Robust and Scalable Consensus for Sharded Distributed Ledgers

Eleftherios Kokoris-Kogias*
*EPFL
eleftherios.kokoriskogias@epfl.ch

***Abstract*—ByzCoin, a promising alternative of Bitcoin, is a scalable consensus protocol used as a building block of many research and enterprise level decentralized systems. In this paper we show that ByzCoin is unsuitable for deployment in an open, adversarial network and instead introduce MOTOR. MOTOR is designed as a secure, robust, and scalable consensus suitable for permisionless sharded blockchains. MOTOR achieves these properties by making four key design choices: (a) it prioritizes robustness in adversarial environments while maintaining adequate scalability, (b) it employees provably correct cryptography that resists DoS attacks from individual nodes, (c) it deploys unpredictable rotating leaders to defend against mildly-adaptive adversaries and prevents censorship, and (d) it creates an incentive compatible reward mechanism. These choices are materialized as (a) a "rotating subleader" communication pattern that balances the scalability needs with the robustness requirements under failures, (b) deployment of provable secure BLS multi-signatures, (c) use of deterministic threshold signatures as source of randomness and (d) careful design of the reward allocation mechanism. We have implemented MOTOR and compare it with ByzCoin. We show that MOTOR can scale similar to ByzCoin with an at most $2x$ overhead whereas it maintains good performance even under high-percentage of faults, unlike ByzCoin.**

I. INTRODUCTION

Blockchain technology has emerged a decade ago with Bitcoin [27], an open, self-regulating cryptocurrency build on top of the idea of decentralization. Bitcoin and the first generation of blockchain consensus protocols that followed have two challenging limitations regarding performance; confirmation latency and transaction throughput. This is due to the consensus used in those blockchains that requires synchronous communication [30] to guarantee persistence of the blocks. As a result clients may need to wait up to 60-minutes [27], before accepting a transaction as valid.

In order to solve this problem a second generation of blockchains emerged [22], [11], [29], which use traditional Byzantine consensus algorithms [10] in order to guarantee strong consistency in seconds. These systems, however, do not scale well since BFT protocols [10], [24] are designed to run among a few (usual deployment is 4-16) nodes. Currently, the most prominent system scaling to more than a few nodes is ByzCoin [22], but this scalability is achieved only in non-adversarial environments as we show in Section III. In order to sidestep the scalability problem a third generation of blockchains is proposed [23], [1] which is based on the idea of sharding the state into multiple sub-blockchains that can process transactions in parallel. This means that no validator audits the full blockchain. Although, sharded blockchains scale better than classic BFT, they still require to run the per-subchain consensus protocol among hundreds of participating nodes [23], especially in strong adversarial environments, showing that the requirement of scalable BFT consensus cannot simply be avoided by changing the architecture. In short, current solutions for scalable BFT consensus do not scale to the required number of participants ($\approx$ 600) [16], [29], are not robust to high adversarial power [15], [22] or adversarial

adaptation [22], [29], and do not consider rational participants [15], [29], [16], [22].

In this work, we introduce MOTOR, a scalable BFT-consensus architecture that is designed for realistic deployment at an open blockchain ecosystem where participants are rational. Our protocol can become an integral part of current multi-million dollar blockchain ventures that deploy either classic BFT consensus (*e.g.,* HyperLedger Fabric [2], [3] or Cypherium [17]) or sharded blockchains (*e.g.,* Zilliqa [37], Harmony [35], or Emotiq [36]).

To provide a viable solution, MOTOR needs to address the following three key challenges. First, a consensus algorithm that is designed to be deployed on the Internet and on potentially anonymous servers needs to be robust to a high percentage of Byzantine faults while maintaining sufficient scalability. Second, the protocol should guarantee censorship-resistance meaning that transactions from honest nodes do not face the risk of starvation in the presence of an adversarial leader. Current BFT protocols do not support this property as a malicious leader can reign for ever proposing blocks that do not include transactions of some specific clients. Finally, MOTOR needs to address incentive-compatibility. Previous works [11], [29], [22] assume that participants are ”honest” and faithfully follow the protocol even if they get higher rewards by deviating from it. In a permissionless blockchain this assumption does not model the real word where participants are rational and want to maximize their rewards.

To address the first challenge MOTOR balances the robust but not scalable *star communication pattern* (see Figure 1), where there is a single leader who collects messages from and broadcasts messages to all nodes (which prior work has extensively studied [31], [24], [16]) and the scalable but not robust *tree communication pattern* (see Figure 2) where every node has a parent and two children which they communicate with (which ByzCoin is using). To this end we design and formally study a hybrid of those two communication patterns which resembles a tree of stars (Figure 3) where the leader is connected to clusters of nodes and rotates the emission of his messages to individual nodes inside every cluster until he finds a non-malicious subleader. For the challenge of censorship resistance, we introduce rotating leaders following existing work in BFT [16], and extend it to withstand round-adaptive adversaries by randomizing the leader schedule. For this purpose we use pairing-based threshold keys to produce deterministic signatures which can be also used as unbiasable random seeds. Finally, we design the reward distribution mechanism such that the consensus nodes are incentivized to participate and the leader to not exclude them.

Our contributions: In summary we make the following contributions:

- We study ByzCoin and show that it has multiple attack surfaces both in the cryptography it uses and in the broadcast communication pattern it employs.
- We propose MOTOR consensus protocol which fixes ByzCoin's issues and remains robust, scalable and functional when deployed among rational participants. MOTOR can be deployed standalone or facilitate sharding.
- We show theoretically MOTOR's safety, liveness, and censorship-resistance and show experimentally its scalability and robustness.

II. BACKGROUND AND SYSTEM MODEL

A. Prior Work

The most closely related work to ours is ByzCoin [22] **ByzCoin** envisions a Bitcoin [27] protocol that uses strongly consistent consensus. In order to scale to hundreds of nodes it uses multi-cast trees and aggregate Schnorr signatures. MOTOR relies on the ideas of ByzCoin. In Section III we extensively study ByzCoin to identify its shortcomings and in Section IV we address those issues.

Parsimonious Broadcast. The majority of BFT algorithms inspired by PBFT [10] use an all-to-all communication pattern making them unable to scale. In order to speed this up, Ramasamy et al. [31] proposed the use of a *strong consistent broadcast* primitive instead of the reliable broadcast PBFT is using, together with a fall-back protocol that guarantees safety and liveness. This reduces the communication complexity of the optimistic case to $O(n)$ instead of $O(n^2)$. Multiple systems such as Zyzzyva [24] use this fast-track communication pattern to perform better than PBFT under normal operation. We call this communication pattern a *star* throughout this paper.

