SoK: Consensus in the Age of Blockchains

Shehar Bano
University College London

Alberto Sonnino
University College London

Mustafa Al-Bassam
University College London

Sarah Azouvi
University College London

Patrick McCorry
PISA Research

Sarah Meiklejohn
University College London

George Danezis
University College London

ABSTRACT

The core technical component of blockchains is *consensus*: how to reach agreement among a distributed network of nodes. A plethora of blockchain consensus protocols have been proposed—ranging from new designs, to novel modifications and extensions of consensus protocols from the classical distributed systems literature. The inherent complexity of consensus protocols and their rapid and dramatic evolution makes it hard to contextualize the design landscape. We address this challenge by conducting a systematization of knowledge of blockchain consensus protocols. After first discussing key themes in classical consensus protocols, we describe: (*i*) protocols based on proof-of-work; (*ii*) proof-of-X protocols that replace proof-of-work with more energy-efficient alternatives; and (*iii*) hybrid protocols that are compositions or variations of classical consensus protocols. This survey is guided by a systematization framework we develop, to highlight the various building blocks of blockchain consensus design, along with a discussion on their security and performance properties. We identify research gaps and insights for the community to consider in future research endeavours.

KEYWORDS

blockchains, consensus, proof-of-work, proof-of-stake, Byzantine Fault Tolerance

ACM Reference Format:
Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. 2019. SoK: Consensus in the Age of Blockchains. In *1st ACM Conference on Advances in Financial Technologies (AFT '19), October 21–23, 2019, Zurich, Switzerland*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3318041.3355458

AFT '19, October 21–23, 2019, Zurich, Switzerland

ACM ISBN 978-1-4503-6732-5/19/10…$15.00
https://doi.org/10.1145/3318041.3355458

1 INTRODUCTION

Blockchains lie at the foundation of Bitcoin and other cryptocurrencies, which have a total global market capital of over $245B as of September 2019 [1]. In addition to the financial industry, blockchains have been employed in a diverse array of use cases, ranging from voting, through social networking, to the sharing economy. Despite their useful properties and applications, adoption of blockchains is nowhere near as ubiquitous as their traditional counterparts due to their performance limitations. These properties are deeply related to the *consensus* protocol—the core component of the blockchain—and we believe this is where future efforts to improve blockchain performance and scalability should be concentrated.

The consensus protocol enables a distributed network of nodes to agree on whether a data item should be added to the blockchain. A plethora of consensus protocols exist—ranging from classical consensus (e.g., Fault Tolerance and Byzantine Fault Tolerance protocols), through probabilistic consensus such as proof-of-work (PoW), to committee-based consensus that repurposes classical protocols to the blockchain setting. We lack a clear understanding of the performance and security trade-offs in the design of systems based on blockchains. A number of studies seek to improve the understanding of consensus protocols, but these efforts are fragmented across two research communities. (1) The distributed systems community is focussed on classical consensus protocols, where the literature is vast and complex, and needs additional effort to be tailoured to blockchains [22, 85, 93]. (2) The network security community is focussed on new blockchain consensus protocols, which is characterized by high-volume, fast-paced work. In this case, systematization efforts are further fragmented across different consensus themes, such as proof-of-work [16, 29, 47, 89, 99] and proof-of-stake [46].

Contributions. This paper presents a unified perspective on the design landscape of blockchain consensus protocols, that spans both the classical and the recent blockchain-driven eras. We develop a systematization framework (Section 3), which is employed to conduct a comprehensive survey that maps how consensus protocols have evolved from the classical distributed systems use case to their application to blockchains. We first discuss key themes in classical consensus protocols (Section 4), and then shift focus to PoW approaches popularized by Bitcoin (Section 5). Section 6 discusses proof-of-X (PoX) schemes, which is an umbrella

term for systems that replace PoW with more useful and energy-efficient alternatives. In the next two sections, we look at hybrid systems based on novel compositions of classical consensus primitives, or that combine classical consensus with PoW or PoX (Sections 7 and 8). We leverage our systematization of knowledge to identify gaps in existing literature and draw insights for future work—these have been highlighted throughout the paper (under the headings 'Gaps' and 'Insights').

Scope. Capturing a longitudinal *and* representative view of a topic as rich as consensus is challenging. We forego in-depth exploration in favour of capturing the breadth. Instead of describing individual works which would be clearly infeasible, we map out the landscape by extracting and evaluating high-level design themes in blockchain consensus protocols, and compare these with the classical literature on consensus where relevant. The primary focus of this paper is consensus in permissionless blockchains, where anyone can join the network; where relevant, we refer to permissioned blockchains explicitly. We focus on seminal and representative works. In the area of blockchains, a large volume of work is published in non peer-reviewed venues and as whitepapers for industrial platforms. Where possible, we prioritize papers published in peer-reviewed venues. The intended audience of this paper is systems and security researchers and engineers to help them understand the building blocks of blockchain consensus design. Therefore we do not attempt to provide formal definitions as are common in the distributed systems and formal security communities. Throughout the paper, we highlight under 'Discussion' the security and performance tradeoffs due to different blockchain consensus design choices. However, we intentionally do not provide a direct comparison between systems. This is because this paper focuses on the bricks that make up the blockchain consensus design space—protocols can suitably combine these bricks to achieve different security and performance properties.

2 BACKGROUND

We present basic concepts related to consensus and blockchains. We refer readers interested in detailed, formal consensus definitions to the work by Garay and Kiayias [45].

Consensus. The consensus protocol enables a distributed network of nodes to agree on the total order of some input values. In the blockchain context, consensus helps reach agreement on whether *transactions* should be accepted or rejected, and in which order. A transaction specifies some transformation on the blockchain state. If a transaction passes validity and verification checks (*transaction validation*), it is included in a candidate *block* (a set of transactions) to be added to the blockchain.

Consensus Leader. Consensus protocols might have a *leader* node that coordinates with other nodes to reach consensus. The leader is usually effective for an interval called an *epoch* or a *round*. If the epoch expires (or upon a fault), a new leader is elected. Leader election refers to the mechanism used to select the leader; it can be chosen deterministically (e.g., round-robin) or non-deterministically (e.g., cryptographic lottery) from the candidate nodes.

Incentives. Incentive compatibility refers to the mechanism designed to financially motivate nodes to participate in the consensus protocol. Typically this is achieved as nodes are rewarded with in-band *coins* (i.e., a block subsidy and transaction fees) for producing blocks. As a result, nodes will naturally compete amongst themselves as their reward is proportional to the number of blocks they can produce. Thus the network is intuitively a free-market as nodes are encouraged to purchase resources and compete amongst themselves, and it is self-sustaining as nodes will only compete while the in-band reward has intrinsic real-world value. Blockchains that use smart contracts require clients to include fees (e.g., 'gas' in Ethereum) to be paid to the nodes that execute the smart contracts. This not only helps to incentivize node participation, but also protects the system from overuse by discouraging clients from submitting long computations that monopolize system resources.

Permissioned vs. Permissionless Blockchains. In *permissioned blockchains*, identities of all the nodes that run consensus are known (trusted or semi-trusted), and are controlled by a single entity or federation. In *permissionless blockchains*, anyone can run a node and join the network. The primary focus of this paper is consensus in permissionless blockchains; where relevant, we refer to permissioned blockchains explicitly. (Permissioned blockchains sometimes imply limited write access. In this paper, we only refer to its meaning within the context of consensus, as defined earlier.)

Consistency. The likelihood that a network of n nodes will reach consensus on a proposed value; it can be either strong or weak. In weak consistency, the shared state across nodes might diverge temporarily leading to *forks*, and additional mechanisms are needed for reconciling forks. This is related to *eventual consistency*—i.e., the blockchain will become consistent eventually. *Finality* refers to the guarantee that a block will be permanently added to the blockchain.

Validation. A number of extensions to consensus protocols include a validation step, that ensures the transactions accepted are valid—however the validation rules must be deterministic and uniform across all nodes, and does not afford nodes any discretion about what constitutes a valid message.

Properties. In terms of the properties expected from a consensus protocol, we consider *liveness* and *safety* as enumerated by Cachin et al. [22]. For liveness, *validity* ensures that if a node broadcasts a message, eventually this message will be ordered within the consensus, and *agreement* ensures that if a message is delivered to one honest node, it will eventually be delivered to all honest nodes. For safety, *integrity* guarantees that only broadcast messages are delivered, and they are delivered only once, and *total order* ensures that all honest nodes extract the same order for all delivered messages.

