

# A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks

Wenbo Wang, *Member, IEEE*, Dinh Thai Hoang, *Member, IEEE*, Peizhao Hu, *Member, IEEE*, Zehui Xiong, *Student Member, IEEE*, Dusit Niyato, *Fellow, IEEE*, Ping Wang, *Senior Member, IEEE* Yonggang Wen, *Senior Member, IEEE* and Dong In Kim, *Fellow, IEEE*

**Abstract**—The past decade has witnessed the rapid evolution in blockchain technologies, which has attracted tremendous interests from both the research communities and industries. The blockchain network was originated from the Internet financial sector as a decentralized, immutable ledger system for transactional data ordering. Nowadays, it is envisioned as a powerful backbone/framework for decentralized data processing and data-driven self-organization in flat, open-access networks. In particular, the plausible characteristics of decentralization, immutability and self-organization are primarily owing to the unique decentralized consensus mechanisms introduced by blockchain networks. This survey is motivated by the lack of a comprehensive literature review on the development of decentralized consensus mechanisms in blockchain networks. In this survey, we provide a systematic vision of the organization of blockchain networks. By emphasizing the unique characteristics of incentivized consensus in blockchain networks, our in-depth review of the state-of-the-art consensus protocols is focused on both the perspective of distributed consensus system design and the perspective of incentive mechanism design. From a game-theoretic point of view, we also provide a thorough review on the strategy adoption for self-organization by the individual nodes in the blockchain backbone networks. Consequently, we provide a comprehensive survey on the emerging applications of the blockchain networks in a wide range of areas. We highlight our special interest in how the consensus mechanisms impact these applications. Finally, we discuss several open issues in the protocol design for blockchain consensus and the related potential research directions.

**Index Terms**—Blockchain, permissionless consensus, Byzantine fault tolerance, mining, incentive mechanisms, game theory, P2P networks.

## I. INTRODUCTION

In the past decade, blockchain networks have gained tremendous popularity for their capabilities of distributively providing immutable ledgers as well as platforms for data-driven autonomous organization. Proposed by the famous grassroots cryptocurrency project “Bitcoin” [1], the blockchain network was originally adopted as the backbone of a public, distributed ledger system to process asset transactions in the form of

digital tokens between Peer-to-Peer (P2P) users. Blockchain networks, especially those adopting open-access policies, are distinguished by their inherent characteristics of disintermediation, public accessibility of network functionalities (e.g., data transparency) and tamper-resilience [2]. Therefore, they have been hailed as the foundation of various spotlight Fin-Tech applications that impose critical requirement on data security and integrity (e.g., cryptocurrencies [3], [4]). Furthermore, with the distributed consensus provided by blockchain networks, blockchains are fundamental to orchestrating the global state machine<sup>1</sup> for general-purpose bytecode execution. Therefore, blockchains are also envisaged as the backbone of the emerging open-access, trusted virtual computers [6] for decentralized, transaction-driven resource management in communication networks and distributed autonomous systems [5], [7]. For these reasons, blockchain technologies have been heralded by both the industry and academia as the fundamental “game changer” [8] in decentralization of digital infrastructures ranging from the financial industry [4] to a broad domain including Internet of Things (IoTs) [9] and self-organized network orchestration [10].

Generally, the term “blockchain networks” can be interpreted from two levels, namely, the “blockchains” which refer to a framework of immutable data organization, and the “blockchain networks” on top of which the approaches of data deployment and maintenance are defined. The two aspects are also considered as the major innovation of blockchain technologies. For data organization, blockchain technologies employ a number of off-the-shelf cryptographic techniques [11]–[13] and cryptographically associate the users’ on-chain identities with the transactions of their tokenized assets. Thus, blockchains are able to provide the proofs of authentication for asset (i.e., token) transfer and then the proofs of asset ownerships. Furthermore, a blockchain maintains an arbitrary order of the transactional records by cryptographically chaining the record subsets in the form of data “blocks” to their chronic predecessors. With the help of cryptographic references, any attempt of data tampering can be immediately detected. From the perspective of network organization, the problem of replicated agreement [14], [15] on a single/canonical transaction history among trustless nodes is creatively tackled by the blockchain consensus protocols in an open-access, weakly synchronized network. Blockchain consensus protocols are

Wenbo Wang, Zehui Xiong, Dusit Niyato and Yonggang Wen are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (email: wbwang@ntu.edu.sg, zxiong002@e.ntu.edu.sg, dniyato@ntu.edu.sg, ygwen@ntu.edu.sg).

Dinh Thai Hoang is with the Faculty of Engineering and Information Technology, University of Technology Sydney, NSW 2007, Australia (e-mail: hoang.dinh@uts.edu.au).

Peizhao Hu is with the Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA 14623 (email: ph@cs.rit.edu).

Ping Wang is with the Department of Electrical Engineering & Computer Science, Lasseonde School of Engineering, York University, 4700 Keele St., LAS 2016 Toronto, ON M3J 1P3, Canada (email: pingw@yorku.ca).

Dong In King is with the School of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, Korea (e-mail: dikim@skku.ac.kr).

<sup>1</sup>Distributed consensus orchestrates the states of replicated program execution on decentralized nodes. It provides the runtime environment for distributively verifying the output of the same program. Therefore, the blockchain network is also known as a distributed Virtual Machine (VM) in the literature [5].

able to offer the agreement on the global blockchain-data state among a large number of trustless nodes with no identity authentication and low messaging overhead [16]. To achieve this, a number of blockchain networks, e.g., Bitcoin, choose to incorporate an incentive-based block creation process known as “block mining” in their protocols. With distributed consensus, the blockchain can be viewed as a universal memory of the blockchain network. Meanwhile, the blockchain network can be viewed as a virtual computer (i.e., distributed VM) comprised by every node therein.

With the rapid evolution in blockchain technologies, the demand for the higher-level quality of services by blockchain-based applications presents more critical challenges in designing blockchain protocols. Particularly, the performance of blockchain networks significantly relies on the performance of the adopted consensus mechanisms, e.g., in terms of data consistency, speed of consensus finality, robustness to arbitrarily behaving nodes (i.e., Byzantine nodes [15]) and network scalability. Compared with the classical Byzantine consensus protocols allowing very limited network scalability in distributed systems [15], [17], most of the existing consensus protocols in open-access blockchain networks (e.g., Bitcoin) guarantee the better network scalability at the cost of limited processing throughput. Also, to achieve decentralized consensus among poorly synchronized, trustless nodes, a number of these protocols incur huge consumption of physical resources (e.g., computing power) [3]. Moreover, to ensure a high probability of consensus finality, the protocols may also impose high latency for transaction confirmation. Out of such concerns, a large volume of research has been conducted with the aim of improving the performance of the open-access blockchain consensus protocols in specific aspects. However, in spite of a few short surveys [16], [18], a comprehensive study on the development of these consensus protocols and the related problems is still missing. Especially, there is a lack of a concise overview on how such a development can be interpreted under a uniform framework and how it impacts the potential applications of blockchain networks.

During the past decade, the scope of blockchain networks has been expanded way further from tamper-evident distributed ledgers. However, due to the recent market frenzy about cryptocurrencies, most of the existing general reviews and surveys on blockchains emphasize narrowly the scenarios of using blockchain networks as the backbone technologies for cryptocurrencies, especially the market-dominant ones such as Bitcoin and Ethereum [2]–[5], [18]–[21]. For example, the issues regarding the client (user)-side application (i.e., wallet), P2P network protocols, consensus mechanisms and user privacy in the scope of Bitcoin are discussed in [3], [4]. In [19], a brief summary of the emerging blockchain-based applications ranging from finance to IoTs is provided. A systematic survey is conducted in [20] with respect to the security in the Bitcoin network including the identified attacks on the consensus mechanisms and the privacy/anonymity issues of the Bitcoin clients. In [5], [21], the special issues regarding

the design, application and security of the smart contracts<sup>2</sup> are reviewed in the context of the Ethereum network. In [7], [16], two brief surveys on consensus protocols in blockchain networks are provided.