B. System Model

In this work, we begin assuming a correct parsimonious-broadcast [31] consensus algorithm, and extend it to be suitable for blockchains. Specifically, we modify the parsimonious broadcast primitive to balance its robustness with our scalablity requirements as described in Section IV-C.

Throughout this paper we consider a system consisting of a fixed set of $n = 3f + 1$ *nodes* of which up to f are Byzantine faulty, meaning that they can behave maliciously, collude or simply crash. The rest of the participants are assumed to be *rational*, meaning that they want to maximize their individual utility functions which is the relative[1] amount of transaction rewards they can collect. We assume a *round-adaptive adversary* that chooses which nodes to corrupt at the end of every consensus round and has control over them at the the end of the next round. We assume that all nodes have already setup both their individual identities (pk_i, sk_i) and a threshold key pk_s. The associated secret key sk_s is not known and every node has a secret share s_i of it. If $f + 1$ nodes use their share to sign a record then a signature sig_s that verifies against sk_s can be generated by pooling those partial signatures together. We do not investigate how the members of the consensus group change through time; a potential way can be through an epoch based Sybil resistant protocol as proposed by OmniLedger [23].

[1]The percentage over the total coins available

C. Network model

We adopt the usual partial synchronous model [10] for our network communication where there exists some unknown time interval named GST (General Stabilization Time) after which all messages between honest nodes are delivered.

D. Cryptography

We make the usual cryptographic assumptions, namely that the adversary is computationally bounded, that cryptographically secure hash functions exist, and that participants sign their messages and verify the signatures of received messages. Furthermore, we assume that BLS [8] signatures are secure and that there is a standard (decentralized) PKI for authentication [22], [29], [15] and for setting up a *(f+1)-out-of-(3f+1)* BLS threshold signature scheme using a partially synchronous DKG [19].

III. ByzCoin's Shortcomings

In this section we delve into ByzCoin and identify the attack vectors it exposes to the adversary as well as general shortcomings that prevent it from being deployable in public blockchain networks. In summary we look into 4 issues as follows: (a) the two-round Schnorr multisgnature scheme used is susceptible to simple DoS attacks by a single malicious node, (b) the tree communication pattern employed to facilitate scalability is fragile under strong adversaries, (c) the leader election mechanism is predictable and cannot defend against a round-adaptive adversary, and (d) the incentives are not aligned with the system design.

A. ByzCoin Protocol

In order to correctly aggregate the Schnorr multi-signatures, ByzCoin relies on a protocol called Collective Signing [34] (CoSi). For each

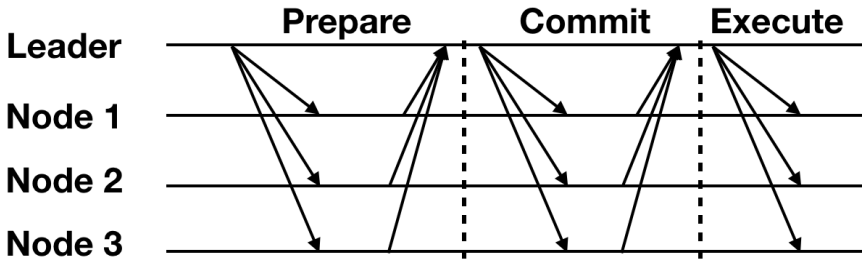

Fig. 1: Parsimonious Broadcast (Star Communication Pattern)

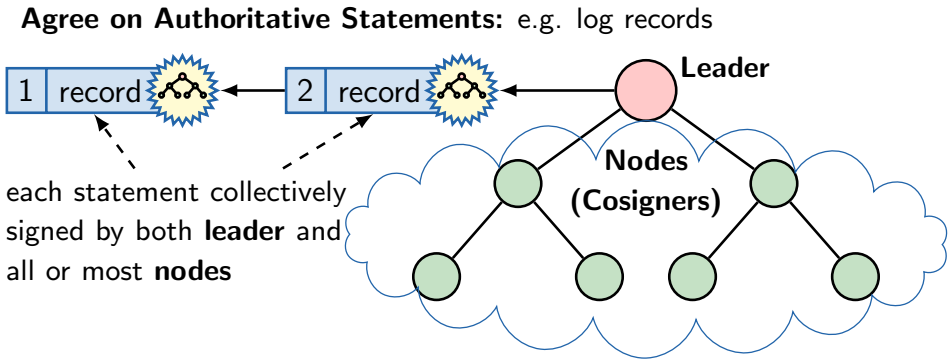

Fig. 2: CoSi protocol architecture

message to be collectively signed, the leader initiates a CoSi four-phase protocol round that requires two round-trips over the communication tree between the leader and its witnesses:

1) **Announcement:** The leader broadcasts an announcement of a new round down the communication tree. The announcement can optionally include the message M to be signed, otherwise M is sent in phase three.
2) **Commitment:** Each node picks a random secret and uses it to compute a Schnorr commitment. In a bottom-up process, each node obtains an aggregate Schnorr commitment from its immediate children, combines those with its own commitment, and passes a further-aggregated commitment up the tree.
3) **Challenge:** The leader computes a collective Schnorr challenge using a cryptographic hash function and broadcasts it down the communication tree, along with the message M to sign, if the latter has not already been sent in phase one.
4) **Response:** Using the collective challenge, all nodes compute an aggregate response in a bottom-up fashion that mirrors the commitment phase.

The result of this four-phase protocol is the production of a standard Schnorr signature. In principle, Kokoris-Kogias et al. implemented a parsimonious broadcast [31] primitive that has a super-fast path, where multiple pipelined broadcasts take place in a hierarchical way using the *tree* communication pattern while the leader can unilaterally decide that he should use the star communication pattern if he detects failures.

B. Schnorr Multi-Signatures

One of the first shortcomings of ByzCoin has been revealed by Drijvers et al [13], where they show that it is impossible to prove two-round Schnorr-based multi-signatures secure. The problem, however, in ByzCoin is not only theoretical. Below we describe a trivial DoS attack against ByzCoin, that an adversary can launch as long as he can control one node, simply by crashing. We note that this attack is on the underlying CoSi protocol and is applicable to all the systems using it [28], [33], [21], [20]. First we define what is a consensus disruption:

Definition 1: We say that the adversary disrupts a consensus round if he can force the leader to block forever or restart the protocol from scratch.