Synchrony Assumptions. Networks may be *synchronous* or *asynchronous*, or offer *eventual synchrony* [36]. In a *synchronous* network the delays messages may suffer can be

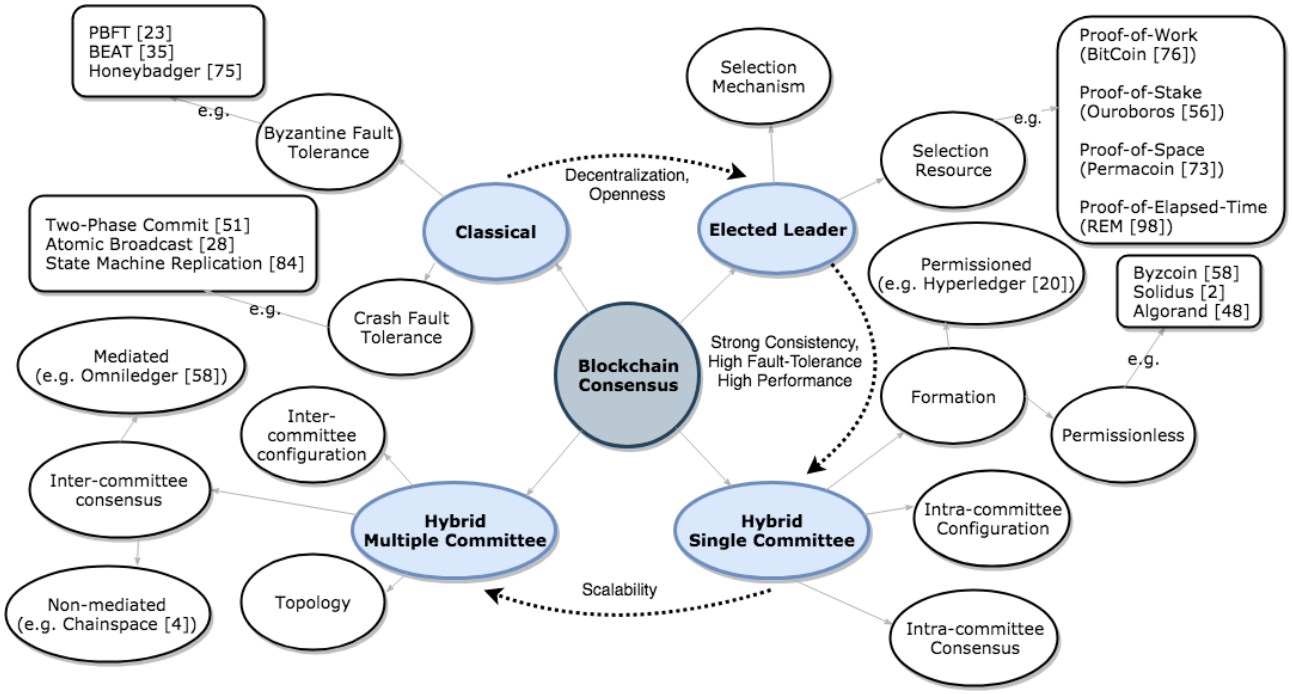

Figure 1: Systematization framework, following the chronological evolution of blockchain consensus protocols, with (not exhaustive) examples.

bound by some time Δ. On the other hand, in *asynchronous* networks messages may be delayed arbitrarily, and there exists no reliable bound Δ for their delay. Networks with partial synchrony (or eventual synchrony, or semi-synchronous networks) assume that the network at some stage will eventually be synchronous despite potentially a long period of asynchrony.

Network Propagation. Consensus protocols make certain assumptions about how messages will be propagated across nodes within the network. In *point-to-point channels*, there is a pairwise connection between all nodes which is both reliable and authenticated. In the peer-to-peer (p2p) messaging model, a node 'diffuses' a message into the network, which is expected to eventually reach all honest nodes with some probability. Every node knows a set of other nodes (*peers*)—when a message is received, nodes diffuse it by passing it on to their peers. A node may not be aware of the identities or number of other nodes in the network. *Gossip-based protocols* rely directly on this assumption by considering that each node has a point-to-point connection with at least a subset of the network; the size of that subset is a security parameter.

Communication Complexity. The communication complexity of a consensus protocol refers to the maximum number of messages exchanged between the nodes in a single run of the consensus protocol. Note that a single run might involve multiple rounds of message exchanges before it completes (i.e., consensus is reached).

Performance. The performance of consensus protocols is usually defined in terms of *throughput* (i.e., the maximum rate at which values can be agreed upon by the consensus protocol), *scalability* (i.e., the system's ability to achieve greater throughput when consensus involves a larger number of nodes) and *latency* (i.e., the time it takes from when a value is proposed, until when consensus has been reached on it).

Adversary Model for Consensus. Adversary model is the fraction of malicious or faulty nodes that the consensus protocol can tolerate (i.e., it will operate correctly despite the presence of such nodes). This is usually referred to as the *failure model* in the distributed systems literature. In the *crash failure* model, nodes may fail at any time—but they fail by stopping to process, emit or receive messages. Usually failed nodes remain silent forever, although a number of distributed protocols consider recovery. On the other hand, in the *byzantine failures* model, failed nodes may take arbitrary actions—including sending and receiving sequences of messages that are specially crafted to defeat properties of the consensus protocol. Another failure model in the context of consensus protocols relates to *network partition*: when network devices fail (or are attacked) such that the network splits into two or more relatively independent subnets.

Adversary Model for Blockchain Consensus. Blockchain consensus has extended the adversarial model to include several new threats. In consensus protocols with weak consistency guarantees, nodes might end up having different views of the blockchain (*forks*) because of latency in propagation of transactions, and faulty or malicious nodes.

A related concept is that of *double-spending* where a transaction consumes an asset which has already been consumed by a previous transaction. *DoS resistance* defines resilience of the node(s) involved in consensus to denial-of-service (DoS) attacks. If the participants of the consensus protocol are known in advance, an adversary may launch a DoS attack against them. In the context of permissionless blockchains, *Sybil attacks* refer to an attacker's ability to create fake identities or subvert existing nodes, and take over majority of the network [34].

Decentralization. This is a key property of the blockchain that enables a number of other properties such as censorship resistance, attack resistance and fault-tolerance. Decentralization has no formal definition, but generally [18] refers to a system that: (*i*) is run by multiple machines and has no architectural choke point (*architectural decentralization*); (*ii*) is run by multiple independent individuals or organisations (*political decentralization*); and (*iii*) comprises multiple interfaces and data structures that can fully operate independently, instead of acting as a single whole (*logical decentralization*). In this paper, we discuss the impact of different consensus design choices on decentralization, where relevant; however, a detailed discussion is beyond the scope of this paper and we refer interested readers to the work by Troncoso et al. [92].

3 SYSTEMATIZATION FRAMEWORK

We employ the systematization framework presented in Figure 1, following the chronological order of how blockchain consensus protocols have evolved.

Classical. Consensus protocols (Section 4) have been studied in the distributed systems community for over two decades. These protocols were intended for closed, small groups of nodes.

Elected Leader. The need to achieve consensus in open, decentralized networks motivated the design of protocols based on *elected leaders* that write to the blockchain. This may involve a combination of steps, usually applied sequentially: (*i*) *selection resource* refers to selecting a set of nodes based on some resource they own, for example via mining power in proof-of-work (Section 5), stakes in proof-of-stake (Section 6), trusted hardware etc.; and (*ii*) *selection mechanism* refers to a technique that is used to non-deterministically elect the leader. This typically takes the form of a cryptographic *lottery*—e.g., a random beacon, a periodically generated pseudo-random number, which allows the nodes to determine if they have been elected as the leader.

Hybrid Single Committee. The key limitation of protocols with elected leaders is their low performance and fault-tolerance, and weak consistency. This led to the design of *hybrid single-committee consensus* protocols (Section 7) where multiple nodes coordinate to collectively write to the blockchain. *Formation* refers to how the members of the committee are chosen; it can be permissioned or permissionless (Section 2). In permissionless blockchains, anyone can run a node and join the network. This may involve a combination of steps—i.e., *selection resource* and *selection mechanism* described earlier for 'elected leader' protocols—usually applied sequentially to elect a committee. Nodes within the committee reach agreement on a value via *intra-committee consensus*. *Intra-committee configuration* means how the nodes are assigned to the committee; either members serve on the committee permanently (static), or they are changed at regular intervals (dynamic).

Hybrid Multiple Committee. Single-committee consensus suffers from poor scalability, which motivated the design of *hybrid multiple-committee* consensus protocols (Section 8). This incurs additional coordination between the committees via *inter-committee consensus* to reach agreement on a value among nodes across multiple committees. The inter-committee consensus protocol may be run entirely by the committees (*non-mediated*), or may be mediated by an external party (*mediated*). *Inter-committee configuration* defines how nodes are assigned to the committees in a multiple committees setting; it can be static or dynamic. When multiple committees are involved in consensus, an important consideration is how they will be organized in terms of *topology*.

4 CLASSICAL CONSENSUS

In this section, we provide an overview of key themes in classical consensus literature studied since the 1970s, with the goal to contextualize rest of the paper. We will revisit some of these concepts when discussing committee-based consensus (Sections 7 and 8).

Two-Phase Commit. Jim Gray, in 1978, proposed the two-phase commit protocol [51], allowing a *transaction manager* to atomically commit a transaction, depending on different resources held by a distributed set of servers called *resource managers*. Transaction commit protocols enable distributed processing, and thus scalability—but do not provide resilience against faulty resource managers, or more generally nodes. In fact, two-phase commit suffers a deadlock in case a resource manager fails to complete the protocol, requiring the introduction of more complex three-round protocols allowing recovery [86]—i.e., the distributed resource managers being able to release the locks held on resources. Since potentially a crucial resource may only be available on a single resource manager, any failures inhibit progress towards accepting dependent transactions.

Consensus, Atomic Broadcast and State Machine Replication. The need for consensus, or atomic broadcast protocols, in distributed systems originates from the need to provide resilience against failures across multiple nodes holding *replicas* of a database. *Atomic broadcast* [28] allows a set of servers to agree on a value associated with an instance of the protocol; and *consensus protocols* extend this to agreeing on a sequence of values. This primitive is closely associated with the state machine replication paradigm [84] for building reliable distributed computations: any computation is expressed as a state machine, accepting messages to mutate its state. Given that a set of replicas start at the

same initial state, and can agree on a common sequence of messages, then they may all privately evolve the state of the computation and correctly maintain consistency across the replicated databases they hold, despite failures or network variations. The underlying consensus protocols are characterized by the communication model, as well as the failure model, assumed (Section 2).

Fischer et al. [42] show that deterministic protocols for consensus are impossible in the fully asynchronous case, and have known solutions in the synchronous case (also known as the "Byzantines General's Problem" [64]). The impossibility theorem does not take into account computational bounds on the work nodes may do—something that is exploited by both Nakamoto consensus, as well as other cryptographic solutions [21], to overcome it.

