Guru: Universal Reputation Module for Distributed Consensus Protocols

Alex Biryukov, Daniel Feher, Dmitry Khovratovich

University of Luxembourg

Abstract. In this paper we describe how to couple reputation systems with distributed consensus protocols to provide high-throughput highly-scalable consensus for large peer-to-peer networks of untrusted validators.
We introduce reputation module Guru, which can be laid on top of various consensus protocols such as PBFT or HoneyBadger. It ranks nodes based on the outcomes of consensus rounds run by a small committee, and adaptively selects the committee based on the current reputation. The protocol can also take external reputation ranking as input. Guru can tolerate larger threshold of malicious nodes (up to slightly above 1/2) compared to the 1/3 limit of BFT consensus algorithms.

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 [17, 25]) and later the ones that tolerate malicious nodes (BFT protocols, for Byzantine fault tolerance [18]) were developed [4, 8]. 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 [23], which revolutionized consensus protocols by using the *Proof-of-Work* concept. 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 [24]. The drawback is low throughput: the two most used protocols, Bitcoin and Ethereum [28], 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 [7] or in private networks [1].

One of the crucial scalability issues for a consensus protocol is the number of nodes that determine the transaction acceptance. On the one hand, a bigger number provides more egalitarian setting and reduces the need of trust in a single or a small set of nodes. On the other hand, smaller number of nodes allows for higher throughput. The issue could have been solved using a committee with some fixed number of nodes, but there is no consensus on how this committee should be selected [21].

It seems natural to do this by *reputation*. 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. 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 malicious behavior.

Our contributions. We design and implement Guru – a module that suggests committee members based on their reputation, which is accumulated over previous rounds of the consensus protocol. In a nutshell, Guru boosts committee member's reputation if a BFT round succeeds, reduces it if the round fails, and selects the committee in a randomized way to enable "forgiveness". 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 one third of the committee and more. The adversary is thus effectively restricted to the takeover scenario, when he either captures a large fraction of nodes at once or gains control slowly and does not act before he has a supermajority. Our scheme is thus secure in the environments where these situations are ruled out. Even when it is not the case we demonstrate that Guru recovers from such events quickly enough. As a module, Guru can be laid on top of many existing distributed consensus protocols such as PBFT [8], HoneyBadgerBFT [22], or Zyzzyva [14].

We have implemented a simulator of Guru, 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. Experiments prove that our approach compares favourably to the detection of maliciousness by sample testing and maximum likelihood.

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 [26] being a classical example. An interested reader is referred to flow-based reputation systems adapted for P2P networks (EigenTrust [11]), subjectiv elogic-based schemes [20], or privacy-preserving designs, both coin-based [3] and not [16]. There are also works considering a rational entity in a Byzantine Agreement protocol like the BAR Primer [9], 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 analysing global events – consensus decisions only. This allows sophisticated methods to apply easily in the decentralized fashion.

2 Preliminaries

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 *replication 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*.

The 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 3.7). If the malicious nodes constitute more than $\lfloor \frac{2N}{3} \rfloor$, they can force an incorrect agreement – *forgery* (usually leads to a malicious takeover).

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 an incorrect agreement.

The network is weakly synchronous, which generally describes a public Internet protocol. By weakly synchronous, we mean that to provide security we do not require any synchrony, and to provide liveness, we require a Δ network delay, that bounds the message delivery for live honest nodes. If a node's messages are delivered after this Δ delay, we consider that node to be malicious. There are distributed consensus protocols with these requirements.

The BFT protocol requires the nodes to digitally sign each message in order to provide non-repudiation. 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.

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 once, or Sybil attack scenario where many possibly malicious nodes are injected at a fast rate.