Theorem 1: An adversary controlling a single node per round can disrupt consensus on ByzCoin.

Proof 1: The consensus disruption attack works during steps 2 and 4 of a collective signing round and exploits the randomness aggregation of step 3. The attack works as follows. First the adversary chooses one node to control before the consensus round begins. Then the leader executes step 1 and announces the new round. The adversary correctly commits in step 2. During step 3 the leader aggregates all the commitments to generate the challenge and sends it to the nodes. At step 4 the adversary crashes his node and does not return a correct response making it impossible to produce a valid signature against the challenge. The leader can either block forever or restart.

This attack is practical as it can be implemented by DDoSing only one node per challenge. Hence, it is clear that MOTOR needs a multi-signature scheme that does not rely on committing and revealing randomness from individual nodes. As we describe in Section IV-B, the BLS pairing based scheme satisfies all our requirements.

C. The Tree Communication Pattern

After looking into the signature scheme of ByzCoin, now it is time to look into the fast-path communication pattern it uses. ByzCoin sends messages using a binary tree. Every non-leaf node forwards the messages to its children, detects invalid replies, and aggregates the commitments and the responses. Below we show that this communication pattern is fragile and the adversary has a high probability of disrupting a consensus.

Theorem 2: An adversary controlling f nodes has a high probability of disrupting fast-path consensus of ByzCoin.

Proof 2: For an adversary to disrupt the fast path consensus he needs to have at least one of his f nodes to be a non-leaf node that is an ancestor of an honest node. This way he will be able to control the information flow to the honest node and eclipse him from the leader. As a result the leader will be unable to collect the necessary quorum to continue. Given that in a full binary tree the number of leaves is equal to the number of internal nodes the probability that all the adversarial nodes are leaves is $0.5^{\lfloor \frac{n}{3} \rfloor}$. Even for 16 nodes the probability of having at least one non-leaf node is an overwhelming 96.88%.

We note that after a few disruptions ByzCoin will fall back to the more robust star topology, however as the authors of ByzCoin showed [22], the star does not scale to more than 100 participants. Thus, it is clear that MOTOR requires a communication pattern that strikes a balance between robustness and scalability.

D. Censorship

In classic consensus algorithms an important property is *validity*, which states that all operations accepted by the consensus layer need to be proposed by a client. Validity, however, does not differentiate between an honest and a dishonest client. This implicitly permits leader-driven censorship, where the leader does not include in his

proposed transactions those of honest clients. A quick-fix [16] to this problem is to have a rotating leader in the hope that some leader will be honest and prevent censorship forever. This fix, however, is a valid defense only under a static adversary that cannot change the compromised nodes throughout the lifetime of the system, a rather weak assumption.

Theorem 3: A round-adaptive adversary can always control the leader of the consensus in ByzCoin.

Proof 3: Currently consensus protocols protect from censorship by changing the leader. This change of leader will signal a new round. Given that the sequence of leaders is well-known the adversary can always choose to corrupt the current and the next-in-line leaders. This means that the adversary will always know the leader of round *r+1* prior to the commencement of round *r*, hence will have control over the leader of round *r+1* and the ability to enforce his censoring rules. Thus, it is clear that MOTOR needs to randomize the schedule of leaders to create a moving target for the adversary.

E. Incentives

In order for ByzCoin to avoid free-riding (or establishing a public goods game [4]), the rewards are not uniformly distributed[2]. Instead the rewards are distributed equally to all participating nodes, whereas the non-participating nodes get zero rewards, which someone would expect to be incentive compatible. However, the exact mechanism of calculating the per node rewards is not well thought, leading to a deviating leader who excludes a minority of nodes to increase his relative rewards. In such a case the aforementioned minority receives zero rewards, which can result to abandoning the system or becoming willing to collude with the adversary in order to break the protocol.

[2]Free-riding is a problem that scalable BFT protocols using threshold [16] signatures might have as everyone is rewarded regardless of participation

Although the fair reward mechanism is a step towards the correct direction as it avoids free-riding, the fate of the rewards that would have normally been assigned to the non-participating nodes require careful design. The two obvious choices that ByzCoin proposes are to either destroy this part of the rewards or redistribute it to the participating nodes. Both possible reward mechanisms are incompatible with the system design. The leader's actual incentive is to achieve consensus with the minimum valid quorum[3] instead of the maximum available quorum.

We model the system as follows. We say that there are n nodes, k of which participated in the consensus round. The total reward of the block is denoted by R and each node gets a reward of value r. Finally the total amount of coins (the scarce commodity that has value) in the system is denoted by N. We say that every node wants to increase his relative amount of coins, meaning that he wants the highest percentage of available coins. This means that the total amount of coins after the block is committed is $N + R$ and each participating node has received a relative reward of $r_{rel} = \frac{r}{R+N}$.

Case 1: The reward is fixed to r coins and the rest of the coins are burned.

Analysis 1: In this case the relative reward of the leader is $r_{rel} = \frac{r}{r*k+N}$. Since r is constant, the r_{rel} increases as k decreases which means that the leader is incentivized to keep k to the lowest value possible. This is the minimum value that enables him to achieve consensus ($2f + 1$), hence he will exclude the rest f nodes.

Case 2: The rewards are redistributed hence $r = \frac{R}{k}$

Analysis 2: In this case the relative reward of the leader is $r_{rel} = \frac{R}{k(R+N)}$. Since R, N are constant, the r_{rel} increases as k decreases which means that

[3]If the leader does not reach a quorum, then a new leader will be elected who might censor the old leader, hence a rational leader wil ask every node to participate but only include 2f+1 contributions

the leader is incentivized to keep k to the lowest value possible, and thus he will exclude f nodes.

Thus, it is clear that MOTOR needs a better reward scheme than the trivial approach taken by ByzCoin.

IV. MOTOR DESIGN

In this section we describe MOTOR. Specifically we discuss how we provide a robust and scalable BFT-consensus protocol, suitable for open blockchains. Section IV-B describes a single round-trip, BLS-based, Collective-Signing protocol that produces provably secure signatures and defends against DoS attacks. Section IV-C describes our *rotating subleader* communication pattern that is a hybrid of the tree and star topology which makes it both robust and sufficiently scalable. Section IV-D describes our randomized view-change that prevents a round-adaptive adversary from breaking liveness and fairness by controlling all leaders. Finally, Section IV-E describes the reward allocation mechanism that provides incentive compatibility for rational participants.

