ReCon: Sybil-Resistant Consensus from Reputation

Alex Biryukov, Daniel Feher

University of Luxembourg

Abstract

In this paper we describe how to couple reputation systems with distributed consensus protocols to provide a scalable permissionless consensus protocol with a low barrier of entry, while still providing strong resistance against Sybil attacks for large peer-to-peer networks of untrusted validators.

We introduce reputation module ReCon, which can be laid on top of various consensus protocols such as PBFT or HoneyBadger. The protocol takes external reputation ranking as input and then ranks nodes based on the outcomes of consensus rounds run by a small committee, and adaptively selects the committee based on the current reputation. ReCon can tolerate larger threshold of malicious nodes (up to slightly above 1/2) compared to the 1/3 limit of BFT consensus algorithms.

Keywords: blockchain, consensus, Sybil, reputation

1. Introduction

Distributed consensus. Distributed consensus protocols, where several equal nodes establish an agreement on a sequence of operations, have been known since at least the 1980s with the appearance of the first distributed databases. Over time, protocols that tolerate faulty nodes (FT protocols [1, 2]) and later the ones that tolerate malicious nodes (BFT protocols, for Byzantine fault tolerance [3]) were developed [4, 5]. However, their application was limited as such databases have been typically constrained to a single enterprise, which can use a trusted leader to facilitate the agreement.

The Byzantine Agreement protocols tolerate up to one third of all nodes being malicious. This is satisfactory for a private system, but does not work when we design a public system with free membership. The situation changed drastically with the introduction of Bitcoin [6], which revolutionized

The research described in this paper was partially supported by the Luxembourg National Research Fund (FNR) through CORE project FinCrypt (C17/IS/11684537).

consensus protocols by using the *Proof-of-Work* concept (PoW). A Bitcoin node solves a computationally hard problem to decide which operations (transactions) to apply. The proof-of-work consensus tolerates malicious nodes as long as they constitute no more than 51% of the computational power or as some more conservative analysis estimates 25% of the computational power [7]. On the other hand there are drawbacks to using a PoW protocols as well: the two most used protocols, Bitcoin and Ethereum [8], support up to 10 transactions per second at most, which is a great difference to thousands of transactions per second achieved in regular Byzantine Agreement protocols such as Tendermint [9] or in private networks [10]; the electricity needed to perform cryptocurrency mining is also reaching new heights, as now it is comparable to the consumption over a year of Ireland [11]; transaction confirmation time may take up to an hour (although it is still faster than wire transfer across continents).

Another crucial issue for open consensus protocols is their vulnerability to Sybil attacks. If there is no cost to join a network, that network will always be prone to classical Sybil attacks. There are multiple protocols that provide safety in an open network against this kind of attack. For example, in PoW this resistance is provided by the cost of mining. In Proof-of-Stake style systems the adversary needs to obtain a large enough stake in the currency to perform such attacks. In general all the different Proof-of-X style protocols require a proof in order to prevent Sybil attacks. We provide a new protocol that leverages node *reputation* in order to enhance Sybil-resistance of consensus protocols.

Reputation systems are abundant in our society from online auctions and marketplaces like eBay, Amazon, credit ratings like Standard & Poor's, Fitch, Moody's to social networks and even academic citations. They are forgeable to different degrees, but in many cases it takes long time and effort to earn reputation and often there is a monetary value associated with it. In some cases monetary stakes can be used directly for ranking or reputation. One may also assume that reputation correlates with the chance of honest behavior if the setting can detect and punish malicious behavior.

Our contributions. We design a proof-of-concept protocol which we call ReCon (*Re*putation *Con*sensus)– a protocol which can bootstrap from some hard to forge reputation source[1], suggests committee members based on their reputation, and this reputation is then updated over the rounds of the consensus protocol. The protocol is built on top of a blockchain, where multiple attributes of blockchains are utilized. In a nutshell, ReCon selects the committee in a randomized way to enable

[1]Maintaining such source of Sybil-resistant reputation is a separate interesting research topic.

diversity and increase the cost of Sybil attack, slightly increases the committee member's reputation if a BFT round succeeds and significantly reduces the reputation if the round fails. Furthermore, the reward mechanism is built in such away that it stops giving rewards after certain number of successful consensus rounds - namely if all goes well keeping the status quo is the best. This is not fair to the newly joined honest nodes, but also keeps any Sybil from increasing reputation of its nodes. The eventual reputation inversely correlates with maliciousness. We show that this method prevents not only a simple adversarial strategy when malicious nodes always try to disrupt the protocol, but also smarter one, when such nodes act only if they constitute 1/3 of the committee or even only after 2/3. Since ReCon quickly penalizes the reputation rating of the validators which couldn't reach consensus the best adversarial strategy is to wait for supermajority of 2/3 of committee nodes. Our scheme is thus secure in the environments where these situations are ruled out, ex. when the chance of obtaining 2/3 majority in any reputation-selected committee is very low (controlled by security parameter λ). As a module, ReCon can be laid on top of many existing distributed consensus protocols such as PBFT [4], HoneyBadgerBFT [12], or Zyzzyva [13] opening them to a larger pool of permissionless validators.

Even though our protocol doesn't solve all the problems with the classical PoW chain, it does provide an alternative which drastically reduces the cost of entry to the network and reduces the energy waste, while still providing strong resistance against Sybil attacks.

We have implemented a proof-of-concept simulator of ReCon, which takes the number of nodes, the committee size, the initial reputation and the prior "maliciousness" probability as inputs and returns the reputation distribution and the posterior probability of being malicious. We also examine the case when external reputation is not available (this could be of interest if Sybil attacker is not present at the early stages of the protocol - allowing the honest users time to acquire reputation) . Experiments prove that our approach compares favourably to the other methods of detection of maliciousness by sample testing and maximum likelihood. Computational complexity of the reputation module itself is constant in each round and for each protocol node (update of reputations for a constant number of validators, typically - 100).

The paper consists of the following sections. First we describe the current state of the art in consensus protocols that either use some form of reputation or uses blockhains to provide an open peer-to-peer consensus protocol (Section 1.1). Afterwards we present the preliminary requirements and assumptions of our protocol (Section 2). We then provide the detailed description of the protocol itself (Section 3), followed by our proof-of-concept simulation results (Section 4). Finally we consider

the different attacks against the protocol based on our assumptions (Section 5).

1.1. Related Work

The literature on the reputation systems is vast and is beyond the scope of this paper. All webpage ranking systems, for example, fall into this category with PageRank [14] being a classical example. An interested reader is referred to flow-based reputation systems adapted for P2P networks (EigenTrust [15]), subjective logic-based schemes [16], or privacy-preserving designs, both coin-based [17] and not [18]. There are also works considering a rational entity in a Byzantine Agreement protocol like the BAR Primer [19], which can be represented as a reputation protocol as well. Our work has a more narrow focus as we do not consider individual ratings but work on the meta-protocol level by analyzing global events – consensus decisions only. This allows sophisticated methods to apply easily in the decentralized fashion.

