[53] M. Vukolic. “The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication”. In: *Open Problems in Network Security*. 2015.