(Invited Paper) on the Security of Blockchain Consensus Protocols

Sourav Das(✉), Aashish Kolluri, Prateek Saxena, and Haifeng Yu

Computer Science Department, School of Computing,
National University of Singapore, Singapore, Singapore
souravdas1547@gmail.com, {aashish7,prateeks,haifeng}@comp.nus.edu.sg

Abstract. In the last decade, several permissionless proof-of-work blockchain protocols have focused on scalability. Since these protocols are very difficult to change once deployed, their robustness and security are of paramount importance. This paper summarizes the desired end properties of blockchain consensus protocols and sheds light on the critical role of theoretical analyses of their design. We summarize the major paradigms in prior constructions and discuss open issues in this space.

1 Introduction

Blockchain protocols, which originated in Bitcoin [57], allow a large network of computers to agree on the state of a shared ledger. Applications utilizing blockchains embrace a semantics of immutability: once something is committed to the blockchain, it can not be reversed without extensive effort from a majority of computers connected to it. These protocols embody the vision of a global "consensus computer" to which arbitrary machines with no pre-established identities can connect for offering their computational resources (in return for a fee), without dependence on any centralized authority. Despite this, the computational infrastructure strives to offer failure resistance against arbitrarily malicious actors. Security is at the heart of these protocols and applications built on them, as they now support an economy valued at several hundred billion dollars[1].

Theoretical frameworks should guide the construction of practical systems. The last decade of work on designing blockchain protocols highlights the importance of this interplay. In this paper, we distill the essence of the problem of designing secure blockchain consensus protocols, which are striving towards lower latencies and scalability. Our goal is to present key results that have surfaced in the last decade, offering a retrospective view of how consensus protocols have evolved. We examine a central question: is Bitcoin's original consensus protocol—often called Nakamoto consensus—secure, and if so, under which conditions?

The authors are sorted alphabetically by the last name.

[1] Total market capitalization of cryptocurrencies is 217, 279, 849, 996 USD at the time of writing [5].

V. Ganapathy et al. (Eds.): ICISS 2018, LNCS 11281, pp. 465–480, 2018.
https://doi.org/10.1007/978-3-030-05171-6_24

There have been many folklore claims, for instance, that Nakamoto consensus is categorically secure up to $\frac{1}{2}$ adversarial power, beyond which "51% attacks" violate its guarantees [57,62]. Careful analysis, however, has dispelled many such claims. The quest for designing more scalable and secure consensus protocols has ensued. We review some of these construction paradigms and open problems. We focus mostly on protocols that are designed to operate in the open or permissionless setting which limit adversaries by computational power only.

2 The Blockchain Consensus Problem

One of the novel algorithmic advances in Bitcoin is its consensus algorithm called Nakamoto consensus. The protocol runs between a set of *miners* (computers), connected via a peer-to-peer (P2P) overlay over the Internet. Miners agree on the state of a globally distributed ledger of *transactions* periodically. Transactions are broken up into sets of constant size called "blocks", and miners broadcast them to other miners continuously. The essence of the blockchain consensus protocol is to reach agreement on the *total order* on a common subset of blocks seen by all the honest miners. The total ordering of blocks is sufficient to achieve a well-defined notion of consistency [43]. With this, for instance in a cryptocurrency application, it is easy to avoid double-spends: the client can always pick the first[2] transaction that spends a coin, ignoring later (conflicting) transactions that spend the same coin.

One way to agree on the total order, as proposed in Nakamoto consensus is to order blocks in a hashchain data structure [2], which coined the term "blockchain". In a blockchain, blocks are chained in a sequence using cryptographic hashes, where one block hash binds it to its predecessor in the total order. Transactions can have any semantics. For instance, in Bitcoin these transactions represent ownership (and payments) of virtual coins. In more recent cryptocurrencies, transactions represent the more traditional notion of atomic state updates for programs called smart contracts [4,68].

2.1 Threat Model and Assumptions

Miners who follow the prescribed protocol are called *honest*. This consensus protocol makes three assumptions, which strikingly differ from prior literature:

(a) honest peers, with no pre-established identities, can broadcast publicly to all other honest nodes a block synchronously, within a delay δ;
(b) the total computational power of the system is approximately known, out of which a known fraction f is assumed malicious (Byzantine [44]);
(c) all peers have an unbiased source of local randomness, and a trusted setup phase creates public parameters in a constant size "genesis block".

[2] Earliest one in the total order.