Key Protocols. In the network security literature byzantine nodes would be considered malicious or collectively controlled by an adversary. Thus the byzantine setting is of relevance to security-critical settings, and traditional consensus protocols tolerating only crash failures such as Paxos [63], viewstamped replication [78] and the more modern Raft [79] or Zab [55] cannot be used, unmodified, in adversarial settings. Practical Byzantine Fault Tolerance (PBFT) by Castro and Liskov [23] is the canonical protocol implementing consensus in the the byzantine and partially synchronous setting.

Discussion.

PBFT and other consensus protocols employ replication to achieve resilience against failures, not scalability. In fact the traditional literature on byzantine consensus does not discuss distribution of resources, in the context of a distributed or sharded database, with the exception of a less known joint work by Gray and Lamport on combining atomic broadcast with atomic commit [50]. As a result, one expects systems employing byzantine consensus to see this protocol become a bottleneck, since its trivial application would require all transactions to be sequenced by the quorum of n nodes—using protocols that are slower than asking a single processor to sequence them.

Some newer BFT protocols [35, 75] even overcome impossibility results [42], and provide both safety and liveness in a fully asynchronous setting, through a randomized consensus algorithm. This breakthrough, building upon the earlier work by Cachin et al. [21], is of notable theoretical value—however, it cannot be extended to permissionless blockchains having open node participation. Such randomized BFT protocols have traditionally been more expensive than deterministic ones, both in terms of communication and cryptographic operation costs.

5 PROOF-OF-WORK CONSENSUS

PoW consensus protocols rely on a computational puzzle to elect a leader that writes to the blockchain. As finding a solution to the puzzle requires a significant amount of computational work, so a valid solution is considered to be a proof-of-work (PoW). PoW was first presented by Dwork and Naor in 1993 as a technique for combatting spam mail, by requiring the email sender to compute the solution to a mathematical puzzle to prove that some computational work was performed [37]. PoW was independently proposed in 1997 for Hashcash by Back, another system for fighting spam [11].

5.1 Nakamoto Consensus

In 2008, Bitcoin [76] was published by a pseudonymous author Satoshi Nakamoto. Its key innovation is the use of PoW as a sybil-resistance mechanism, combined with a rule to choose between different versions of the blockchain (fork-choice rule), to achieve consensus—also called Nakamoto consensus after its originator—in an open, permissionless network. It was not until 2015—7 years after Bitcoin was first released— that it was formally proved that Bitcoin PoW is a consensus protocol [44]. While the technical components of Bitcoin originate in previous academic literature [77], their composition in Bitcoin to achieve consensus is novel.

Nakamoto consensus is based on a PoW puzzle derived from Hashcash [11], which requires finding a hash of a block that is less than a target integer value t. As the hashing algorithm is pre-image resistant, the puzzle can be solved only by including random nonces in the block until the resulting hash is valid (i.e., less than t). The difficulty of the puzzle is therefore adjustable: decreasing t increases the number of guesses (and thus work) required to generate a valid hash. The nodes that generate hashes are called *miners* and the process is referred to as *mining*. Miners calculate hashes of candidate blocks of transactions to be added to the blockchain, and are rewarded with new coins if they find a valid block. The value t is reset by the network every 2016 blocks such that miners are successful (and can append a block to the blockchain) probabilistically every 10 minutes (also called the *inter-block interval*).

Insight 1. *Nakamoto consensus relies on the cryptographic paradigm of provers and verifiers. Miners take on the role of provers who mint blocks, and every other node is a verifier who validates (and potentially rejects) blocks according to a list of globally agreed consensus rules. This is the "trust, but verify" paradigm.*

5.2 Forks in Proof-of-Work Blockchains

Forks allow an attacker to potentially double-spend assets on the blockchain. A fork occurs if two miners find two different blocks that build on the same previous block. An attacker must have sufficient computing power to be able to create a fork of the blockchain that has more accumulated work than the chain that is to be overridden. Thus the threat model assumes an adversary that has the majority of the computing power on the network (referred to as a *51% attack*). The *security threshold* of the network is the percentage of computing power required to conduct a 51% attack.

Insight 2. *Nakamoto consensus is a fork-tolerant protocol as all nodes reach eventual consistency about the blockchain's content, whereas classical consensus focuses on fork-avoidance*

protocols as nodes must have a consistent view after every epoch.

Resolving Forks: Nakamoto consensus resolves forks by accepting the 'longest chain, which has the greatest PoW effort invested in it' as the correct one. In practice, this is implemented as the chain with most accumulated work, as it is possible for a shorter chain to have more PoW than a longer chain. New policies have been proposed for the selection of the main chain in the forked blockchain to obtain a more resilient and scalable system than Bitcoin. GHOST [88] exploits blocks that are not on the main chain, achieving higher transaction rates without undermining Bitcoin security. Unlike Bitcoin's linear blockchain, GHOST organizes blocks in a tree structure. The tree is shaped by the blocks that successful miners choose to extend. The chain selection algorithm chooses the heaviest path as main chain, where a block's weight depends on how dense its subtree is.

5.3 Scaling Proof-of-Work Consensus

One approach for increasing the throughput of Nakamoto consensus is to increase the block size (i.e., the capacity to confirm new transactions) or reduce the inter-block interval (i.e., increase the block frequency). Decker and Wattenhofer showed this scaling approach is limited due to delays in block propagation [33]. Intuitively, if the block size is increased or the frequency decreased, then this provides time for a miner to produce a competing block while the solved block is in transit across the network and as a result this directly reduces the network's reliability as it increases the chance of forks occurring. Gervais et al. [47] further demonstrated that Bitcoin's block frequency can be reduced to 1 block per minute without reducing the security threshold of the existing network, modelling the bandwidth distribution of the network around real-world broadband data.

Bitcoin-NG [39] shares Bitcoin's trust model, but improves performance by separating leader election from transaction serialization (i.e., appending them to the blockchain). In each epoch, a leader is selected via PoW as in Bitcoin. Unlike Bitcoin, the leader can continue to append transactions to the blockchain for the duration of its epoch, until a new leader is elected. This allows the network's throughput to be limited only by the network's propagation delay, bandwidth and the processing capacity of the nodes.

Another approach for improving scalability, used by Spectre [87], is to allow miners to mine blocks concurrently by replacing the 'linear' blockchain structure with a Directed Acyclic Graph (*block-DAG*). Off-chain approaches to improve Bitcoin scalability such as the Lightning Network [81] have also been proposed, where parties can execute transactions off the main consensus path, and submit only the final state to the blockchain. A more detailed discussion of off-chain solutions is outside the scope of this work.

5.4 Mining Centralization in Proof-of-Work Blockchains

To reduce the variance of rewards, miners form mining pools to aggregate resources and share rewards amongst themselves. Mining pools are typically operated by a pool master who decides the content of a block and distributes the mining task to every member in the pool. Mining pools undermine the goal of decentralization as it empowers a small set of pool masters to control what is added to the blockchain.

To mitigate centralization, the PoW mechanism should be *fair*: the number of valid blocks mined by a miner should be proportional to its computing power in the network. A number of techniques have been proposed to create decentralized mining pools. SmartPool [69] implements a practical decentralized mining pool through an Ethereum smart contract, with the smart contract replacing the traditional pool manager. On the other hand, Miller et al. [74] discourage mining pools by proposing non-outsourceable proof-of-work puzzles, in which rewards can be entirely stolen from the pool manager by the entity solving the puzzle, without producing any evidence of its implication. DECOR+HOP [66] enforces fairness between miners by allowing them to share the profit when competing blocks are generated.

Some PoW blockchains like Ethereum and Monero pursue *ASIC-resistant* PoW algorithms, which have been designed such that implementing these in an ASIC will yield no significant speedup in solving the puzzle. The aim is to prevent centralization due to specialized mining hardware that only a few companies can produce. For example, Bitmain—one of the largest producers of mining hardware used in Bitcoin—was found responsible for Antbleed, a notorious backdoor that let Bitmain shut down all hardware remotely [26].

5.5 Incentives in Proof-of-Work Consensus Protocols

The security of Nakamoto consensus relies on economically incentivizing miners to validate and mine blocks, by rewarding them with new coins.

Insight 3. *Nakamoto consensus aligns financial incentives to self-enforce rational behaviour of miners. This provides a self-sustaining network as miners are rewarded in-band coins for extending the heaviest chain and for including recent transactions.*

However, previous work has shown that Nakamoto consensus is not completely incentive-compatible, and there is a potential tragedy of the commons. Collectively, miners should have an interest in the long-term success of the cryptocurrency—but in the short term, miners may deviate from the honest mining strategy to maximize their profit [15, 38, 68]. Moreover, protocol-level attacks exist that lower the security threshold of Bitcoin below 51%, such as selfish mining, stubborn mining, Fork After Withholding (FAW) attack, and eclipse attack. We only discuss here the most notable of these, selfish mining (see Gervais et al. [47] for a detailed analysis of the security of PoW blockchains).

Selfish mining [40] allows selfish miners to generate more valid blocks than their computing power would normally allow them to if they were following the honest mining protocol. In selfish mining, instead of broadcasting a solved block immediately to the network, the miner withholds their solved block from the rest of the network and begins solving the next block. This provides the selfish miner a head start on solving the next block and effectively wastes the remaining network's computational power as they attempt to solve an already solved block. If the network is close to catching up with the selfish miner, then the selfish miner can release a portion of their withheld blocks to the network (i.e., the selfish miner can be 1 block ahead for the longest and heaviest chain) and overtake the honest network. Using this mining strategy, it is possible to conduct a 51% attack against the network with as little as 25% of the network's computing power.

