SoK: Consensus in the Age of Blockchains

Shehar Bano[1], Alberto Sonnino[1], Mustafa Al-Bassam[1], Sarah Azouvi[1], Patrick McCorry[1], Sarah Meiklejohn[1], and George Danezis[12]

[1]University College London, United Kingdom
[2]The Alan Turing Institute

***Abstract*—The blockchain initially gained traction in 2008 as the technology underlying Bitcoin [104], but now has been employed in a diverse range of applications and created a global market worth over $150B as of 2017. What distinguishes blockchains from traditional distributed databases is the ability to operate in a decentralized setting without relying on a trusted third party. As such their core technical component is *consensus*: how to reach agreement among a group of nodes. This has been extensively studied already in the distributed systems community for closed systems, but its application to open blockchains has revitalized the field and led to a plethora of new designs.**

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 systematic and comprehensive study of blockchain consensus protocols. After first discussing key themes in classical consensus protocols, we describe: (*i*) protocols based on proof-of-work (PoW), (*ii*) proof-of-X (PoX) protocols that replace PoW with more energy-efficient alternatives, and (*iii*) hybrid protocols that are compositions or variations of classical consensus protocols. We develop a framework to evaluate their performance, security and design properties, and use it to systematize key themes in the protocol categories described above. This evaluation leads us to identify research gaps and challenges for the community to consider in future research endeavours.

I. Introduction

Blockchains—the technology at the foundation of Bitcoin and other cryptocurrencies—have been hailed as a major disruptive innovation with the potential to transform most industries. The total global market capital of blockchain-based tokens and cryptocurrencies has reached over $150B as of 2017, and is expected to grow further [112]. While the projected capabilities and value of the blockchain might seem overly optimistic, its key properties of integrity, resilience, and transparency make it an attractive option for a number of applications. The blockchain is a decentralized, replicated, immutable and tamper-evident log: data on the blockchain cannot be deleted, and anyone can read data from the blockchain and verify its correctness. An important implication of this architecture is *disintermediation*: multiple untrusted or semi-trusted parties can *directly* and *transparently* interact with each other without the presence of a trusted intermediary. This makes blockchains immediately relevant to banks and financial institutions which incur huge middleman costs in settlements and other back office operations. A number of big players are actively exploring the feasibility of blockchains, including the Bank of England [22], the Bank of America [46] and the IMF [63]. However, blockchains are not just restricted to the financial industry; the list of use cases is long [79], ranging from voting [5] and government and public records [? ?], to the sharing economy [3, 7, 8] and social media [1, 11].

We are at a crucial point in the evolution of blockchains. The major hurdle in the widespread adoption of blockchains is their performance and scalability—while improvements have been made, they are nowhere near as ubiquitous as their traditional counterparts. 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 specifies how to get multiple nodes to agree on a value—that is, if a data item should be added to the blockchain. Two key properties of a consensus protocol are: (*i*) requests from correct clients are eventually processed (*liveness*), and (*ii*) if an honest node accepts (or rejects) a value then all other honest nodes make the same decision (*safety/consistency*). Consensus is not a new problem: the distributed systems community has extensively studied it for decades, and developed robust and practical protocols that can tolerate faulty and malicious nodes [43, 88]. However, these protocols were designed for closed groups.

Bitcoin's fundamental innovation was to enable consensus among an open, decentralized group of nodes. This was achieved via a leader election based on proof-of-work (PoW): all nodes attempt to find the solution to a hash puzzle and the node that wins adds the next block to the blockchain. Due to its probabilistic leader election process combined with performance fluctuations in decentralized networks, Bitcoin offers only weak consistency: different nodes might end up having different views of the blockchain leading to *forks*. Additionally, Bitcoin suffers from poor performance which cannot be remedied without fundamental redesign [48] and its PoW consumes a huge amount of energy [129]. This has led to a plethora of proposals for new consensus protocols [23]. Some replace Bitcoin's PoW with more energy-efficient alternatives [101], while others modify the original design of Bitcoin for better performance [60]. To achieve strong consistency and similar performance as mainstream payment processing systems like Visa and PayPal, a number of recent proposals seek to repurpose classical consensus protocols for use in decentralized blockchains [131].

To date there has been no systematic and comprehensive study of blockchain consensus protocols (though there exist a few short surveys based on selected systems which we discuss in Section II-B). This incurs two major challenges. First, a comprehensive survey of blockchains would doubtless include a discussion of classical consensus protocols. However, the literature is vast and complex, which makes it hard to be tailored to blockchains. Second, conducting a survey of consensus protocols in blockchains has its own difficulties. Though young, the field is characterised by high-volume, fast-paced work. Since 2014, on average about 250 papers per year have appeared on the topic of blockchains. A reasonable approach is to only consider work published in reputable venues, but here the bulk of the work is published in non peer-reviewed venues and as white papers for industrial platforms.

We fill this gap by making three contributions. First, we conduct a comprehensive survey mapping 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 IV), and then shift focus to PoW approaches popularized by Bitcoin (Section V). Section VI 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 VII and VIII). Our second contribution is a common evaluation framework to visualize the capabilities of blockchain consensus protocols (Table I). Instead of considering individual protocols which would be clearly infeasible, we map out the landscape by extracting and evaluating high-level design themes in blockchain consensus protocols. Finally, we present a discussion of open research challenges and potential directions in the design of future blockchain consensus protocols (Section IX).

II. Background and Related Work

A. Background

We describe key concepts in blockchains. For details, we refer the readers to the excellent work by Bonneau *et al.* [32]. The *blockchain* is a decentralized, replicated, resilient and transparent data store that allows anyone to read data and verify its correctness. In *permissioned* blockchains, all the node identities are known (trsuted or semi-trusted), and are controlled by a single entity or federation. *Permissionless* blockchains are fully decentralized and anyone can run a node and join the network. Data is stored on the blockchain as *blocks*. The blockchain is typically implemented as a linked list in which pointers to previous blocks have been replaced with the cryptographic hash of the previous block. The hash serves as the id of the previous block, and also verifies its integrity. This pattern is repeated in each block, resulting in a *hash chain* in which each block implicitly verifies integrity of the entire chain before it, and tampering with previous data is detectable. It is also possible to store the blockchain as a tree-like structure called the *hash tree* or the *Merkle tree* [133]. A *transaction* specifies some transformation on the state of the blockchain. 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. Nodes in the network participate in a collaborative protocol (*consensus*) to agree on whether the block should be added to the blockchain. In probabilistic consensus protocols like Bitcoin's PoW, 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. Consensus protocols might have a *leader* node that coordinates with other nodes to reach consensus, and for appending a final, committed value to the blockchain. 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. Bitcoin transactions can include reference to well-known deterministic scripts that operate on the transaction inputs and produce some outputs. To make the blockchain a general-purpose platform, scripts are being replaced with *smart contracts*. A smart contract is self-executing code that enforces a digital contract.

B. Related Work

We believe that our study represents the most comprehensive systematic investigation of consensus protocols in blockchains to date. Below we list work that illuminates different subsets of this space and supports our study.

1) Surveys and Systematization: Bonneau *et al.* [32] present a comprehensive systematization of Bitcoin and other cryptocurrencies. Narayanan and Clark [105] trace the academic pedigree of Bitcoin technical components. Zohar [139] provides an overview of scalability and security issues in cryptocurrencies especially Bitcoin, emphasizing the role of incentivization in PoW blockchains to enforce security. Cachin and Vukolić [40] discuss key concepts in classical consensus and describe a selection of permissioned blockchains. Vukolić [131] emphasizes the weak consistency of PoW systems. He advocates the shift to classical BFT protocols that offer strong consistency, but notes challenges in their scalability which are discussed in a previous paper by the same author [130]. Both these papers focus on permissioned blockchains. Yli-Huumo *et al.* [136] conduct a survey (based on 41 papers) of topic trends in blockchain research and find that 80% of the papers are on Bitcoin and the remaining 20% are dominated by security and privacy in blockchains. They highlight concrete evaluation criteria and scalability as neglected areas of research.