Bitcoin's assumptions, especially the combination of (a) and (b), are novel and minimalistic in a sense. Prior works in the literature study asynchronous networks which can lose connectivity between honest miners in the P2P overlay for indefinite periods of time [10,28,51]. We say that the network is "partitioned" if honest nodes lose connectivity to a significant fraction of other honest miners. Under this asynchronous model, a deterministic consensus is classically impossible [28] and most probabilistic consensus algorithms in the classical model have exponential round complexity [39][3]. This suggests that some more assumptions are necessary to avoid well-known impossibility results and long-standing problems. Assumption (a) of δ-synchronous broadcast is stronger than assuming an arbitrarily asynchronous network, but the protocol designer can estimate an acceptable network delay δ, and the Nakamoto consensus protocol can be instantiated with a block generation time that is large enough to accommodate it [15,56]. Different blockchains use this flexibility of picking different tolerance to network partitions [3,4]. Many prior protocols in the literature have assumed much more complex communication models of strongly synchronized clocks across nodes, pre-established identities attached to each message, global directories of identities participating (e.g. PKI), secret communication channels between peers, and so on [29]. Bitcoin takes a fresh approach assuming none of these.

Some form of sybil resistance is necessary to an open system where any number of computers or miners can connect [23]. Assumption (b) is a form of Sybil resistance, which is substantially different from prior protocols that assume pre-established identities or PKI [52]. For instance, popular Byzantine agreement protocols achieve consensus in a setup that assumes that the set of participants in the protocol are known to each other in advance [19,44]. Bitcoin does not assume that miners know identities of other miners in advance. More recently, many "Proof-of-Stake" (PoS) proposals assume that identities are pre-established and have an agreed upon fractional ownership in virtual coins (or stake) [20,32,38]. Such staking assumptions can be bootstrapped from Assumption (b).

Assumption (c) is assumed only once at the start of the blockchain. However, we believe this assumption is not necessary; it can be constructed directly from assumptions (a) and (b) using recent works as building blocks [7,34]. It is convenient, however, to assume this to avoid complexity of bootstrapping.

Attacking the Assumptions. A number of works have shown direct attacks on these assumptions. Assumption (a) states that all messages from honest nodes reach other honest within time δ; however, partitioning and eclipse attacks subvert these directly [8,33]. In partitioning attacks, malicious nodes aim to disconnect honest miners from each other at the P2P or ISP level. Similarly, eclipse attacks allow certain malicious miners to delay the propagation of network messages selectively to other miners. Protecting against these attacks is directly

[3] King at el. have presented the first theoretical result with polynomial round complexity recently in the model where no secret channels are constructed; the construction tolerates less than 1% Byzantine adversary [12,39].

important to fixing the parameters of the consensus algorithms; however, these are outside the scope of the design of the consensus protocol itself. It does motivate building "hard-to-partition" P2P overlays, and defenses to avoid centralization at the ISP-level on the Internet, upon which blockchain overlays operate.

Assumption (b) has been challenged as well. The assumption that the adversary controls no more than fraction f of the compute power has been subject to much debate, since centralization of mining power is an acknowledged concern [25,48]. Mining protocols that force mining pools to run fairly, such as by executing a smart contract, have been investigated as a practical solution [50]. Prior work has proposed dis-incentives against forming mining pools or coalitions through non-outsourceable puzzles [53]. However, recently, there have been reports of real attacks that they require short-lived capital to carry out the attacks on specific public blockchains [14]. Addressing these attacks effectively, through incentives or technical means, is an open problem. Nonetheless, we argue that some forms of Sybil resilience and network delivery guarantees seem necessary; therefore, Bitcoin's assumptions are an acceptable starting point.

2.2 Nakamoto Consensus

Bitcoin's consensus protocol is a concrete example of a blockchain protocol. The protocol uses a specific computational puzzle or "proof-of-work" (PoW) puzzle [2,24]. The puzzle asks miners to find a nonce such that $H(nonce||seed||\dots) < 2^d$, where d is tunable "puzzle difficulty" parameter and H is a cryptographically strong hash function. Anyone with the solution to the puzzle $<nonce, seed, d>$ can verify in one hash evaluation whether the solution is valid. The *seed* serves the role of a randomized value for instantiating new puzzles over time. It is useful to think of PoW puzzles as a procedure to sample from the computational power distribution in the mining network.

Each miner in the protocol keeps minimal state, i.e., the longest chain of blocks in its local view. Each miner solves a PoW puzzle, which is stateless computation. The inputs of a puzzle are taken only from miner's local view, specifically, the latest block hash value serves as *seed* of the PoW puzzle. If a miner receives a block from the network, it inspects the validity of the block. If the block has a valid PoW solution, the miner extends its local view of the blockchain by one block, and the next round of mining starts with this new seed. If the miner receives a valid chain longer than its present chain, the miner switches its local view to it immediately. A block is confirmed after a constant number of blocks (k) extend it in the longest chain. The protocol sets k internally ($k = 6$ in Bitcoin). This consensus protocol is orthogonal to the representations of transaction data (UTXO [57] vs. accounts [72]), DoS-prevention checks on network messages [71], and validity checks (double-spend validation) [17,47,71].