There is a vast and quickly evolving body of work on consensus protocols for open peer-to-peer protocols that use the blockchain primitives to achieve their goals. The first major category of these protocols are the proof-of-X (PoX) protocols, originating from the PoW protocol. These protocols try to utilize the idea of the PoW protocol while replacing the wasteful hashing with something that is either useful or simply much less wasteful. The most well known variant is proof-of-stake (PoS), where the lottery is not computation-based but instead uses the controlled stake per user. PoS is implemented in the Ouroboros protocols [20, 21] or in Tendermint [**?**]. Other solution include proof-of-space, where the miner has to prove the existence of hard drive space, proof-of-burn where miners have to transfer coins to an unspendable address and thus burning the coins, etc.

Another family of protocols usually referred to as Hybrid protocols are more similar to our construction. These hybrid protocols often use a PoX construction which is then used to generate a small committee. This small committee then runs some form of Byzantine Agreement protocol. ByzCoin [22] uses the most recent miners of a proof-of-work puzzle at every block to define the next committee. Algorand [23] uses global randomness to create a new committee where the likelihood to be selected is based on the controlled stake in the blockchain. Meanwhile some protocols generate multiple protocols at the same time, using either proof-of-work in the case of the Elastico [24] protocol, or a plug-in of any kind of proof-of-X in case of OmniLedger [25].

Comparatively our construction uses a form of lottery as well to generate the committees, but this lottery is based on reputation instead of well-established protocols that natively prevent Sybil attacks.

2. Preliminaries

2.1. Generic

We work in the following model. The network is composed of N public *nodes*, which maintain a consistent state by applying *transactions* of certain type in the same order. Transactions are supplied to the network by clients in a pre-specified format, but we do not make any assumptions on their size or structure, nor on the number of clients and their connectivity.

Each node is an equal participant in a *consensus protocol*, which specifies the action sequence so that eventually the nodes agree on the transaction order (*safety*) and every valid transaction is accepted at some point (*liveness*). A protocol is called *Byzantine fault tolerant* (BFT) if it provides safety and liveness despite some *malicious* nodes violating the protocol secretly or openly. The number of malicious nodes tolerated by a BFT protocol can not exceed $\lfloor \frac{N-1}{3} \rfloor$ (one of our goals is to go beyond this bound). Protocols can involve random coin tosses or be *deterministic*.

Byzantine Agreement protocols typically operate in rounds. If the number of malicious nodes exceeds $\lfloor \frac{N-1}{3} \rfloor$, there may be no agreement (the round is wasted), or with equivocation the malicious nodes can create two valid blocks, which is the equivalent of a *fork* in a blockchain protocol. Dealing with these types of forks is discussed later in the paper (Section 3.8). If the malicious nodes constitute more than $\lfloor \frac{2N}{3} \rfloor$, they can force an incorrect agreement – *forgery* (which usually leads to a malicious takeover).

2.2. Assumptions

We assume *smartly malicious* nodes, which act so that in the case of round failure an external observer can not detect who disrupted the protocol. Malicious nodes can communicate with each other to detect if they constitute the necessary $\lfloor \frac{N-1}{3} \rfloor + 1$ nodes to disrupt the round, force an equivocation or the 2/3 fraction for an incorrect agreement.

The network is considered to be asynchronous. As a non-probabilistic protocol can not provide safety and liveness at the same time in an asynchronous network, these properties of ReCon are dependent on the permissioned consensus protocol used. However the validators are selected via reputation-based rule from a much larger permissionless set of nodes.

The BFT protocol requires the nodes to digitally sign each message in order to provide the required integrity and authentication. Even though early BFT designs used MACs, they all can use fast

signatures such as Ed25519 or similar. Given that the transactions are signed in batches, the performance overhead due to signatures is negligible. There has been previous work on how to create more efficient BFT consensus using digital signatures in Byzcoin [22] and this approach is applicable to our case as well (but without using the PoW to select the validators). Similar to Byzcoin this would allow ReCon to have at least an order (and possibly two order) of magnitude improvement in transaction throughput compared to Bitcoin.

We study both scenarios where malicious nodes are determined before the protocol run and thus no honest node can become malicious, as well as a dynamic case in which nodes can become malicious or can become honest (cleaned), new nodes entering the system at certain rate, etc.. We also study the botnet takeover scenario, in which many nodes can become malicious (at random, including some high reputation nodes), or Sybil attack scenario where many malicious nodes but with zero or low reputation are injected at a fast rate.

The motivation for these assumptions is based on real life observations in open peer-to-peer blockchain based consensus protocols, as openness allows any level of malicious behaviour, thus the assumptions have to be as hard as possible. Our synchrony assumption is the hardest and thus the protocol is secure against network-based attacks in any circumstances which also covers any observed attacks in blockchain networks as well. The same logic led to our assumption of malicious nodes, where the attacker's goal is to thwart or take over the network, but it wants to achieve that in the least detectable way. Furthermore with our assumptions, an external observer can not distinguish honest and malicious nodes in a committee in case of a halted consensus.

2.3. Nodes

Node-to-node connection is authenticated with public keys. The corresponding PKI system is maintained by the chain with transactions in the network. In order to register a new node, revoke or refresh a key the user has to send a transaction with all the necessary data included. Such blockchain based PKI systems were shown in [26, 27]. Nodes participating in successful rounds of the protocol are rewarded by increase in their reputation score and potentially by cryptocurrency minted. Such cryptocurrency rewards as well as reputation score (which might have value outside of the protocol) motivate the economically rational behavior. Our protocol is permissionless, apart from the initial commitment of registering the public keys by the nodes. Generic node-to-node communication of distributing the new candidate transactions and the new blocks is done via the gossip protocol.

3. Reputation module

In this section we describe the reputation module ReCon, which can be plugged into any Byzantine Agreement protocol with the following rules:

- The protocol consists of (arbitrarily many) rounds.
- At each round N nodes decide the fate of one or many transactions.
- At each round the nodes may reach the consensus or not, and both outcomes are visible to all nodes.
- Each round a decision is made by a public committee C of m nodes, which does not necessarily include all the nodes. The committee decision is unforgeable[2].
- All the committee messages are signed by the transmitting node.
- At the end of each round the results are published as the new block of the chain.