A. Goals.

To sum up, MOTOR has the following goals.

- **Robustness:** The adversary cannot disrupt the consensus round unless he controls the leader node.
- **Scalability:** The protocol performs well even when run among hundreds of nodes[4].
- **Fairness:** The probability of a malicious leader being elected is equal to the percentage of malicious nodes in the system. This prevents the adversary from covertly censoring specific clients, by always controlling the leader.
- **Incentive Compatibility:** The most rewarding strategy is to follow the protocol.

We do not aim to design a consensus algorithm from scratch but to define the set of extensions that need to be deployed in order for an existing classic BFT algorithm such as ByzCoin to be converted into a blockchain-consensus algorithm. We focus on ByzCoin because it provides the backbone of multiple decentralized application [23], [22], [28], [33], [20], [36], [35]. Nevertheless, our individual protocols can be deployed to other BFT-consensus algorithms to achieve robustness in scale or incentive compatibility.

B. BLS-CoSi

The first design step towards MOTOR is to abandon the DoS-susceptible two-round CoSi and instead use BLS signatures. BLS signatures are provably secure and can be aggregated into compact multi-signatures. Because of their non-interactive and deterministic nature BLS signatures are a powerful tool MOTOR employees in multiple components.

1) Protocol: In this section we describe MOTOR's agreement protocol assuming an honest leader. In order to get consensus we need to combine two rounds of this protocol (prepare and commit) just like ByzCoin and Hybrid Consensus [22], [29]. The protocol needs a setup phase where all participants prove knowledge of their secret keys sk_i by signing pk_i in order to eliminate rogue-key attacks [6]. Furthermore there is an array of bits $b[i]$ which is used to signal which keys take part in the final aggregate signature. [5] We also assume that all the keys are publicly available for anyone. The protocol works in two steps or one round-trip of Figure 1:

- **Announcement:** The leader broadcast the message M to be signed.
- **Response:** Every node verifies M against some policy (*e.g.,* if all transactions in a block are valid) and on agreement signs message M with his key pk_i creating σ_i. The leader verifies and aggregates the signatures while setting the correct bits of $b[:]$ as 1. After

[4]Enough to support secure sharding (600)

[5]We note that our agreement protocol is equivalent to the independently proposed ASM protocol of Boneh et al [7]

aggregation, the leader holds an aggregate signature $\hat{\sigma_M}$ of M and $b[:]$. He can then send $\hat{\sigma_M}$ to everyone, who can verify it by aggregating all the keys pk_i that have been marked as 1 in $b[:]$.

The protocol above is a modification of CoSi for BLS-multi-signatures, which is DoS-resistant as there is no randomness aggregation step to exploit.

Theorem 4: An adaptive adversary controlling f nodes (out of 3f+1) cannot disrupt consensus in MOTOR.

Proof 4: The consensus disruption attack on ByzCoin targeted the interactivity of the Schnorr multi-signatures. By using BLS signatures we remove this attack surface. During step 1 the leader will announce the new round. At step 2 the adversary can decide to sign, which helps the leader, or abort. Even if all f nodes abort the leader will still collect $2f + 1$ responses which represent a sufficient quorum for liveness.

Finally, to achieve consensus we need two rounds of the above protocol (which implements the consistent broadcast abstraction that parsimonious broadcast employs), however, there is no need for the quorums to be the same nodes. As a result even if the adversary decides to compromise different sets of f nodes for the prepare and the commit phase there will still be $2f + 1$ honest nodes per phase which will enable the leader reach consensus.

C. Rotating-Subleader Communication

After applying BLS-CoSi on Parsimonious [31], we get a more scalable algorithm because of the compaction of signatures which is close to $O(N)$ (there is a bit-vector that still increases linear to the size of the system, but is small compared to the size of N signatures). However, the leader is still the bottleneck to our new scalable consensus algorithm.

The next challenge MOTOR aims to solve is to find a suitable communication pattern to avoid the communication bottleneck that classic BFT protocols have when run among many servers over a limited bandwidth network such as the Internet. More specifically we design a *rotating-subleader (*RS*)* communication pattern which is a hybrid of the robust *star* communication pattern that many BFT protocols use and the scalable *tree* communication pattern that ByzCoin employs.

The RS communication pattern is a star of star topologies. The leader first splits the nodes into c random groups and then chooses ephemeral subleaders who route his messages and aggregate signatures (see Figure 3). As illustrated the set of subleaders together with the leader form a robust star topology, similarly the subleader and his group's nodes form a second star. If all subleaders are honest (or rational as we show later), they will collect as many signatures as possible and reply to the leader. If, however, they are malicious they can crash or not forward the message to the group. If a subleader either replies with very few accept messages or does not reply within some timeout (we define later) the leader will choose a new subleader for every failed group and retry. This will continue until the leader manages to collect $2f + 1$ positive replies.

Timers. To prove liveness we first need to define the timers that decide when to rotate subleaders and when to view change. Every node maintains two timers. The first timer Δ_1 is used to detect a crashed leader, while the second timer Δ_2 is used by the leader to rotate his unresponsive subleaders. When a leader is assumed faulty a new leader is elected and both timers are doubled. There is a fixed ratio (see below) between Δ_1 and Δ_2 such that the leader has enough time to rotate through all his potential subleaders before being assumed faulty.

Theorem 5: An honest leader will eventually find a sufficient quorum to commit his proposal.

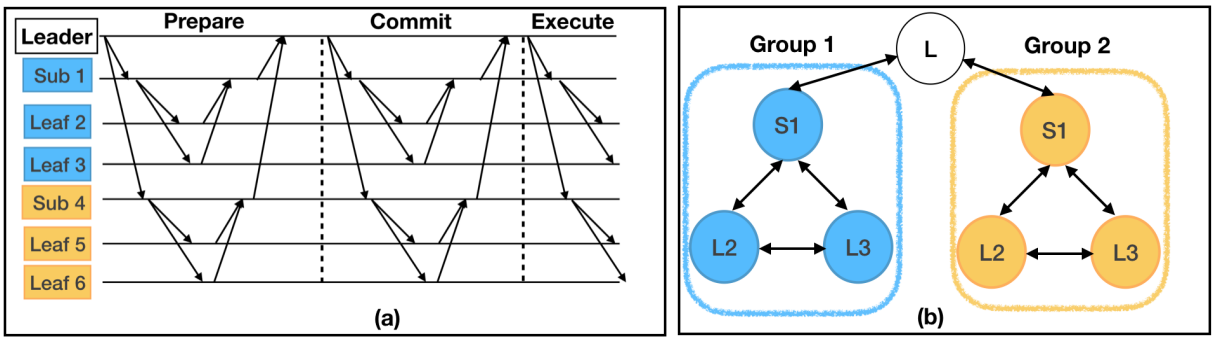

Fig. 3: (a): Rotating Subleader Communication Pattern. The Leader splits the nodes in the blue and the yellow group. Then he chooses subleaders and only communicates with them. If Subleader 1 fails to reply with enough messages, the Leader will ask Leaf 2 or Leaf 3 to take over Subleader 1 and broadcast the message again in the blue group. (b): Rotating Subleader Network Topology. The leader splits the nodes into two fully connected groups and randomly elects one subleader per group to be his proxy. If the proxy fails the group rotates ($L2- > L3, L3- > S1, S1- > L2$)