2.3 The Problem

The blockchain consensus protocol allows each miner to periodically output a set of blocks that it deems as *confirmed* or final. The security goal of the consensus protocols is to ensure that honest miners (a) agree on the same total order for all confirmed blocks, and (b) the set of confirmed blocks includes those proposed by all miners fairly, i.e., in proportion to their contributed computation power for mining. We consider a protocol secure up to a fraction f of adversarial power if it can guarantee its security goals with high probability (w.h.p.)[4]. The underlying constraint is δ, the time taken for honest miners to receive a fixed size block, which is pre-determined by the network bandwidth of the miners. The performance criterion is how quickly blocks proposed by miners are agreed upon by the honest network.

Protocols can be compared both on their *block confirmation rate* and their tolerance to adversarial fraction f. If a protocol A includes strictly more blocks in its agreed total order per unit time than protocol B, tolerating the same adversarial power, then A is strictly better in performance. Likewise, if protocol A agrees on the same number of blocks per unit time as B, but tolerates strictly more adversarial power, then A is strictly better in security. One can even compare different configurations of the same protocol. Taking Nakamoto consensus as an example, the parameter k (number of confirmation blocks) offers a tradeoff between security tolerance and confirmation time. If we compare two configurations of Nakamoto consensus, with different values of k, it turns out the configuration with larger values of k offer slower confirmation times but higher security tolerance.

Security Properties of Blockchain consensus Protocols. The foremost question for any blockchain consensus protocol is which security guarantees it provides when a fraction f of power is controlled by a Byzantine adversary. One can think of the blockchain the protocol as a continuous time protocol, where at any time instant, each miner reports a set of blocks as confirmed and a total ordering relation over them. The first security goal is to ensure that an honest miner does not change its set of confirmed blocks over time, captured by a *stability* property:

Stability: For any honest miner, the set of confirmed blocks output at time t_1 is a subset of the set of confirmed blocks at time t_2 w.h.p, if $t_2 > t_1$. The order of confirmed blocks does not change over time w.h.p.

One oft-cited strategy for the attacker to subvert the stability property in Nakamoto consensus is to introduce an alternate longer chain starting at a block that is at least k blocks deep. If successful, this causes honest miners to switch their view on what is confirmed. This strategy was analyzed in the original Bitcoin paper [57]. However, this is not the only strategy to consider; the protocol must remain secure under all adversarial strategies [16].

The second security goal is to prove that miners following the protocol reach *agreement*: for any two honest miners, the confirmed blocks of one are also confirmed by the other, and that the order of the confirmed blocks is identical for both. Specifically, the following agreement property captures this:

[4] For any security parameter $\lambda > 0$, an event happening *with high probability* (w.h.p) implies that event happens with probability $1 - O(1/2^{\lambda})$.

Agreement: Let C_1 and C_2 be the set of confirmed blocks reported by any two honest miners, then w.h.p:

(A) Either $C_1 \subseteq C_2$ or $C_2 \subseteq C_1$; and
(B) the blocks in $C_1 \cap C_2$ are ordered identically by both miners.

At any time instant, note that requirement (A) above allows one miner to not have confirmed all the blocks of the other honest miner. But, it disallows the case where two honest miners confirm two blocks, each one of which is only confirmed by one miner and not the other. Requirement (B) ensures blocks that are confirmed by both will necessarily be in the same order.

In Nakamoto consensus, satisfying the agreement property implies that the longest chain, discarding the last k blocks, of an honest miner should be a prefix of the longest chain of other honest miners [17]. Ensuring a common prefix satisfies both requirement (A) and (B) above. These properties (and others) are used to prove rigorous analytical bounds on the fraction f tolerated under different attack models by Nakamoto consensus and its variants [30,36,58].

A third critical property of the blockchain protocols is *fairness*. In a fair protocol, if the adversary controls fraction f of the computational power, the expected fraction of blocks contributed by it in the confirmed set blocks should be close to f. However, the adversary can deviate from the honest protocol to mine more blocks [27,31,63]. It may do so to increase its mining rewards, to favor or censor transactions of its choice, or bias the fairness of the application running on top of the blockchain in some way. The following property captures this security notion of fairness:

Fairness: There is a negligible probability that the fraction of blocks proposed by the adversary in the set of confirmed blocks, over any time interval $t > c \ . \ \delta$, for some constant c, is more than f.