The protocol parameters can be found in Table 1. The default values in the table are not fixed, their purpose is to provide a general view on the protocol. We explore the different choices for most of the values in our simulations in Section 4. Furthermore Figure 1 is an example of a single round in our protocol. An exponential committee selection function chooses the committee members based on the node ranking per reputation, and if there are more than 2/3 honest nodes (with green), then the round will succeed, and every committee member's reputation increases, otherwise if the honest nodes are less than 2/3, then the protocol halts and the members' reputation is penalized.

3.1. External Reputation

ReCon instructs the protocol how to select the committee and maintains the reputation ranking $\mathcal{R} : \mathcal{N} \rightarrow [0, 1]$, where $\mathcal{N}$ is the set of nodes, so that the nodes with high reputation have low posterior probability of being malicious. The prior probability of being malicious is given to the module and is called *external reputation*. If there is no external source of reputation, or, equivalently, all nodes have equal probability α_0 to be malicious, then we set: $R(n) \equiv 0 \quad \forall n \in \mathcal{N}$.

[2]The implementation of the secure committee broadcast is protocol-dependent [9, 22].

[3]If p is probability of forgery in one round (i.e. malicious 2/3 majority), then $\lambda = -log\ p$.

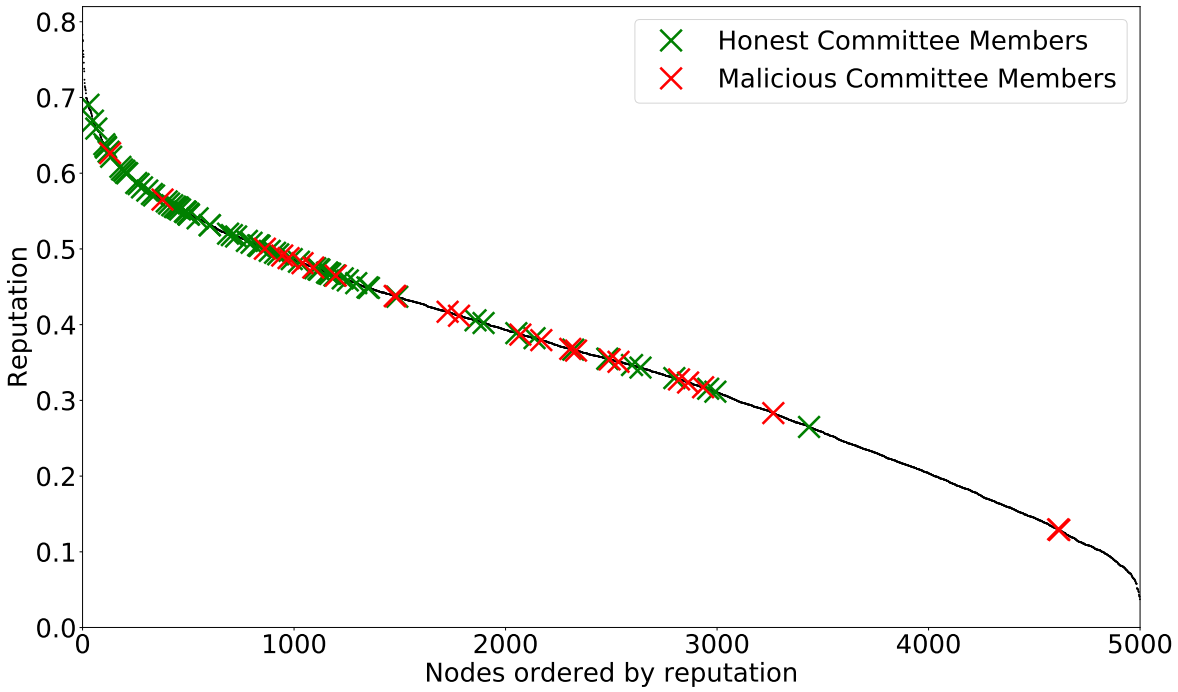

Figure 1: The reputation curve after 10,000 rounds, and the exact chosen committee members in that round with the 'X' markers (with green the honest and with red the malicious ones) using the exponential selection function. In this case 76 of the members are honest, which means the committee will reach consensus and the committee members' reputation will increase.

Parameter	Notation	Default value
Total nodes	N	$5,000$
Committee size	m	100
External reputation distribution	F	Discrete Uniform Normal Exponential
Ongoing reputation	$\mathcal{R}$	$-$
Default malicious rate	α_0	0.4
Minimum malicious rate	α_1	0.05
Committee selection rule	D	Exponential Triangular
Security parameter[3]	λ	30

Table 1: The protocol parameters

If the probability of being malicious varies from α_0 (default) to α_1 (minimum possible), then we normalize as

$$R(n) = 1 - \frac{P(n) - \alpha_1}{\alpha_0 - \alpha_1},$$

where $P(n)$ is the probability that node n is malicious. Equivalently,

$$P(n) = (\alpha_0 - \alpha_1)(1 - R(n)) + \alpha_1.$$

We denote the initial distribution of $R()$ by F, and consider various distribution functions (since $R()$ and $P()$ are affine equivalent, their distribution functions are similar). For instance, when $R()$ follows the $(0.5, 0.15)$-normal distribution constrained to $[0, 1]$, there are 23% malicious nodes in the top 10% nodes by reputation. The value Ω stands for the overall fraction of malicious nodes in N. ReCon outputs a new reputation ranking $\mathcal{R}$, for which we experimentally estimate the posterior maliciousness probability. The results for the top and bottom 10% of nodes by reputation are in Tables 4-9.

3.2. Committee selection

The decision in a round is made by a committee, which runs a round of a BFT protocol on the current set of transactions, and decides to either apply each of them or not. If the committee comes to a consensus, the transactions are applied to the state. The committee is selected based on the current

reputation $\mathcal{R}$ the nodes inherited or earned during the previous rounds. $C[r]$ denotes the committee of the r-th round.

The selection of the committee is based on some distribution D, where the higher reputation value $R(n)$ would result in a higher chance of selection (e.g. exponential distribution, exponential power distribution, triangular distribution). Here P is the probability of being selected into a committee.

$$\forall n, l \in N : R(n) \geq R(l) \Rightarrow P(n|D) \geq P(l|D)$$

This selection algorithm (Figure 1) is implemented in the following way. First, we sort the nodes based on their reputation in a descending order. Then, based on D, m random numbers are generated in $[0, N)$, and then taking the floor of all of them, we receive the selected nodes. In order to avoid double selection we select the closest yet unselected node with a higher reputation. If such node does not exist, then we do the same going towards the lower reputed nodes.

We consider two different selection distributions: exponential and triangular. Exponential gives priority to the highly reputed nodes and can strongly suppress the lower ranked ones, depending on its variance. The triangular distribution is the more fair one for the new low reputation nodes – it gives priority proportionally to the reputation but at a cost of slower convergence and lower cost for a Sybil attack.