A number of mitigations to selfish mining have been proposed. For example, Fruitchain [80] mitigates selfish mining by using two independent mining processes on top of each other: in addition to the PoW to create blocks, Fruitchain requires an additional PoW to mine a new type of block, called 'fruit'. Blockchain transactions are included into these fruits, and the fruits are included into the blocks created by the first mining process. This mechanism prevents selfish miners from dropping honest blocks from the blockchain by releasing their withheld blocks because eventually, an honest block will be created and will include back all the dropped fruits.

6 PROOF-OF-X CONSENSUS

One of the biggest criticisms of Bitcoin is that it is based on power-intensive PoW that has no external utility, and makes it prone to centralization (Section 5.4). These limitations of PoW motivated a new class of consensus protocols based on proof-of-X (PoX) that replace wasteful computations with useful work derived from alternative commonly accessible resources, or remove computational work altogether. We discuss the most prominent PoX protocols, based on stake, space and time—out of a number of other alternatives [16], notably proof-of-deposit where a miner's voting power is proportional to the amount of coins locked (e.g., Tendermint [62]), and proof-of-coin-age where the quantity of coins is weighed by their coin-age (adopted by Peercoin[1]).

6.1 Proof-of-Stake

In proof-of-stake, participants vote on new blocks weighted by their in-band investment such as the amount of currency held in the blockchain (*stake*). A number of recent systems have provably secure proof-of-stake protocols [30, 32, 56]. A common theme in these systems is to randomly elect a leader from among the stakeholders (participants) via lottery, which then appends a block to the blockchain. Leader election may be public, that is the outcome is visible to all the participants [30, 56]. Alternatively, in a private election the participants use private information to check if they have been selected as the leader, which can be verified by all other participants using public information [32].

Insight 4. *Leader election based on private lottery is resilient to DoS attacks because participants privately check if they are elected before revealing it publicly in their blocks, at which point it is too late to attack them.*

The nature of the lottery varies across different systems, but broadly it is either collaborative (i.e., requires coordination between the participants) or independent. In Ouroboros [56], the participants (a random subset of all stakeholders) run a multiparty coin-tossing protocol to agree on a random seed. The participants then feed this seed to a pseudo-random function defined by the protocol, that elects the leader from among the participants in proportion to their stake. The same random seed is used to elect the next set of participants for the next epoch. In Ouroboros Praos [32] and Snow-White [30] participants independently determine if they have been elected. Snow-White selects participants for each epoch based on the previous state of the blockchain, who independently check if they have been elected as the leader. Snow-White uses similar criteria for leader election as Bitcoin, that is finding a pre-image that produces a hash below some target. However, participants are limited to compute only one hash per time step (assuming access to a weakly synchronized clock) and the target takes into account each participant's amount of stake. In Ouroboros Praos, participants generate a random number using a verifiable random function (VRF). If the random number is below a threshold, it indicates that the participant has been elected as the leader, who then broadcasts the block along with the associated proof generated by the VRF to the network. Ouroboros and Ouroboros Praos distribute rewards among all the participants regardless of whether or not they win the election. Snow-White employs the incentive structure of Fruitchain [80] (Section 5.5).

Insight 5. *PoW's leader election eligibility is out-of-band, and all nodes verify the leader election's result only so far as to find the longest and heaviest chain. Whereas in proof-of-stake the entire leader-election protocol transcript is recorded in-band which increases the nodes' storage, bandwidth and validation overhead for every block.*

A challenge for proof-of-stake systems is to keep track of the changing stakes of the stakeholders. Ouroboros requires that shift in stakes is bounded, meaning the statistical distance is limited over a certain number of epochs. Additionally, Snow-White looks at stakes sufficiently far back in time to ensure that everyone has agreed on the stake distribution.

Outside academia, some deployed cryptocurrencies incorporate proof-of-stake (e.g., Peercoin), but their designs have not been rigorously studied. Ethereum Foundation has been considering using proof-of-stake for some time [14], and some systems like EOS[2] use delegated proof-of-stake, where participants elect delegates of their choice for mining.

Attacks and Mitigation. Proof-of-stake results in three new attacks compared to Nakamoto consensus [24]. The

[1] https://peercoin.net/

[2] https://eos.io

first is called the *nothing-at-stake attack* where miners are incentivized to extend every potential fork. Since it is computationally cheap to extend a chain, in the case of forks rational miners mine on top of every chain to increase the likelihood of getting their block in the right chain. One way of dealing with this is to introduce a penalty mechanism: a miner producing blocks on different forks is penalized by having part of their stake taken [30]. Another mitigation against nothing-at-stake is to remove the opportunity for forks in the consensus protocol altogether as proposed by Algorand [48]. The second attack is called the *grinding attack* where a miner re-creates a block multiple times until it is likely that the miner can create a second block shortly afterwards. This attack can be thwarted by ensuring that a miner is not able to influence the next leader election by using an unbiasable source of randomness or a non-deterministic leader election. In the third attack called the *long-range attack*, an attacker can bribe miners to sell their private keys. If these keys had considerable value in the past, then the adversary can mine previous blocks and re-write the entire history of the blockchain. This is possible because the bribed miners have already received their external utility for these coins (i.e., sold the coins for fiat currency), and no longer have a stake in the system. Thus the bribed miners can send their keys to the adversary at almost no cost. This can be thwarted by central checkpointing: some entity (e.g., one of the main developers) declares that some blocks are final if they are sufficiently far in time, or by requiring participants to lock their coins for a longer period of time than the duration of their participation.

6.2 Proof-of-Space

In proof-of-space, participants vote on new blocks weighted by their capacity to allocate a non-trivial amount of disk space. PermaCoin [73] repurposes Bitcoin's PoW with a more broadly useful task: providing a robust, distributed storage. In PermaCoin, eligibility for the leader election requires participants to also store segments of a large file. The file is distributed by an authoritative 'dealer' who signs file blocks. To provide censorship-resistant file storage, the file is fully recoverable from the participants in the event of a dealer failure or shutdown. SpaceMint [53] employs a consensus protocol based on a non-interactive variant of proof-of-space, where participants generate and commit to a unique hard-to-pebble graph. PermaCoin and SpaceMint have the same basic model as Nakamoto consensus, so they inherit Bitcoin's incentivization mechanism, as well as its resilience against censorship and DoS.

Attacks and Mitigation. Proof-of-capacity is vulnerable to centralization due to participants outsourcing the file storage to an external provider. To mitigate this problem, the proof-of-retrievability in PermaCoin requires sequential read access to blocks in a pseudorandom order: this directly increases the bandwidth latency in case of outsourced storage, which reduces the miner's chance of finding a solution.

6.3 Proof-of-Elapsed-Time

Using the trusted enclave in Intel SGX, it is possible to replace computational work with proof-of-elapsed-time [54]. Participants request a wait time from their enclave and the chip with the shortest wait time is elected as the leader. The newly elected leader can provide an attestation alongside the new block to convince other participants that: (*i*) it indeed had the shortest wait time, and (*ii*) that it did not broadcast the block until after the wait time had expired.

An alternative approach is called Resource-Efficient Mining (REM) [98] that proposes computing useful PoW using trusted hardware. Every instruction cycle for the useful PoW can be seen as a lottery ticket: if a cycle wins the lottery, the participant is authorized to mint a new block. To extend this model to arbitrary work, the authors introduce a two-layer hierarchical attestation. The first layer certifies that useful PoW was performed, and the second layer attests that the program (and its input) incremented the counter for instruction cycles appropriately. A hash of both layers is sent alongside a new block to prove that the participant was authorized to mint it.

Attacks and Mitigation. Both proof-of-elapsed-time approaches suffer from two limitations. First, breaking a single piece of trusted hardware enables the attacker to always win the lottery. Both Sawtooth and REM argue that a statistical analysis of newly minted blocks suffices to detect whether a chip can be compromised. Second, the *stale chip problem* highlights that it is advantageous to collect chips as this increases the probability of minting a new block (i.e., every new chips is an additional lottery ticket). REM provides an economic analysis to show that a miner's revenue source originates from useful work, and not farming chips.

Discussion.

Centralization of the consensus protocol is an important issue in PoX protocols (previously discussed in the context of PoW in Section 5.4). To thwart the centralization problem, Brünjes et al. [17] propose a reward scheme that achieves a desirable number of pools in proof-of-stake blockchains.

In a proof-of-stake system, the "rich get richer" problem is particularly important as stake can get reinvested in the mining process as soon as it is received. To solve this problem Fanti et al. [41] propose a reward scheme that ensures that stake holders amplify their stakes in an *equitable* way in order to ensure fairness in the long term.

A general problem is the lack of suitable evaluation criteria to compare PoX systems. Gauba et al. [46] provide a preliminary investigation, but a formal model has not yet been adopted.

7 HYBRID CONSENSUS: SINGLE COMMITTEE

The elected leader approach suffers from poor performance as well as safety limitations such as weak consistency and low fault-tolerance. This has resulted in a shift towards consensus protocols where a *committee*—rather than a single node—collectively drives the consensus.

7.1 Committee Formation

This refers to the criteria used to allow nodes to join a committee. *Permissioned* blockchains like Hyperledger [20] operate in a trusted environment where nodes are granted committee membership based on the organizational policy. In permissionless blockchains, the committee is formed so as to thwart sybil attacks. Nodes are usually allowed to join the committee based on a *selection resource* such as *PoW*. In ByzCoin [57], the consensus committee is dynamically formed by a window of recent miners. Each miner has voting power proportional to its number of mining blocks in the current window, which is proportional to its hash power. When a miner finds a solution to the puzzle, it becomes a member of the committee and receives a share in the consensus. In addition to PoW, Omniledger [58] also supports *proof-of-stake* to allocate committee membership based on directly invested stake. Some permissionless blockchains employ a further *selection mechanism* such as a *lottery* to form the committee. In Algorand [48], all the nodes that have PoX run a verifiable random function—they are promoted to the committee if the output is below a certain value.