The existing surveys on the fast-developing studies of blockchain technologies rarely provide a global view on the issues related to consensus protocols. Our work aims to fill this gap by providing a comprehensive survey on this specific topic. To distinguish our study from the existing works, we present our survey on blockchain networks from the perspective of consensus formation, especially in open-access<sup>3</sup> P2P networks. In analogy to the distributed database, blockchain consensus is perceived as a process of collaborative state transitions among distributed nodes in the framework of blockchain-specified data organization. We emphasize that such a viewpoint brings the taxonomy of blockchain networks into a paradigm that is comparable to the classical problems of global state maintenance in distributed systems [22]. Therefore, we are able to cast our analysis of blockchain networks into the context of classical fault-tolerant studies by focusing on the standard consensus properties in distributed systems (i.e., the Agreement-Validity-Termination properties [22, Chapter 13.1]). We provide a uniform view of blockchain networks by presenting a number of implementation stacks and revealing the interconnection between different protocol components therein. We align our survey on blockchain consensus protocols with a uniform framework based on Zero-Knowledge (ZK) prover-verifier systems [12], [13] in Section III. By focusing on the blockchain protocols for data organization, network organization, and consensus maintenance, our survey contributes in the following aspects:

- (1) providing a brief overview on the data organization and network protocols of blockchain networks,
- (2) providing a generic paradigm for the consensus mechanisms using cryptographic techniques in open-access blockchain networks,
- (3) reviewing the studies on the behaviors of the rational (profit-driven) nodes in the consensus processes of blockchain networks,
- (4) providing an in-depth review on the research effort toward addressing the concerns (e.g., performance vs. scalability) for blockchain networks with different roadmaps of consensus protocol design, and
- (5) providing an outlook of the research in the emerging decentralized applications built on top of the consensus layer, which may not be limited to the framework of the prevalent blockchain technologies (cf. our discussion in Sections III-VI).

The rest of this survey is organized as follows. Section II provides an introductory overview on the protocol organization of blockchain networks. Section III provides an in-

<sup>2</sup>A smart contract is a deterministic program stored as executable bytecode on the blockchain [5], [21]. Its replicas are independently executed in the local VMs/containers on some or all nodes in the network, where the same triggering transactions produce the same output on all the honest nodes.

<sup>3</sup>We consider the property of opens access to all network functionalities instead of only open-access blockchain data. Throughout the survey, we use the terms “opens-access” and “permissionless” interchangeably.

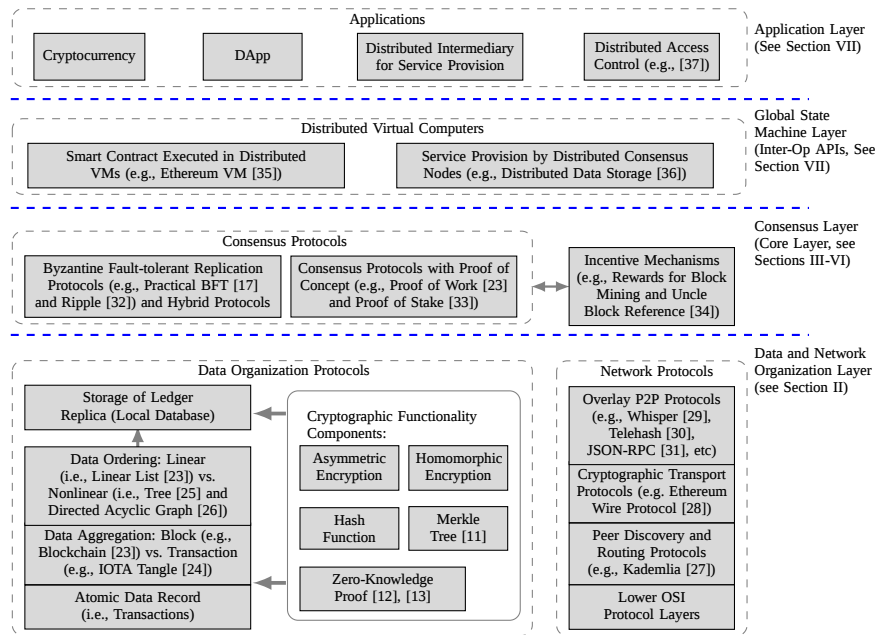


Figure 1. An overview of the blockchain network implementation stacks. The arrow direction indicates the influence on protocol component selection.

depth survey on the popular approaches of consensus protocol design for open-access networks using linear blockchains. Consequently, Section IV provides a survey on the studies of the rational nodes' strategies in these consensus processes and their impact on the performance of blockchain networks. Section V extends our survey on blockchain consensus protocols to the emerging fields including virtual block-mining (i.e., blockchain-extension) mechanism and hybrid consensus. Section VI briefly reviews the emerging cross-layer design regarding the data organization and consensus protocols, namely, the "next-generation blockchains" which may have different roadmaps for scalability and performance other than the prevalent blockchain paradigm. Section VII provides a short review of the emerging applications of blockchains as well as an outlook of the potential research directions in the context of telecommunication networks. Section VIII concludes this survey by summarizing the contributions.

## II. PROTOCOL OVERVIEW AND PRELIMINARIES

### A. Overview of Blockchain Network Protocols

The core task of a blockchain network is to ensure that the trustless nodes in the network reach the agreement upon a single tamper-proof record of transactions. The network is expected to tolerate a portion of the nodes deviating from this canonical record with their local views of data (i.e., replica). From the perspective of system design, a blockchain network can be abstracted into four implementation levels. These levels are the protocols of data and network organization, the protocols of distributed consensus, the framework of autonomous organization relying on smart contracts [5] executed in distributed VMs and the implementation of human-machine interfaces (i.e., application). Following the approach of protocol layer definition in the Open Systems Interconnection (OSI) model, we provide in Figure 1 an overview of these layers in blockchain networks and the related ingredient technologies.

The data organization protocols provide a number of ingredient cryptographic functionalities [11]–[13] to establish unique and secured node identities in a blockchain network. The protocols also define the approaches to form the cryptographic dependence among all the records, e.g., transaction records and account balances, in a local blockchain replica for ordering and tamper proof. From the perspective of data representation, the term "blockchain" is named as such partly for historical reason. In early networks such as Bitcoin [1], the digitally signed transactional records are arbitrarily "packed up" into a cryptographically tamper-evident data structure known as the "block". The blocks are then organized in a chronological order as a "chain of blocks", or more precisely, a linear list of blocks linked by tamper-evident hash pointers. Nevertheless, to improve the processing efficiency, network scalability and security, the linear data organization framework has been expanded into the nonlinear forms such as trees and graphs of blocks [26], [38]. As in linear blockchains, the partial orders are also determined by the chaining direction between blocks. Furthermore, block-less, nonlinear data structures are also adopted in recent protocol design [24]. Despite the different forms of block organization, cryptographic data representation provides the fundamental protection of privacy and data integrity for blockchain networks. When compared with conventional database, it also provides more efficient on-chain storage without harming the data integrity.

On the other hand, the network protocols provide the means of P2P network organization, namely, peer/route discovery and maintenance as well as encrypted data transmission/synchronization over P2P links. Given reliable data synchronization over P2P connections, the consensus layer provides the core functionality to maintain the originality, consistency and order of the blockchain data across the network. From the perspective of distributed system design, the consensus protocols provide Byzantine agreement [15] in blockchain

networks. More specifically, the nodes in the network expect to agree on a common update, i.e., consensus, of the blockchain state that they copy as the local replicas even in the presence of possible conflicting inputs and arbitrary faulty (Byzantine) behaviors of some nodes. When choosing the permissioned access-control schemes of network functionalities, blockchain networks usually adopt the well-studied Byzantine Fault-Tolerant (BFT) consensus protocols such as Practical BFT (PBFT) [17] for reaching the consensus among a small group of authenticated nodes (e.g., HyperLedger Fabric v0.5 [39]). On the contrary, in open-access/permissionless blockchain networks, probabilistic Byzantine agreement is achieved by combining a series of cryptographic techniques, e.g., cryptographic puzzle systems [13], [40], and incentive mechanism design. As pointed out in [18], permissioned consensus protocols rely on a semi-centralized consensus framework and a higher messaging overhead to provide immediate consensus finality and thus high transaction processing throughput. In contrast, permissionless consensus protocols are more appropriate for a blockchain network with loose control on the synchronization and behaviors of the nodes, but may only guarantee probabilistic finality. In the condition of bounded delay and honest majority, permissionless consensus protocols provide significantly better support for network scalability at the cost of a lower processing efficiency.