The constant c in the above definition specifies whether the protocol is fair over a small time windows or larger ones. Protocols that minimize c are desirable, as they sample from the computational power distribution in the mining network frequently. To understand the importance of minimizing this constant, consider the following proposal: one could run any fair consensus protocol to agree on a leader (say) once a week who broadcasts a massive "block" for the rest of the week. This is sufficient to utilize the bandwidth available, and in expectation over a large time window (say 1 year), it would be fair in picking leaders. However, such a protocol is *not* desirable as that leader may favor its own transaction blocks for a week. Further, the leader may be targeted for Denial-of-Service (DoS) or eclipse attacks during its tenure. Therefore, blockchain protocols that agree of (lots of) small blocks, sampling often from the computation power distribution, are better as fairness holds over shorter time windows.

Existing blockchain protocols can be compared directly on the minimum time window (or c) over which their fairness holds. Existing scalable blockchain protocol compete on lowering c for better decentralization and DoS-resilience.

3 Security Analysis of Nakamoto Consensus

Stability & Agreement Properties. Different strategies to subvert the stability and agreement properties of Nakamoto consensus have been studied, both experimentally and analytically, in prior works [30,58,66]. One key observation from these analyses is that Nakamoto consensus protocol exhibits poor tolerance to adversarial power when the block interval reduces significantly, especially as it starts to approach the broadcast latency. Intuitively, at low block intervals, many miners will start to mine blocks nearly simultaneously; these will be received in an unpredictable order by other miners. Consequently, some miners will mine on one block while the others on other blocks. This results having temporary "forks" in the chain. The rate of forks, often measured by the creation rate of "stale blocks" (which do not end up on the longest chain), is measured empirically by Gervais et al. for various configurations of Nakamoto consensus [31].

The security of Nakamoto consensus protocol hinges primarily on the ratio of the block interval to the broadcast latency. Several analyses have shown that there exists a large enough k (number of block confirmations) for which the protocol is secure for some large values of the block interval [30,58,66]. For certain high block interval rates (e.g., 10 min as in Bitcoin) for broadcast delay δ of a few tens of seconds observed empirically [22], prior analysis shows that the agreement property holds close to $f = \frac{1}{2}$ adversarial power fraction. However, this adversarial power tolerated drops as block interval rates reduce. Specific attack strategies have shown that the f drops to well-below 40%, even when the ratio of block interval rate to δ is close to 1, as in Ethereum [36,58]. The theoretical security tolerance thresholds for which security is guaranteed drops quickly as block interval decreases further. These results explain that Nakamoto consensus is not categorically secure under arbitrary block interval rates, unlike what folklore claims portray. More effective attack strategies and models than those proposed in prior works are possible and an open area of investigation.

Fairness Property. The fairness property has been extensively studied as well. The selfish mining and short-term block withholding strategies (c.f. Eyal and Sirer [27]) provide prominent results. This work shows that even a miner with 25% of the computation power can bias the agreed chain with its blocks (gaining more reward than expected). This shows that Nakamoto consensus cannot withstand a $\frac{1}{3}$ or $\frac{1}{2}$ adversarial power, as is assumed by the folklore claims of "51% attacks". This is relevant because a number of works rely on this fairness property for application-specific security guarantees (e.g. beacons [13,16], lotteries [7,13], bounties [18], samplers [6,41,47,73]), assuming that fairness property holds for certain adversarial power.

When studying the fairness property, many works have emphasized a subclass of rational adversaries, i.e., miners incentivized to optimize some utility function (e.g. maximizing their expected profits, maximize blocks mined by it, censor certain transactions, and so on) [27,31,63]. There have been various results showing the coin reward structures are not incentive-compatible, and rational miners can maximize their utility by deviating from the protocol [35,49,69,70,74].

Remarks. We remind readers that assuming that no miners are Byzantine, just that miners are either rational or honest, has some limitations. Rationality arguments are often made in virtual coin incentives and there are real markets today where coins are traded for fiat currencies. An attack may seem irrational (not incentivized) viewed from the objective of an assumed utility function for the attacker, whereas it may be incentivized as it may impact the valuation of virtual coins. Early works on the Goldfinger attacks [42] and feather-forking [1] discuss this issue of how reasonable it is to assume that miners will be rational versus Byzantine. Nakamoto protocol safety merits a study independently of the model of incentives, directly in the threat model of Byzantine adversaries.

4 Scalability Extensions to Nakamoto Consensus

Increasing Block Sizes. One natural way of increasing transaction rates is to have large blocks in Nakamoto consensus that consume all the bandwidth available. In such a design, optimal bandwidth utilization is achieved by picking large block sizes. Therefore, in such a solution, the constant c of the fairness property is directly dependent on the block sizes sufficient to saturate the entire network's bandwidth. As discussed earlier, this may not guarantee fairness in short time windows, and the attacker can target a single block proposer in each epoch. Several proposals utilize this design choice, such as the use of key blocks in Bitcoin-NG [26] and the identity establishment step of [21], implicitly inheriting the issue of ensuring fairness.