2) Evaluation: Blockbench [55] is a framework for evaluating the security and performance of private blockchains. Their evaluation reveals that due to design gaps, popular blockchains lag far behind traditional database systems when processing traditional data processing workloads. Their recommendations include systematic benchmarking, improved usability, and revitalizing classical database design principles such as modularity, exploiting hardware primitives, sharding, and support for declarative languages. Gervais *et al.* [67] present a quantitative framework to evaluate the security and performance of PoW blockchains. They focus on optimal adversarial strategies for double-spending and selfish mining, while accounting for network propagation, different block sizes, block generation intervals, information propagation mechanism, and the impact of eclipse attacks. Croman *et* *al.* [48] present metrics to evaluate the resource costs and performance of Bitcoin with a focus on scalability. They show that even with reparametrization, Bitcoin can only achieve a maximum throughput of 27 tx/s with a latency of 12 seconds.

III. Systematization Methodology

Capturing a longitudinal *and* representative view of a topic as rich as consensus is challenging. We describe our methodology for compiling the literature on which this work is based, and describe the review process and the evaluation framework. We consider consensus in classical systems and blockchains separately because there is a significant difference in the maturity of these two fields.

A. Classical consensus

The area of classical consensus is well-established and spans decades. Our goal is to present sufficient background on classical consensus to contextualize its subsequent application to blockchains. The surveyed literature comprised of well-known seminal works in the area, based on their influence and subsequent citations. It was also supported by Cachin *et al.*'s comprehensive book on this topic [37] and Schneider's classical survey on state machine replication [120].

B. Consensus in Blockchains

Consensus in blockchains is more involved because it is a high-volume, high-churn evolving area of research.

1) Compiling and Reviewing Survey Material: We used a combination of sources to compile our survey material because the bulk of the work in this area originates in non-academic venues, and the usual metrics such as impact (the number of citations) cannot be employed in this young field. We first compiled a seed-list of literature to survey based on a creditable, actively maintained bibliographic repository on blockchain research by Christian Decker from ETH Zürich.[1] We augmented this list with work that cited peer-reviewed papers in it, and with other relevant papers of which we were aware. After further refinement, we ended up with the papers listed in Appendix A, which are categorized by PoW, PoX, or hybrid. For each category, a subset of representative papers were selected, prioritizing papers published in academic venues, and papers that significantly advance the field.

2) Evaluation Framework: Our evaluation framework describes systems along three broad themes: security, performance, and design aspects. In terms of security, we consider three properties: *consistency* (*i.e.,* whether or not the system will reach consensus on a proposed value), *transaction censorship resistance* (*i.e.,* the system's resilience to malicious nodes suppressing transactions), and *DoS resistance* (*i.e.,* the system's resilience to DoS attacks against nodes involved in consensus). In terms of performance, we consider *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). In terms of design, some properties are relevant only to their associated categories, so we defer explanation of them to sections where they are relevant. A complete glossary is included in Appendix B.

We use Table I as a common reference throughout our discussion on PoW, PoX, and hybrid consensus, focusing on parts of the table relevant to each category. Unless explicitly stated, we always assume partial synchrony (*i.e.,* messages might be delayed in the network but eventually arrive within some bound). The wide view captured by this table aids in visualizing evaluation of the field.

IV. Classical Consensus

Safety in distributed systems has been studied since the 1970s, alongside the rise of distributed databases and transactions. Jim Gray, in 1978, proposed the two-phase commit protocol [72], allowing a transaction manager to atomically commit a transaction, depending on different resources held by a distributed set of 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 [123]—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.

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 databases. The primitive is closely associated with the *state machine replication* paradigm [120] 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.

According to the taxonomy by Dwork *et al.* [57], networks may be *syncronous* or *asynchronous*, or offer *eventual synchrony*. In a *synchronous* network the delays messages may suffer can be 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 partially synchronous, or eventually synchronous networks, assume that the network at some stage will eventually be synchronous despite potentially a long period of asynchrony. Fischer *et al.* [62] show that deterministic protocols for consensus are impossible in the

[1]Other similar repositories by Aljosha Judmayer [80] and Brett Scott [122] (discontinued after 2016) are also note-worthy; Decker's collection overlaps with both of these.

fully asynchronous case, and have known solutions in the synchronous case (also known as the "Byzantines General's Problem"). The impossibility theorem is also not taking into account computational bounds on the work nodes may do—something that is exploited by both Nakamoto consensus, as well as other cryptographic solutions [38] to overcome it.

Different failure models have also been considered in the 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. In the network security literature those 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 [88], viewstamped replication [109] and the more modern Raft [110] or Zab [81] cannot be used, unmodified, in adversarial settings.

In terms of the properties expected from a consensus protocol, we consider *liveness* and *safety* as enumerated by Cachin *et al.* [40]. 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.

Consensus refers to 'agreement' by all nodes, not 'choice': consensus protocols are not voting protocols ensuring that all or a majority of nodes agree to the total order, or any single message—the order may be arbitrary or even controlled by an adversary. 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.

An exemplary protocol implementing consensus in the the byzantine and partially synchronous setting is Practical Byzantine Fault Tolerance (PBFT) by Castro and Liskov [42]. The protocol operates in a sequence of views, each coordinated by a leader—a pattern also used in Paxos [88]. Within each view the leader orders messages, and propagates them through a three step reliable broadcast to the replicas. Replicas monitor the leader for safety, as well as for liveness, and can propose a view change in case the leader is unavailable or malicious. Safety is guaranteed within the asynchronous network setting; liveness on the other hand is only guaranteed within a partially synchronous setting, since replicas rely on time-outs to detect a faulty leader. The key complexity of PBFT lies in the view-change sub protocol, that needs to ensure agreement on the new leader and view, as well as guarantee safety of messages agreed in previous views. The basic protocol requires $\mathcal{O}(n^2)$ messages for n replicas to achieve consensus, where n is the number of nodes. The properties of the protocol are guaranteed if $n = 3f + 1$, where f is the number of byzantine nodes.

The issue of storage efficiency, a topic of great relevance to blockchain protocols, is discussed in PBFT: a naive implementation of state machine replication, based on consensus, would store the full sequence of actions. The proposed solution relies on replicas agreeing *checkpoint* actions. Those checkpoints are co-signed by all replicas, and allow them to only store the current state of the system, and discard the past sequence that led to the checkpoint state.

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 [71]. 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.

V. Proof-of-Work Consensus

In 2008, Bitcoin [104] was published by a pseudonymous author Satoshi Nakamoto; it has since gone on to become one of the most successful cryptocurrencies of modern times. The key innovation of Bitcoin is its use of proof-of-work (PoW) to achieve consensus—also called Nakamoto consensus after its originator—in a fully decentralized, permissionless network.

A. Nakamoto consensus

While the technical components of Bitcoin originate in previous academic literature, their composition in Bitcoin to achieve consensus is novel. The idea of proof-of-work 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 [58].

PoW was independently proposed in 1997 for Hashcash by Back, another system for fighting spam [21]. In Hashcash, the computational puzzle is finding a SHA-1 hash of a header including the email recipient's address and current date, such that the hash contains at least 20 bits of leading zeros. As the hashing algorithm is pre-image resistant, the puzzle can be solved only by including random nonces in the header until the resulting hash meets the leading zeros requirement. These guesses require a significant amount of computational work, so a valid hash is considered to be a PoW.

Nakamoto consensus is derived from Hashcash [21]. It replaces Hashcash's SHA-1 hashing with two successive SHA-2 hashes, and requires valid hashes to have a value below a target integer value 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

TABLE I: Evaluation of blockchain consensus protocols. Notation for binary values: ✓ has property, ✗ does not have property. Notation for non-binary values: ● has property, ◒ partially has property, ○ does not have property. Notation for meta-information: – the property does not apply to the given category, ? the value could not be extracted, ! the value is missing. The rows correspond to selected systems in each protocol category; a full list of the corresponding citations is provided in Appendix A. A list of terms is included in Appendix B. In the *Msg.* column (message complexity), n refers to the number of participants, and c is the size of the committee.