Proof 5: In the worst case the system will have exactly $2f+1$ honest nodes and one group that has only one honest node, whose reply is necessary. In order to get his answer at the worst case the leader will rotate all $\frac{n}{c}$ nodes as subleaders of the specific "bad-case" group before asking the honest node to be the subleader. In RS we set the view-change timer Δ_1 to be $(\frac{n}{c}+1)*\Delta_2$, giving the leader enough time to try all possible subleaders. If the leader does not reach consensus within Δ_1 this means that he is slower than GST and should be changed.

Theorem 6: The RS communication pattern preserves safety of MOTOR.

Proof 6: The RS communication pattern does not affect the quorum requirements and as a result has the same safety guarantees.

Message Complexity: In this section we analyze the best-case, average-case and worst-case message complexity of MOTOR. What we care about is both the total number of messages as well as the maximum load (bottleneck) on any given node.

In the best case scenario (Figure 3.(a)) the leader is going to send $3*c$ messages to his subleaders who are going to forward it to the $\lceil\frac{n}{c}\rceil$ members of their group. The leaf nodes will send 2 messages each. As a result the total number of messages is $O(n)$. The max load on any given node will be $O(n)$. Depending on c, this max load will appear either on the leader or on the subleaders. In the case that $c=\sqrt(n)$ the max load on any node will be $O(\sqrt(n))$.

In the average case the leader will have to re-elect a constant number of subleaders. For example if the adversary controls a fraction P_{adv} of the nodes and the system has c groups. Then we need to find y in the equation:

$$(1-P_{adv}^{y})^{c}=\frac{1}{2}$$

which says that with probability $\frac{1}{2}$ (average case) the adversary will find at least one honest leader after y retries in every one of the c groups (union bound for disjoint probabilities). Hence,

$$y=\frac{\log\left(1-2^{-\frac{1}{c}}\right)}{\log P_{adv}}$$

retries per group. For example if $c=18$ ($c^2=324$) and $P_{adv}=\frac{1}{3}$ the leader will find an honest

subleader at every group 51% of the time after 3 retries.

In this average case the load of the leader increases by y (needs to contact y times more subleaders), the load of the subleaders remains the same as before and the load of leaves increases by y (need to reply to y subleaders). This increases both the max load of the leader and the total number of messages by $O(y)$ which is expected to be constant and low under weak adversaries (normal execution environment).

Finally, the worst case will be as described in *Proof 5*, in which case the leader will have to do $(\frac{n}{c}+1)$ retries. As a result the total load of the leader will increase to $O(n)$ and the load of the leaves to $O(\frac{n}{c})$. Given that there are $O(n)$ leaf-nodes the total message complexity will be $O(\frac{n^2}{c})$, which for $c = O(\sqrt(n))$ leads to a message complexity of $O(n\sqrt(n))$.

To sum up, the RS communication pattern increases the overall load of the network by y in the average case and by $\frac{n}{c}$ in the worst case. However, it always distributes the load in a more egalitarian way, such that the leader (the bottleneck of the star), has best-case $O(c)$ load and worst case $O(n)$ load. On the contrary, the more robust star has always $O(n)$ load on the leader.

Discussion: We can observe the RS communication pattern as a generalization of the star [24], the tree [22] or even the chain [38] communication pattern where c defines the robustness-scalability trade-off. With $c = n$ we have a star which is robust but adds too much burden on the leader. With $n < c < \lfloor\sqrt(n)\rfloor$ we have the RS communication pattern; it decreases the load of the leader at the optimal and average case on the expense of increasing the load on the rest of the nodes and on the expense of increasing the total number of messages in the average and bad case. If we set $c = 2$ we create a binary tree and with $c = 1$ we create a chain [38]. Because of the recursive nature of the chain and the tree, the bandwidth load is equally distributed between all non-leaf nodes, but any non-leaf failure breaks the recursion completely.

The reason why RS is robust, whereas the tree or the chain are not is because when failures occur in RS the leader knows to blame the subleader, hence there is only a constant number of retries ($\frac{n}{c} \leq \sqrt{n}$) he needs to do to find a good per-group subleader (only the subleaders change, the groups remain static). On the contrary, in the chain or the tree, the failure can happen at any non-leaf node and the leader would need to retry an exponential number of configurations to find a good permutation.

D. Rotating Leader

After defining how one round of MOTOR works, we now proceed to describe the inter-round protocol which enables unpredictable leader rotation.

In MOTOR we adopt the view-change protocol from SBFT [16] (which is an improvement against parsimonious broadcast) and focus on the problem that current view-change protocols assume a predetermined schedule of leaders. This requirement makes them susceptible to mildly adaptive adversaries who can predict the next f leaders and choose to compromise them (only need to adapt once every f rounds).