Provided that the robustness of the consensus protocols is guaranteed, smart contracts are deployed on the distributed virtual computer layer. In brief, this layer abstracts away the details of data organization, information propagation and consensus formation in blockchain networks. As the interoperation layer between the lower-layer protocols and the applications, the virtual computer layer defines the high-level programming language implementation (e.g., Solidity in Ethereum [21]) for encoding smart contracts. It also provides the sandboxed runtime environment (e.g., Ethereum VMs) to ensure the correct execution of the replicated smart contracts on the network level. The virtual computer layer may adopt different levels of Turing-completeness for smart contract implementation, ranging from stateless circuits in Bitcoin [1] to fully Turing-complete state machines in Ethereum [35] and HyperLedger Fabric [39]. Full Turing-completeness enables blockchain networks to perform general-purpose computation in a replicated manner. For this reason, a blockchain network is able to not only provide the services of trusted data recording and timestamping, but also facilitate the functionalities of general-purpose autonomous organization. Therefore, blockchain networks are able to work as the backbone of autonomous organization systems for managing data or transaction-driven interactions among the decentralized entities in the network. On top of the virtual computer layer, the application layer provides the end-user-visible interfaces such as Distributed Applications (DApps) [41], [42] and cryptocurrencies.

## B. Cryptographic Data Organization

When viewed as a data structure, a blockchain can be abstracted as an infinitely-growing, append-only string that is canonically agreed upon by the nodes in the blockchain network [23]. For data organization, the local blockchain replica

of each node is organized in a hierarchical data structure of three levels, namely, the transactions, the blocks and the chain. Each level requires a different set of cryptographic functionalities for the protection of data integrity and authenticity.

1) *Transactions, Addresses and Signatures*: Transactions are the atomic data structure of a blockchain. Generally, a transaction is created by a set of users or autonomous objects (i.e., smart contracts) to indicate the transfer of tokens from the senders to the specified receivers. A transaction specifies a possibly empty list of inputs associating the token values with the identities (i.e., addresses) of the sending users/objects. It also specifies a nonempty list of outputs designating the redistribution result of the input tokens among the associated identities of the receivers. A transaction can be considered as a static record showing the identities of the senders and the receivers, the token value to be redistributed and the state of token reception. To protect the authenticity of a transaction record, the functionalities of cryptographic hashing and asymmetric encryption are activated:

- *Hash Function*: A cryptographic hash function maps at random an arbitrary-length binary input to a unique, fixed-length binary output (i.e., image). With a secure hash function (e.g., SHA-256), it is computationally infeasible to recover the input from the output image. Also, the probability to generate the same output for any two different inputs is negligible.
- *Asymmetric Key*: Each node in the blockchain network generates a pair of private and public keys. The private key is associated with a digital signature function, which outputs a fixed-length signature string for any arbitrary-length input message. The public key is associated with a verification function, which takes as input the same message and the acclaimed signature for that message. The verification function only returns *true* when the signature is generated by the signature function with the corresponding private key and the input message.

The nodes in the network or the autonomous objects identify themselves by revealing their public keys, namely, the hash-code of their public keys, as their permanent addresses (also known as their pseudo-identities) on the blockchain<sup>4</sup>. Since each input tuple in a transaction is signed by the associated sending account, the network is able to publicly validate the authenticity of the input through verifying the signature based on the sender's public address.

2) *Block Organization, Hash Pointer and Merkle Tree*: A block is a container of an arbitrary subset of transaction records and can only be created by a node participating in the consensus process. To protect the integrity of the transaction records and to specify the ordering of adjacent blocks in a consensus node's local view, a data field known as the hash pointer is kept in the block's data structure. In addition, to reduce the on-chain storage, the cryptographic data structure of Merkle tree is also enabled to generate the tamper-evident digest in the transaction set of a block (see Figure 2):

<sup>4</sup>Some cryptocurrency systems (e.g., Monero [43] and ZCash [44]) incorporate cryptographic techniques such as one-time signature and group signature to create ephemeral addresses for enhancing anonymity.

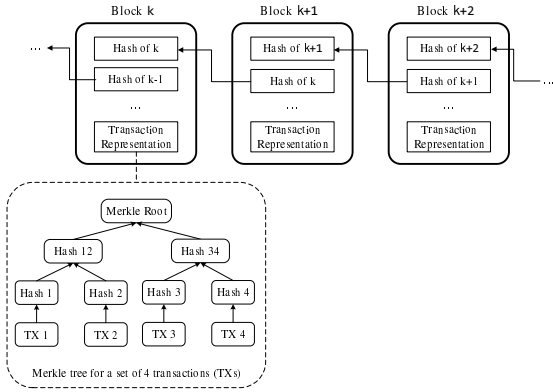


Figure 2. Illustration of a chain of blocks, where the transactions in a single block is represented by a Merkle root.

- **Hash pointer:** A hash pointer to a block is the hashcode of the concatenated data fields in that block. The hashcode of the current block is stored as the header of that block. The hashcodes of the reference blocks are stored as the hash pointers of a block to indicate that at the local view, the block recognizes that the transactions in the reference blocks are created earlier than those in the current block.
- **Merkle Tree [11]:** A Merkle tree represents a transaction set in the form of a binary tree. Therein, each leaf is labeled with the hashcode of a transaction and a non-leaf node is labeled with the hashcode of the concatenated labels of its two child nodes. The root node of the Merkle tree is known as the Merkle digest/root. A block storing only the Merkle root of the selected transactions is known to be in a lightweight form, which is sufficient for quick validation and synchronization. When using the lightweight-form storage, the node has to query its peers to retrieve the complete transaction records in the blocks.

In addition to the Merkle digest, block header and the hash pointers, a block may also contain auxiliary data fields, whose definition varies with the adopted protocol of block generation based on different consensus schemes. At a local view of the blockchain, the blocks are organized based on the hash pointers to their references/predecessors. Every blockchain admits a unique block with no reference as the “genesis block”, namely, the common ancestor block of all valid blocks in the chain. According to the number of hash pointers to the predecessors that are allowed to be kept by a block, the block organization can vary from a linear linked list to a tree of blocks (e.g., GHOST [25]) or a Directed Acyclic Graph (DAG) (e.g., SPECTRE [26]). Without specification, we limit most of our discussion on blockchains to the linear-list case, where the total order of the blocks is guaranteed (see Figure 2).

### C. Blockchain Networks

In a Byzantine environment, the identity management mechanism plays a key role in determining how the nodes in a blockchain network are organized. In an open-access (i.e., public/permissionless) blockchain network, a node can freely join the network and activate any available network functionalities. Notice that the term “node” refers to a logical entity (i.e., the identity of a blockchain user) rather than to a physical device. For example, multiple “nodes” associated

with different network functionalities can be hosted on the same physical machine. In alternative words, a physical device may appear in multiple identities in the network. Without any authentication scheme, the nodes are organized as overlay P2P networks. Comparatively, in a consortium (i.e., permissioned) blockchain network, only the authorized nodes are allowed to enable the core functionalities such as consensus participation or data propagation. The authorized nodes may be organized in different topologies, e.g., fully connected networks or P2P networks, according to the consensus protocols that the networks adopt. In this paper, we mainly focus on the network protocols in the permissionless cases.