	Systems	**code avail.**	**Committee Formation (Resources)**	**Strong consistency**	**Single Committee**				**Multiple Committee**			**Safety**			**Performances**			
					Committee Configura-tion	**Inter-Committee Consensus**			**Intra-Committee Configura-tion**	**Intra-committee Consensus**		**Transaction Censorship Resistance**	**DoS Resistance**	**Adversary Model**	**Throughput**	**Scalable**	**Latency**	**Exp. Setup**
						Incentives (Join,Participate)	**Leader**	**Msg.**		**Mediated**	**Incentives**							
hybrid	**ByzCoin [83]**	✓	PoW	✓	Rolling (single)	✓✗	Internal	$O(n)$	-	-	-	✓	◒	33%	1000 tx/s [1]	✗	10–20s [1]	Real
	Solidus [15]	✗	PoW	✓	Rolling (single)	✓✓	External	$O(n^2)$	-	-	-	✗	◒	33%	-	-	-	-
	Algorand [69]	✗	Lottery	✓	Full swap	✗✗	Internal	$O(n^2)$	-	-	-	✗	●	33%	90 tx/h [2]	✗	40s [2]	Real
	Hyperledger [132]	✓	Permissioned	✓	Static	-	Flexible	Flexible	-	-	-	✓	●	33%	110k tx/s [3]	✗	<1s [3]	Real
	RSCoin [50]	✓	Permissioned	✓	Static	-	Internal	$O(n)$	✗	Client	✗	✓	●	33%	2k tx/s [4]	✓	<1s [4]	Real
	Elastico [94]	✗	PoW	✓	Full swap	✓✗	Internal	$O(n^2)$	Dynamic (Random)	!	!	✗	●	33%	16 blocks in 110s [5]	✓	110s for 16 blocks [5]	Real
	Omniledger [84]	✗	PoW/PoX	✓	Rolling (subset)	✓✗	Internal	$O(n)$	Dynamic (Random)	Client	✗	✓	●	33%	≈10k tx/s [6]	✓	≈1s [6]	Real
	Chainspace [17]	✓	Flexible	✓	Flexible	✗✗	Internal	$O(n^2)$	✗	✗	✗	✓	◒	33%	350 tx/s [7]	✓	<1s [7]	Real
proof-of-X	**Ouroboros [82]**	✗	Lottery	✗	Full swap	✓✓	Internal	$O(nc)$	–	–	–	✗	◒	50%	257.6 tx/s [9]	✗	20s	Simulation
	Praos [51]	✗	Stake	✗	Rolling (subset)	✓✓	Internal	O(1)	–	–	–	✗	●	50%	–	–	–	–
	Snow-white [49]	✗	Stake	✗	Full swap	✓✓	Internal	O(1)	–	–	–	✗	◒	50%	100-150 tx/s [9]	✓	?	Simulation
	PermaCoin [101]	✓	PoW/PoR[11]	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	50%	–	✗	–	–
	SpaceMint [76]	✓	PoS	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	50%	?	✗	600s	Simulation
	Intel PoET [78]	✓	TH[12]	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	TH[12]	1000 tx/s [10]	✓	–	Real
	REM [137]	✗	TH[12]	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	TH[12]	!	✓	–	Real
proof-of-work	**Bitcoin [104]**	✓	PoW	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	50%	7 tx/s	✗	600s	Real
	Bitcoin-NG [60]	✗	PoW	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	◒	50%	7 tx/s	✗	<1s	Simulation
	GHOST [125]	✗	PoW	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	50%	–	✗	–	–
	DECOR+HOP [91]	✗	PoW	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	50%	30 tx/s [8]	✗	60s	Simulation
	Spectre [124]	✗	PoW	✗	Rolling (single)	✗✓	Internal	$O(1)$	–	–	–	✓	●	50%	–	✗	–	–

[1] 144 nodes/committee.
[2] 50k nodes/committee.
[3] 4 nodes/committee (corresponding to BFTSmart [13]).
[4] 3 nodes/committee. 10 committees.
[5] 100 nodes/committee. 16 committees.
[6] 72 nodes/committee (12.5% adversary). 25 committees.
[7] 4 nodes/committee. 15 committees.
[8] 1 minute average interval; 1 block = 1 MB.
[9] 40 nodes.
[10] As reported in a blog post [6].
[11] proof-of-retrievability.
[12] Trusted Hardware.

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*).

B. Forks

Forks in the blockchain may occur if two miners find two different blocks that build on the same previous block. This is resolved by PoW consensus, which orders transactions and makes double-spending expensive. In the original paper, forks are resolved in the consensus rules by accepting the 'longest chain, which has the greatest proof-of-work 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 proof-of-work than a longer chain.

To double-spend assets on a PoW blockchain, 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*) can outpower the remaining computational power in generating a chain with the most accumulated work. The *security threshold* of the network is the percentage of computing power required to conduct a 51% attack. Decker and Wattenhofer showed that due to the delays in blocks propagation in the Bitcoin network, increasing the block size and decreasing the inter-block interval increases the chance of forks occurring [53], as delayed miners may waste effort in attempting to mine on top of blocks that are no longer the latest ones. As a result, the network becomes more susceptible to 51% attacks from a miner that does not suffer from delays.

C. Scaling Bitcoin

Bitcoin currently has a hardcoded blocksize limit of 1MB per block, and a 10 minute block frequency target. Gervais *et al.* [67] showed that Bitcoin's block frequency can be reduced to 1 minute per block without reducing the security threshold of the existing network, modelling the bandwidth distribution of the network around real-world broadband data.

Forks might still occur in PoW blockchains despite countermeasures to avoid them. 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 [125] 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.

The challenge of scaling PoW blockchains is that the more one increases throughput (block size) or the more one decreases latency (block frequency), the lower the resilience of the network to 51% attacks. A common theme in enhancing PoW consensus is to improve performance while maintaining the security threshold of the network without requiring nodes and miners to upgrade their network connections.

Bitcoin-NG [60] 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 latency to be limited only by the network's propagation delay, and bandwidth to be limited only by the processing capacity of the nodes. Another approach for improving performance, used by Spectre [124], is to allows miners to mine blocks concurrently by replacing the 'linear' blockchain structure with a block-DAG. Off-chain approaches to improve Bitcoin scalability such as the Lightning Network [115] 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.

D. Mining Centralization

To reduce the variance of miners' rewards, miners often aggregate resources and share rewards among themselves *via* pooled mining protocols. However, mining pools undermine decentralization and are vulnerable to transaction censorship by a malicious pool manager (needed to map transactions to blocks) [96]. To mitigate such attacks, 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 [96, 102]. SmartPool [96] 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.* [102] 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 [91] enforces fairness between miners by allowing them to share the profit when competing blocks are generated. Such efforts aim to avoid centralization by giving miners the same reward and guaranteeing low variance as if they were mining with centralized pools. Moreover, DECOR+HOP improves Bitcoin performance by using 'header-first propagation', where the block header is sent first, and nodes attempt to reconstruct the full block from transactions that they have already heard about. If there are missing transactions, the node fetches them from its peers.

E. Incentives

The security of Nakamoto consensus relies on economically incentivising miners to validate and mine blocks, by rewarding them with new coins. However, previous work has shown that Nakamoto consensus is not completely incentive compatible [31, 59, 95].

Aside from incentives, protocol-level attacks exist that lower the security threshold of Bitcoin below 51%. Selfish mining [61] allows colluding miners to generate more valid blocks than their computing power would normally allow them to if they were following the standard protocol. In selfish mining, colluding miners withhold blocks that they have found, which allows them to maintain a lead over the rest of the network, who may waste their computational power on stale blocks. When the network is about to catch up with the colluding miners, the colluding miners release a portion of their withheld blocks to the 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.

Systems like Fruitchain [113] aim to mitigate 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 an new type of block, called 'fruits'. 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 withhold blocks because eventually, an honest block will be created and will include back all the dropped fruits.