Nodes Node-to-node connection is authenticated with public keys. We assume there is a public service that lists the keys and IP addresses of the nodes. The key list can be updated by an organization (example would be *consensus* in Tor maintained by Tor authorities) or in a decentralized fashion, which typically requires a separate ledger [2]. We do not specify the details and assume only that each node can learn which nodes are valid participants at each step. An example would be creating an Ethereum smart contract, where a node would have to pay the required gas price to store a signature on the chain, and can only join the Guru protocol, if it registered its key. This way an actor has also some financial commitment to joining the protocol. Nodes participating in successful rounds of the protocol are rewarded by increase in their reputation score and potentially by cryptocurrency minted. Such cryptocurrency rewards motivate the

economically rational behavior. Our protocol is permissionless, apart from the initial commitment of registering the public keys by the nodes.

3 Reputation module

In this section we describe the reputation module Guru, 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[1].
- All the committee messages are signed by the transmitting node.

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

Table 1: The protocol parameters

3.1 Reputation: external and final

Guru instructs the protocol how to select the committee and maintains the reputation ranking $\mathcal{R} : \mathcal{N} \to [0,1]$, 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

$$R(x) \equiv 0 \quad \forall x \in \mathcal{N}.$$

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

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

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

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

[1] The implementation of the secure committee broadcast is protocol-dependent [7, 13].

We denote the initial distribution of $R()$ by F, and consider various distribution functions (as $R()$ 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. Guru 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 $R(x)$ 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.

$$\forall x, y \in N : R(x) \geq R(y) \Rightarrow P(x|D) \geq P(y|D)$$

This selection algorithm 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 – it gives priority proportionally to the reputation but at a cost of slower convergence.

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 Source of randomness

Since the random values determine the nodes participating in each committee, a deterministic PRNG would allow adversaries to predict the committee members and thus plan the strategy for distant future. Even if the round outcome is added as entropy the attackers control it as they control the outcome.

A protocol thus needs a random beacon [27] – the common randomness regularly coming from an external trusted source. For example, a system that relies on the security

of the Bitcoin or Ethereum blockchain can take random values from the respective block hashes.

We suggest instantiating an external beacon with a distributed key generation protocol, which assumes that there is never more than $m/2$ malicious nodes in the committee. At each round committee members produce randomness for the next round with the following properties:

- Honest parties receive the same value $y = g^x \bmod p$ for a large prime p and prefixed g.
- Any $m/2 + 1$ honest parties can reconstruct x.
- x and y are distributed uniformly in their respective spaces.

The protocol by Gennaro et al. [10] is a perfect example, though alternatives are possible [12]. The committee nodes engage for 2-step message exchange, where they start with generating a secret value, and end with collectively producing the uniform output y, which serves as a random seed for the next round from a PRNG such as HKDF [15]. This can be easily set for an arbitrary long output. Its outputs are taken as 128-bit strings $b_0, b_1, \ldots$, which are treated as fixed-point values in $[0, 1]$ as $b_i/2^{128}$.

If the probability of a committee with more than $m/2$ malicious nodes is not negligible, then the security proof does not work. However, adversary's life can be still made harder. We suggest passing the seed through a computationally difficult and ASIC-unfriendly function such as Argon2 [5] or, if fast verification is needed, the more recent Equihash [6]. Then malicious nodes do not have sufficient computing power to affect the distribution of the random number to their favor. In addition, periodically taking (every s rounds) the randomness from public proof-of-work blockchains such as Bitcoin prevents infinitely long chains of rounds under adversarial control.

3.4 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 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 takeoverand 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. It is not a requirement to give a lot of nodes a high reputation. 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 optimisation, 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).

We describe the behaviour of our reward and penalty values, to reason with our approach of inversely proportional functions. The question is, which one is bigger, the reward or the penalty, and in what circumstances.

$$\frac{(1-s)(1-R(n))}{d} \lessgtr \frac{s \cdot R(n)}{d} \tag{3}$$

which can be reduced to:

$$1 - (s + R(n)) \lessgtr 0$$
$$s + R(n) > 1 \Rightarrow \text{penalty} > \text{reward}$$
$$s + R(n) < 1 \Rightarrow \text{penalty} < \text{reward}$$

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 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.