Reducing Block Interval. For achieving fairness in shorter intervals, one prominent re-configuration of Nakamoto consensus is to reduce the block interval. In Nakamoto consensus, this is achieved by lowering the puzzle difficulty for the known computational power. This tack is utilized in many cryptocurrencies. Bitcoin fixes the interval to be approximately 10 min, whereas Litecoin reduces it by four times and Ethereum brings it to 10–17 s. The lower the block interval rate, the better the fairness in choosing how many block proposal are generated per unit time. However, as explained in Sect. 2.3, the longest chain rule for selecting the total order does not remain secure. Lowering the block interval lowers the adversarial power f that the protocol tolerates significantly.

When block intervals are reduced in Nakamoto consensus, the open problem is how to order the blocks received by a miner. Various ordering rules have been proposed in the literature, but a solution that provably achieves optimal security and confirmation rate is not yet known. We summarize existing proposals next.

GHOST Rule. Sompolinsky and Zohar proposed the GHOST rule, an alternative to the longest chain rule of counting the number of blocks [66]. In the GHOST rule, miners retain information of all blocks they obtain from the network in their local view. These blocks thus form a block tree-like structure i.e., each block contains a subtree of blocks mined upon it. The weight of a block is the number of blocks in all forks belonging to its subtree. The GHOST rule dictates that heaviest chain consists of the heaviest block, by weight, at each depth in the tree of forks, thus, allowing blocks that are *not* on the heaviest to contribute to the weights of blocks on it. Hence, in essence, it picks the "heaviest" chain (with evidence of the most mining work contributed to it), rather than the longest.

A security analysis of the GHOST rule for certain attack strategies is presented in the proposal of Sompolinsky and Zohar [66]. Kaiyias et al. establish that GHOST rule is secure for certain parameters when block intervals are large [37]. The security analysis of the GHOST rule, especially when the block interval is smaller than the broadcast delay, merits a careful analysis, like in the case of Bitcoin's longest chain rule. Specifically, the security depends on how ties between heaviest blocks of equal weight are resolved. Several tie-breaking rules have been proposed, picking (a) uniformly between candidate blocks, (b) the first one that the miner receives, (c) the one with the smallest timestamp, and (d) the one with the smallest PoW puzzle solution. Different attacker models have been studied showing that GHOST is not unilaterally superior to the long chain rule [37,66]. It has been suggested that strategy (a) is preferred to strategy (b) for certain range of block intervals when the adversary uses a selfish mining strategy for fairness [27]. We conjecture that, in fact, strategy (a) is not universally better because it splits the available honest mining power across various forks. This reduces the power necessary for the adversary to create the heaviest chain by mining selfishly on its own fork, impacting the stability and agreement property.

Much like the longest chain rule, the final selected chain in GHOST discards all blocks that are not along the heaviest chain. So, the throughput of the GHOST protocol is within a small factor of that resulting from the longest chain rule. This is sub-optimal since many blocks seen by the honest miners are eventually discarded, lowering the average confirmation rate per block.

Directed Acyclic Graphs (DAGs). Recent works have proposed mechanisms to include blocks that are *not* on the longest or the heaviest chain in Nakamoto consensus. One line of work proposes that instead of keeping a chain, the miners can keep a directed acyclic graph (DAG) of blocks seen in their local view [45,46,60,64,65]. Each miner has its view of the blocks it has seen, partially ordered in the DAG. The DAG has edges called "reference edges" that point to those blocks that the miner saw before it mined the present block, in addition to usual hashchain edges. The protocol specifies how miners order the blocks at the same depth in their local views of the DAG, and agree on their diverging DAG views.

A number of rules have been proposed to agree and order DAGs, such as SPECTRE [65], PHANTOM [64], and Conflux [46]. For instance, the Conflux protocol shows one mechanism for achieving this by finding a "pivot chain" using the GHOST rule and then topologically sorting the blocks that are at the same depth as a block on the pivot chain. Miners union the DAG views they receive from other nodes. Blocks at the same depth are ordered on hash value of PoW puzzle difficulty solved. The security of Conflux reduces to that of the GHOST rule. The PHANTOM protocol selects a subtree rather than a single chain and sorts topologically. The Conflux paper presents a liveness attack on the PHANTOM protocol, which is shown to be effective with an adversary that controls 15% computational power. DAG based schemes are relatively recent, and their rigorous scrutiny deserves further attention.

5 Scalability Solutions Based on Byzantine Agreement

The difficulty of securing variants of Nakamoto consensus has led to an alternative line of protocols that leverage classical Byzantine agreement (or BA) protocols instead. Consensus in a Byzantine network has been extensively studied, see surveys [9,29,39]. However, directly applying BA algorithms in the assumption model of Bitcoin is not straightforward. One key difficulty is that commonly BA protocols assume a pre-established set of identities known to all participants running the protocol.