To prevent this attack in MOTOR, we rely again on the power of BLS signatures. Specifically we use their deterministic nature to generate an unpredictable and unbiasable random number which will be used to randomize the schedule of the leaders. We achieve this using the a BLS-threshold key pk_s and we minimally change the view-change protocol of [16] (this works on any correct view-change protocol), such that:

1) When node i sends his view-change message to the leader of view v (who became known

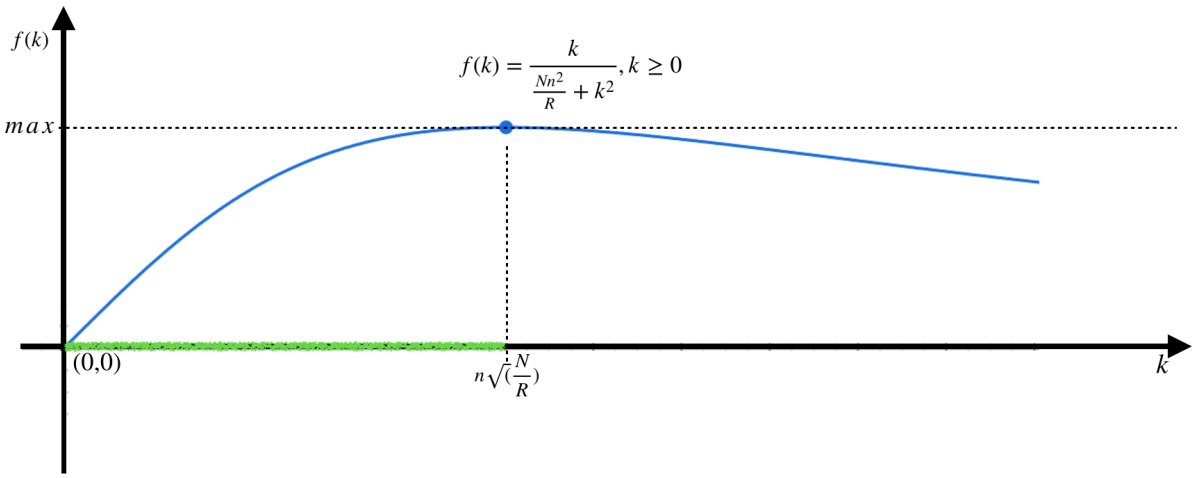

Fig. 4: Equation 2. In order to have correct incentives we want the function to be monotonically increasing within the value space of k (green range).

only during the previous round[6]) he produces a normal BLS Signature (for safety of the view-change protocol) and also produces a partial signature of v using his secret share s_i.

2) When the leader of view v collects $2f+1$ view change messages he can claim his new view only after combining $f+1$ the partial signatures to generate a signature verifiable against pk_s on v. This signature is by definition unforgeable hence unpredictable. This random number is used to define a permutation over the current schedule and as a result elect the leader of view $v+1$. The leader of $v+1$ can prove his election when needed (at the end of round v) by broadcasting sig_s, the aggregate $2f+1$ BLS-signature, and with new-view message that includes the fully signed v.

The rest of the protocol remains the same. If we assume a more adaptive adversary, then we would need to randomize the leader election during the current round. This is possible with MOTOR's rotating leaders however we would need to broadcast the view-change message since the leader of view v would be unknown. This would lead to $O(n^2)$ message complexity, which might be an acceptable trade-off.

Theorem 7: The adversary cannot predict nor bias the output of threshold signature.

Proof 7: The unpredictability property follows directly from the unforgeability property of the signing algorithm. The unbiasability property follows from the deterministic nature of the BLS-signature. Given that the signature scheme is deterministic and the message to be signed is well defined (the view number v), the adversary cannot change the resulting signature. This is not true in Schnorr signatures where the commitment is random, hence the signature is non-deterministic.

Theorem 8: A round-adaptive adversary cannot control the consensus decision forever.

Proof 8: Given the unpredictability of the leader election the only plausible strategy of the adversary is to randomly choose the nodes he controls and hope that one is elected a leader. Hence, the probability that the adversary controls d consecutive

[6] The protocol resistant to a round adaptive adversary, meaning that the leader of the current round is well-known and can receive view-change messages.

leaders is $1/3^d$. Since $\lim_{d\to\infty} 1/3^d = 0$, the adversary cannot always control the leader, hence he cannot always control the consensus decision (*e.g.,* to censor specific users).

E. Mechanism Design

A final challenge of MOTOR is to design the reward allocation so that the nodes are incentivized to participate. The key idea to our design is to bind the amount of rewards to the percentage of participation, which results to an incentive compatible reward allocation scheme.

1) Reward Allocation Incentives: We iterate our definitions. We say that there are n nodes, k of which participated in the consensus round. The total reward of the block is R and each node gets a reward of value r. Finally, the total amount of coins is N. First, we define the max reward per node as $r_{max} = \frac{R}{n}$ and then we bind the percentage of the reward the nodes receive with the participation, meaning $r = \frac{R}{n}\frac{k}{n}$. This leads to an inflation of $r * k = \frac{R*k^2}{n^2}$ which leads to a relative reward of

$$r_{rel} = \frac{\frac{R*k}{n^2}}{N + \frac{R*k^2}{n^2}}, k \geq 0 \quad (1)$$

The variable that the nodes control is k so we can transform Equation 1 to

$$r_{rel} = \frac{k}{\frac{N*n^2}{R} + k^2}, k \geq 0 \quad (2)$$

Equation 2 has a global maximum at $k = n\sqrt{(\frac{N}{R})}$ and increases monotonically for $0 \leq k \leq n * \sqrt{(\frac{N}{R})}$ (see Figure 4).

To make sure that our protocol is incentive compatible we need to move the tipping point of the curve such that even for $k = n$ the system is still on the monotonically increasing part (green part of x-axis in Figure 4) which means that $n * \sqrt{(\frac{N}{R})} \geq n$ or $\sqrt{(\frac{N}{R})} \geq 1$, hence $N > R$. As a result the incentives of MOTOR align with our system design as long as the system has inflation rate of less than 100% per block.

2) Liveness Incentives: A challenge that arises from our mechanism design is that a rational leader might be unwilling to finish committing the block unless he has 100% participation (since increasing k maximizes his reward). This incentive will eventually be overtaken by the incentive to get rewards from the next block, but in a system with a finite supply of transactions, allocating enough rewards for the next block can take a long time, hindering latency.

To prevent this non-cooperative behavior we introduce *delayed signature aggregation*. The idea is simple, the amount of the reward that block h pays the participating nodes is not defined at the time of the block commitment but at a later time which we call block encashment. Between the block commitment and the block encashment any leader can collect delayed signatures from nodes that failed to sign on time and add them to his current proposal removing the liveness-deterring incentive we described above.

At the time of encashment (*e.g.,* after 100 blocks like in Bitcoin [27]), nodes can collect all individual signatures from the chain and produce a new aggregate signature that includes the delayed signatures to increase their claimed reward. The aggregation is possible due to the non-interactivity of BLS signatures and would not be possible in CoSi. The same protocol could work with individual signatures in ByzCoin, however the encashment proof-size would then grow linearly instead of remaining constant.

V. IMPLEMENTATION

We have built a working prototype of MOTOR in Go, which includes a working implementation of BLS-based Collective Signing, running on top of the RS communication pattern. Furthermore, we modified ByzCoin to work with BLS signatures both in tree and star communication pattern and use it as a baseline for MOTOR.