The inversely proportional changes in the values also provides us the feature, that if a node has a high reputation and participates in a bad round, it will be penalised 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.

3.5 Probability of selecting a malicious node

Now we calculate the probability of selecting a bad node into a committee with our model of an external reputation system, as well as a probability for maliciousness per node. This can be calculated using the law of total probability in the following way. Let X be the event of selecting a bad node. Let Y_n be the event of selecting the n-th node. Then $P(X)$ is the probability of selecting a bad node, and $P(Y_n)$ is the probability of selecting the n-th node. Now if we apply the law of total probability, we get:

$$P(X) = \sum_n P(X \cap Y_n) = \sum_n P(X|Y_n) \cdot P(Y_n)$$

From the selection distribution, we can compute the exact values for $P(Y_n)$ case-by-case. On the other hand, if $P(Z_n) = P(X|Y_n)$, then it is easy to see, that Z_n is the event of the n-th node being malicious. We can compute the $P(Z_n)$ probabilities from sampling reputation values from the external reputation distribution, and then compute the exact probability for a node being malicious based on our model (Section 3.1). This means we can calculate the value of $p = P(X)$ on a case-by-case basis, and then use p as the parameter for a $Binomial(m, p)$ distribution, where m is the committee size. With this we can compute the probability of selecting a good committee ($P(k \leq \frac{m}{3})$), which is also the success rate of the protocol.

We can also apply the same method after a β-convergence (Section 3.8), to see how much did the protocol improve on the initial probabilities. For this we use the probabilities of choosing a node (Y_n), but the probability of the n-th node being malicious (Z_n) can be done with an indicator random variable, as we know exactly which nodes are malicious and which are not.

We also have to consider what is the probability of having a forgery, which means that over $\frac{2}{3}$ of the committee members are malicious ($P(k > \frac{2m}{3})$). Using the same $Binomial(m, p)$ distribution as before, we can tell the probability of this event, and how small does p have to be in order to achieve a certain security. For that we introduce a security parameter λ which is the upper bound on the probability of forgery. Then:

$2^{-\lambda}$	2^{-30}	2^{-60}	2^{-120}
p	0.368	0.248	0.124

Table 2: The λ security parameters and the corresponding p values, where p is the probability of selecting a bad node.

Another interesting approach is converting these values into success rates, which then can be used inversely to find an approximation of the value p. The same λ security parameters converted into success rates are giving a better impression of what do these p values mean:

$2^{-\lambda}$	2^{-30}	2^{-60}	2^{-120}
Success rate	24.8%	97.5%	99.99999%

Table 3: The λ security parameters and the corresponding success rates based on the p values from Table2. It is easily observable, that for $\lambda = 30$ security level even 25% success rate is enough.

3.6 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:

$$F = \int_0^N |f(x) - \frac{1}{N}| dx \tag{4}$$

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 $N = 5000$.

$$F_{triangular} = 0.357; \quad F_{exponential} = 0.671$$

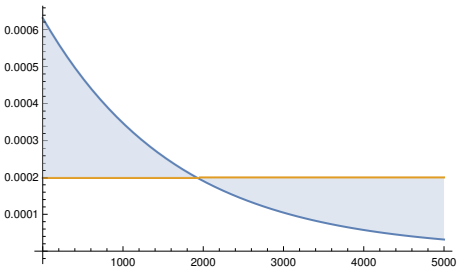

Fig. 1: The filled area is the fairness of the exponential selection

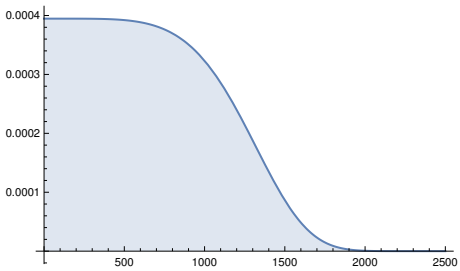

Fig. 2: The exponential power distribution, or sometimes also called the generalized normal distribution with the parameters (5,0,1000)

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:

$$F_{triangular} = 0.5; \quad F_{exponential} = 0.918$$

One would 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 2). 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.