To achieve this starting point, several protocols propose different designs to establish identities from the assumptions (a)–(c). Wattenhofer et al. use the Nakamoto consensus protocol to arrive at a common prefix of blocks, which contain public keys (identities) of the participants [21]. The security of this step directly relies directly on the security of Nakamoto consensus variants utilized.

The use of Nakamoto consensus to establish identities as a pre-step is not necessary. A number of works including Elastico [47], Omniledger [41], and RapidChain [73] directly establish identities from PoW or related cryptographic constructs[5]. A number of these solutions further "shard" identities, i.e., assign different clusters/committees to identities implicitly which can operate in parallel [6,41,47,73]. More recent works show how to use more general cryptographic puzzles to bootstrap a "reconciled view" of the set of participants in a mining network without Nakamoto consensus and even without assumption (c) outlined in Sect. 1 [7,34,53].

The security of these designs depends directly on the size of the set of identities established to run the BA protocol. The larger the size of the identity set, the higher is the communication cost of establishing the identity sets between the participants and subsequently running the BA protocol instances. This sample size establishes limits on how often the identity establishment protocol can run, which is directly related to the constant c for which the fairness property holds. There is a trade-off in choosing the sample size that different designs make. The sample sizes picked in various designs vary, but typically are in hundreds, for acceptable levels of security and confirmation times in tens of seconds [32,41,47].

The set of identities is supposed to be chosen randomly by sampling the computational power or stake distribution. Therefore, the fraction of adversarial identities in the chosen set is f in expectation. For sets of size s, the probability of the adversarial identities deviating from the mean f is bounded by a function exponentially small in s, which follows from the standard Chernoff bounds (Chap. 4. [55]). We point out that these analyses of sample sizes for establishing identity sets are often the same for proof-of-work systems and proof-of-stake systems [32]. This is because the process of creating identities based on random

[5] Verifiable random functions (VRFs) have been used to probabilistically select identity sets without eagerly revealing the identities selected [32,41].

sampling, and counting how many identities (Byzantine and honest) end up in an identity set, is the same for many PoS- or PoW-based systems. In all these different protocols for establishing identities, the role of a formal framework to model the sampling process (often a Binomial random variable[6]) guides the robust choice of sample size parameters.

A second factor that dictates the set size is the fraction of adversaries the BA algorithm can tolerate in one instance. BA protocols designed original for fully asynchronous networks like PBFT [19] tolerate $\frac{1}{3}$ adversary or their more efficient versions (ByzCoin [40], Omniledger [41]). Recent works use synchronous BA protocols which can tolerate the optimal $\frac{1}{2}$ Byzantine fraction [29,61]. Protocols that can tolerate better adversarial fractions (e.g. $\frac{1}{2}$ vs. $\frac{1}{3}$) require further smaller sets of identities [73].

The use of BA agreement in blockchains has spurred further research in designing faster BA protocols. The trade-offs in designing BA protocols which are fast when the network delay δ is small while degrading gracefully on slower networks have been actively studied [41,54,59,61]. Several works have improved the communication costs of BA agreement protocols, trading off the performance between the honest case and when the overlay P2P graphs have Byzantine adversaries [40,67]. More efficient broadcast primitives have emerged, for instance, using collective signing [67] or erasure-coded information dispersal techniques [54,73].

As blockchains run continuously, multiple rounds of BA protocol are implicitly composed in sequence. In sharded blockchains, BA protocol instances are often composed in parallel as well. Some care must be taken when composing instances, especially for BA protocols that have probabilistic termination time like the BA* algorithm [32] or PBFT[7] [19]. When BFT protocol instances are running in parallel—as in sharding-based blockchain protocols—the expected running time for all of the instances generation may not be constant in expectation, as the slowest instance (out of many) dominates the stopping time [11]. Optionally, to mitigate this delay, a protocol may choose to run a BA protocol instance to synchronize the output of the parallel instances running on each shard [47]. Specifically, a final committee determines whether a shard has agreed upon a block or not within a predetermined time bound. This bounds the delay at each shard, however, it admits the possibility that in some rounds, empty blocks will be mined. However, the probability that the protocol does not make progress for a few rounds under the assumption (a) of Sect. 2.3 is negligibly small. Protocols may not choose to synchronize outputs of shards at each epoch,

6 The probability of a picked identity being Byzantine in the sample set is f, and honest is $1 - f$. The analysis examines two Binomial random variables, the number of honest and Byzantine adversaries picked in an indentity set, such that their ratio does not exceed the tolerance of the BA algorithm. When Nakamoto-style PoW is used to create identities, the number of identities created per unit time (by setting an appropriate puzzle difficulty), is approximated well by a Poisson random variable.