VI. 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 V-D). 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.

A. 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. A number of recent systems have provably secure proof-of-stake protocols [49, 51, 82]. A common theme in these systems is to randomly elect a leader from among the stakeholders, which then appends a block to the blockchain. Leader election may be public, that is the outcome is visible to all the participants [49, 82]. 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 [51]. Private leader election is resilient to DoS attacks because candidates privately check if they are elected before revealing it publicly in their blocks, at which point it is too late to DoS them. A malicious leader can censor transactions during its epoch. But as leaders are re-elected sufficiently often, a subsequent leader will add the censored transaction to the blockchain (albeit some delay).

In Ouroboros [82], 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. Ouroboros distributes rewards among all the participants regardless of whether or not they win the election.

In Ouroboros Praos [51] and Snow-White [49] participants independently determine if they have been elected. 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. Snow-White employs the incentive structure of Fruitchain [113]: payouts are distributed equally among fruits (Section V-E). 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 Praos inherits the incentive structure of Ouroboros.

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 [4, 9], but their designs have not been rigorously studied and they are not very popular. Ethereum Foundation has been considering using proof-of-stake for some time, but their work is still in progress [30].

1) Attacks and Mitigation: PoX results in three new attacks compared to Nakamoto consensus [44]. The 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 [49]. Another mitigation is to remove forks, at the cost of a bigger overhead [69]. 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 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.

2) Alternatives: Bonneau *et al.* [32] describe informal (and unpublished) consensus protocols based on proof-of-stake that have been proposed in the cryptocurrency community. Broadly, these system require miners to hold or prove the ownership of coins. We list three variations of this theme, though we note that this area has not seen significant advances.

- *Proof-of-deposit:* Miners 'lock' a certain amount of coins, which they cannot spend for the duration of their mining. One such system is Tendermint [87], where a miner's voting power is proportional to the amount of coins they have locked.
- *Proof-of-burn:* Miners prove that they have destroyed a quantity of coins, for example by sending them to a verifiably unspendable address [10]. Slimcode [111] implemented this approach in 2014 but has recently been discontinued.
- *Proof-of-coin-age:* Miners show possession of a quantity of coins, where the quantity of coins is weighted by their *coin-age*—the time since the coins were last moved. Peercoin [9] adapts this approach.

B. Proof-of-Capacity

In proof-of-capacity, participants vote on new blocks weighted by their capacity to allocate a non-trivial amount of disk space. PermaCoin [101] 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 provides censorship-resistant file storage, the file is fully recoverable from the participants in the event of a dealer failure or shutdown. SpaceMint [76] employs a consensus protocol based on a non-interactive variant of proof-of-capacity (called 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 inherit Bitcoin's incentivization mechanism, as well as its resilience against censorship and DoS.

1) 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.

C. Proof-of-Elapsed-Time

Using the trusted enclave in Intel SGX, it is possible to replace computational work with proof-of-elapsed-time [78]. 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) [137] 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.

1) 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.

VII. Hybrid Consensus: Single Committee

A single consensus node 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.

A. Committee Formation

Committee formation refers to the criteria used to allow nodes to join a committee. This is an important aspect of decentralized, permissionless systems to thwart sybil attacks.

1) Permissioned: Permissioned blockchains operate in a trusted environment where nodes are granted committee membership based on organizational policy. Hyperledger [36] is one such system that supports smart contracts. There is a hierarchy of trust, with some nodes being fully trusted while others only partially trusted. This 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. In Hyperledger, clients first submit their transactions to the endorsers who execute the smart contract. A transaction is only submitted to a subset of endorsers according to the policy of the respective smart contract. As different smart contracts can designate different endorsers, execution can take place in parallel. Clients collect matching signed results and smart contract state updates from sufficient number of endorsers, and submit these to the ordering nodes which append it to the blockchain using a consensus protocol.

2) Proof-of-work: In these systems, nodes are allowed to join the committee based on PoW. In ByzCoin, 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. Solidus and Omniledger have a similar model for committee formation. Omniledger also supports proof-of-stake to allocate committee membership based on directly invested stake instead of power-wasteful work. A public randomness or cryptographic sortition protocol is run within the current committee to select the next committee from the current stakeholder distribution defined in the ledger.

3) Lottery: Candidates are promoted to committee membership based on the outcome of a lottery. In Algorand, all candidates have a public key, and get chosen to become a committee member using cryptographic sortition. This involves the candidates running a verifiable random function and seeing if the output is below a certain value.

B. Committee Configuration

The way a committee is configured has safety and performance implications. Permissioned systems usually assume static committee members, but sybil resistance in a permissionless and decentralized setting requires dynamic membership.

1) Static: In static setting, the committee members are not periodically changed. This is the typical configuration in permissioned systems like Hyperledger and RSCoin where committee members have known, trusted identities and the threat model does not include sybil attacks.

2) Rolling (Single): The committee is updated in a sliding window fashion: new miner(s) are added to the current committee and the oldest members are ejected. In ByzCoin, each miner has voting power proportional to the number of mining blocks it has 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 current consensus group and receives a share in the current window which moves one step forwards (ejecting the oldest miner).

An important aspect of reconfiguration is *wedging*, that is to stop the old committee from approving more transactions without losing any transactions it is processing at the time of reconfiguration. Solidus updates its committee similarly to ByzCoin, but 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. Another issue is how to resolve *leader contention* when two miners simultaneously solve a PoW puzzle. Solidus uses a Paxos-style leader election where a higher ranked leader can interrupt a lower ranked leader. Ranks are derived from leaders' PoW solutions and supplementary epoch numbers. To ensure safety, the new leader must propose a value that has been (or may be) committed.

A reconfigurable committee needs some mechanism to track committee membership. In Peercensus [52], a new member is allowed to join the committee following a collective decision which involves validating that the member is reachable over the network. Committee members use a failure detector (e.g., by sending regular ping messages) to detect when a member has left the committee. If a member finds another to be unreachable, it can propose 'leave' for the absent member and the committee membership is updated after a collective decision is made by the committee. A limitation of this approach is that malicious members can slow down or stall the system 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.

3) Full: Lottery-based systems like Algorand and Snow-White select the committee members for each epoch using randomness generated based on previous blocks.

4) Rolling (Multiple): 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. This also has the benefit that the system is operational during reconfiguration as the operational members can continue to process transactions while a fraction of the committee is being reconfigured and bootstrapped.

C. Consensus Protocol

Most committee-based systems use classical BFT consensus protocols such as PBFT. In this section we focus on modifications to classical BFT protocols or their novel compositions to tailor them for use in blockchains.

In Solidus, the leader is external to the committee and can propose transactions and PoW to nominate itself as a committee member only once to the committee. If the committee agrees, they approve the proposed transactions and allow the miner to join the committee in the next round. The proposal, that has now become a decision, also serves as the next puzzle and is propagated to all miners. This approach is motivated by 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.

ByzCoin organizes the consensus committee into a communication tree where the most recent miner (the leader) is at the root. The leader runs PBFT [43] to get all members to agree on the next block. However, it replaces PBFT's $O(n^2)$ MAC-authenticated all-to-all communication with a primitive called scalable collective signing (CoSi) that reduces messaging complexity to $O(n)$. The outcome of running two rounds of PBFT with CoSi is a fixed 64 byte collective signature that proves that at least two-thirds of the committee members witnessed and attested the block. A node in the network can verify in $O(1)$ time that a block has been validated.

A malicious committee leader can potentially censor transactions by not proposing them; this does not compromise safety, but it can negatively affect fairness. Omniledger deals with this issue by allowing non-leader committee members to propose a transaction if they suspect that it has been censored by the leader (since they can 'hear' all messages because of the gossip protocol for information dissemination). In Chainspace and ByzCoin, transaction-censorship triggers leader re-election (or view change). Elastico does not discuss censorship by committee leader. Classic BFT protocols often rely on a timeout to detect censorship or unreliability by a leader, and thus liveness and censorship resistance rest on a partial synchronous network assumption.