Though the actual distributions prioritize the higher reputed nodes, they will still allow low reputed nodes to be selected into the committees. In the exponential case, $\xi = -log(0.05)/N$. The distribution itself is truncated to the $[0, N)$ interval. This ξ value means that $\int_0^N exp.dist.(\xi) = 0.95$, or in other words 95% is the chance of randomly getting an integer that is in the interval $[0, N)$.

In a similar fashion, the triangular distribution is actually a distribution from 0 to $N + \frac{N}{5}$, truncated to the $[0, N)$ interval, to give chance to be included in a committee even to nodes that have a low reputation value.

3.3. Rewards and penalties

The reputation module observes whether the committee has reached consensus. In the "smart malicious" model we imply that these two outcomes are the result of the following configurations:

- The committee has reached consensus if there are fewer than $m/3$ (no influence) or more than $2m/3$ (total control) malicious nodes in the committee.

- The committee has not reached consensus if the number of malicious nodes is between $m/3$ (non-inclusive) and $2m/3$ (inclusive).

Thus in our simulation we model the protocol execution as follows:

- If $C[r]$ has fewer than $m/3$ malicious nodes, then the round is declared success and every node in $C[r]$ gets their reputation increased.
- If $C[r]$ has $m/3$ or more malicious nodes, but less than $2m/3$ malicious nodes, then the round is declared failure and every node in $C[r]$ gets their reputation decreased. This event is undesirable (round time is wasted) but not catastrophic.
- If $C[r]$ has $2m/3$ or more malicious nodes, then the round is declared forgery. Since we can not detect externally if the decision is malicious or not, every node in $C[r]$ gets their reputation increased. However in most cases this would mean a hostile takeover and such event should be avoided by the proper parameter choice in the protocol.

The exact rewards and penalties are calculated in the following way per node. In case of a reward, the reward function for node n is

$$R_r(n) = R(n) + \frac{(1-s)(1-R(n))}{d}, d \geq 1. \tag{1}$$

The penalty function is

$$R_p(n) = R(n) - \frac{s \cdot R(n)}{d}, d \geq 1, \tag{2}$$

where s is the proportion of success rounds in the last 100 rounds (if 56 were successful, then $s = 0.56$). The idea behind this adaptive parameter is the following. Our goal is to sort the participating nodes based on their likelihood of maliciousness. Thus we choose values, that will increase and decrease the reputation values by the same amount on average, but the nodes will be reordered based on their behaviour.

The divisor d is for optimization, as for different selection functions a different d will result in the best behaviour in our protocol. For example, in the case of exponential selection $d = 10$, but for triangular selection $d = 35$. These values are the results from our empirical testing of the protocol,

where we simulated the behaviour of the nodes (Section 4). Below we explain our choice of these reward and penalty functions.

$$\frac{(1-s)(1-R(n))}{d} \lesseqgtr \frac{s \cdot R(n)}{d}$$

$$1-(s+R(n)) \lesseqgtr 0$$

$$s+R(n) > 1 \Rightarrow \text{penalty} > \text{reward}$$

$$s+R(n) < 1 \Rightarrow \text{penalty} < \text{reward}$$

In the case of $s+R(n) > 1$, notice that it is only true, if none of the values are 0, which means that there are definitely several consensus successes. Also notice, that if $s = 0$, then the value of penalty is 0, and if $s = 1$, then similarly the value of reward is 0.

These kind of changes in the values also provide us the feature, that if a node has a high reputation and participates in a bad round, it will be penalized more than a lower reputed node in the same failed round. It is true in the opposite direction as well, as the reward is higher for lower reputed nodes in successful rounds.

3.4. Probability of a forgery

We have to consider what is the probability of having a forgery ($P(k > \lfloor \frac{2m}{3} \rfloor)$, where m is the committee size and k is the number of malicious nodes in a committee). We model this as a Bernoulli trial where each member of the committee independently has probability p to be malicious node. With this model the binomial distribution $B(m, p)$ describes the committee selection. We also introduce a security parameter λ which describes the upper bound on the probability of forgery as $2^{-\lambda}$ (Tables 2). Then:

$$P\left(k > \left\lfloor \frac{2m}{3} \right\rfloor\right) < 2^{-\lambda} \iff 1 - \sum_{i=0}^{\lfloor \frac{2m}{3} \rfloor} \binom{m}{i} p^i (1-p)^{m-i} < 2^{-\lambda}$$

Furthermore, we can observe these p parameter values from the protocol attributes as well. In Table 3 we show what is the relative success rate (s, introduced in Section 3.3) of the protocol calculated from the p parameter values found in Table 2. This also means that if the protocol achieves a higher success rate then 24.5% it is safe with $\lambda = 30$.

$2^{-\lambda}$	2^{-30}	2^{-60}	2^{-120}
p	0.364	0.248	0.124

Table 2: For $m = 100$, the λ security parameters and the corresponding p values, where p is the probability of selecting a bad node.

$2^{-\lambda}$	2^{-30}	2^{-60}	2^{-120}
Successful rounds (s)	24.5%	97.5%	99.99999%

Table 3: For $m = 100$, the λ security parameters and the corresponding percentage of successful rounds based on the p values from Table 2. It shows that for $\lambda = 30$ security parameter even if only 25% of the rounds reach consensus, the protocol is still safe.

If we increase the committee size m, the value of p increases as well, where $\lim_{m\to\infty} p = \frac{2}{3}$.

This only shows the values for cases of uniform choice, but in our protocol we use a ranking based on reputation and a weighted selection (Figure 1). In the following we show how to calculate the p parameter from an external reputation system and the selection algorithm using the law of total probability for a fixed N sized set. Let X be the event of selecting a bad node and Y_n the event of selecting the n-th node (the n-th based on the reputation ranking, see Section 3.2). Then:

$$p = P(X) = \sum_{n=1}^{N} P(X \cap Y_n) = \sum_{n=1}^{N} P(X|Y_n) \cdot P(Y_n) \text{ where}$$

$$P(X|Y_n) := P(Z_n) \text{ then}$$

$$P(Z_n) = P(\text{n-th node is malicious})$$

$$p = \sum_{n=1}^{N} P(Z_n) \cdot P(Y_n)$$

For the value of $P(Z_n)$ in case of an external reputation see Section 3.1, otherwise for an observed state:

$$P(Z_n) = \begin{cases} 1 \text{ if n-th node is malicious} \\ 0 \text{ if n-th node is honest} \end{cases}$$

3.5. Types of Blocks

The protocol allows 2 types of blocks to be added to the chain. The first type is the regular block which contains the transactions agreed upon by the committee. However we want our protocols to work beyond the 1/3 maliciousness limit of BFT protocols. To avoid protocol stalls over this threshold

we introduce a mechanism to detect stalls but which does not introduce strict synchrony. We use a second type of block that acts as a timing epoch. To produce this block, we use the Verifiable Delay Functions (VDFs, [28, 29, 30]), which are a new cryptographic technique that combines time-lock puzzles with fast practical verification. In essence, these are proofs of sequential work, which return unique results (thus - functions) that can not be solved faster than time T, but which allow fast verification of the result.