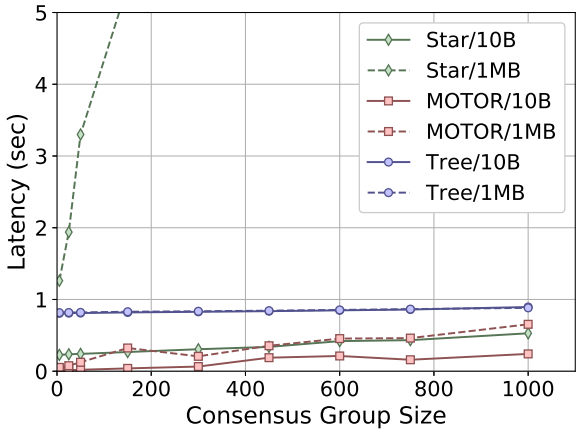

Fig. 5: Scalability

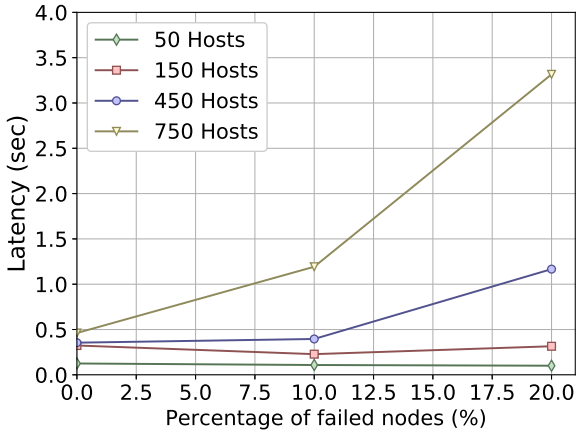

Fig. 6: Failures

Our code can be found on Github. We build on top of the BN-256 curve from Kyber[7] and evaluate MOTOR with real Bitcoin transactions creating the same test infrastructure as prior work [23], [22].

VI. EVALUATION

To simulate consensus groups of up to 1000 nodes, we oversubscribed the available 25 physical machines (see below) and ran up to 40 separate MOTOR processes on each server. Realistic wide-area network conditions are mimicked by imposing a round-trip latency of 200 ms between any two machines and a link bandwidth of 25 Mbps per simulated host. Note that this simulates only the connections between miners of the consensus group and not the full Bitcoin network. Full nodes and clients are not part of the consensus process and can retrieve signed blocks only after consensus is reached.

The key questions we want to answer with the evaluation is whether MOTOR scales well enough to support consensus group of 600 nodes, which are necessary in sharded blockchains [23], and whether it maintains this scalability even under a high percentage of failed nodes. We also provide evaluation for the star communication pattern in order to show that it is not sufficient in realistic conditions.

Figure 5 illustrates how different communication patterns scale under low (10 Bytes) and normal (1MB, which is Bitcoin's block size) load. From the graph it is obvious that MOTOR provides the best latency even against the tree (ByzCoin), because of the low roundtrip time it takes to send a block and receive a reply. The best scalability is still provided by the tree communication pattern (ByzCoin) as it is not affected neither by the increasing number of hosts nor by the increased load. This is expected and it comes in the expense of fault tolerance, as we showed in Section III-C. MOTOR comes second as it is slightly affected by the increasing number of host and nearly doubles in latency when we increase the load to 1MB, maintaining it, nevertheless, in an acceptable range. The star communication pattern comes last as it is only scalable under low load, but once the load is increased to a normal range (*i.e.,* Bitcoin's block size) the latency sky-rockets as the connection of the leader is the scalability bottleneck.

Figure 6 depicts the fault tolerance of different MOTOR configurations. It is apparent that failures do take a toll on MOTOR's performance but the system remains functional and still manages to reach consensus in a few seconds even in large scale deployments.

[7]https://github.com/dedis/kyber

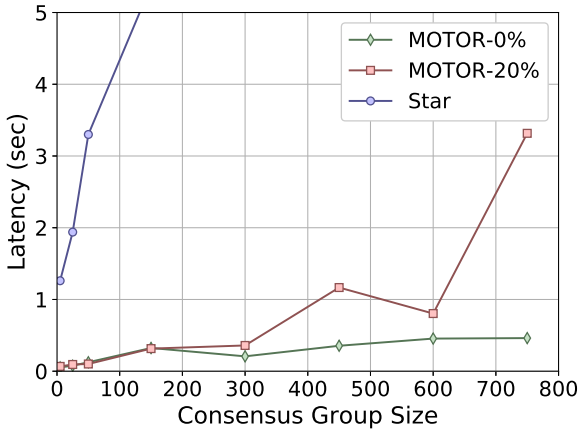

Fig. 7: Robustness

Finally, Figure 7 compares MOTOR scalability with and without faults against the star communication pattern that achieves maximum robustness. Here again we can observe that MOTOR under faults is not as fast as the normal case, however it still outperforms the optimal robustness of the star, hence becoming the best choice for large scale deployment in an adversarial environment.

VII. RELATED WORK

OmniLedger [23] was the first scalable sharding protocol that remained secure in a long term execution. However, its reliance on ByzCoin actually makes it susceptible to DoS attacks from a strong networking adversary. Concurrently, Elastico [25] and Chainspace [1] proposed sharded blockhcains for value exchange and smartcontracts, but they both relied on PBFT [10] for their evaluation which cannot scale to the necessary size for secure permissionless sharding [23]. Finally, Rapidchain [39] takes the road of changing the network assumption and restricting the adversary to sychrony. This enables higher fault thresholds for BFT, hence lower commitee sizes, where performant consensus protocols do exist. Nevertheless, in the presence of asychrony the safety of the system might be broken, hence RapidChain might be risky to deploy over the Internet.