In the context of BFT-based committees, DoS means that all the honest members in the committee are taken offline. This is challenging, and made further difficult by how frequently and what fraction of committee membership changes (epoch, dynamism). ByzCoin has medium DoS resistance because the committee configuration is rolling (single). Elastico has high resistance within single committees because full committees are reconfigured (full swap), but in the absence of an intra-shard consensus mechanism, an adversary can flood the system with transactions that touch multiple committees causing a deadlock. Omniledger further enhances ByzCoin's efficient BFT protocol with $O(n)$ messaging complexity by using a more robust group communication pattern. This addresses an issue in the original protocol where node failures cause ByzCoin to fall back on a more robust all-to-all communication pattern that significantly degrades system performance. Chainspace does not include details on intra-committee configuration, hence it only provides medium level of resistance against DoS attacks.

Systems based on proof-of-stake consensus (*e.g.,* Ouroboros Praos and Algorand) achieve DoS protection by privately electing committees. This ensures that participants cannot learn whether another participant is a committee member and only learns this when the newly elected member announces this. In Algorand, users check for themselves whether or not they should play a role in the committee for the next round by seeing if, on input a seed known only to the user, the output of a verifiable random function is less than a certain value; this ensures that only they know what roles (if any) they should play in the committee. Once the roles are fixed, users then participate according to their role in the *BA*$\star$ consensus protocol, which is a variant of PBFT that allows the set of participating servers to rotate. As participants start playing their roles, they can include information in their messages that allows other participants to check that they are in fact eligible.

Hyperledger uses *pluggable and modular* consensus in which the consensus protocol can be specified by the smart contract policy. For example, Hyperledger supports a CFT service based on Apache Kafka [2] and its ZooKeeper unit [77], and more recently a PBFT-variant BFTSmart [29]. Because of their trust assumptions, permissioned systems are resilient to DoS and censorship of transactions by the committee.

D. Incentives

Classical BFT protocols assume two kinds of players: cooperative and byzantine. This assumption works well in centralized settings where nodes are controlled by the same entity or federation. However, decentralized networks that rely on volunteer nodes need to provide incentives for participation.

Most committee-based systems such as ByzCoin use the same incentive model as Bitcoin; however, instead of the most recent miner receiving all reward and fee, it is shared between members of the committee in proportion to their shares. ByzCoin states that to ensure that members remain active after joining the committee, they will also be rewarded for participation (*e.g.,* upon completion of PBFT pre-prepare and commit phases); however, details have not been provided.

In general, consensus protocols assume two kind of players: honest and byzantine. Solidus argues that with no clear incentives, the honest (or altruistic) committee members have nothing to gain from participating in the consensus. To alleviate this, Solidus introduces a third kind of player: a *rational* player that assesses its expected utility in terms of Solidus coins. Solidus argues that equal distribution of rewards between all committee members can lead to a situation where members can suppress reconfiguration to stay on the committee and continue to collect rewards. To address this, Solidus rewards the committee members that are the fastest to endorse reconfiguration creating a competition. Moreover, the external leader gets all the transaction fees, while the mining reward is split between the external leader and the committee members. This ensures that committee members do not have an incentive to delay reconfiguration due to a possible high transaction fee. Solidus includes incentives for information propagation and present a game-theoretic analysis that a miners best strategy is to propagate the PoW puzzle and charge a small fee.

Smart contract platforms require clients to include fees 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. Ethereum clients have to pay 'gas' in proportion to the cost of executing the contract [134].

VIII. Hybrid Consensus: Multiple Committees

While single-committee consensus significantly improves performance over single-node consensus, a major limitation is that it is not scalable: adding more members to the committee decreases throughput. This motivated the design of consensus based on multiple committees. To make the system scalable, transactions are split among multiple committees (shards) which then process these transactions in parallel.

A. Committee Topology

When multiple committees are involved in consensus, an important question is how they will be organized in terms of topology. Chainspace and Omniledger have flat topologies, that is all committees are at the same level. Elastico 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 [50] (a permissioned blockchain), the central bank controls all monetary supply, while 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.

B. Intra-committee Configuration

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. This should be done randomly to stop an adversary from concentrating its presence in one committee and exceeding the byzantine-tolerance threshold. 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 [47, 70]. However, generating good randomness in a distributed way is a known hard problem: current best solutions tolerate up to $1/6$ fraction of byzantine peers, while incurring a high message complexity [19]. Among the more recent solutions, RandHerd [127] provides a scalable, secure multi-party computation protocol that offers unbiasable, decentralized randomness in a byzantine setting.

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. The swap operation is done such that liveness is maintained during reconfiguration events because a subset of committee members continues to be operational.

Elastico operates in epochs: assignment of nodes to committees is valid only for duration of the epoch. At the end of the epoch, nodes compute solution to a puzzle seeded by a random string generated by the final committee and sends the solution to the final committee to be assigned to a committee. As a result, in each epoch a node is paired with different nodes in a committee managing a different set of transactions. The number of committees scales linearly in the amount of computational power available in the system, but the number of nodes within a committee is fixed.

Chainspace has abstracted details of committee reconfiguration and it is up to policy enforced *via* a smart contract to decide how nodes will be allocated to committees. Nodes can be added (and removed) to committees by their members through majority $(2f + 1)$ voting.

C. Intra-committee Consensus

In a multi-committee system, some transactions might involve coordination between multiple committees. Such transactions might require access and manipulation of state that is handled by different committees. The intra-committee consensus ensures that this takes place consistently and atomically across all concerned committees.

Omniledger uses an atomic commit protocol to process transactions across committees. A transaction submitted by a client is processed by the committees that manage its 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.

In RSCoin, communication between committee members takes place indirectly through the client (similar to Omniledger), and it also relies on the client to ensure completion of transactions. A client first gets signed 'clearance' from majority of the mintettes that manage the transaction inputs. Next the client sends the transaction and signed clearance to mintettes corresponding to transaction outputs. The mintettes check validity of the transactions and verify signed evidence from input mintettes that the transaction is not double-spending any inputs. If the checks pass, the mintettes adds the transaction to be included in the next block. The system operates in epochs: at the end of each epoch, mintettes send all cleared transactions to the central bank which collates transactions into blocks that are added to the blockchain.

Client-driven atomic commit protocols like Omniledger and RSCoin are vulnerable to DoS if the client stops participating and the inputs are locked forever. These systems 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. Instead of a client-driven approach, Chainspace runs 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.

IX. Discussion

A. Integrating BFT protocols into blockchains

The renewed interest in BFT protocols, in the context of blockchain, has led to more mature and efficient variants of those protocols, or variants that leverage new assumptions. Here we discuss several open problems that still remain.

1) 'Open' vs 'closed' asynchronous protocols: A number of recent scalable blockchain protocols, such as RSCoin [50], Omnileדger [84] and Chainspace [17], employ traditional byzantine consensus protocols for scalability and sharding. However, those consensus protocols are inherently 'closed', in the sense that replicas need to have authenticated channels between them, long term interactions with each other, and can only tolerate f byzantine nodes. Thus, traditional consensus protocols cannot accommodate open participation of nodes and high churn, and are vulnerable to sybil attacks [56].

Newer BFT protocols, such as Honeybadger [103], even overcome impossibility results, and provide both safety and liveness in a fully asynchronous setting, through a randomized consensus algorithm. While this breakthrough, building upon the earlier work by Cachin *et al.* [38] is of notable theoretical value, it does not resolve the issue of the need for a 'closed' group and therefore those solutions cannot be a drop-in replacement for open Nakamoto consensus. Such randomized BFT protocols have traditionally been more expensive than deterministic ones, both in terms of communication and cryptographic operation costs. Byzantine consensus protocols, besides Nakamoto consensus, in the context of open group participation is still an open research problem.

2) Exploiting advances in hardware and cryptography: The most mature current implementation of BFT is the Java BFTSmart [29] library with message complexity $\mathcal{O}(N^2)$ in the size of the quorum N. However, Byzcoin [83] uses modern signature schemes to optimistically relay all messages through a leader, reducing the common case of BFT consensus to $\mathcal{O}(N)$. The XFT [93] protocol, on the other hand, improves the efficiency of consensus by relaxing the threat model. It considers that byzantine nodes may act arbitrarily, however links between honest nodes are reliable and eventually synchronous. This leads to a simplification of the view change and steady state BFT protocol. Finally, some consensus protocols are now leveraging secure hardware executions environments: 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 [116].