In ReCon VDFs are used in the following way. The T value is set to be a large enough constant (e.g. a few minutes), that would give more than enough time for an honest committee to reach consensus. Then as soon as the committee is selected based on the previous block, any member[4] of the entire network can start computing the VDF based on the last block. If a consensus is reached in time it will be distributed in the network and a new committee will be created for the next block. On the other hand, if a node creates a block with a correct VDF before it sees a new block with a correct consensus, it can gossip it in the network as the next block as a proof that the committee did not reach consensus in time.

In order to keep the protocol fair, we have to choose the time T very carefully based on comprehensive testing on what is the expected time for an honest committee to reach consensus and to avoid consequences of secret VDF computation optimization by the attacker which may allow him to penalize honest committees[5]. With this technique, there is no direct known Δ time that is given to the committee to reach consensus, but instead and unknown Δ time until the first block with a valid VDF appears, acting as a timing epoch. This way we do not require strong synchrony, as the Δ is unknown, which is in line with protocols working in asynchronous networks like PBFT.

3.6. Source of randomness

In our protocol we use a deterministic PRNG as randomness for the selection of the committee. The drawback of this approach is that successful committees might be able to manipulate the randomness with the list of chosen transactions (so called grinding). VDFs do provide unpredictable results, which makes them even more useful for our protocol as a re-randomization seed, but if every committee is successful there is no VDF computation. In order to still have a regular influx of hard to predict

[4]Or alternatively a smaller set of high reputation nodes, not participating in the current committee - for extra Sybil protection.

[5]Optimization free implementation of VDF is a non-trivial problem in itself. However if the protocol gains traction we expect that there will be public optimized hardware to compute VDFs.

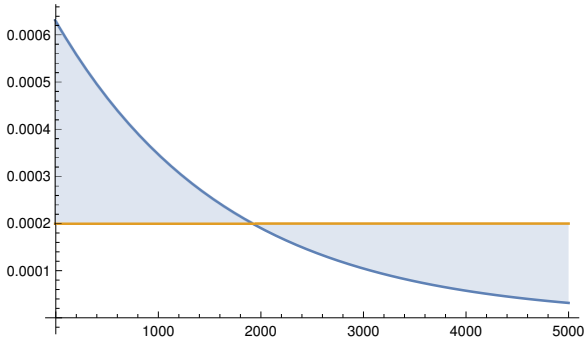

Figure 2: The filled area is the fairness of the exponential selection

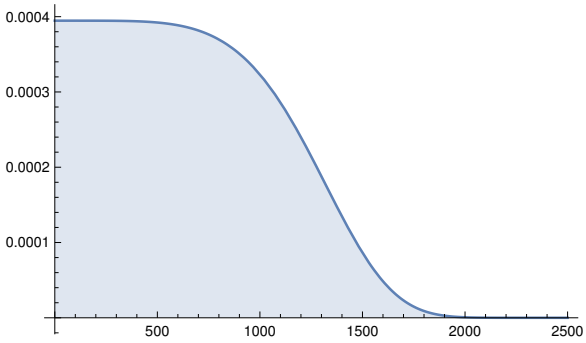

Figure 3: The exponential power distribution, or sometimes also called the generalized normal distribution with the parameters (5;0;1,000)

randomness we suggest that VDF computation is also performed at regular intervals (e.g. every 50 blocks).

3.7. Fairness

We define the fairness F of a selection function to be the L_1 distance between the uniform distribution over N nodes and the selection distribution over the same interval, namely:

$$F = \int_0^N |f(x) - \frac{1}{N}| dx \tag{3}$$

Where $f(x)$ is the probability density function of the selection distribution. The idea behind the definition is to describe how close is the selection distribution to the uniform distribution, which would be considered as perfectly fair. It is the most fair, because as an observer of the protocol we do not know which nodes are malicious, thus we should give the same probability to every node to be chosen into a committee. This way our selection functions would produce the following fairness values when $N = 5,000$: $F_{triangular} = 0.357; F_{exponential} = 0.671$.

This is also the reason why both of our selection functions (triangular and exponential) are selected in such a way, that even the node with the lowest reputation will have a chance of being selected, instead of completely ignoring the last few nodes. If we would design our selection function with the last nodes only having close to zero chance of being selected, then the fairness values would be much worse: $F_{triangular} = 0.5; F_{exponential} = 0.918$.

One could consider other distributions as a selection function, e.g. a selection that would act as a filter, which selects almost only from the highest reputed nodes. An example for that can be the

exponential power distribution (Figure 3). This selection function, however, would be very unfair, as the fairness value would be $F_{exponentialpower} = 1.35$ and the bottom nodes have no chance to be selected for the committee.

Implicitly we consider a fairness value of above 1 as unfair based on our empirical data. However such selection function might be useful during botnet takeover or Sybil attack events.

3.8. Dealing with forks

As we noted earlier, if the adversary has control over $1/3$ of the committee, different scenarios can happen. Even though each BFT protocol may have its own method to resolve these situations, we list some solutions here.

The first option for the adversary is halting the protocol by not participating in the Byzantine Agreement, and thus there will be no new blocks created in that round, as $\lfloor \frac{2N}{3} \rfloor + 1$ signatures are required for a block to be accepted. In this case a valid VDF timeout block will be created. This might lead to forks as well, if a consensus succeeds, but another member of the network created a VDF block meanwhile. In this case we let the network handle the fork and simply use the longest chain rule for choosing the valid chain.

The second one is equivocation. The adversary splits the honest nodes into two subgroups, such that he has $2/3$ majority with either of them combined with himself. Then he communicates different transactions to these groups, thus creating two valid blocks in the same round. Both blocks contain only valid transactions, as they need signatures from honest nodes, and honest nodes will only approve valid transactions. However, as all protocol messages are signed, an evidence of signing both blocks can be presented in the next committee rounds and the malicious nodes will have their reputation score reset to 0.

Notice the difference between the fork by VDF blocks and a fork by equivocation. In case of equivocation the committee creates two separate valid consensuses with overlapping verifiers, while with a VDF only one of the blocks is a valid consensus at the same chain height. This approach makes the protocol probabilistic, as there is a possibility for forks but keeps it asynchronous. It is well known that a protocol can not be deterministic in an asynchronous setting, either the safety or liveness would break. Considering this, forks in the blockchain *temporarily* break the safety of the protocol, but they keep the liveness property in any circumstances while the protocol converges to a single chain and restores the safety. This is the opposite compromise compared to classic BFT protocol like PBFT.