3.7 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.

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 be penalized.

3.8 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 determine how big success rate is guaranteed by the Guru ranking, and the value l determines the length of the bootstrap phase needed to rank the nodes.

Algorithm 1 Guru Reputation module

procedure ROUND(r) ▷ The main round function
  $C[r] := gen\_comm(rnd\_seed, N, m, \mathcal{R})$ ▷ If r is divisible by s, take a sample from the external beacon as well
  if $distr\_cons(C[r])$ **then** ▷ Whether the distributed consensus is successful or not
    $gossip\ new\_block(succ, txs, rew, pen, rnd\_seed)$
    $Round(r+1)$
  else
    $gossip\ new\_block(fail, pen, rnd\_seed)$
    $Round(r+1)$
  end if
end procedure

procedure DISTR_CONS($C[r]$)
  $start\_cons\_alg(C[r])$
  if $fork(k)$ **then** ▷ A fork is proved to have happened in round k
    $mal\_nodes :=$ (nodes that signed both chains)
    $reset(mal\_nodes)$
  end if
  if $consensus$ **then** ▷ The consensus algorithm was successful
    $reward(C[r])$
    $rnd\_seed := gen\_new\_rnd\_seed()$
    $return$ **TRUE**
  end if
  if $no\_consensus$ **then** ▷ Nodes have detected the failure of the consensus algorithm
    $penalise(C[r])$
    $rnd\_seed := gen\_new\_rnd\_seed()$
    $return$ **FALSE**
  end if
end procedure

4 Simulation Results

We ran our simulations[2] for 10000 rounds[3] with default values $N = 5000, m = 100$ 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 ΩN 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 $\xi = 0.3$, as according to [19] 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.

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 $P(X < 5000) \geq 0.9$, where X is the random exponential variable with $\xi = \frac{1}{N-500i}$. This means that the exponential distribution is mostly restricted to the $[0, 5000)$ 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. The number l rounds to converge is mostly determined by the variance of the exponential distribution and stays around 4500. We also list the fraction of malicious nodes (called malicious rate) for the top and bottom 10% of nodes by the final reputation.

Then we consider the triangular distribution for its seemingly 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 1000 rounds, we settle with the aforementioned (Section 3.2)length of $N + \frac{N}{5}$ truncated to N.

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

[3] Note that if we take a conservative estimate of 15 seconds per round, 10000 rounds would take 42 hours. Thus a bootstrap phase of a few thousand rounds is reasonable for the convergence of reputations.

Ω	Success Rate	Malicious Rate	
		Top 10%	Bottom 10%
0.1	100%	10%	10%
0.2	99.95%	18.6%	26.9%
0.25	99.7%	11.5%	26.6%
0.33	99.6%	4.4%	43.8%
0.4	98.7%	2.4%	58.8%
0.45	96.5%	3.2%	70.3%

Table 4: No external reputation, exponential selection rule: success rates and reputation effects after 10000 rounds.

Ω	Success Rate	Malicious Rate	
		Top 10%	Bottom 10%
0.1	100%	10%	10%
0.2	99.92%	19.8%	20%
0.25	98.8%	15.3%	27.9%
0.33	96.3%	1%	53.3%
0.4	89.1%	3%	77.4%
0.45	60%	7%	84.4%

Table 5: No external reputation, triangular selection rule: success rates and reputation effects after 5000 rounds.

The difference in success rates and the amount of malicious nodes in the top and bottom 10% 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 this way 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).

In this case the column "rounds" refers to the required number of rounds to achieve the presented success rate, or in other words the convergence. Note, that in multiple cases it says 10000 rounds, as we ran the simulation only that long, but based on further tests running it for more rounds can still slightly increase the success rate of the protocol.