3) Identity management: In BFT consensus protocols, a malicious member can potentially generate spoofed responses on behalf of other members to bias majority in its favour. To counter this attack, BFT committees assume that there exist point-to-point, authenticated channels between all members, which requires some mechanism to track committee members and their keys. Tracking membership and key distribution in dynamic permissionless committees is challenging, and most systems abstract these details.

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 [94]. 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 [84], is to record committee members for each round in a separate 'identity' blockchain—however, its details are not provided. In multi-committee systems, intra-committee interaction further requires each committee to have a collective identity, and some way for the committees to discover each other.

B. Committee-based approaches

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 [47, 70, 90]. The key challenge in these systems is to ensure linearizability by atomically executing operations that span multiple committees. More recent optimizations enable *elastic* SMR so that committees can dynamically merge (scale up) and split (scale out) their state for load-balancing purposes [108]. 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. However, sharding permissionless blockchains with byzantine adversaries is challenging and tackled by only a few recent systems [17, 84, 94]. 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. This is an outstanding issue shared by all multi-committee blockchains. Future research should focus on developing robust mechanisms to detect malicious committees and to recover from them.

Chainspace starts mitigating this issue by making the author of the smart contract responsible to designate the parts of the infrastructure that are trusted to maintain the integrity of its contract; the contract's integrity only depend on their correctness (as well as the correctness of contract sub-calls). Moreover, Chainspace provides an auditing mechanism allowing honest node in honest committees to detect inconsistencies and discover the malicious committee; there are however no systems today providing a recovery mechanism.

Finally, sharded solutions achieve a different notion of verifiability from solutions that rely on a single committee (or are fully decentralized), as it is no longer clear how to define a global set of transactions. In Omniledger and Chainspace, for example, every committee defines its own blockchain, and in RSCoin the separate sets of transactions agreed upon by each shard are combined only through the use of a central entity. We leave it as an interesting research problem to quantify the difference, in terms of public verifiability, between sharded and non-sharded solutions.

2) Bootstrapping committees: The biggest threat to the integrity of a permissionless committee is from an adversary that might create sybil identities and take over the whole committee. As discussed in Section VII-A, prominent approaches include using PoW or PoX to allow nodes to join the committee. A limitation here 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 have been proposed but these suffer from similar issues.

Multi-committee systems raise the additional issue of how to map nodes to committees. One approach is to randomly map nodes to committees [84, 94]. However, this prohibits finer governance. General-purpose platforms like Chainspace might have different policies within committees; for example some committees can be permissioned while others are 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.

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. Systems such as Tor [54] 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. Furthermore, like blockchains, they only handle high-integrity operations—not privacy-sensitive ones—making their audit and also replacement in case of unreliability, easier despite being manual. This is a hopeful example, illustrating that even small closed groups may, through careful selection of participants, provide sufficient protection against coercion.

C. Incentives and governance in consensus protocols

Decentralized networks need to incentivize nodes for active participation in different operations such as consensus [83, 84, 94], information propagation [15, 20, 27], and executing smart contracts [17, 134]. Recently there has been a shift towards incentive-compatible consensus protocols, where incentives are built into the core of the protocol. Solidus argues that cooperative players in the classical BFT protocols are replaced by rational players in the decentralized setting, who are motivated to maximize their utility [15]. To ensure that rational players participate in all phases of the protocol, incentives should be distributed among them such that they can only be claimed after the completion of each phase. This is a new but important observation. This implies that 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).

An important question is: who distributes the incentives? In 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 decentralized networks where members located farther from the leader are at a natural disadvantage.

Broadly, incentivization in PoW blockchains has seen some study [86]: major limitations have been identified [41, 61, 106, 119], and possible solutions have been proposed [138]. Similarly, in the context of protocols where creating a block is cheap, good incentives are crucial to prevent attacks on the system, but have not been carefully analyzed. The investigation of these issues in BFT protocols is likewise far from mature, and non-existent in multi-committee protocols where the incentives need to be extended to intra-committee operations. This area will benefit from combining formal economic and game theoretic analysis with cryptography, such as has already been done in the blockchain community [85, 115]. Techniques such as rational cryptography [35, 64] and the BAR model [16], which considers Byzantine, altruistic, and rational agents, could also be adapted to work here.

Beyond concrete incentivization to participate in the protocol, it is important to consider also what makes protocols attractive to participants in the first place, and what makes them think their investment in a given system will be repaid. These broader types of incentive are both related to the governance of the system, in terms of identifying the entities who define its rules, and the extent to which the protocol is able to evolve. It is ultimately a relatively unstudied question at this point what types of governance structures would provide the strongest incentivization, or the extent to which these structures are taken into account when participants are deciding which protocol to join. This area would benefit from social science-based analysis.

D. Privacy in consensus

By their very nature transparent distributed ledgers pose significant privacy challenges with respect to both the information contained in them, as well as the privacy of transactions and their meta-data. The original Bitcoin announcement promised 'anonymity' as a property of the new system; however, the weak form of pseudonymity offered can be bypassed by tracing attacks [100].

Permissioned systems, such as BigchainDB [99], have the ability to protect privacy by restricting the set of core participants in the consensus to a small vetted set—that are assumed to be trusted both for the integrity (safety) of the systems, its liveness, and can also be trusted for keeping secrets (privacy). However, trusting a set of entities for privacy is of a different nature than trusting them for integrity: if information is replicated any node may violate the property, and such a violation cannot be detected within the system—since it only involves leaking secrets. Furthermore, relying on permissioned ledgers for privacy forces the system to rely on closed groups, for reasons besides efficiency of consensus—making this design choice hard to change.

Protocol-layer techniques for protecting privacy also have implications for scalability. The most established exemplar of this family is Zcash [25], where a transaction contains a succinct non-interactive zero-knowledge proof (zk-SNARK) [73], proving that it is spending an existing unspent coin, but without publicly specifying which one. The cost of constructing such SNARKs and verifying them, as well as their size, affects the efficiency and scalability of protocols. Furthermore, since coins are spent in private, it is impossible to prune past transactions to maintain a smaller Unspent Transaction Output (*utxo*), and checkpointing is ineffective. Thus the state necessary to validate transactions grows indefinitely.

From a systems perspective, such systems expose the minimum amount of information for validation, and to agree on a consensus on the ordering of transactions—while the actual construction and execution of transactions happens off-chain between the parties having visibility into the full secrets. The Chainspace platform applies this privacy pattern to general smart contracts [17]. Others [132] 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. Such architectures can scale at the same rate as the core ordering protocol, but do not provide any universal end-to-end verifiability.

X. Conclusions

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, and evaluated their performance and security properties using a novel framework. In a broader context, this work has highlighted a number of open areas and challenges related to: (*i*) gaps between BFT and blockchains, (*ii*) security vs. performance tradeoffs, (*iii*) incentives, and (*iv*) privacy. 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 security and privacy requirements.

Acknowledgements. George Danezis, Shehar Bano and Alberto Sonnino are 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, Shehar Bano (in part) and Patrick McCorry are supported by EPSRC grant EP/N028104/1.