3.9. Convergence

We say that the BFT protocol β-*converges after* l *rounds* if the success rate (fraction of successful rounds) never goes below β after l rounds.

Concrete convergence parameters depend on the application. The values α_0, α_1 (from Table 1) determine what success rate s can be guaranteed by the ReCon ranking, and the value l determines the length of the bootstrap phase needed to rank the nodes.

4. Simulation Results

We ran our simulations[6] for $10,000$ rounds[7] with default values of total nodes[8] $N = \{5,000, 10,000, 20,000, 30,000\}$ committee size $m = 100$ and various combinations of external reputation and selection rule. Every combination is tested 100 times and the results are averaged.

We note that if there is an external reputation, we set α_1 to 0.05. We also introduce a new variable, namely Ω, which is the overall malicious rate of the nodes (so that in N nodes there are $\Omega \cdot N$ malicious ones).

We study three cases for the external reputation: (a) no external reputation, equivalent to the uniform zero reputation; (b) normally distributed reputation and (c) exponentially distributed reputation. Normal distribution of reputation is natural in scenarios where ranking or reputation is determined by many independent factors. We chose one with parameters $N(0.5, 0.15)$ so that its restriction to the $[0, 1]$ interval covers more than 99% of events (the 3σ rule).

We take exponential distribution with $\xi = 0.3$, since according to [31] the reputation distribution in an online consumer-to-consumer network as well as in most social networks is exponential.

4.1. External reputation: discrete (no information)

In this case every node has equal chance α_0 of being malicious, and the initial reputation is zero for all nodes. We consider two different selection rules and every 100 rounds we increase or decrease the variance of the selection distribution by a certain value.

[6] The simulator is available with a user friendly interface at `https://github.com/cryptolu/ReCon`.

[7] Note that if we take a conservative estimate of 60 seconds per round, 10,000 rounds would take 166 hours. Thus a bootstrap phase of a few thousand rounds is reasonable for the convergence of reputations.

[8] At the time of writing the maximum number of nodes in the Bitcoin network was around 10,000, while in Ethereum around 30,000.

First, we consider the exponential selection rule. We start with a variance of $1/N = \xi^{-2}$, and then increase it every 100 rounds by $\frac{1}{N-500i}$ starting with $i = 1$. We do this until we reach a variance, for which $P(X < 5,000) \geq 0.9$, where X is the random exponential variable with $\xi = \frac{1}{N-500i}$. This means that the exponential distribution is mostly restricted to the $[0; 5,000)$ interval. Thus at the start of the protocol every node has a similar chance to gain reputation, and later more trusted nodes have more significance.

Our results (Table 4) show that the protocol converges to a correct behaviour even if 45% of the nodes are malicious. However, the success rate decreases as the initial malicious rate Ω grows.

Then we consider the triangular distribution for its more fair selection process, as even the node with the lowest reputation score should have a real chance of participating in a committee (results in Table 5). In this case, we start with a length of 10 time N, truncate it to N, and reduce this length by N every 100 rounds. After a $1,000$ rounds, we settle with the aforementioned (Section 3.2) length of $N + \frac{N}{5}$ truncated to N.

Ω	Success Rate			
	$N = 5,000$	$N = 10,000$	$N = 20,000$	$N = 30,000$
0.1	100%	100%	100%	100%
0.2	99.95%	99.8%	99.8%	99.8%
0.25	99.7%	99.5%	99.7%	98.7%
0.33	99.6%	99.5%	98.6%	98.2%
0.4	98.7%	98.2%	96.5%	95.3%
0.45	96.5%	94.2%	89.9%	76.9%

Table 4: No external reputation, exponential selection rule: success rates after 10,000 rounds.

Ω	Success Rate			
	$N = 5,000$	$N = 10,000$	$N = 20,000$	$N = 30,000$
0.1	100%	100%	100%	100%
0.2	99.92%	99.9%	99.9%	99.9%
0.25	98.8%	98.1%	98%	96.7%
0.33	96.3%	95.9%	92%	87%
0.4	89.1%	85.8%	78%	60.1%
0.45	60%	50.3%	23.2%	9.9%

Table 5: No external reputation, triangular selection rule: success rates after 10,000 rounds.

The difference in success rates can be explained with our introduced F fairness value. The triangular distribution has a better fairness value, which means it will choose lower reputed nodes more often, and such it can sort them better as well. On the other hand, the same can be said about the exponential distribution, as it has a worse fairness value, and it will choose higher reputed nodes more often, but because of that it will not be able to sort out the nodes that well.

4.2. External reputation with normal distribution

We repeat our previous tests with external reputation distributed normally and the maliciousness probability varying from α_0 to $\alpha_1 = 0.05$. First we consider the selection rule based on exponential distribution (see Table 6).

α_0	Ω	Success Rate			
		$N = 5,000$	$N = 10,000$	$N = 20,000$	$N = 30,000$
0.4	0.225	99.99%	99.9%	99.9%	99.9%
0.6	0.325	99.8%	99.7%	99.6%	99.6%
0.7	0.375	99.5%	99.3%	98.8%	98.8%
0.8	0.425	98.8%	98.8%	98%	95.7%
0.9	0.475	93%	90%	84.8%	77%
1	0.525	50%	47.9%	41.4%	40.7%

Table 6: External normally distributed reputation, exponential selection rule: success rates after 10,000 rounds.

α_0	Ω	Success Rate			
		$N = 5,000$	$N = 10,000$	$N = 20,000$	$N = 30,000$
0.4	0.225	99.98%	99.9%	99.9%	99.9%
0.6	0.325	98.1%	97.5%	97%	94.7%
0.7	0.375	97%	96.7%	93.5%	86.7%
0.8	0.425	91%	89.8%	79%	65.2%
0.9	0.475	50%	39.7%	9.2%	9.0%
1	0.525	1%*	1%*	0.8%*	0.7%*

Table 7: Simulation results in the case of external normal distribution, selection with triangular distribution. The last simulation has an asterisk, as it produced a forgery in one of the runs.

The results in Table 6 show, that even in heavily adversarial settings of $\Omega = 0.475$ the protocol 0.93-converges and 0.5-converges for $\Omega = 0.525$. The difference based on the selection distributions between Tables 6,7 can be explained with the same reasoning as in the previous case. We can achieve better success rates compared to Tables 4,5 because of the pre-sorting of the nodes based on the external reputation. This is natural and demonstrates that trusted external reputation enhances Sybil resistance of the protocol.