Discussion.

7.1.1 Sybil Resistance. A limitation of using PoW or PoX for sybil resistance in permissionless committees is that the biggest miners will have a greater likelihood of dominating the committee, though at the cost of significantly more hashing power than required for single-leader PoW systems. Other PoX alternatives, relying on space, memory, or space-time, have been proposed but these suffer from similar issues.

Gap 1. *Protocols have also been proposed for sybil detection based on the analysis of social networks and trust graphs [6], but those have not been adapted to blockchains, besides the definitional framework for Federated Byzantine Agreement Systems proposed by Stellar [72].*

7.1.2 Coercion Resistance. Another consideration for bootstrapping committees is to achieve coercion resistance, in the form of requiring enormous effort for an adversary to suppress the overall operation of the system. Coercion resistance properties are also key to the success of other decentralized systems, such as BitTorrent [25], that are subject to take-down pressures by publishers. Bitcoin itself was coincidentally proposed in 2008, the same year when E-gold was declared illegal by the US Department of Justice and taken offline—illustrating that value exchange systems, and monetary systems that are transnational and unregulated, will come under fire by national monetary and law enforcement authorities. Systems such as Tor[3] have survived in a highly adversarial environment despite parts of its infrastructure, namely directory authorities, being a closed consensus group. These authorities are distributed geographically, and are under different jurisdictions and managed by different organizations to preclude both collusion and single jurisdiction attacks.

[3] https://www.torproject.org

Insight 6. *Single-committee blockchains may, through careful selection of nodes, achieve coercion resistance.*

7.2 Intra-Committee Configuration

Intra-committee configuration means how nodes are assigned to the committee. In *static* configuration, nodes are statically assigned to the committee, and are allowed to stay on indefinitely. Static configuration is typically employed in permissioned blockchains like Hyperledger and RSCoin. In *dynamic* configuration, committee members are changed periodically. This model is typically used in permissionless blockchains as this helps thwart sybil attacks. Dynamic committee membership can take three forms. (1) In *rolling (single)* membership, the committee is updated in a sliding window fashion, i.e., a new node replaces the oldest committee member periodically. In ByzCoin, when a miner finds a solution to the puzzle, it becomes a member of the committee and receives a share in the current consensus window which moves one step forward (ejecting the oldest miner). (2) *Rolling (multiple)* committee membership is a similar concept, where multiple committee members are replaced periodically. Omniledger uses cryptographic sortition to select a subset of the committee to be swapped out and replaced with new members. (3) Some systems replace the *full* committee, e.g., Algorand and SnowWhite select the committee members for each epoch using randomness generated based on previous blocks.

Discussion.

7.2.1 Identity Management. A number of recent blockchains [4, 31, 58] employ BFT for reaching consensus among committee nodes. However, traditional BFT protocols are inherently 'closed', in the sense that committee members need to have point-to-point, reliable and authenticated channels between them (i.e., the *view* of the network). In traditional BFT, nodes may employ a *failure detector* that sends regular ping messages to detect when a member has left the committee or an unreliable leader, and initiate a *view change* if required. A limitation of this approach is that malicious members can slow down or stall the committee by constantly generating false alarms for eviction of legitimate members. Addressing this would require rate-limiting the number of leave operations a member can propose in a given time interval.

Gap 2. *Traditional BFT protocols cannot accommodate open node participation and high churn, especially in the dynamic committee configuration which requires additional mechanisms to track committee members and their keys across reconfiguration events.*

A naïve solution is for all nodes to regularly broadcast their identity to the entire network (along with evidence that they have been granted permission to join the committee) resulting in $O(n^2)$ messages. A better approach to is to form a special committee that offers directory services to new committee members [67]. However, this presents a dilemma: a static committee undermines decentralization, but forming a decentralized directory committee suffers from the same

challenges as the committee aims to solve in the first place. Another technique, used by Omniledger [58], is to record committee members for each round in a separate 'identity' blockchain—however, its details are not provided.

Insight 7. *The challenges in identity management in BFT-based permissionless blockchains has motivated a shift to gossip-based consensus protocols [83] where a node is only required to know (and send messages to) a small set of other nodes, and messages are diffused across the network.*

7.2.2 Liveness in Dynamic Intra-Committee Configuration. Dynamic intra-committee configuration improves security by raising the bar for sybil attacks, but introduces a new challenge: how is liveness maintained during reconfiguration? One approach is to only do rolling configuration, which has the benefit that the committee is operational during reconfiguration as the operational members can continue to process transactions while a fraction of the committee is being reconfigured and bootstrapped. Omniledger uses cryptographic sortition to select a subset of the committee to be swapped out and replaced with new members. This is done in such a way that the ratio between honest and byzantine members in a committee is maintained. In Solidus [2], a new miner joining the committee can propose transactions only once. This binds transaction proposals to reconfiguration, so it is no longer possible for an old committee to approve transactions concurrent to a reconfiguration event.

Gap 3. *Maintaining liveness and security in dynamic intra-committee configuration setting is an open and neglected research area in the design of single-committee blockchains.*

7.2.3 Denial-of-Service (DoS) Attack. An adversary can launch DoS attack on the blockchain committee. Small statically configured committees are particularly vulnerable. Systems that do a full swap [32, 48, 67] of the committee, and have small epochs are highly resilient against DoS attacks—but this may introduce liveness challenges (Section 7.2.2). Rolling configuration [57, 58] can be tuned to provide optimal tradeoffs between DoS resilience and liveness.

Insight 8. *A blockchain committee's vulnerability to DoS attacks is directly related to the intra-committee configuration, i.e., how frequently and what fraction of committee membership changes (epoch, dynamism).*

7.3 Intra-Committee Consensus Protocol

The intra-committee consensus protocol ensures that the committee members reach agreement on state of the blockchain. A number of committee-based blockchains [4, 58, 67] use PBFT. The messaging complexity of PBFT's MAC-authenticated all-to-all communication is $O(n^2)$. This is problematic for permissionless blockchains where a committee can potentially have thousands of nodes. Another approach to reach intra-committee consensus is based on gossip protocols. These protocols are suited to permissionless blockchains as point-to-point connections between the n nodes of the committee are no longer needed. Upon reception of a new transaction, nodes query a subset of k randomly select others nodes; each of those nodes replies with its view of the state of the system, and initiates a similar query. The requesting node weights the replies and potentially updates its own view of the state; this process is repeated until global consensus is reached. Avalanche [83] introduces a gossip-based family of BFT protocols that have a communication complexity of $O(k \times n)$, where $k << n$ is a security parameter. These protocols are leaderless and present strong DoS and censorship resistance.

Discussion.

7.3.1 Committee Leadership. Traditional BFT protocols proceed in rounds, where consensus in each round is led by a committee leader. The concept of a committee leader is not compatible with permissionless blockchains that aspire to achieve the design goal of decentralization. An adversary can concentrate its DoS attack on committee leaders which are easy to discover by joining the committee. As the leader is responsible for proposing transactions, a malicious leader can prioritize transactions from which it can benefit. While committee members can potentially detect a malicious leader and trigger leader re-election (i.e., view change), this severely degrades performance [7]. Solidus highlights a safety problem in PBFT's 'stable' leader which can potentially manipulate reconfiguration by waiting for a malicious miner to solve the puzzle, and later nominating it on to the committee—allowing the committee to gradually become dominated by corrupt members.

Insight 9. *The concept of a leader in committee-based blockchains introduces a number of challenges with respect to scalability, and security (DoS attack, transaction censorship, and centralization). This has motivated the design of leaderless consensus protocols.*

7.3.2 Optimizations. A number of optimizations have been proposed to improve the performance of BFT consensus protocols. *Scheduling optimizations* involve techniques to identify and execute non-conflicting transactions in parallel (and thus achieve high throughput) by leveraging application-specific information [61]. *Execution optimizations* reduce latency by allowing clients [95] or replicas [60] to speculatively execute transactions based on predicted results—if a fault is detected (i.e., the speculation turns out to be incorrect), the client/replica rolls back its state to the last checkpoint and re-executes the transactions based on the correct results. *Protocol optimizations* refer to the committee's ability to switch between suitable BFT protocols according to varying network conditions and performance requirements [52]. Hyperledger uses *pluggable and modular* consensus in which the consensus protocol can be specified by the smart contract policy. *Cryptographic optimizations* leverage advances in cryptography to optimize the communication complexity of BFT. ByzCoin organizes the consensus committee into a communication tree that uses a primitive called scalable collective signing [91] which reduces PBFT's messaging complexity to $O(n)$. *Hardware optimizations* enable consensus protocols to achieve high performance by exploiting advances in hardware. The Intel Sawtooth lake system uses the Intel SGX and related trusted

execution environments to perform the duties related to ordering transactions, while ensuring safety and liveness [82]. Finally, *architectural optimizations* improve performance by distributing different consensus duties across independent subsets of replicas. A useful paradigm (employed by Hyperledger) is to separate ordering from execution [96], which allows for a modular design where transaction validation is performed by the fully trusted nodes (or endorsers) while the semi-trusted nodes (ordering nodes) order the transactions and add these to the blockchain. Others [94] argue that distributed ledgers can decouple the ordering—performed in public on cryptographic commitments of transactions—from the validation containing private information, that is only checked by a trusted cabal.