References

[1] Akasha. https://akasha.world.
[2] Apache kafka. https://kafka.apache.org/.
[3] Arcade City. https://arcade.city.
[4] Blackcoin. https://blackcoin.co/.
[5] Follow My Vote. https://followmyvote.com.
[6] Hyperledgers Sawtooth Lake Aims at a Thousand Transactions per Second. https://www.altoros.com/blog/hyperledgers-sawtooth-lake-aims-at-a-thousand-transactions-per-second/
[7] Lazooz. http://lazooz.org.
[8] LemonWay. https://www.lemonway.com/en/.
[9] Peercoin. https://peercoin.net/.
[10] Proof of burn. https://en.bitcoin.it/wiki/Proof_of_burn.
[11] Steem. https://steem.io.
[12] The Swirlds hashgraph consensus algorithm: fair, fast, Byzantine fault tolerance. Technical Report SWIRLDS-TR-2016-01, Swirlds, Inc., 2016.
[13] hyperledger-bftsmart. https://github.com/jcs47/hyperledger-bftsmart, 2017.
[14] I. Abraham and D. Malkhi. Bvp: Byzantine vertical paxos. 2016.
[15] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. https://arxiv.org/abs/1612.02916, Dec 2016. Accessed: 2017-02-06.
[16] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. Bar fault tolerance for cooperative services. *SIGOPS Oper. Syst. Rev.*, 39(5):45–58, Oct. 2005.
[17] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis. Chainspace: A Sharded Smart Contracts Platform. In *To appear in Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.
[18] ArtForz. Re: Possible way to make a very profitable 50 plus ish attack for pools? https://bitcointalk.org/index.php?topic=43692.msg521772#msg521772, 2011.
[19] B. Awerbuch and C. Scheideler. Robust random number generation for peer-to-peer systems. In *Proceedings of the 10th International Conference on Principles of Distributed Systems*, OPODIS'06, pages 275–289, Berlin, Heidelberg, 2006. Springer-Verlag.
[20] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM, 2012.
[21] A. Back. A partial hash collision based postage scheme. http://www.hashcash.org/papers/announce.txt, 1997.
[22] Bank of England. Digital Currencies. http://www.bankofengland.co.uk/research/Pages/onebank/cbdc.aspx.
[23] S. Bano, M. Al-Bassam, and G. Danezis. The Road to Scalable Blockchain Designs. *USENIX ;login: magazine*, December 2017. To appear.
[24] A. Barger, Y. Manevich, B. Mandler, V. Bortnikov, G. Laventman, and G. Chockler. Scalable communication middleware for permissioned distributed ledgers. In *Proceedings of the 10th ACM International Systems and Storage Conference*, page 23. ACM, 2017.
[25] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.
[26] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. *CoRR*, abs/1406.5694, 2014.
[27] I. Bentov, P. Hubáček, T. Moran, and A. Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. http://eprint.iacr.org/2017/300, 2017. Accessed: 2017-06-29.
[28] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoins proof of work via proof of stake. Cryptology ePrint Archive, Report 2014/452, 2014. http://eprint.iacr.org/2014/452.
[29] A. Bessani, J. a. Sousa, and E. E. P. Alchieri. State machine replication for the masses with bft-smart. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '14, pages 355–362. IEEE Computer Society, 2014.
[30] E. Blog. Introducing Casper "the Friendly Ghost". https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/, 2015.
[31] J. Bonneau. Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.
[32] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.
[33] E. Buchman. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*. PhD thesis, 2016.
[34] V. Buterin. Soft-forking the block time to 2 min. https://www.reddit.com/r/btc/comments/428tjl/softforking_the_block_time_to_2_min_
2016.
[35] P. Caballero-Gil, C. Hernández-Goya, and C. Bruno-Castañeda. A rational approach to cryptographic protocols. *CoRR*, abs/1005.0082, 2010.
[36] C. Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.
[37] C. Cachin, R. Guerraoui, and L. Rodrigues. *Introduction to Reliable and Secure Distributed Programming*. Springer Publishing Company, Incorporated, 2nd edition, 2011.
[38] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 123–132. ACM, 2000.
[39] C. Cachin, S. Schubert, and M. Vukolić. Non-determinism in byzantine fault-tolerant replication. *arXiv preprint arXiv:1603.07351*, 2016.
[40] C. Cachin and M. Vukolić. Blockchain Consensus Protocols in the Wild. *preprint, arXiv:1707.01873 [cs.DC]*, https://arxiv.org/pdf/1707.01873.pdf, 2017.
[41] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 154–167, New York, NY, USA, 2016. ACM.
[42] M. Castro. *Practical Byzantine Fault Tolerance*. Ph.D., MIT, Jan. 2001. Also as Technical Report MIT-LCS-TR-817.
[43] M. Castro, B. Liskov, et al. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.
[44] A. Chepurnoy. Interactive proof-of-stake. *CoRR*, abs/1601.00275, 2016.
[45] A. Churyumov. Byteball: A decentralized system for storage and transfer of value.
[46] CoinDesk. Bank of America, Microsoft Partner on Blockchain Trade Finance. https://www.coindesk.com/bank-america-microsoft-partner-blockchain-trade-finance/, 2016.
[47] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.*, 31(3):8:1–8:22, Aug. 2013.
[48] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. Gün. On scaling decentralized blockchains. In *3rd Workshop on Bitcoin and Blockchain Research, Financial Cryptography 16*, 2016.
[49] P. Daian, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016. http://eprint.iacr.org/2016/919.
[50] G. Danezis and S. Meiklejohn. Centrally banked cryptocurrencies. In *Network and Distributed System Security*. The Internet Society, 2016.

[51] B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Cryptology ePrint Archive, Report 2017/573, 2017. http://eprint.iacr.org/2017/573.

[52] C. Decker, J. Seidel, and R. Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 13. ACM, 2016.

[53] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *P2P*, pages 1–10. IEEE, 2013.

[54] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.

[55] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan. Blockbench: A framework for analyzing private blockchains. https://arxiv.org/pdf/1703.04057.pdf, 2017. Accessed: 2017-03-22.

[56] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[57] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[58] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, 1992.

[59] I. Eyal. The miner's dilemma. In *36th IEEE Symposium on Security and Privacy, S&P 2015*, 2015.

[60] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 45–59, Berkeley, CA, USA, 2016. USENIX Association.

[61] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2013.

[62] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[63] I. M. Fund. Fintech and Financial Services : Initial Considerations. http://www.imf.org/en/Publications/Staff-Discussion-Notes/Issues/2017/06/16/Fintech-and-Financial-Services-Initial-Considerations-44985, 2017.

[64] J. Garay, J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. Cryptology ePrint Archive, Report 2013/496, 2013. http://eprint.iacr.org/2013/496.

[65] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.

[66] A. E. Gencer, R. van Renesse, and E. G. Sirer. Service-oriented sharding with aspen. *arXiv preprint arXiv:1611.06816*, 2016.

[67] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 3–16. ACM, 2016.

[68] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 3–16. ACM, 2016.

[69] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. http://eprint.iacr.org/2017/454, 2017.

[70] L. Glendenning, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Scalable consistency in scatter. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 15–28. ACM, 2011.

[71] J. Gray and L. Lamport. Consensus on transaction commit. *ACM Transactions on Database Systems (TODS)*, 31(1):133–160, 2006.

[72] J. N. Gray. Notes on data base operating systems. In *Operating Systems*, pages 393–481. Springer, 1978.

[73] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 321–340, 2010.

[74] S. Guo and C. Mate Jr. Crysto: A scalable and permission-less blockchain platform.

[75] M. Hearn. Corda–a distributed ledger. *Corda Technical White Paper*, 2016.

[76] T. Hønsi. SpaceMint: A Cryptocurrency Based on Proofs of Space. *IACR Cryptology ePrint Archive*, 2017.

[77] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. Zookeeper: Wait-free coordination for internet-scale systems. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 11–11, Berkeley, CA, USA, 2010. USENIX Association.

[78] Hyperledger. Sawtooth. https://intelledger.github.io/introduction.html.

[79] C. INSIGHTS. Banking Is Only The Beginning: 30 Big Industries Blockchain Could Transform. https://www.cbinsights.com/research/industries-disrupted-blockchain/, 2017.

[80] A. Judmayer. blockchainbib. https://allquantor.at/blockchainbib/.

[81] F. P. Junqueira, B. C. Reed, and M. Serafini. Zab: High-performance broadcast for primary-backup systems. In *Dependable Systems and Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 245–256. IEEE, 2011.

[82] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889, 2016. http://eprint.iacr.org/2016/889.

[83] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing.

[84] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford. Omniledger: A secure, scale-out, decentralized ledger. http://eprint.iacr.org/2017/406, 2017.

[85] A. Kothapalli, A. Miller, and N. Borisov. Smartcast: An incentive compatible consensus protocol using smart contracts. In *1st Workshop on Trusted Smart Contracts, Financial Cryptography*, 2017.