In permissionless blockchain networks, the main goal of the network protocol is to induce a random topology among the nodes and propagate information efficiently for blockchain replica synchronization. Most of the existing blockchain networks employ the ready-to-use P2P protocols with slight modification for topology formation and data communication. For peer discovery and topology maintenance, the nodes in Bitcoin-like blockchain networks rely on querying a hard-coded set of volunteer DNS servers, which return a random set of bootstrapping nodes’ IP addresses for the new nodes to initialize their peer lists [45], [46]. Nodes then request or advertise addresses based on these lists. In contrast, the Ethereum-like networks adopt a Kademlia-inspired protocol based on Distributed Hash Tables (DHTs) [27] for peer/route discovery<sup>5</sup> through UDP connections. In blockchain networks, the connection of a node to a peer is managed based on reputation using a penalty score. A node will increase the penalty score of the peer sending malformed messages until the IP address of the faulty node is locally banned [28], [46].

To replicate the blockchain over all nodes in the network, the messages of transactions and blocks are “broadcast” through flooding the P2P links in a gossip-like manner. Typically, a P2P link in blockchain networks is built upon a persistent TCP connection after a protocol-level three-way handshake, which exchanges the replica state and the protocol/software version of each node [28], [47]. After the connections to the peer nodes are established, another three-way handshake occurs for a node to exchange new transactions/blocks with its neighbors. The node first notifies its peers with the hashcode of the new transactions/blocks that it receives or generates. Then, the peers reply with the data-transfer request specifying the hashcode of the information that they need. Upon request, the transfer of transactions/blocks is done via individual transfer messages<sup>6</sup>. The data transfer in blockchain networks is typically implemented based on the HTTP(s)-based Remote Procedure Call (RPC) protocol, where the messages are serialized following the JSON protocol [28].

An open-access blockchain network does not explicitly specify the role of each node. Nevertheless, according to the enabled functionalities, the nodes in the network can be categorized as the lightweight nodes, the full nodes and the consensus nodes [48]. Basically, all nodes are required to enable

<sup>5</sup>Kademlia measures the node distance using XOR distance of the node addresses (hash values). The  $k$ -closest nodes are selected as neighbors.

<sup>6</sup>For example, the details of handshake and synchronization in the Ethereum network are defined in the DEVp2p Wire Protocol [28].

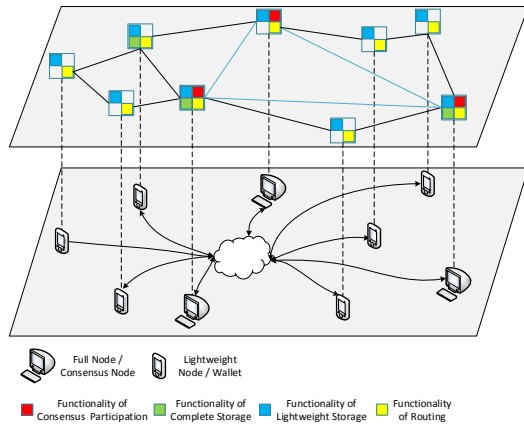


Figure 3. Illustration of the nodes' roles in a permissionless blockchain network. The P2P links between consensus nodes are shown in blue.

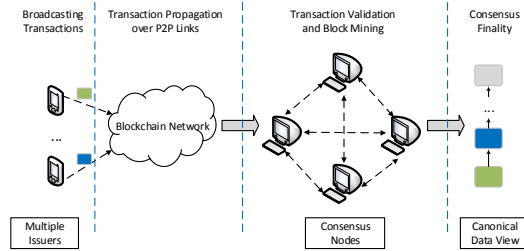


Figure 4. The life cycle of blockchain transactions. Note that transaction validation and blockchain mining may happen at the same time with transaction propagation, depending on the consensus protocol adopted by the blockchain.

the routing functionality for message verification/propagation and connection maintenance. A lightweight node (e.g., wallets) only keeps the header of each block in its local storage. A full node stores locally a complete and up-to-date replica of the canonical blockchain. Compared with the lightweight nodes, a full node is able to autonomously verify the transactions without external reference. A consensus node enables the functionality of consensus participation. Therefore, it is able to publish new blocks and has a chance to influence the state of the canonical blockchain. A consensus node can adopt either complete storage or lightweight storage. In Figure 3, we present an example of different node types in a public blockchain network. Meanwhile, the lifecycle of a new transaction is shown in Figure 4. It is worth noting that the consensus nodes are often referred to as the “miners” or “mining nodes” of blocks in the context of blockchain consensus formation, especially when token rewards of block proposal are involved. Meanwhile, different roles of nodes lead to the inconsistency in their interests. Namely, the transaction-issuing nodes (e.g., lightweight nodes) may not be the transaction-approving nodes (i.e., consensus nodes). For this reason, caution needs to be taken in protocol design to ensure that the consensus nodes act on behalf of the others in a trustless environment, especially on the consensus layer.

#### D. Consensus in Blockchain Networks

In the context of distributed system, the issue of maintaining the canonical blockchain state across the P2P network can be mapped as a fault-tolerant state-machine replication problem [14]. In other words, each consensus node maintains a local replicate (i.e., view) of the blockchain. An agreement

(i.e., consensus) on the unique common view of the blockchain is expected to be achieved by the consensus nodes in the condition of Byzantine/arbitrary failures<sup>7</sup>. In blockchain networks, Byzantine failures cause faulty nodes to exhibit arbitrary behaviors including malicious attacks/collusions (e.g., Sybil attacks [49] and double-spending attacks [20]), node mistakes (e.g., unexpected blockchain fork due to software inconsistency [50]) and connection errors. We can roughly consider that the sequence of blocks represents the blockchain state, and the confirmation of a transaction incurs a blockchain state transition. According to [14], [51], a blockchain updating protocol is said to achieve the (probabilistic) consensus (a.k.a. atomic broadcast<sup>8</sup> [14], [52], [53]) in a Byzantine environment if the following properties are (probabilistically) satisfied [16]:

- *Validity (Correctness)*: If all the honest nodes activated on a common state propose to expand the blockchain by the same block, any honest node transitioning to a new local replica state adopts the blockchain headed by that block.
- *Agreement (Consistency)*: If an honest node confirms a new block header, then any honest node that updates its local blockchain view will update with that block header.
- *Liveness (Termination)*: All transactions originated from the honest nodes will be eventually confirmed.
- *Total order*: All honest nodes accept the same order of transactions as long as they are confirmed in their local blockchain views.

The consensus protocols vary with different blockchain networks. Since the permissioned blockchain networks admit tighter control on the synchronization among consensus nodes, they may adopt the conventional Byzantine Fault-Tolerant (BFT) protocols (c.f., the primitive algorithms described in [54], [55]) to provide the required consensus properties. A typical implementation of such protocols can be found in the Ripple network [32], where a group of synchronized Ripple servers perform blockchain expansion through a voting mechanism. Further, if an external oracle is introduced to designate the primary node for block generation (e.g., with HyperLedger Fabric v0.5 [39]), Practical BFT (PBFT) [17] can be adopted to implement a three-phase commit scheme for blockchain expansion. In a network of  $N$  consensus nodes, the BFT-based protocols are able to conditionally tolerate  $\lfloor \frac{N-1}{5} \rfloor$  (e.g., [32]) to  $\lfloor \frac{N-1}{2} \rfloor$  (e.g., [56]) faulty nodes.

On the contrary, permissionless blockchain networks admit no identity authentication or explicit synchronization schemes. Therefore, the consensus protocol therein is expected to be well scalable and tolerant to pseudo identities and poor synchronization. Since any node is able to propose the state transition with its own candidate block for the blockchain header, the primary goal of the consensus protocol in permissionless networks is to ensure that every consensus node adheres to the “longest chain rule” [3]. Namely, when the blocks are organized in a linked list, at any time instance, only the longest chain can be accepted as the canonical state of the blockchain.

<sup>7</sup>See [15], [17] for the formal definition of Byzantine failures.

<sup>8</sup>Here, the semantic of “broadcast” is consistent with that in the context of distributed system/database. Namely, a message is atomically broadcast when it is either received by every nonfaulty node, or by none at all.

Due to the lack of identity authentication, the direct voting-based BFT protocols do not fit in permissionless blockchain networks. Instead, the incentive-based consensus schemes such as the Nakamoto consensus protocol [1] are widely adopted.