Insight 10. *Separating ordering from execution [96] allows committee-based blockchains to scale at the same rate as the core ordering protocol, but does not provide any universal end-to-end verifiability and undermines decentralization.*

7.4 Incentivization

Classical BFT protocols assume two kinds of players: *cooperative* and *byzantine*. This assumption works well in 'closed' group settings where nodes are controlled by the same entity or federation. However, permissionless blockchains need to provide incentives to nodes for active participation in consensus [57, 58, 67] as well as information propagation [2, 10, 13]. However, with no clear incentives, the cooperative committee members have nothing to gain from participating in the consensus, which introduces a third kind of player: a *rational* player that, for each action it performs, assesses its expected utility in terms of the rewards it will receive.

This has led to the design of incentive-compatible consensus protocols, where incentives are built into the core of the protocol (e.g., Solidus [2]). Classical techniques such as rational cryptography [19, 43] and the BAR model [3] could be adapted to work here. Due to the unavoidable selfish behaviour observed in distributed systems, the BAR model was introduced to construct systems that can tolerate both Byzantine and rational players. The general approach is to analyze the classical BFT protocol in the presence of rational players, and modify the sources of cost and benefits to make it resilient to rational behaviour.

Insight 11. *The committee should be reconfigured regularly to maintain a suitable committee size: large committees might result in trivial rewards for individual committee members (or might lead to inflation of client fees to account for the difference).*

Discussion.

An important question in the context of committee-based blockchains is who distributes the incentives? In leader-based systems like Solidus, the leader of the committee distributes incentives among the first $2f + 1$ responders. This approach has several limitations: (*i*) a faulty or malicious leader might not divide the rewards; (*ii*) there is no way to enforce that the leader rewards the genuinely fast responders, so the leader can instead wait for its favourite members to reply; and (*iii*) the notion of 'fast' is problematic in open, permissionless networks where members located farther from the leader are at a natural disadvantage.

Gap 4. *Broadly, incentivization in committee-based blockchain consensus protocols (with leader) has started to see some study, but is far from mature. Incentivization in leaderless consensus protocols remains largely unexplored.*

This area will benefit from combining formal economic and game theoretic analysis with cryptography, such as has already been done in the blockchain community [9, 12, 59, 81].

8 HYBRID CONSENSUS: MULTIPLE COMMITTEES

Single-committee consensus is not scalable and adding more nodes to the committee decreases throughput—leading to the design of consensus based on multiple committees. Transactions are split among multiple committees (called *shards*) which then process these transactions in parallel.

8.1 Committee Topology

When multiple committees are involved in consensus, an important question is how they will be organized in terms of topology. Chainspace [4] and Omniledger [58] have flat topologies, that is all committees are at the same level. Elastico [67] has a hierarchical topology in which a number of 'normal' committees validate transactions, and a leader committee orders these transactions and extends the blockchain. In RSCoin [31], a permissioned blockchain, the central bank controls all monetary supply while committees (called mintettes) authorized by the bank validate a subset (shard) of transactions. The transactions that pass validation are submitted to the central bank which adds them to the blockchain.

Insight 12. *Hierarchical topology facilitates configuration and management of committees in multi-committee blockchains, but undermines decentralization.*

8.2 Committee Formation

Multi-committee blockchains raise the additional issue of how to map nodes to committees. In permissioned systems, the process of assigning nodes to committees is usually done *statically* according to the policy of the federation. Another approach is to *dynamically* allocate nodes to committees. Permissioned systems like RSCoin can use a trusted source of randomness for committee reconfiguration, but this can be problematic in a permissionless setting which would require a shared random coin [27, 49]. Generating good randomness in a distributed way is a known hard problem: current best solutions tolerate up to 1/6 fraction of byzantine nodes, while incurring a high message complexity [8]. Among the more recent solutions, RandHerd [90] provides a more scalable, secure multi-party computation protocol that offers unbiasable, decentralized randomness while tolerating a third of Byzantine faults. It brings down the communication complexity to $O(c^2\log(n))$, where c is the size the subgroups it uses.

Insight 13. *In multi-committee blockchains, nodes should be assigned to committees in a non-deterministic way to stop an adversary from concentrating its presence in one committee and exceeding the byzantine-tolerance threshold.*

Discussion.

8.2.1 Secure committees. The idea of scaling services built on state machine replication (SMR) by splitting state (or sharding) among multiple committees (also called partitions or shards) has been well-studied in the context of traditional distributed systems [27, 49, 65]. These systems employ fault-tolerant BFT protocols at their core as the nodes are controlled by a single entity or a group of entities that collectively govern the system. Due to similar governance assumptions, these techniques can be extended to permissioned blockchains.

Sharding permissionless blockchains with byzantine adversaries is challenging and tackled by only a few recent systems [4, 58, 67]. Individual committees can tolerate up to 33% of malicious members, but if this is not the case then the malicious committee can compromise all the transactions that touch the bad committee. Chainspace starts mitigating this issue by making the author of the smart contract responsible for designating the parts of the infrastructure that are trusted to maintain the integrity of its contract. Moreover, Chainspace provides an auditing mechanism allowing honest node in honest committees to detect inconsistencies and discover the malicious committee.

Gap 5. *In multi-committee blockchains, a single malicious committee can compromise security of the entire system. There is some preliminary work on detecting a malicious shard, however there are no systems today providing a recovery mechanism.*

8.2.2 Committee Governance. Randomly mapping nodes to committees improves security, but prohibits finer governance. General-purpose platforms like Chainspace might have different policies within committees; for example some committees can be permissioned while others can be permissionless. In this case it might be useful to enforce node-to-shard mapping via smart contracts that allow a node to join a committee trusted by the smart contract provider.

8.2.3 Coercion Resistance. It is crucial for a value exchange system based on blockchains that its clients believe it will exist as a medium of value in the future—and thus the potential for future disruption of the network, would reduce its value significantly. Thus, if the key feature of PoW schemes is the robustness and coercion resistance resulting from their openness, multi-committee blockchains that sacrifice this property may fail regardless of their superior performance. Similar to single-committee systems (Section 7.1.2), multi-committee systems have to consider forming committees in such a way that the system is resilient against coercion.

Insight 14. *Multi-committee blockchains can achieve coercion resistance within each committee via careful selection of nodes; and across committees by creating 'backup' committees that mirror the 'primary' committees, and can replace primaries that are taken down.*

8.3 Inter-committee Configuration

Inter-committee configuration means whether node assignment to committees in a multi-committee blockchain remains *static* or is periodically changed (*dynamic*). Omniledger periodically reconfigures committees to ensure that a committee is never compromised. This is achieved by a secure shard reconfiguration protocol, based on RandHerd, that committee members run periodically and autonomously. In every epoch, a random subset of members is replaced with new set of members that registered their interest in the previous epoch.

Insight 15. *Dynamic inter-committee configuration prevents an adversary from subverting existing nodes in a committee and exceeding the byzantine-tolerance threshold.*

Discussion.

8.3.1 State Transfer in Dynamic Configuration. Multi-committee blockchains achieve a different notion of verifiability from single-committee blockchains, as it is no longer clear how to define a global set of transactions. For example, in Omniledger and Chainspace every committee is responsible for managing a subset of transactions, and defines its own blockchain corresponding to those transactions. If nodes are dynamically reconfigured across committees, there needs to be a mechanism for 'blockchain handover'.

Gap 6. *Multi-committee blockchains achieve scalability by sharding transactions and state, but tend to overlook the issue of state transfer when node membership changes in dynamic inter-committee configuration.*

8.3.2 Liveness Issues in Dynamic Configuration. Similar to liveness in dynamic intra-committee configuration (Section 7.2.2), multi-committee systems need mechanisms to ensure liveness during inter-committee reconfiguration. Omniledger addresses this issue by swapping only a subset of committee members at a time, so every committee has enough nodes to remain operational despite ongoing partial reconfigurations.

Insight 16. *To minimize system-wide liveness issues, inter-committee reconfiguration events should involve only a few committees at a time so the system is at least partially operational.*

8.3.3 Denial-of-Service (DoS) Attack on Committees. An adversary can launch Denial-of-Service (DoS) attack on one or more committees. Small statically configured committees are more vulnerable, while dynamic configuration makes committees more resilient to DoS attacks.

Insight 17. *Multi-committee blockchains are highly resilient against DoS attacks—a successful attack will only affect the transactions managed by the 'victim' committee, and rest of the system will remain operational.*

8.3.4 Committee Discovery and Identity Management. Elastico has an explicit overlay setup (a fully-connected subgraph) for committees that describes how members in the same committee will discover each other. Instead of broadcasting information which has $O(n^2)$ messaging complexity, they provide a methodology that requires $O(nc)$ broadcast messages, where c is the number of committees. A special committee serves as a set of directories which can be queried by a new member to find other members in its committee. The directories and the committee members can tolerate different views of the member set up to a threshold. RapidChain [97] uses an inter-committee routing protocol based on Kademlia [71] allowing node discovery and message routing in $O(\log(n))$ steps (where n is the number of nodes in the system).

Gap 7. *Multi-committee blockchains require each committee to have a collective identity, and some way for the committees to discover each other; most systems abstract these details.*

8.4 Inter-Committee Consensus

In a multi-committee system, some transactions might manipulate state that is handled by different committees. The inter-committee consensus ensures that such transactions are processed consistently and atomically across all the concerned committees. One approach is to mediate the inter-committee consensus protocol via the client. Omniledger uses an atomic commit protocol to process transactions across committees. A transaction submitted by a client is processed by the committees that manages the transaction inputs. Each related committee validates the transaction, and returns a proof-of-acceptance (or rejection) to the client, and locks the transaction inputs. To unlock the inputs, the client sends proof-of-accepts to the committees that manage the transaction outputs, who add the transaction to the next block to be appended. If the transaction fails the validation test, the client can send proof-of-rejection to the input committees to roll back the transaction and unlock the inputs.