There are several proposals that, like MOTOR, target the consensus mechanism and try to improve different aspects. Ripple [32] implements and runs a variant of PBFT that is low-latency and based on collectively-trusted subnetworks with slow membership changes. The degree of decentralization depends on the concrete configuration of an instance. Tendermint [9] also implements a PBFT-like algorithm, but evaluates it with at most 64 "validators". Furthermore, Tendermint does not address important challenges such as the link-bandwidth between validators, which we found to be a primary bottleneck. PeerCensus [11] and Hybrid Consensus [29] are interesting alternative that shares similarities with MOTOR, but are only of theoretical interest. Algorand [15] is another existing solution for BFT consensus in cryptocurrencies, it relies on the use of verifiable random functions [26] to achieve adaptivity under fully adaptive adversaries. However, in order to achieve this property it has to trade-off security or performance (*e.g.,* withstands a 20% adversary and uses the network bandwidth of 50.000 nodes to run consensus among 2.000 nodes in order to achieve good performance). Furthermore, Algorand is only deployable in the honest/malicious environment making it suitable mostly for permissioned settings. Another scalable protocol is SBFT [16], which scales better than PBFT using threshold cryptography and parsimonious broadcast, the use however of threshold signatures creates a public good's game during consensus which incentivizes free-riding and might harm the liveness of the system.

Off-chain transactions, an idea that originated from the two-point channel protocol [18], are another alternative to improve latency and throughput of the Bitcoin network. Other similar proposals include state channels [14], [5] and micro-payment channels [12], which allow transactions without a trusted middleman. They use contracts so that any party can generate proof-of-fraud on the main blockchain and thereby deny revenue to an attacker. Although these systems enable faster cryptocurrencies, they do not address the core problem

of scaling SMR systems, thus sacrificing the open and distributed nature of Bitcoin.

VIII. Conclusion

In this work we studied ByzCoin, a widely deployed blockhain consensus algorithm and showed that it is unsuitable for deployment in a real-world scenario where nodes fail and behave rationally. Then we presented MOTOR, a drop-in replacement of ByzCoin that is suitable for real-world deployment. MOTOR achieves this property by (a) prioritizing robustness in adversarial environments while maintaining adequate scalability, (b) employing provably correct cryptography that resists DoS attacks from individual nodes, (c) deploying unpredictable rotating leaders to defend against mildly-adaptive adversaries and prevents censorship, and (d) creating an incentive compatible reward mechanism. To support our claims, we proved theoretically MOTOR's safety, liveness, and censorship-resistance and showed experimentally that it can scale to 1000 with an at most $2x$ overhead and is also able to maintain this scalability under strong adversaries (up to 20%).

References

[1] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis. Chainspace: A sharded smart contracts platform. *CoRR*, abs/1708.03778, 2017.

[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15, 2018.

[3] E. Androulaki, C. Cachin, A. De Caro, and E. Kokoris-Kogias. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In *European Symposium on Research in Computer Security*, pages 111–131. Springer, 2018.

[4] M. Archetti and I. Scheuring. Game theory of public goods in one-shot social dilemmas without assortment. *Journal of theoretical biology*, 299:9–20, 2012.

[5] G. Avarikioti, E. K. Kogias, and R. Wattenhofer. Brick: Asynchronous state channels. *arXiv preprint arXiv:1905.11360*, 2019.

[6] A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *Public Key Cryptography – PKC 2003*. Springer, 2002.

[7] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains.

[8] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.

[9] E. Buchman. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains, 2016.

[10] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Feb. 1999.

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

[12] C. Decker and R. Wattenhofer. A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. In *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18. Springer, Aug. 2015.

[13] M. Drijvers, K. Edalatnejad, B. Ford, and G. Neven. On the provable security of two-round multi-signatures. Technical report.

[14] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski. Perun: Virtual payment channels over cryptographic currencies. *IACR Cryptology ePrint Archive*, 2017:635, 2017.

[15] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies, Oct. 2017.

[16] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu. Sbft: a scalable decentralized trust infrastructure for blockchains. *arXiv preprint arXiv:1804.01626*, 2018.

[17] S. Guo. Cypherium: A scalable and permissionless smart contract platform. *Draft v1. 0. www. cypherium. io/wp-content/uploads/2017/03/cypherium_whitepaper. pdf*, 2017.

[18] M. Hearn and J. Spilman. Rapidly-adjusted (micro)payments to a pre-determined party, 2015.

[19] A. Kate and I. Goldberg. Distributed key generation for the internet. In *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*, pages 119–128. IEEE, 2009.

[20] E. Kokoris-Kogias, E. C. Alp, S. D. Siby, N. Gailly, P. Jovanovic, L. Gasser, and B. Ford. Hidden in plain sight: Storing and managing secrets on a public ledger. Technical report, Cryptology ePrint Archive: 209 https://eprint. iacr. org/2018/209. pdf, 2018.

[21] E. Kokoris-Kogias, L. Gasser, I. Khoffi, P. Jovanovic, N. Gailly, and B. Ford. Managing Identities Using

Blockchains and CoSi. Technical report, 9th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2016), 2016.

[22] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *Proceedings of the 25th USENIX Conference on Security Symposium*, 2016.

[23] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *Security and Privacy (SP), 2018 IEEE Symposium on*, pages 19–34. Ieee, 2018.

[24] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: Speculative Byzantine Fault Tolerance. In *21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*. ACM, Oct. 2007.

[25] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 17–30, New York, NY, USA, 2016. ACM.

[26] S. Micali, S. Vadhan, and M. Rabin. Verifiable Random Functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 120–130. IEEE Computer Society, 1999.

[27] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[28] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford. CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1271–1287. USENIX Association, 2017.

[29] R. Pass and E. Shi. Hybrid Consensus: Efficient Consensus in the Permissionless Model. Cryptology ePrint Archive, Report 2016/917, 2016.

[30] R. Pass, C. Tech, and L. Seeman. Analysis of the Blockchain Protocol in Asynchronous Networks. *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2017.

[31] H. V. Ramasamy and C. Cachin. Parsimonious asynchronous Byzantine-fault-tolerant atomic broadcast. In *9th International Conference on Principles of Distributed Systems (OPODIS)*, Dec. 2005.

[32] D. Schwartz, N. Youngs, and A. Britto. The Ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, page 5, 2014.

[33] E. Syta, P. Jovanovic, E. Kokoris-Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable Bias-Resistant Distributed Randomness. In *38th IEEE Symposium on Security and Privacy*, May 2017.

[34] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *37th IEEE Symposium on Security and Privacy*, May 2016.

[35] The Harmony Team. Open Consensus for 10 Billion People. 2018.

[36] The Team. Emotiq Yellowpaper (v1.0). 2018.

[37] The ZILLIQA Team. The zilliqa technical whitepaper. 2018.

[38] R. Van Renesse and F. B. Schneider. Chain replication for supporting high throughput and availability. In *OSDI*, volume 4, pages 91–104, 2004.

[39] M. Zamani, M. Movahedi, and M. Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 931–948. ACM, 2018.