α_0	Ω	Success Rate	Rounds	Malicious Rate	
				Top 10%	Bottom 10%
0.4	0.225	99.99%	0	12.5%	31.4%
0.6	0.325	99.8%	5400	5.9%	50.1%
0.7	0.375	99.5%	5800	4.1%	61.5%
0.8	0.425	98.8%	7200	3.1%	77.6%
0.9	0.475	93%	9500	4.7%	88.3%
1	0.525	50%	10000	10.7%	88.6%

Table 6: External normally distributed reputation, exponential selection rule: success rates, convergence, and reputation effects.

α_0	Ω	Success Rate	Rounds	Malicious Rate	
				Top 10%	Bottom 10%
0.4	0.225	99.98%	0	12.5%	31.4%
0.6	0.325	98.1%	8500	2.6%	52.1%
0.7	0.375	97%	9000	0.7%	64.4%
0.8	0.425	91%	9000	0.4%	77.3%
0.9	0.475	50%	10000	1.8%	81.6%
1	0.525	1%*	10000	24.7%	78.6%

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 show, that even in heavily adversarial settings the protocol 0.9-converges with consistently high success rate.

The difference based on the selection distributions can be explained with the same reasoning as in the previous case. We can achieve better success rates because of the pre-sorting of the nodes based on the external reputation.

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	Rounds	Malicious Rate	
				Top 10%	Bottom 10%
0.4	0.307	99.6%	7800	5.1%	39.5%
0.5	0.381	99.1%	8000	2.6%	55.2%
0.6	0.456	97%	9000	3%	75.9%
0.7	0.529	51.2%	10000	10.1%	90.5%
0.75	0.565	28.5%	10000	10.3%	89.5%
0.8	0.605	5%*	10000	16.4%	81.1%

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

α_0	Ω	Success Rate	Rounds	Malicious Rate	
				Top 10%	Bottom 10%
0.4	0.307	97%	6000	3.5%	42.3%
0.5	0.381	93%	7200	0.4%	62.3%
0.6	0.456	67%	10000	0.7%	79.3%
0.7	0.529	1%*	10000	19%	71%

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. The success rates are lower, but the sorting of the malicious nodes is better, as seen in the previous cases.

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.

Mitigation We have simulated these attacks, and in the case of a large takeover of 1000 random nodes, where $N = 5000$, the success rate 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 3.5, 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.

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.

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, which would be an unnatural behaviour, and would lead to suspicion by the honest members of the protocol.

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.

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.

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.

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. Note that DoS attack would be very noticeable by the sharp decrease of the success rate 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.

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 5000 bitcoin full nodes currently in the network (more than 7000 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), 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 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. 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 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 way to solve the scalability problem of Byzantine fault tolerance (BFT) consensus protocols using the reputation feedback. Our solution allows Bitcoin-style egalitarian peer-to-peer networks of thousands of validator nodes, while keeping the benefit of high throughput of BFT-style consensus. We can also tolerate a larger threshold of malicious nodes than the 1/3 limit of the BFT consensus. Our solution can be used as an enhancement module for the deployed BFT-based protocols, leveraging the information gathered by now widely used reputation systems.

References

1. Visa inc. at a glance, 2015. `https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf`.
2. ALLEN, C., AND ET AL. Decentralized public key infrastructure: whitepaper, 2015. `https://danubetech.com/download/dpki.pdf`.
3. ANDROULAKI, E., CHOI, S. G., BELLOVIN, S. M., AND MALKIN, T. Reputation systems for anonymous networks. In *International Symposium on Privacy Enhancing Technologies* (2008), Springer, pp. 202–218.
4. AUBLIN, P., MOKHTAR, S. B., AND QUÉMA, V. RBFT: redundant byzantine fault tolerance. In *ICDCS* (2013), IEEE Computer Society, pp. 297–306.
5. BIRYUKOV, A., DINU, D., AND KHOVRATOVICH, D. Argon2: New generation of memory-hard functions for password hashing and other applications. In *EuroS&P* (2016), IEEE, pp. 292–302.
6. BIRYUKOV, A., AND KHOVRATOVICH, D. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. In *NDSS* (2016), The Internet Society.
7. BUCHMAN, E. Tendermint: Byzantine fault tolerance in the age of blockchains. master thesis, 2016. `https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf?sequence=7`.