Another approach, used by Chainspace, is to run an atomic commit protocol collaboratively between all the concerned committees. This is achieved by making the entire committees act as resource managers for the transactions they manage.

Discussion.

Client-driven inter-committee consensus protocols make the assumption that clients are incentivized to proceed to the unlock phase. Such incentives may exist in a cryptocurrency application where an unresponsive client will lose its own coins if the inputs are permanently locked, but do not hold for a general-purpose platform where transaction inputs may have shared ownership.

Insight 18. *Client-driven inter-committee consensus protocols are vulnerable to DoS attack if the client stops participating midway, resulting in the transaction inputs being locked forever.*

Gap 8. *Generally, inter-committee consensus protocols are relatively immature, and their security has not been rigorously evaluated.*

For example, some preliminary results [5] show the susceptibility of these protocols to replay attacks that allow an attacker to double-spend or lock resources with minimal effort, and without colluding with any nodes. The attacker records a target committee's responses to the consensus protocol, and replays them during another instance of the protocol. The attacks succeed even if the individual committees satisfy the byzantine safety criteria.

8.5 Incentivization

Multi-committee blockchains need to provide incentives to committee members for active participation in the inter-committee consensus protocol. Manshaei et al. [70] analyze the strategic behaviour of rational nodes within committees using a game-theoretic model. They propose an incentivization model where a shard coordinator splits block rewards among participating nodes. This degrades the DoS-resistance and censorship-resistance properties of the consensus protocol, and undermines decentralization. Furthermore, they relax the threat model to only consider rational adversaries, where each node aims at maximizing its reward and at minimizing its cost in the protocol participation; but do not consider the traditional byzantine nodes which can arbitrarily deviate from the protocol. Finally, fair incentivization for inter-shard communication remains an open question.

Gap 9. *There has been little investigation into how to build incentives into inter-committee consensus protocols in the context of multi-committee blockchains.*

9 CONCLUSION

The last few years have seen a dramatic surge in blockchain consensus protocols, as a result of which the field has grown increasingly complex. We presented a comprehensive systematization of blockchain consensus protocols. In a broader context, this work has highlighted a number of open areas and challenges related to: (*i*) gaps between classical consensus protocols and their applications to blockchains, (*ii*) security vs. performance tradeoffs, and (*iii*) incentives. This longitudinal perspective makes a timely contribution to the prolific and vibrant area of blockchain consensus protocols: the wide-scale adoption of blockchains is constrained by their performance and scalability limitations, and is desperately in need of new and faster consensus protocols that can cater to varying requirements and use cases.

ACKNOWLEDGMENTS

George Danezis, Shehar Bano and Alberto Sonnino were supported in part by EPSRC Grant EP/M013286/1 and the EU H2020 DECODE project under grant agreement number 732546. Mustafa Al-Bassam is supported by a scholarship from The Alan Turing Institute. Sarah Meiklejohn and Sarah Azouvi are, and Patrick McCorry and Shehar Bano (in part) were, supported by EPSRC grant EP/N028104/1. Thanks to Adrian Colyer for inspiring Figure 1.

REFERENCES

[1] 2018. CoinMarketCap: Global Charts. https://coinmarketcap.com/charts/.

[2] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. 2016. Solidus: An Incentive-compatible Cryptocurrency Based on Permissionless Byzantine Consensus.

[3] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. 2005. BAR Fault Tolerance for Cooperative Services. *SIGOPS Operating Systems Review* 39, 5 (Oct. 2005), 45–58.

[4] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. 2018. Chainspace: A Sharded Smart Contracts Platform. In *The Network and Distributed System Security Symposium*.

[5] Mustafa Al-Bassam Alberto Sonnino, Shehar Bano and George Danezis. 2019. Replay Attacks and Defenses Against Cross-shard Consensus in Sharded Distributed Ledgers. https://arxiv.org/abs/1901.11218.

[6] Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. 2013. SoK: The Evolution of Sybil Defense via Social Networks. In *IEEE Symposium on Security and Privacy*.

[7] Yair Amir, Brian A. Coan, Jonathan Kirsch, and John Lane. 2011. Prime: Byzantine Replication under Attack. *IEEE Transactions on Dependable and Secure Computing* 8, 4 (2011), 564–577.

[8] Baruch Awerbuch and Christian Scheideler. 2006. Robust Random Number Generation for Peer-to-peer Systems. In *International Conference on Principles of Distributed Systems*. 275–289.

[9] Sarah Azouvi, Patrick McCorry, and Sarah Meiklejohn. 2018. Betting on Blockchain Consensus with Fantomette. *CoRR* abs/1805.06786 (2018).

[10] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. 2012. On Bitcoin and Red Balloons. In *ACM Conference on Electronic Commerce*. 56–73.

[11] Adam Back. 1997. A Partial Hash Collision Based Postage Scheme. http://www.hashcash.org/papers/announce.txt.

[12] Christian Badertscher, Juan A. Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. 2018. But Why Does It Work? A Rational Protocol Design Treatment of Bitcoin. In *Advances in Cryptology - EUROCRYPT 2018 - Conference on the Theory and Applications of Cryptographic Techniques*. 34–65.

[13] Iddo Bentov, Pavel Hubáček, Tal Moran, and Asaf Nadler. 2017. Tortoise and Hares Consensus: the Meshcash Framework for Incentive-Compatible, Scalable Cryptocurrencies. http://eprint.iacr.org/2017/300.

[14] Ethereum Blog. 2015. Introducing Casper "the Friendly Ghost". https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/.

[15] Joseph Bonneau. 2016. Why Buy When You Can Rent? - Bribery Attacks on Bitcoin-Style Consensus. In *Financial Cryptography Workshops (Lecture Notes in Computer Science)*, Vol. 9604. 19–26.

[16] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. 2015. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *IEEE Symposium on Security and Privacy*.

[17] Lars Brünjes, Aggelos Kiayias, Elias Koutsoupias, and Aikaterini-Panagiota Stouka. 2018. Reward Sharing Schemes for Stake Pools. *arXiv preprint arXiv:1807.11218* (2018).

[18] Vitalik Buterin. 2017. The Meaning of Decentralization. https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

[19] Pino Caballero-Gil, Candelaria Hernández-Goya, and Carlos Bruno-Castañeda. 2007. A Rational Approach to Cryptographic Protocols. *Mathematical and Computer Modelling* 46, 1-2 (2007), 80–87.

[20] Christian Cachin. 2016. Architecture of the Hyperledger Blockchain Fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

[21] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2000. Random Oracles in Constantipole: Practical Asynchronous Byzantine Agreement using Cryptography. In *ACM symposium on Principles of Distributed Computing*. 123–132.

[22] Christian Cachin and Marko Vukolić. 2017. Blockchain Consensus Protocols in the Wild. preprint, arXiv:1707.01873 [cs.DC].

[23] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *Symposium on Operating Systems Design and Implementation*, Vol. 99. 173–186.

[24] Alexander Chepurnoy. 2016. Interactive Proof-of-stake. *CoRR* abs/1601.00275 (2016).

[25] Bram Cohen. 2003. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer systems*, Vol. 6. 68–72.

[26] CoinDesk. 2015. Antbleed: Bitcoins Newest New Controversy Explained. https://www.coindesk.com/antbleed-bitcoins-newest-new-controversy-explained/.

[27] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. 2013. Spanner: Google's Globally Distributed Database. *ACM Transactions on Computing Systems* 31, 3, Article 8 (2013), 22 pages.

[28] Flaviu Cristian, Houtan Aghili, H. Raymond Strong, and Danny Dolev. 1995. Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement. *Information and Computation* 118, 1 (1995), 158–179.

[29] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün. 2016. On Scaling Decentralized Blockchains. In *Workshop on Bitcoin and Blockchain Research, Financial Cryptography*.

[30] Phil Daian, Rafael Pass, and Elaine Shi. 2016. Snow White: Provably Secure Proofs of Stake. Cryptology ePrint Archive, Report 2016/919. http://eprint.iacr.org/2016/919.

[31] George Danezis and Sarah Meiklejohn. 2016. Centrally Banked Cryptocurrencies. In *Network and Distributed System Security*.

[32] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In *EUROCRYPT (2) (Lecture Notes in Computer Science)*, Vol. 10821. Springer, 66–98.

[33] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the Bitcoin network. In *IEEE Conference on Peer-to-Peer Computing*. 1–10.

[34] John R Douceur. 2002. The Sybil Attack. In *Springer International Workshop on Peer-to-Peer Systems*. 251–260.

[35] Sisi Duan, Michael K. Reiter, and Haibin Zhang. 2018. BEAT: Asynchronous BFT Made Practical. In *ACM SIGSAC Conference on Computer and Communications Security*. 2028–2041.

[36] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the Presence of Partial Synchrony. *J. ACM* 35, 2 (1988), 288–323.

[37] Cynthia Dwork and Moni Naor. 1992. Pricing via Processing or Combatting Junk Mail. In *CRYPTO*.

[38] Ittay Eyal. 2015. The Miner's Dilemma. In *IEEE Symposium on Security and Privacy*.

[39] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-NG: A Scalable Blockchain Protocol. In *USENIX Conference on Networked Systems Design and Implementation*. 45–59.

[40] Ittay Eyal and Emin Gun Sirer. 2013. Majority is not Enough: Bitcoin Mining is Vulnerable. In *Financial Cryptography*.