4.3. External reputation with exponential distribution

In the case of an external exponential reputation system, the number of rounds for convergence values is bigger for the same α_1, but the overall malicious rate is much higher. For $\alpha_0 = 0.6$ we have $\Omega = 0.45$, and in the case of $\alpha_0 = 0.7$ it is 0.53. Also notice that we do not achieve our required success rate, but the 0.7 and 0.75 cases still converged to a lower value, and they never produced a forgery in our simulations. First we show the results for the selection based on exponential distribution (Table 8).

α_0	Ω	Success Rate			
		$N = 5,000$	$N = 10,000$	$N = 20,000$	$N = 30,000$
0.4	0.307	99.6%	99.5%	99.4%	98.9%
0.5	0.381	99.1%	99.0%	98.9%	96.6%
0.6	0.456	97%	96.8%	93.4%	87.1%
0.7	0.529	51.2%	51.1%	50.7%	47.7%
0.75	0.565	28.5%	25.7%	18.7%	8.4%
0.8	0.605	5%*	3.6%*	0.1%*	0.1%*

Table 8: External exponentially distributed reputation, exponential selection rule. The last simulation has an asterisk, as it produced a forgery in one of the runs

α_0	Ω	Success Rate			
		$N = 5,000$	$N = 10,000$	$N = 20,000$	$N = 30,000$
0.4	0.307	97%	96.9%	93.6%	93.3%
0.5	0.381	93%	92.8%	82.2%	74.6%
0.6	0.456	67%	62.7%	31.1%	18.4%
0.7	0.529	1%*	1%*	0.1%*	0.1%*

Table 9: Simulation results in the case of external exponential distribution, selection with triangular distribution. The last simulation has an asterisk, as it produced a forgery in one of the runs

In triangular selection case we have similar results as with the external normal reputation system.

5. Attacks and their mitigation

We consider attacks, based on examples from real world financial blockchains, such as Bitcoin and Ethereum. We also consider what is a good mitigation against them.

5.1. Botnet takeover

Our first example is a botnet takeover, where an attacker takes over the control of a large subset of nodes, and tries to either block the protocol (DoS), or even create a forgery. The success of the attack largely depends on the number of nodes taken over, but the results can be vastly different based on the overtaken nodes' reputation value.

5.1.1. Mitigation

We have simulated these attacks, and in the case of a large takeover of 1,000 random nodes, where $N = 5,000$, the success rate s of the protocol dropped heavily at first from above 95% to a minimum of 40%, but it recovered in a few hundred rounds, and got close to its previous success rate. As discussed in Section 3.4, even a success rate of 25% achieves $\lambda = 30$ security, as we can revert the success rate into a binomial distribution. If a large enough subset is taken over, that can cause a forgery, but that would mean the overall number of malicious nodes would be probably above 50%. Note also that botnet takeover would be noticeable by the rapid drop in the success rate of the protocol - which any node can efficiently and locally measure and which can be used to trigger a temporary switch to less fair but more robust selection rules.

5.2. Sybil attack: saturation

In this version of the well-known Sybil attack, a large number of new malicious nodes (more then $N/5$) join the protocol, and try to subvert the performance, or even create a forgery. However such nodes would have zero initial reputation.

5.2.1. Mitigation

The protocol may require a new node to participate only in communications without any eligibility for selection into a committee for a set amount of time (e.g. 2 weeks). Then every new node would start from reputation value 0. This way for an attacker to gain a large enough probability of one of its nodes being selected into a committee would require either buying and running many dedicated servers, or controlling a botnet for weeks.

It is also easily detected if a lot of nodes are joining the network at the same time and could be also a trigger for a switch to more conservative selection rules.

5.3. Sybil attack: lie and wait strategy

A more dangerous version of a Sybil attack would be if the malicious nodes only act badly, if they have $2/3$ majority in a committee. At this point they just take over the network.

5.3.1. Mitigation

Due to random selection even nodes with high reputation might have to wait for long before getting a chance of creating a forgery. Thus the adversary has to control a high number of nodes and have to keep up them active until that round. This would be costly and we choose the security parameter λ so that probability of this attack is negligible (ex. below 2^{-30} in any given round).

5.4. Attacks on randomness

Another attack would be simply DoS-ing the committee members, as their participation is publicly known to all the nodes in the protocol. If an attacker is a node, and learns the members of the next committee quickly enough, he can DoS a portion of them, which would stall the protocol.

5.4.1. Mitigation

A defense in case of a DoS attack could be generating multiple committees (in the limit every node being in some committee), making it harder and more expensive for the attacker to DoS more then $1/3$ of the nodes in all of them. As for which committee will produce the actual block it could be decided by an external unpredictable beacon (possibly based on VDF). Note that DoS attack would be very noticeable by the sharp decrease of the success rate s of the protocol, and thus this mitigation can be switched on only when it is really needed.

5.5. Honest majority

Another problem could be the fact, that we require only an honest majority in the committees, and there is no rational reason for acting honestly.

5.5.1. Mitigation

This can be mitigated in two different ways. Firstly, there are real world examples (e.g. Bitcoin or Tor), where there is no direct reward for running a full node (or Tor relay), only the indirect reward, that the user can personally monitor the validity of transactions. Even this way there are more then $10,000$ bitcoin full nodes currently in the network (more than 7,000 Tor relays).

Secondly, we can introduce a small reward for participating in a correct committee (for example, by minting a cryptocurrency in the BFT process or by distributing transaction fees), which would introduce some economic rationale for acting honestly. The problem with that is, that it would decrease the cost of a Sybil lie and wait strategy (Section 5.3), as running nodes would not be that expensive, or would even partly pay for themselves. Because of that these rewards would have to stay either relatively small so that running even a highly reputed node would not pay for itself or the opposite, so that attacking the network would be against the economic interest of the adversary (similar to the current situation with mining in Bitcoin).

5.6. Detection based on the success rate

A lot of attacks are detectable by simply monitoring the success rate s (Section 3.3). If there is a significant drop (e.g. 10% at least) in the number of successful rounds, the protocol can automatically employ a stricter selection rule (e.g. exponential power rule), which would quickly penalize bad nodes at the cost of being temporarily unfair to some of the honest nodes. Switching back to a more democratic triangular selection rule when the success rate improves.

6. Conclusions

In this paper we have described a novel approach for more scalable permissionless blockchain consensus protocols that are resilient against Sybil-attacks. Our protocol ReCon utilizes external reputation ranking to select from a large set of nodes a small subset of validators for the fast permissioned BFT protocol. This in turn would help to improve transaction throughput by one-two orders of magnitude compared to Bitcoin's Nakamoto consensus. Our solution allows Bitcoin-style egalitarian peer-to-peer networks of thousands of validator nodes without the energy waste of a proof-of-work based blockchain. Our protocol also tolerates a larger threshold of malicious nodes than a BFT consensus - 1/2 instead of 1/3.