#### E. Nakamoto Consensus Protocol and Incentive Compatibility

To jointly address the problems of pseudonymity, scalability and poor synchronization, Nakamoto proposed in [1] a permissionless consensus protocol based on a framework of cryptographic block-discovery racing game. This is also known as the Proof of Work (PoW) scheme [2], [3]. From a single node's perspective, the Nakamoto consensus protocol defines three major procedures, namely, the procedure of chain validation, the procedure of chain comparison and extension and the procedure of PoW solution searching [23]. The chain validation predicate provides a Boolean judgment on whether a given chain of blocks has the valid structural properties. It checks if each block in the chain provides valid PoW solution and no conflict between transactions as well as the historical records exists. The function of chain comparison and extension compares the length of a set of chains, which may be either received from peer nodes or locally proposed. It guarantees that an honest node only adopts the longest proposal among the candidate views of the blockchain. The function of PoW solution searching is the main “workhorse” of the protocol and defines a cryptographic puzzle-solving procedure in a computation-intensive manner.

In brief, PoW solution requires exhaustively querying a cryptographic hash function for a partial preimage generated from a candidate block, whose hashcode satisfies a pre-defined condition. For simplicity of exposition, let  $\mathcal{H}(\cdot)$  denote the hash function and  $x$  denote the binary string assembled based on the candidate block data including the set of transactions (e.g., Merkle root), the reference hash pointers, etc. Then, we can formally define the PoW puzzle and solution as follows:

**Definition 1.** Given an adjustable hardness condition parameter  $h$ , the process of PoW puzzle solution aims to search for a solution string, *nonce*, such that for a given string  $x$  assembled based on the candidate block data, the hashcode (i.e., the target block header  $bh$ ) of the concatenation of  $x$  and *nonce* is smaller than a target value  $D(h)$ :

$$bh = \mathcal{H}(x||nonce) \leq D(h), \quad (1)$$

where for some fixed length of bits  $L$ ,  $D(h) = 2^{L-h}$ .

The Nakamoto protocol is computation-intensive since to win the puzzle solving race, a node needs to achieve a hash querying rate as high as possible. This property financially prevents the Sybil attacks of malicious nodes by merely creating multiple pseudo identities. On the other hand, the economic cost (mainly electricity consumption) also renders it impractical for any node to voluntarily participate the consensus process at a consistent economic loss. To ensure proper functioning of a permissionless blockchain network, the Nakamoto protocol introduces incentives to probabilistically award the consensus participants based on an embedded mechanism of token supply and transaction tipping [1]. From

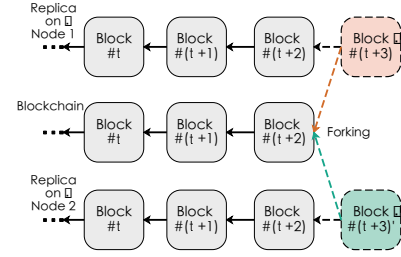


Figure 5. A (temporary) fork happens at nodes 1 and 2 when their local PoW processes lead to different proposals of the new blockchain header, i.e.,  $(t+3)$  and  $(t+3)'$  at the same time. Both  $(t+3)$  and  $(t+3)'$  satisfy (1).

a game theoretic point of view, an implicit assumption adopted by the Nakamoto consensus protocol is that all the participant nodes are individually rational [57]. In return, the consensus mechanism is expected to be *incentive compatible*. In other words, the consensus protocol should ensure that any consensus node will suffer from final loss whenever it deviates from truthfully following the protocol.

However, the incentive compatibility of the Nakamoto protocol has been openly questioned [58]–[61]. Since the Nakamoto protocol allows nodes to propose arbitrary blocks from their local pending transaction set, it is inevitable for the network to experience blockchain expansion race with a (temporary) split, i.e., fork, in the local views of the blockchain state [3], [20] (see Figure 5). To guarantee the consensus properties and thus convergence to one canonical blockchain state, the Nakamoto protocol relies on the assumption that the majority of the consensus nodes follow the longest chain rule and are altruistic in information forwarding. It has been found in [58], [62] that rational consensus nodes may not have incentive for transaction/block propagation. As a result, the problem of blockchain forking may not be easily resolved in the current framework of the Nakamoto protocol. Special measures should be further taken in the protocol design, and a set of folklore principles has been suggested to gear the consensus mechanism towards a protocol for secured and sustainable blockchain networks [4], [63]–[65]:

- The consensus mechanism should enforce that propagating information and extending the longest chain of block are the monotonic strategies of the consensus nodes [65]. In other words, all the sub-stages in the consensus process should be incentive-compatible in an open environment with the tolerance to Byzantine and unfaithful faults.
- The consensus mechanism should encourage decentralization and fairness. Namely, it should not only discourage coalition, e.g., botnets and mining pools [23], [66], but also make the consensus process an uneasy prey of the adversaries with cumulated computation power.
- The consensus mechanism should strike a proper balance between processing throughput and network scalability [53], [67].

### III. DISTRIBUTED CONSENSUS MECHANISMS BASED ON PROOF OF CONCEPTS

Based on the technical components of permissionless blockchain networks introduced in Section II, now we are ready to review the details about the designing methodologies of the consensus protocol for permissionless blockchains. In

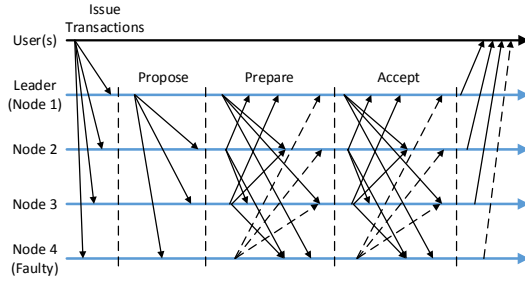


Figure 6. BFT-based message pattern of three-way handshake in permissioned blockchains, e.g., Hyperledger Fabric using BFT-SMaRt [69]. The message is formed based on the granularity level of blocks, i.e., a batch of transactions.

this section, we start by presenting the consensus protocols in the most prevalent blockchain networks in a uniform framework. Then, we explore the different approaches of extending/modifying the protocol to meet a series of specific performance requirement.

#### A. Permissionless Consensus via Zero-Knowledge Proofs

For traditional BFT consensus protocols, e.g., Byzantine Paxos [68] and PBFT [17], it is generally necessary to assume a fully connected topology among the consensus nodes as well as a leader-peer hierarchy for block proposal. The BFT consensus process is organized explicitly in rounds of three-way handshakes, thus synchronization between nodes with bounded execution time and message latency is also required. As illustrated in Figure 6, only the leader is responsible for proposing new blocks to a consortium of peer nodes at the proposal (pre-prepare) phase. This is followed by two all-to-all messaging phases, where a peer node only accepts the proposal (i.e., commit) when it receives more than a certain number of proposal approvals from the other peers (e.g.,  $\lfloor \frac{n+f+1}{3} \rfloor$  with PBFT for a network of  $n$  honest nodes and  $f$  Byzantine nodes). These classical state-machine replication approaches guarantee the properties of deterministic agreement and liveness in Byzantine environment, and are well-known for their low processing latency [18]. However, the characteristics of leader-peer hierarchy and high communication complexity in  $\Theta(n^2)$  [68] naturally require the BFT-based blockchain consensus protocols to be implemented in a small-scale permissioned network with centralized admission control. In order to achieve full decentralization and high consensus scalability, alternative approaches such as Nakamoto protocols become critical in the design of blockchain's consensus layer.

According to our discussion in Section II-E, the primary functionality of PoW in the Nakamoto protocol is to simulate the leader election in the traditional BFT protocols. The PoW process abstracted by Definition 1 is essentially a verifiable process of weighted random coin-tossing, where the probability of winning is no longer uniformly associated with the nodes' identities but in proportion to the resources, e.g., hashrate casted by the nodes. Then, we can consider that each new block is generated by a time-independent "lottery", where the probability of being elected as the leader for block proposal depends on the ratio between the casted resource of a node (or a node coalition) and the total resources presented in the entire network. Let  $w_i$  denote the resource held by node  $i$  in a

network of node set  $\mathcal{N}$ , then, the probability of node  $i$  winning the leader-election in a PoW-like process should follow:

$$\Pr_i^{\text{win}} = \frac{w_i}{\sum_{j \in \mathcal{N}} w_j}, \quad (2)$$

where  $w_i$  generalizes the share of any verifiable resource such as computational power [1], memory [44], storage [70], etc. In contrast to the BFT protocols, the peer nodes accept the received block proposal following the longest-chain-rule therein. Since no all-to-all messaging phase is needed, the Nakamoto protocol may have a much smaller message complexity  $\Omega(n)$  when the majority of the peers are honest [54].