[41] Giulia Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, and Gerui Wang. 2018. Compounding of Wealth in Proof-of-Stake Cryptocurrencies. *arXiv preprint arXiv:1809.07468* (2018).

[42] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM* 32, 2 (1985), 374–382.

[43] Juan Garay, Jonathan Katz, Ueli Maurer, Bjoern Tackmann, and Vassilis Zikas. 2013. Rational Protocol Design: Cryptography Against Incentive-driven Adversaries. Cryptology ePrint Archive, Report 2013/496.

[44] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *Advances in Cryptology-EUROCRYPT 2015*. Springer, 281–310. http://courses.cs.washington.edu/courses/cse454/15wi/papers/bitcoin-765.pdf

[45] Juan A. Garay and Aggelos Kiayias. 2018. SoK: A Consensus Taxonomy in the Blockchain Era. *IACR Cryptology ePrint Archive* 2018 (2018), 754.

[46] Alexis Gauba, Aparna Krishnan, Zubin Koticha, Maaz Uddin, and Mahnush Movahedi. 2018. A MetaAnalysis of

Proposed Alternative Consensus Protocols for Blockchains. https://github.com/Mechanism-Labs/MetaAnalysis-of-Alternative-Consensus-Protocols.
[47] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the Security and Performance of Proof of Work Blockchains. In *ACM SIGSAC Conference on Computer and Communications Security*. 3–16.
[48] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *ACM Symposium on Operating Systems Principles*. 51–68.
[49] Lisa Glendenning, Ivan Beschastnikh, Arvind Krishnamurthy, and Thomas Anderson. 2011. Scalable Consistency in Scatter. In *ACM Symposium on Operating Systems Principles*. 15–28.
[50] Jim Gray and Leslie Lamport. 2006. Consensus on Transaction Commit. *ACM Transactions on Database Systems* 31, 1 (2006), 133–160.
[51] James N Gray. 1978. Notes on Database Operating Systems. In *Springer Operating Systems*. 393–481.
[52] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. 2010. The Next 700 BFT Protocols. In *Proceedings of the 5th European Conference on Computer Systems*. 363–376.
[53] Trond Hønsi. 2017. SpaceMint: A Cryptocurrency Based on Proofs of Space. *IACR Cryptology ePrint Archive* (2017).
[54] Hyperledger. [n.d.]. Sawtooth. https://intelledger.github.io/introduction.html.
[55] Flavio Paiva Junqueira, Benjamin C. Reed, and Marco Serafini. 2011. Zab: High-performance Broadcast for Primary-backup Systems. In *IEEE Dependable Systems and Networks*. 245–256.
[56] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *CRYPTO (1) (Lecture Notes in Computer Science)*, Vol. 10401. Springer, 357–388.
[57] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. 2016. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *USENIX Security Symposium*. 279–296.
[58] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *IEEE Symposium on Security and Privacy*. 583–598.
[59] Abhiram Kothapalli, Andrew Miller, and Nikita Borisov. 2017. SmartCast: An Incentive Compatible Consensus Protocol Using Smart Contracts. In *Workshop on Trusted Smart Contracts, Financial Cryptography*.
[60] Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. 2009. Zyzzyva: Speculative Byzantine fault tolerance. *ACM Transactions on Computing Systems* 27, 4 (2009), 7:1–7:39.
[61] Ramakrishna Kotla and Mike Dahlin. 2004. High Throughput Byzantine Fault Tolerance. In *Conference on Dependable Systems and Networks*.
[62] Jae Kwon. 2014. Tendermint: Consensus without Mining. https://tendermint.com/static/docs/tendermint.pdf.
[63] Leslie Lamport. 1998. The Part-Time Parliament. *ACM Transactions on Computer Systems (TOCS)* 16, 2 (1998), 133–169.
[64] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 382–401.
[65] L. H. Le, C. E. Bezerra, and F. Pedone. 2016. Dynamic Scalable State Machine Replication. In *ACM Symposium on Operating Systems Principles*.
[66] Sergio Demian Lerner. [n.d.]. DECOR+ HOP: A Scalable Blockchain Protocol. https://scalingbitcoin.org/papers/DECOR-HOP.pdf. ([n. d.]).
[67] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A Secure Sharding Protocol For Open Blockchains. In *ACM SIGSAC Conference on Computer and Communications Security*. 17–30.
[68] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. 2015. Demystifying Incentives in the Consensus Computer. In *ACM SIGSAC Conference on Computer and Communications Security*. 706–719.
[69] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. 2017. SmartPool: Practical Decentralized Pooled Mining. In *USENIX Security Symposium*. 1409–1426.
[70] Mohammad Hossein Manshaei, Murtuza Jadliwala, Anindya Maiti, and Mahdi Fooladgar. 2018. A Game-theoretic Analysis of Shard-based Permissionless Blockchains. *arXiv preprint arXiv:1809.07307* (2018).
[71] Petar Maymounkov and David Mazieres. 2002. Kademlia: A Peer-to-Peer Information System based on the XOR Metric. In *Springer International Workshop on Peer-to-Peer Systems*. 53–65.
[72] David Mazieres. 2015. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus. *Stellar Development Foundation* (2015).
[73] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. 2014. Permacoin: Repurposing Bitcoin Work for Data Preservation. In *IEEE Symposium on Security and Privacy*. 475–490.
[74] Andrew Miller, Ahmed Kosba, Jonathan Katz, and Elaine Shi. 2015. Nonoutsourceable Scratch-off Puzzles to Discourage Bitcoin Mining Coalitions. In *ACM SIGSAC Conference on Computer and Communications Security*. 680–691.
[75] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The Honey Badger of BFT Protocols. In *ACM SIGSAC Conference on Computer and Communications Security*. 31–42.
[76] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.
[77] Arvind Narayanan and Jeremy Clark. 2017. Bitcoin's Academic Pedigree. *ACM Queue* 15, 4, Article 20 (Aug. 2017), 30 pages.
[78] Brian M Oki and Barbara H Liskov. 1988. Viewstamped Replication: A New Primary Copy Method to Support Highly-available Distributed Systems. In *ACM Symposium on Principles of Distributed Computing*. 8–17.
[79] Diego Ongaro and John K Ousterhout. 2014. In Search of an Understandable Consensus Algorithm.. In *USENIX Annual Technical Conference*. 305–319.
[80] Rafael Pass and Elaine Shi. 2017. Fruitchains: A Fair Blockchain. In *ACM Symposium on Principles of Distributed Computing*. 315–324.
[81] Joseph Poon and Thaddeus Dryja. 2015. The Bitcoin Lightning Network: Scalable Off-chain Instant Payments. *Technical Report (draft)* (2015).
[82] Giulio Prisco. 2016. Intel develops Sawtooth Lakedistributed ledger technology for the Hyperledger project. *Bitcoin Magazine* (2016).
[83] Team Rocket. 2018. Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies. https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV.
[84] Fred B. Schneider. 1990. Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial. *Comput. Surveys* 22, 4 (1990), 299–319.
[85] Atul Singh, Tathagata Das, Petros Maniatis, Peter Druschel, and Timothy Roscoe. 2008. BFT Protocols Under Fire. In *USENIX Symposium on Networked Systems Design and Implementation*.
[86] Dale Skeen. 1981. Nonblocking Commit Protocols. In *ACM SIGMOD International Conference on Management of Data*. 133–142.
[87] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. 2016. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *IACR Cryptology ePrint Archive* (2016), 1159.
[88] Yonatan Sompolinsky and Aviv Zohar. 2015. Secure High-Rate Transaction Processing in Bitcoin. In *Financial Cryptography and Data Security*, Rainer Böhme and Tatsuaki Okamoto (Eds.). 507–527.
[89] Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Alexei Zamyatin, and Edgar R. Weippl. 2018. Agreement with Satoshi - On the Formalization of Nakamoto Consensus. *IACR Cryptology ePrint Archive* 2018 (2018), 400.
[90] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. 2017. Scalable Bias-resistant Distributed Randomness. In *IEEE Symposium on Security and Privacy*. 444–460.
[91] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. 2016. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *IEEE Symposium on Security and Privacy*. 526–545.
[92] Carmela Troncoso, Marios Isaakidis, George Danezis, and Harry Halpin. 2017. Systematizing Decentralization and Privacy: Lessons from 15 Years of Research and Deployments. *PoPETs* 2017, 4 (2017), 404–426.

[93] Marko Vukolić. 2015. The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. In *International Workshop on Open Problems in Network Security*. 112–125.

[94] Marko Vukolić. 2017. Rethinking Permissioned Blockchains. In *ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. 3–7.

[95] Benjamin Wester, James A. Cowling, Edmund B. Nightingale, Peter M. Chen, Jason Flinn, and Barbara Liskov. 2009. Tolerating Latency in Replicated State Machines Through Client Speculation. In *USENIX Symposium on Networked Systems Design and Implementation*. 245–260.

[96] Jian Yin, Jean-Philippe Martin, Arun Venkataramani, Lorenzo Alvisi, and Mike Dahlin. 2003. Separating Agreement from Execution for Byzantine Fault Tolerant Services. In *ACM Symposium on Operating Systems Principles*. 253–267.

[97] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling Blockchain via Full Sharding. In *ACM SIGSAC Conference on Computer and Communications Security*. 931–948.

[98] Fan Zhang, Ittay Eyal, Robert Escriva, Ari Juels, and Robbert Van Renesse. 2017. REM: Resource-Efficient Mining for Blockchains. In *USENIX Security Symposium*. 1427–1444.

[99] Aviv Zohar. [n.d.]. Securing and Scaling Cryptocurrencies. In *Joint Conference on Artificial Intelligence*.