[86] J. A. Kroll, I. C. Davey, and E. W. Felten. The Economics of Bitcoin Mining or, Bitcoin in the Presence of Adversaries. *Workshop on the Economics of Information Security*, 2013.

[87] J. Kwon. Tendermint: Consensus without mining. https://tendermint.com/static/docs/tendermint.pdf, 2014.

[88] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[89] B. Laurie. An efficient distributed currency. *Practical*, 2011.

[90] L. H. Le, C. E. Bezerra, and F. Pedone. Dynamic scalable state machine replication. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, DSN '16. IEEE/IFIP, 2016.

[91] S. D. Lerner. Decor+ hop: A scalable blockchain protocol.

[92] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame. Towards scalable and private industrial blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 9–14. ACM, 2017.

[93] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolic. Xft: Practical fault tolerance beyond crashes. In *OSDI*, pages 485–500, 2016.

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

[95] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 706–719, New York, NY, USA, 2015. ACM.

[96] L. Luu, Y. Velner, J. Teutsch, and P. Saxena. Smart pool: Practical decentralized pooled mining. *IACR Cryptology ePrint Archive*, 2017:19, 2017.

[97] W. Martino. Kadena—the first scalable, high performance private blockchain. whitepaper. http://kadena.io/docs/Kadena-ConsensusWhitePaper-Aug2016.pdf, 2016.

[98] D. Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 2015.

[99] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB*, 2016.

[100] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 127–140, New York, NY, USA, 2013. ACM.

[101] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 475–490. IEEE, 2014.

[102] A. Miller, A. Kosba, J. Katz, and E. Shi. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 680–691. ACM, 2015.

[103] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.

[104] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, Dec 2008. Accessed: 2015-07-01.

[105] A. Narayanan and J. Clark. Bitcoin's academic pedigree. *Queue*, 15(4):20:20–20:49, Aug. 2017.

[106] K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *1st IEEE European Symposium on Security and Privacy, 2016*. IEEE, 2016.

[107] K. Nikitin, E. Kokoris Kogias, P. S. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, J. Cappos, and B. A. Ford. Chainiac: Proactive software-update transparency via collectively signed skipchains and verified builds. In *Proceedings of the 26th Usenix Security Symposium*, number EPFL-CONF-229405, 2017.

[108] A. Nogueira, A. Casimiro, and A. Bessani. Elastic state machine replication. *IEEE Trans. Parallel Distrib. Syst.*, 28(9):2486–2499, 2017.

[109] B. M. Oki and B. H. Liskov. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, pages 8–17. ACM, 1988.

[110] D. Ongaro and J. K. Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319, 2014.

[111] P4Titan. Slimcoin: A Peer-to-Peer Crypto-Currency with Proof-of-Burn. http://www.slimcoin.club/whitepaper.pdf, 2014.

[112] D. Palmer. $150 billion: Total cryptocurrency market cap hits new all-time high. https://www.coindesk.com/150-billion-total-cryptocurrency-market-cap-hits-new-all-time-high/, 2017.

[113] R. Pass and E. Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324. ACM, 2017.

[114] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 91. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[115] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *Technical Report (draft)*, 2015.

[116] G. Prisco. Intel develops sawtooth lakedistributed ledger technology for the hyperledger project. *Bitcoin Magazine*, 2016.

[117] L. Ren, K. Nayak, I. Abraham, and S. Devadas. Practical synchronous byzantine consensus. *arXiv preprint arXiv:1704.02397*, 2017.

[118] Z. Ren, K. Cong, J. Pouwelse, and Z. Erkin. Implicit consensus: Blockchain with unbounded throughput. *arXiv preprint arXiv:1705.11046*, 2017.

[119] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. http://arxiv.org/pdf/1507.06183.pdf, 2015. Accessed: 2016-08-22.

[120] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, Dec. 1990.

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

[122] B. Scott. Bitcoin academic research. https://docs.google.com/spreadsheets/d/1VaWhbAj7hWNdiE73P-W-wrl5a0WNgzjofmZXe0Rh5sg/htmlview?usp=sharing&pli=1&sle=true.

[123] D. Skeen. Nonblocking commit protocols. In *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pages 133–142. ACM, 1981.

[124] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[125] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013(881), 2013.

[126] T. Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. *Report, available online, Apr*, 2015.

[127] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable bias-resistant distributed randomness. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 444–460. IEEE, 2017.

[128] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 526–545, 2016.

[129] D. Trends. The world's cryptocurrency mining uses more electricity than iceland. https://www.digitaltrends.com/computing/bitcoin-ethereum-mining-use-significant-ele 2017.

[130] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[131] M. Vukolic. Eventually returning to strong consistency. https://pdfs.semanticscholar.org/a6a1/b70305b27c556aac779fb65429db9c2e1ef2.pdf, 2016.

[132] M. Vukolić. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, BCC '17, pages 3–7, New York, NY, USA, 2017. ACM.

[133] Wikipedia. Merkle tree. https://en.wikipedia.org/wiki/Merkle_tree.

[134] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[135] G. Wood. Polkadot: Vision for a heterogeneous multi-chain framework, 2016.

[136] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander. Where Is Current Research on Blockchain Technology? – A Systematic Review. volume 11, page e0163477. Public Library of Science, 2016.

[137] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. V. Renesse. REM: Resource-efficient mining for blockchains. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1427–1444, Vancouver, BC, 2017. USENIX Association.

[138] R. Zhang and B. Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*, pages 277–292. Springer, 2017.

[139] A. Zohar. Securing and scaling cryptocurrencies. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 5161–5165, 2017.

Appendix

A. List of Papers

PoW	[104], [21], [18], [34], [61], [53], [60], [65], [58], [96], [68], [115], [134], [125], [96], [102], [91], [113]
PoX	[26], [28], [82], [51], [49], [9], [4] [44] [30] [101] [76] [78] [137]
HYBRID	[98], [83], [128], [50], [69], [94], [17], [84], [94], [132], [15], [33], [45], [103], [114], [14], [24], [12], [107], [126], [75], [74], [89], [118], [39], [**?**], [135], [117], [121], [66], [97], [92], [36], [99]

B. Glossary

- *Adversary model*: 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).
- *Code available*: Whether the code implementing the system is publicly available.
- *Committee*: How the participants work together to participate in the consensus protocol; either they all work together (single committee), or they are divided in multiple subgroups (multiple committees).
- *Committee Formation*: How the members of the committee are chosen, for example *via* proof-of-work, proof-of-stake, trusted hardware etc.
- *Consistency*: The likelihood that the system will reach consensus on a proposed value; it can be either strong or weak.
- *DoS resistance*: 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.

- *Experimental setup*: The configuration used to generate the numbers reported for throughput and latency.
- *Incentives*: The mechanisms that keep nodes motivated to participate in the system and follow its rules.
- *Inter-committee Configuration*: How the members are assigned to the committee in a single committee setting; either members serve on the committee permanently (static), or they are changed at regular intervals (rolling, or full swap).
- *Inter-committee Consensus*: Reaching agreement on a value among nodes in a single committee.
- *Intra-committee configuration*: How the members are assigned to the committees in a multiple committees setting; it can be static or dynamic.
- *Intra-committee Consensus*: Reaching agreement on a value among nodes across multiple committees; this can be optionally mediated by an external party (*e.g.,* the client).
- *Latency*: The time it takes from when a transaction is proposed until consensus has been reached on it.
- *Leader*: The leader of the consensus protocol, which can be either elected among the current committee (internally), externally, or flexibly (*e.g.,* through arbitrary smart contracts).
- *Participants*: The nodes that participate in the consensus protocol.
- *Permissioned blockchain*: Only participants selected by the appropriate authorities can participate in the consensus protocol.
- *Permissionless blockchain*: Anyone can join the system and participate in the consensus protocol.
- *Scalability*: The system's ability to achieve greater throughput when consensus involves a larger number of nodes.
- *Throughput*: The maximum rate at which transactions can be agreed upon by the consensus protocol (transactions per second/hour).
- *Transaction censorship resistance*: The system's resilience to proposed transactions being suppressed (*i.e.,* censored) by malicious node(s) involved in consensus.