As the core component of the Nakamoto protocol, the PoW scheme originates from the idea of indirectly validating nodes' identities in pseudonymous P2P networks through an identity pricing mechanism [71], [72]. More specifically, the PoW scheme described by Definition 1 is originally designed to measure the voting power or the trustworthiness of a node according to the constrained resources presented by the node in the P2P network. Thus, the tolerable fraction of Byzantine nodes in BFT protocols is replaced by a limited fraction of the total computational power of the network [72]. Compared with the original design, the PoW scheme in blockchain networks is no longer used for direct identity verification between peers. Instead, the PoW processes of all the nodes in a blockchain network are expected to collectively simulate a publicly verifiable random function to elect the leader of block proposal following the distribution given by (2). Based on such a design paradigm, PoW can be generalized into the framework of Proof-of-Concepts (PoX) (cf. [3]). With PoX, the nodes in the network are required to non-interactively prove the possession or commitment of certain measurable resources beyond hashrates in PoW. Furthermore, their collective behavior should also yield a stochastic process for leader assignment following the distribution given in (2).

From a network-level perspective, PoX generally relies on a pseudorandom oracle to provide the property of verifiable unpredictability. It also needs to implement a one-way cryptographic puzzle for the proof of resource devoting in the framework of non-interactive ZK Proofs (ZKPs). A conventional ZKP system consists of two parties, namely, the prover executing a computationally unbounded strategy to generate the proof of an assertion without releasing it and the verifier executing a probabilistic polynomial-time strategy to verify it. A party is non-interactive when it can only choose between publishing messages to the network and remaining passive. Otherwise it is interactive. In the context of blockchain consensus protocols, the ZKP framework is extended from proving a private input (i.e., knowledge) to proving possession/consumption of a minimum amount of resource (e.g., computational work). Recent studies haven shown that with specific puzzle design, proof of knowledge and proof of work can be incorporate into a single framework of indistinguishable Proofs of Work or Knowledge (PoWoK) [40], where the prover of work makes calls to a certain puzzle solving algorithm instead of sampling from a non-polynomial language witness relation distribution. In general, the adopted puzzle

has to satisfy the basic soundness and completeness properties [12], [13]. Namely, an invalid proof should always be rejected by nonfaulty verifying nodes while a valid proof should always be accepted by nonfaulty verifiers. A complexity gap is expected such that the puzzle is easy to verify (in polynomial-time) but (moderately) hard for adversaries to invert/solve [73]. Furthermore, in permissionless blockchain networks, any node is able to publish arbitrary block proposals. In this situation, a 3-step interactive prover-verifier ZK scheme with verifier-designated challenges will lead to excessive message overhead. This is the critical reason for requiring a non-interactive puzzle design. Following the generation-computation-verification paradigm of non-interactive puzzles (cf. the verifiable random function defined in [74]), we can abstract a PoX process into the three stages described in Table I.

Table I  
THREE-STAGE ABSTRACTION OF A POX PROCESS

|                                                   |                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Initialization (generator of random seed or keys) | The <i>initialization stage</i> provides the prover and the verifier the necessary information to run in subsequent stages according to the PoX specifications. Typical non-interactive ZKP systems, e.g., zk-SNARK [75] have to query a trusted third-party key/random seed generation protocol to produce a common reference string for both the prover and the verifier. |
| Execution (challenge and proof generator)         | For non-interactive ZKP, the <i>execution stage</i> requires the prover to generate according to the common reference string a random challenge that constitutes a self-contained, uncompromisable computational problem, namely, the puzzle. Meanwhile, a corresponding proof (a.k.a. witness or puzzle solution) is also generated.                                       |
| Verification                                      | In the <i>verification stage</i> , a verifier checks about the proof's correctness, which is determined solely based on the information issued by the prover.                                                                                                                                                                                                               |

With the paradigm of PoX described above, we are now ready to investigate the puzzle design problem for different PoX schemes, which can be seen as modification or extension to the existing PoW-based Nakamoto protocol (see [36], [76]–[79] for examples). Since a trusted third party does not exist in a permissionless blockchain network, special caution should be taken in the puzzle design such that the freshness of the puzzle is guaranteed at the execution stage. Namely, the puzzle solution is unpredictable and the proof is non-reusable. Theoretical analyses of blockchain networks, e.g., [77] may assume such a property on the condition that the network has access to a universal random sampler (a.k.a., random oracle) or an ideal randomness beacon<sup>9</sup>. Nevertheless, due to full decentralization of the permissionless blockchain networks, a case-by-case study for different PoX schemes is usually needed for practical implementation of the random oracle in order to prevent puzzle grinding and leader election manipulation. Apart from the aforementioned properties of non-invertibility, completeness, soundness and freshness, the other requirements for puzzle design in PoX may include but are not limited to the following:

- The puzzle should be resistant to the aggregation [81] or outsourcing [82] of the computational resources.
- The puzzle-solving process should be eco-friendly [33], [76], [78], [79], [83].

<sup>9</sup>The concept of random beacon service is first proposed in [80], where a trusted third party periodically emits random integers to the public.

- In addition to providing incentive based on resource pricing mechanism, the puzzle-solving process should provide useful services in the meanwhile [36], [84].

## B. Nakamoto Protocol Based on Primitive Proof of Work

As we have reviewed in the previous discussion, the primitive PoW scheme proposed in [1] works to financially disincentivize the Sybil attacks on block proposal and maintains a biased random leader election process in proportion to the hashrate casted by each node. Recall that the input string  $x$  to the PoW puzzle is a concatenation of the previous block's hash pointer and the payload data of the proposed block. For the puzzle design of PoW, the reason of choosing the hash function  $\mathcal{H}(\cdot)$  in (1), e.g., SHA-256 in practice lies in the fact that a hash function is computationally indistinguishable from a pseudorandom function, if it preserves the properties of collision resistance<sup>10</sup> and pre-image resistance [85]. Since the random output of  $\mathcal{H}(\cdot)$  is time-independent and only determined by the input string, it plays the role of an uncompromisable random oracle and outputs a unique, unpredictable result every time when it is queried with a different  $x$  [86]. This means that a node in the blockchain network is able to construct a fresh random challenge solely based on its block proposal without referring to any designated verifier or third-party initializer. Meanwhile, it is well-known that with a proper cryptographic hash function, the search for a preimage  $(x, \text{nonce})$  satisfying the condition  $\mathcal{H}(x \parallel \text{nonce}) \leq 2^{L-h}$  in (1) cannot be more efficient than exhaustively querying the random oracle for all  $\text{nonce} \in [0, 2^L]$ . This leads to a puzzle time complexity of  $\mathcal{O}(2^h)$  [64]. On the other hand, verifying the puzzle only requires a single hash query. Therefore, the properties of non-invertibility, completeness, soundness and freshness are all satisfied by the PoW puzzle given by Definition 1.

For a given difficulty level  $D(h)$  in (1), each single query to  $\mathcal{H}(\cdot)$  is an i.i.d. Bernoulli trial with a success probability

$$\Pr(y : \mathcal{H}(x \parallel y) \leq D(h)) = 2^{-h}. \quad (3)$$

We adopt the typical assumption of loosely network synchronization for analyzing PoW-based blockchains [23], [86]. Namely, all messages are delivered with bounded delay in one round. Then, (3) indicates that the frequency for a node to obtain the puzzle solutions during a certain number of loosely synchronized rounds is a Bernoulli process. Since the probability given in (3) is negligible for a sufficiently large  $h$  with cryptographic hash functions  $\mathcal{H}(\cdot)$ , the Bernoulli process of node  $i$  converges to a Poisson process as the time interval between queries/trails shrinks [54].