8. CASTRO, M., AND LISKOV, B. Practical byzantine fault tolerance. In *OSDI* (1999), USENIX Association, pp. 173–186.
9. CLEMENT, A., LI, H., NAPPER, J., MARTIN, J.-P., ALVISI, L., AND DAHLIN, M. Bar primer. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on* (2008), IEEE, pp. 287–296.
10. GENNARO, R., JARECKI, S., KRAWCZYK, H., AND RABIN, T. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptology 20*, 1 (2007), 51–83.
11. KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 640–651.
12. KIAYIAS, A., KONSTANTINOU, I., RUSSELL, A., DAVID, B., AND OLIYNYKOV, R. A provably secure proof-of-stake blockchain protocol. *IACR Cryptology ePrint Archive 2016* (2016), 889.
13. KOKORIS-KOGIAS, E., JOVANOVIC, P., GAILLY, N., KHOFFI, I., GASSER, L., AND FORD, B. Enhancing bitcoin security and performance with strong consistency via collective signing. *USENIX'16* (2016).
14. KOTLA, R., ALVISI, L., DAHLIN, M., CLEMENT, A., AND WONG, E. L. Zyzzyva: Speculative byzantine fault tolerance. *ACM Trans. Comput. Syst. 27*, 4 (2009).
15. KRAWCZYK, H. Cryptographic extraction and key derivation: The HKDF scheme. In *CRYPTO* (2010), vol. 6223 of *Lecture Notes in Computer Science*, Springer, pp. 631–648.
16. LAJOIE-MAZENC, P., ANCEAUME, E., GUETTE, G., SIRVENT, T., AND TONG, V. V. T. Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratings. `https://hal.archives-ouvertes.fr/hal-01104837/document`.
17. LAMPORT, L. The part-time parliament. *ACM Trans. Comput. Syst. 16*, 2 (1998), 133–169.
18. LAMPORT, L., SHOSTAK, R. E., AND PEASE, M. C. The byzantine generals problem. *ACM Trans. Program. Lang. Syst. 4*, 3 (1982), 382–401.
19. LIN, Z., LI, D., AND HUANG, W. 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.
20. LIU, Y., LI, K., JIN, Y., ZHANG, Y., AND QU, W. A novel reputation computation model based on subjective logic for mobile ad-hoc networks. *Future Generation Computer Systems 27*, 5 (2011), 547–554.
21. MAZIERES, D. The stellar consensus protocol: A federated model for internet-level consensus. *Draft, Stellar Development Foundation, 15th May, available at: https://www. stellar. org/papers/stellarconsensus-protocol. pdf (accessed 23rd May, 2015)* (2015).
22. MILLER, A., XIA, Y., CROMAN, K., SHI, E., AND SONG, D. The honey badger of BFT protocols. *IACR Cryptology ePrint Archive 2016* (2016), 199.
23. NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system, 2009. `http://www.bitcoin.org/bitcoin.pdf`.
24. NAYAK, K., KUMAR, S., MILLER, A., AND SHI, E. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *EuroS&P* (2016), IEEE, pp. 305–320.
25. ONGARO, D., AND OUSTERHOUT, J. K. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference* (2014), USENIX Association, pp. 305–319.
26. PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford InfoLab, 1999.
27. RABIN, M. O. Transaction protection by beacons. *J. Comput. Syst. Sci. 27*, 2 (1983), 256–267.
28. WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* (2014). `http://gavwood.com/paper.pdf`.