7. Acknowledgement

We thank Dmitry Khovratovich for his contribution to the early stages of this research.

References

[1] L. Lamport, The part-time parliament, ACM Trans. Comput. Syst. 16 (2) (1998) 133–169.

[2] D. Ongaro, J. K. Ousterhout, In search of an understandable consensus algorithm, in: USENIX Annual Technical Conference, USENIX Association, 2014, pp. 305–319.

[3] L. Lamport, R. E. Shostak, M. C. Pease, The byzantine generals problem, ACM Trans. Program. Lang. Syst. 4 (3) (1982) 382–401.

[4] M. Castro, B. Liskov, Practical byzantine fault tolerance, in: OSDI, USENIX Association, 1999, pp. 173–186.

[5] P. Aublin, S. B. Mokhtar, V. Quéma, RBFT: redundant byzantine fault tolerance, in: ICDCS, IEEE Computer Society, 2013, pp. 297–306.

[6] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, `http://www.bitcoin.org/bitcoin.pdf` (2009).

[7] K. Nayak, S. Kumar, A. Miller, E. Shi, Stubborn mining: Generalizing selfish mining and combining with an eclipse attack, in: EuroS&P, IEEE, 2016, pp. 305–320.

[8] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper`http://gavwood.com/paper.pdf`.

[9] E. Buchman, Tendermint: Byzantine fault tolerance in the age of blockchains. master thesis, `https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf?sequence=7` (2016).

[10] Visa inc. at a glance, `https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf` (2015).

[11] A. de Vries, Bitcoin's growing energy problem, Joule 2 (5) (2018) 801–805.

[12] A. Miller, Y. Xia, K. Croman, E. Shi, D. Song, The honey badger of BFT protocols, IACR Cryptology ePrint Archive 2016 (2016) 199.

[13] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. L. Wong, Zyzzyva: Speculative byzantine fault tolerance, ACM Trans. Comput. Syst. 27 (4).

[14] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Tech. rep., Stanford InfoLab (1999).

[15] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proceedings of the 12th international conference on World Wide Web, ACM, 2003, pp. 640–651.

[16] Y. Liu, K. Li, Y. Jin, Y. Zhang, W. Qu, A novel reputation computation model based on subjective logic for mobile ad-hoc networks, Future Generation Computer Systems 27 (5) (2011) 547–554.

[17] E. Androulaki, S. G. Choi, S. M. Bellovin, T. Malkin, Reputation systems for anonymous networks, in: International Symposium on Privacy Enhancing Technologies, Springer, 2008, pp. 202–218.

[18] P. Lajoie-Mazenc, E. Anceaume, G. Guette, T. Sirvent, V. V. T. Tong, Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratingshttps://hal.archives-ouvertes.fr/hal-01104837/document.

[19] A. Clement, H. Li, J. Napper, J.-P. Martin, L. Alvisi, M. Dahlin, Bar primer, in: Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on, IEEE, 2008, pp. 287–296.

[20] A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A provably secure proof-of-stake blockchain protocol, in: Annual International Cryptology Conference, Springer, 2017, pp. 357–388.

[21] B. David, P. Gaži, A. Kiayias, A. Russell, Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2018, pp. 66–98.

[22] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, Enhancing bitcoin security and performance with strong consistency via collective signing, USENIX'16.

[23] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling byzantine agreements for cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017, 2017, pp. 51–68.

[24] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, P. Saxena, A secure sharding protocol for open blockchains, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, 2016, pp. 17–30.

[25] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, B. Ford, Omniledger: A secure, scale-out, decentralized ledger via sharding, in: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA, 2018, pp. 583–598.

[26] J. Bonneau, Ethiks: Using ethereum to audit a coniks key transparency log, in: International Conference on Financial Cryptography and Data Security, Springer, 2016, pp. 95–105.

[27] C. Allen, et al., Decentralized public key infrastructure: whitepaper, `https://danubetech.com/download/dpki.pdf` (2015).

[28] D. Boneh, J. Bonneau, B. Bünz, B. Fisch, Verifiable delay functions, in: Annual International Cryptology Conference, Springer, 2018, pp. 757–788.

[29] B. Wesolowski, Efficient verifiable delay functions., IACR Cryptology ePrint Archive 2018 (2018) 623.

[30] K. Pietrzak, Simple verifiable delay functions., IACR Cryptology ePrint Archive 2018 (2018) 627.

[31] Z. Lin, D. Li, W. Huang, Current security management & ethical issues of information technology, IGI Global, Hershey, PA, USA, 2003, Ch. Reputation, Reputation System and Reputation Distribution: An Exploratory Study in Online Consumer-to-consumer Auctions, pp. 249–266. URL `http://dl.acm.org/citation.cfm?id=949953.949968`

Appendix A.

The protocol description as a pseudocode (Algorithm 1), where N is the number of nodes in the network, m is the committee size and $\mathcal{R}$ is the reputation values. $C[r]$ is the selected committee in round r generated by the $gen\_committee$ function. The new gossiped block $block_r$ contains the exact reward and penalty values based on the result of the consensus or VDF block. To keep the description simpler we don't specify which nodes compute the VDF, but this can be easily determined based on the previous block. Important consideration is whether we allow any node to compute the VDF, or to a small set of high reputation nodes (who are not in the current committee). The latter will provide additional Sybil resistance.

Algorithm 1 ReCon Reputation module

procedure *Round*(r, $block_{r-1}$) ▷ The main round function
  $C[r] := gen\_committee(N, m, \mathcal{R}, block_{r-1})$ ▷ Apply reput. selection rule
  If r is divisible by l, only create a VDF block, no committee selected
  if $distr\_cons(C[r], block_r)$ **then** ▷ Whether the consensus is successful
    $gossip(block_r)$ ▷ Contains all information from a successful consensus
    $Round(r+1, block_r)$
  else
    $gossip(block_r)$ ▷ Contains all information from a failed consensus
    $Round(r+1, block_r)$
  end if
end procedure

---

procedure *distr_cons*($C[r], new\_block$) ▷ Returns new block and a flag
  if $myNode\ in\ C[r]$ **then** $start\_consensus\_alg(C[r], myNode)$
  end if
  while $!new\_block$ **do** wait() ▷ Wait till new block is returned
  end while
  if $fork(r)$ **then** ▷ A fork has happened in some round $k < r$
    Let mal_nodes be the nodes that signed both chains
    $reset(mal\_nodes)$ ▷ Set their reputation to 0, or even delete them
  end if
  if $new\_block.type = CONSENSUS$ **then** ▷ Consensus was reached
    $reward(C[r])$
    $return$ **TRUE**
  else if $new\_block.type = VDF$ **then** ▷ VDF was faster than consensus
    $penalise(C[r])$
    $return$ **FALSE**
  end if
end procedure