To analyze the PoW scheme, let  $w_i$  in (2) refer to the number of queries that node  $i$  can make to  $\mathcal{H}(\cdot)$  in a single round. Then, we can approximate the rate of the Poisson process for node  $i$ 's puzzle solution by  $\lambda_i = w_i/2^h$  [87]. Note that every node in the network is running an independent puzzle-solving process. Since a combination of  $N$  independent Poisson processes is still a Poisson process, then, the collective

<sup>10</sup>The collision probability of  $\mathcal{H}(\cdot)$  is  $e^{-\Omega(L)}$  and thus negligible [23].

PoW process of a network with  $N$  nodes has a rate

$$\lambda = \sum_{i=1}^N \lambda_i = \frac{\sum_{i=1}^N w_i}{2^h}. \quad (4)$$

The property of the combined Poisson processes in (4) leads to the probability distribution for leader election in (2). From a single node's perspective, the repeated PoW puzzle-solving processes take the form of a block-proposal competition across the network. From the perspective of the network, for a given difficulty level  $D(h)$ , this puzzle-solving race simulates a verifiable random function for leader election and guarantees to follow the distribution in (2). Most importantly, it tolerates any fraction of the Byzantine nodes in the network.

Nevertheless, the PoW by itself cannot guarantee any of the principle Byzantine consensus properties as described in Section II-D. On top of the designed PoW puzzle and the P2P information diffusion functionality, three external functions are abstracted in [23] to describe the Nakamoto consensus protocol from a single node's perspective. These functions are

- 1) the *chain reading function* that receives as input a blockchain and outputs an interpretation for later use;
- 2) the *content validation function* that validates a blockchain replica and checks the data consistency with the applications (e.g., Bitcoin) on top of the blockchain;
- 3) the *input contribution function* that compares the local and the received views of the blockchain and adopts the "best" one following the rule of longest chain.

The input contribution function realizes the puzzle execution stage and the content validation function realizes the puzzle verification stage in Table I. Due to the independent Poisson processes in the block-proposal competition, more than one node may propose to extend the blockchain using different blocks with corresponding valid PoW solutions at the same time. As a result, the nodes may read from the network multiple valid views of the blockchain and choose different forks as their "best" local views (see also Figure 5). Theoretically, it has been shown in [88] that deterministic consensus in permissionless blockchain networks cannot be guaranteed unless all non-faulty nodes are reachable from one to another and the number of consensus nodes is known. For this reason, in [23], [86], [89], Garay et al. propose to capture the properties of validity, agreement and liveness of the Nakamoto consensus protocol by the three chain-based properties in Table II. Then, the PoW-based Nakamoto protocol can be modeled as a probabilistic Byzantine agreement protocol.

In order to quantify the Byzantine agreement properties for blockchains, three conditions, i.e., the upper-bounded information diffusion delay, a "flat network" with equal and limited hashrates and the upper-bounded number of Byzantine nodes are assumed in [23], [86], [89]. It is shown in [23] that the three properties in Table II are quantified by three parameters, namely, the collective hashrates of the honest nodes, the hashrate controlled by the adversaries and the expected block arrival rate of the network-level Poisson process given in (4). It has been further proved in [23] that under the condition of honest majority, the basic properties of validity and agreement are satisfied by the Nakamoto protocol with overwhelming

Table II  
THREE PROPERTIES OF NAKAMOTO PROTOCOLS FOR BLOCKCHAINS

| Nakamoto Protocol-Specified Properties | Corresponding Properties of Byzantine Agreement | Explanation in Details                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Common-prefix property                 | Agreement (and permanent order)                 | In the condition of multiple local blockchain views due to forking, the <i>common-prefix property</i> indicates that after cutting off (pruning) a certain number of block from the end (header) of the local chain, an honest node will always obtain a sub-chain that is a prefix of another honest node's local view of the blockchain. |
| Chain-quality property                 | Validity                                        | Among a given length of consequent blocks in the local blockchain view of an honest node, the number of blocks that is proposed by Byzantine nodes (adversaries) is upper-bounded.                                                                                                                                                         |
| Chain-growth property                  | Liveness                                        | For any given rounds of block proposals, the number of blocks appended to the local view of any honest node is lower-bounded.                                                                                                                                                                                                              |

probability. Furthermore, the common-prefix property and the chain-growth property formalize the presumption in [1] that a transaction is secured when a sufficient length of subsequent blocks is appended to the chain. In other words, when a block is a certain number of blocks deep from the end of the chain, or equivalently, the repeated block-proposal competition has passed sufficiently many rounds, the transaction data in that block is non-reversible/persistent and thus guaranteed to be double-spending proof. It is worth noting that the studies in [23], [89] provide a generalizable approach for evaluating the security and the efficiency of the PoX-based Nakamoto protocols in permissionless blockchains. Based on the quantitative analysis of the properties in Table II, the same framework of security evaluation has been adopted by the studies in consensus protocols using other types of puzzle design such as Proof of Stakes (PoS) [77], [90].

Due to the open access nature of permissionless blockchains, the hashrate presented in a practical blockchain network is generally unstable. As indicated by Figure 7, since the introduction of the Application Specific Integrated Circuit (ASIC) for hash acceleration in 2013, the practical PoW-based blockchain networks, e.g., Bitcoin, have experienced an explosive increase of the total hashrate with huge fluctuation [91]. Practically, blockchain networks adopt a heuristic, periodic difficulty-adjustment policy to maintain a roughly fixed time interval, i.e.,  $\lambda^{-1}$  in (4), between two neighbor blocks. However, the expected value of  $\lambda^{-1}$  is usually chosen in an arbitrary manner and is frequently reduced in favor of a higher transaction throughput (see Litecoin [92] and ZCash [44] for example). Following the assumption of partial synchronization [23], the roughly fixed time interval indeed implies an upper bound for the information dissemination latency in the P2P network [93].

With such a consideration in mind, a theoretical study is provided in [94] between the upper bound of the information latency and the persistence of the block data in a node's local view of the blockchain. Consider a flat network of  $N$  nodes with a maximum block propagation delay of  $T$ . It is found in [94] that for a given fraction of adversary node  $\rho$  ( $0 \leq \rho < 0.5$ ), the block generation probability for each node should satisfy the following condition in order to ensure the property

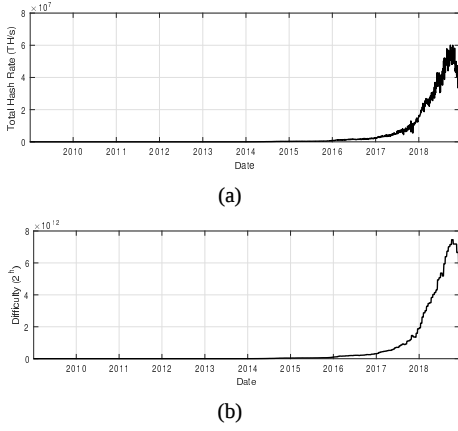


Figure 7. Evolution of (a) the total hash rate and (b) the PoW puzzle difficulty in the Bitcoin network over time. Data source: <https://www.blockchain.com>.

of data persistence (Theorem 1.1 in [94]):

$$\mathbf{Pr}_i^g \leq \frac{1}{T\rho \sum_{i=1}^N w_i}, \quad (5)$$

where  $\mathbf{Pr}_i^g$  can be calculated based on (3) and a given hashrate.

Furthermore, the block interval rules the trade-off between security and efficiency. The formal refers to the degree of fulfillment (i.e., the probabilistic consistency) of the Byzantine agreement properties, whereas the latter refers to the transaction throughput, which can be measured in the number of confirmed transactions per second. In [45], [93], examination on the block propagation delay  $T$  in (5) shows that a safe upper bound on  $T$  is jointly determined by the block size, the network scale measured in hop counts, and the average round-trip time of the links. The empirical study in [45] reveals that for small-size blocks, e.g., less than 20kB for Bitcoin, the round-trip delay is the dominant factor of the block propagation delay. Otherwise, transaction validation time becomes the major factor of the block propagation delay, which grows linearly with respect to the size of a block, e.g., 80ms/kB for Bitcoin. In [95], an implicit metric to capture the impact of network scale on the block propagation delay is adopted. Therein, the ratio between the block size and the propagation time required to reach a certain percentage of the nodes in the network is measured for the Bitcoin network. The experiments show that in the Bitcoin network with 55kb/s propagation rate for 90% of the nodes, the block interval should not be smaller than 12s, which leads to a peak transaction throughput of 26TX/s for 250Byte transactions.