7 PBFT is a leader-based protocol and may have multiple rounds, which depends on the probability of a dishonest leader being chosen at a particular round triggering a "view change" sub-step.

but then additional mechanisms to ensure atomicity of cross-shard commits in an epoch are often utilized [41].

6 Conclusions

We survey known results about how well Nakamoto consensus guarantees desired security, when configured for faster confirmations. Guided by theoretical analyses, new designs and variants of the Nakamoto consensus protocol are under active investigation, searching for an optimal protocol. Careful analyses have dispelled folklore claims of safety against 51% attacks hold categorically when re-configuring the Nakamoto consensus protocol. We further summarize another recent paradigm of constructions that are based on using established Byzantine agreement protocols. We explain some of the commonalities and the factors that determine their confirmation latencies and security trade-offs.

Acknowledgements. We thank Hung Dang for his helpful comments on the work. We thank sponsors of the Crystal Center at NUS, which has supported this work. All opinions presented in this work are those of the authors only.

References

1. Feather-forks: enforcing a blacklist with sub-50. https://bitcointalk.org/index.php?topic=312668.0
2. Hash chain wiki. https://en.wikipedia.org/wiki/Hash_chain
3. Litecoin wiki. https://en.wikipedia.org/wiki/Litecoin
4. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper
5. Total market capital of cryptourrencies (2018). https://coinmarketcap.com
6. Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., Danezis, G.: Chainspace: a sharded smart contracts platform. arXiv preprint arXiv:1708.03778 (2017)
7. Andrychowicz, M., Dziembowski, S.: PoW-based distributed cryptography with no trusted setup. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 379–399. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_19
8. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking Bitcoin: routing attacks on cryptocurrencies. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 375–392. IEEE (2017)
9. Aspnes, J.: Randomized protocols for asynchronous consensus. Distrib. Comput. **16**(2–3), 165–175 (2003)
10. Ben-Or, M.: Another advantage of free choice (extended abstract): completely asynchronous agreement protocols. In: Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing, pp. 27–30. ACM (1983)
11. Ben-Or, M., El-Yaniv, R.: Resilient-optimal interactive consistency in constant time. Distrib. Comput. **16**(4), 249–262 (2003)
12. Ben-Or, M., Pavlov, E., Vaikuntanathan, V.: Byzantine agreement in the full-information model in O (log n) rounds. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 179–186. ACM (2006)

13. Bentov, I., Gabizon, A., Zuckerman, D.: Bitcoin beacon. arXiv preprint arXiv:1605.04559 (2016)
14. Bitcoinst: 51 percent attack on Bitcoin cash (2018). https://bitcoinist.com/roger-ver-bitpico-hard-fork-bitcoin-cash/
15. Bolot, J.C.: End-to-end packet delay and loss behavior in the internet. In: ACM SIGCOMM Computer Communication Review, vol. 23, pp. 289–298. ACM (1993)
16. Bonneau, J., Clark, J., Goldfeder, S.: On bitcoin as a public randomness source. IACR Cryptology ePrint Archive 2015, 1015 (2015)
17. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: research perspectives and challenges for Bitcoin and cryptocurrencies. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 104–121. IEEE (2015)
18. Breidenbach, L., Daian, P., Tramer, F., Juels, A.: Enter the hydra: towards principled bug bounties and exploit-resistant smart contracts. In: Proceedings of the 27th USENIX Conference on Security Symposium. USENIX Association (2018)
19. Castro, M., Liskov, B., et al.: Practical Byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation, pp. 173–186. USENIX Association (1999)
20. Daian, P., Pass, R., Shi, E.: Snow white: robustly reconfigurable consensus and applications to provably secure proofs of stake (2017)
21. Decker, C., Seidel, J., Wattenhofer, R.: Bitcoin meets strong consistency. In: Proceedings of the 17th International Conference on Distributed Computing and Networking, p. 13. ACM (2016)
22. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P), pp. 1–10. IEEE (2013)
23. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_24
24. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_10
25. Eyal, I.: The miner's dilemma. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 89–103. IEEE (2015)
26. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-NG: a scalable blockchain protocol. In: NSDI, pp. 45–59 (2016)
27. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (2018)
28. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM (JACM) **32**(2), 374–382 (1985)
29. Garay, J., Kiayias, A.: SoK: a consensus taxonomy in the blockchain era. Cryptology ePrint Archive, Report 2018/754 (2018). https://eprint.iacr.org/2018/754
30. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
31. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16. ACM (2016)

32. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68. ACM (2017)
33. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on Bitcoin's peer-to-peer network. In: USENIX Security Symposium, pp. 129–144 (2015)
34. Hou, R., Jahja, I., Luu, L., Saxena, P., Yu, H.: Randomized view reconciliation in permissionless distributed systems (2017)
35. Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S.M., Felten, E.W.: Arbitrum: scalable, private smart contracts. In: Proceedings of the 27th USENIX Conference on Security Symposium, pp. 1353–1370. USENIX Association (2018)
36. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols (2015)
37. Kiayias, A., Panagiotakos, G.: On trees, chains and fast transactions in the blockchain. (2016)
38. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
39. King, V., Saia, J.: Byzantine agreement in expected polynomial time. J. ACM (JACM) **63**(2), 13 (2016)
40. Kogias, E.K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing Bitcoin security and performance with strong consistency via collective signing. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 279–296 (2016)
41. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Ford, B.: OmniLedger: a secure, scale-out, decentralized ledger. IACR Cryptology ePrint Archive 2017, 406 (2017)
42. Kroll, J.A., Davey, I.C., Felten, E.W.: The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In: Proceedings of WEIS, vol. 2013, p. 11 (2013)
43. Lamport, L.: How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans. Comput. **28**(9), 690–691 (1979). https://doi.org/10.1109/TC.1979.1675439
44. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. (TOPLAS) **4**(3), 382–401 (1982)
45. Lewenberg, Y., Sompolinsky, Y., Zohar, A.: Inclusive block chain protocols. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 528–547. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47854-7_33
46. Li, C., Li, P., Xu, W., Long, F., Yao, A.C.: Scaling Nakamoto consensus to thousands of transactions per second. arXiv preprint arXiv:1805.03870 (2018)
47. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: Asecure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30. ACM (2016)
48. Luu, L., Saha, R., Parameshwaran, I., Saxena, P., Hobor, A.: On power splitting games in distributed computation: The case of Bitcoin pooled mining. In: 2015 IEEE 28th Computer Security Foundations Symposium (CSF), pp. 397–411. IEEE (2015)
49. Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying incentives in the consensus computer. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 706–719. ACM (2015)
50. Luu, L., Velner, Y., Teutsch, J., Saxena, P.: Smart pool: practical decentralized pooled mining. IACR Cryptology ePrint Archive 2017, 19 (2017)

51. Lynch, N.A.: Distributed Algorithms. Elsevier, Amsterdam (1996)
52. Maurer, U.: Modelling a public-key infrastructure. In: Bertino, E., Kurth, H., Martella, G., Montolivo, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 325–350. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61770-1_45
53. Miller, A., Kosba, A., Katz, J., Shi, E.: Nonoutsourceable scratch-off puzzlesto discourage Bitcoin mining coalitions. In: Proceedings of the 22nd ACMSIGSAC Conference on Computer and Communications Security, pp. 680–691. ACM(2015)
54. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of BFT protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–42. ACM (2016)
55. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)
56. Moon, S.B., Skelly, P., Towsley, D.: Estimation and removal of clock skew from network delay measurements. In: INFOCOM 1999 Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 227–234. IEEE (1999)
57. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
58. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22
59. Pass, R., Shi, E.: Thunderella: blockchains with optimistic instant confirmation. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 3–33. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_1
60. Popov, S.: The tangle. cit. on, p. 131 (2016)
61. Ren, L., Nayak, K., Abraham, I., Devadas, S.: Practical synchronous byzantine consensus. arXiv preprint arXiv:1704.02397 (2017)
62. Rosenfeld, M.: Analysis of hashrate-based double spending. arXiv preprint arXiv:1402.2009 (2014)
63. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in Bitcoin. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 515–532. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_30
64. Sompolinsky, Y., Zohar, A.: PHANTOM: a scalable BlockDAG protocol (2018)
65. Sompolinsky, Y., Lewenberg, Y., Zohar, A.: SPECTRE: a fast and scalable cryptocurrency protocol. IACR Cryptology ePrint Archive 2016, 1159 (2016)
66. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in Bitcoin. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 507–527. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47854-7_32
67. Syta, E., et al.: Keeping authorities "honest or bust" with decentralized witness cosigning. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 526–545. IEEE (2016)
68. Szabo, N.: Smart contracts (1994). http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html
69. Teutsch, J., Jain, S., Saxena, P.: When cryptocurrencies mine their own business. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 499–514. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_29
70. Teutsch, J., Reitwießner, C.: A scalable verification solution for blockchains (2017). https://people.cs.uchicago.edu/teutsch/papers/truebitpdf

71. Vasek, M., Thornton, M., Moore, T.: Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 57–71. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_5
72. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Proj. Yellow Pap. **151**, 1–32 (2014)
73. Zamani, M., Movahedi, M., Raykova, M.: RapidChain: scaling blockchain via full sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948. ACM (2018)
74. Das, S., Ribeiro, V.J., Anand, A.: YODA: enabling computationally intensive contracts on blockchains with Byzantine and Selfish nodes. arXiv preprint arXiv:1811.03265 (2018)