Furthermore, the studies in [96], [97] also consider the impact of the propagation delay on the incidence of abandoning a proposed block with valid PoW solution. More specifically, finding a valid puzzle solution does not necessarily mean that the proposed block will be finally accepted by the network. Due to the propagation delay, a blockchain fork (see Figure 5) can only be adopted as the canonical blockchain state when it is first disseminated across the network. By considering both the round-trip delay and the block verification delay, the average block propagation delay across a P2P network is modeled as a function of the block size  $s$  in [97]:

$$T(s) = T_p(s) + T_v(s) = \frac{s}{aC} + bs, \quad (6)$$

where  $a$  is a network scale-related parameter,  $C$  is the average effective channel capacity of each link [98] and  $b$  is a coefficient determined by both the network scale and the average verification speed of each node (cf. [45]). Based on (6), the probability for the network to abandon/orphan a valid block proposal of size  $s$  due to the delay of block diffusion is modeled as follows [96], [97]:

$$\mathbf{Pr}^{\text{Orphan}}(s) = 1 - e^{-\lambda T(s)}, \quad (7)$$

where  $\lambda$  is the expected block arrival rate.

From a user's perspective, it is insufficient to know only the network-level probability of block orphaning due to the latency. Alternatively, it is of more interest to determine the safe time interval between locally observing on the chain a transaction and confirming it. With this in mind, the study in [94] considers a scenario where the adversary gets additional computation time by delaying the block propagation with a certain number of rounds  $\Delta$ . Based on the analysis of the common-prefix property [23], a new metric, i.e.,  $K$ -consistency is proposed in [94] to examine whether any two honest nodes are able to agree on the blockchain state that is at least  $K$  blocks deep from the end of the chain. Let  $\alpha$  and  $\beta$  denote the probabilities that an honest node and the attackers can propose a valid block within a round, respectively. The analytical study in [94] (cf. [93, Lemma 8]) shows that the required waiting time  $T$  is jointly determined by  $\alpha$ ,  $\beta$ ,  $\Delta$  and the parameter determining the searching space of the hash function, i.e.,  $L$  in Definition 1. More specifically, as long as the following condition is satisfied with an arbitrarily small constant  $\delta > 0$  (see [94, Theorem 1.2])

$$\alpha(1 - (2\Delta + 2)\alpha) \geq (1 + \delta)\beta, \quad (8)$$

and  $K > K_0(L) = c \log(L)$  for some constant  $c$ , the Nakamoto protocol satisfies the property of  $K$ -consistency (except with negligible probability in  $K$ ). However, the closed-form threshold  $K_0(L)$  for  $K$ -consistency is not provided in [94].

### C. Proof of Concepts Attached to Useful Resources

Under the framework of Nakamoto protocol, a number of alternative PoX schemes have been proposed to replace the original PoW scheme in permissionless blockchain networks. Generally, these PoX schemes aim at two major designing goals, i.e., to incentivize useful resource provision, e.g., [36], [70], [84], [99], [100] and to improve the performance, e.g., in terms of security, fairness and eco-friendliness [83], [101], [102] of the blockchain networks. Starting from this subsection, we will focus on the principles of puzzle design discussed in Section III-A and provide a close examination on different PoX schemes in the literature.

With the purpose of useful resource provision, the idea of "Proof of Useful Resources" (PoUS) has been proposed to tackle the resource wasting problem of PoW. Instead of enforcing the consumption of computational cycles for merely hash queries, a number of studies are devoted to the design of puzzles that are attached to useful work. An early attempt, i.e., Primecoin [103], proposed to replace the PoW puzzle in (1) by the puzzle of searching three types of prime number

chains, i.e., the Cunningham chain of the first/second kind or the bi-twin chain [104]. However, the verification stage of Primecoin puzzle is based on classical Fermat test of base two (pseudoprime) [103], hence violates the principle of soundness in non-interactive ZKP. Meanwhile, since the induced solution arrival does not follow the i.i.d. Bernoulli model in (3), the Primecoin puzzle does not simulate the random distribution for leader selection as required by (2).

In [105], a similar scheme, i.e., the proof of exercise is proposed to replace the preimage searching problem in PoW with the useful “exercise” of matrix product problems. The scheme uses a pool of task proposals to replace the PoW-based puzzle solving processes by the computation tasks offered by non-authenticated clients. Each consensus node needs to bid for a specific task to determine its puzzle. For this reason, the puzzle solution-generating scheme behaves more like a Computation as a Service (CaaS) platform. Since the matrix problems in the task pool may present different complexity levels, the puzzle competition does not fully simulate on the network level the random distribution in (2). Also, the solution verification can only be done probabilistically due to the lack of  $O(n)$  verification schemes. Therefore, the proposed scheme in [105] suffers from the same problems as in the Primecoin [103].

In [84], a new puzzle framework, i.e., useful Proof of Work (uPoW) is designed to replace the primitive PoW puzzle in (1) with a specific set of problems satisfying not only the properties of completeness, soundness and non-invertibility (hardness), but also the additional requirement of usefulness. Here, the usefulness is implied in the execution stage of the puzzle (cf. Table I). Formally, by assuming completeness and soundness, the properties of usefulness can be defined as follows (cf. [84, Definition 1]):

**Definition 2** (Usefulness). *Suppose that a challenge  $c_x$  and an accompanying puzzle solution (proof)  $s$  are generated from an input string  $x$ . If there exists an algorithm  $\text{Recon}(c_x, s)$  such that for a target function  $F(\cdot)$  its output satisfies  $\text{Recon}(c_x, s) = F(x)$ , the challenge is known to be useful for delegating the computation of  $F(x)$ .*

The study in [84] proposes to replace preimage searching in (1) with a family of one-way functions satisfying the property of fine-grained hardness [106] for uPoW puzzle design. Namely, the PoW puzzle is proposed to be replaced by the problem of known worst-case-to-average-case complexity reduction. A special case of uPoW puzzles based on the problem of  $k$ -Orthogonal Vectors ( $k$ -OV) is discussed. In brief, the solution to  $k$ -OV performs an exhaustive search over  $k$  sets of identical-dimension vectors and determines whether for each set there exists a vector such that these  $k$  vectors are  $k$ -orthogonal. In order to construct non-interactive proofs, uPoW in [84] employs the hash function  $\mathcal{H}(\cdot)$  as a random oracle. Simply put, given the number of vectors in each set, non-interactive uPoW treats the elements of each vector as the random coefficients of polynomials with the identical order. uPoW initializes the first element of each vector, i.e., the lowest order coefficient with a publicly known input string  $x$  and then uses it as the input to  $\mathcal{H}(\cdot)$  for generating the next-

order coefficient. The output of  $\mathcal{H}(x)$  will then be iteratively used as the input for generating the next-order coefficient. This can be considered as a typical example of applying the Fiat-Shamir scheme<sup>11</sup> to construct non-interactive PoW out of interactive ZKP schemes. With such an approach, uPoW does not need to explicitly define the vector sets. It also guarantees that the solutions of  $k$ -OV found by each prover follow a Bernoulli distribution. Therefore, the uPoW scheme fits well in the existing Nakamoto protocols by simulating a provable random function. As stated in [84], besides  $k$ -OV, uPoW is compatible with computation delegation for other problems such as 3SUM [106], all-pairs shortest path [106], and any problem that reduces to them<sup>12</sup>.

Schemes that are similar to uPoW can also be found in [100]. In [100], the problem of untrusted computational work assignment is addressed in a Trusted Execution Environment (TEE). The TEE can be constructed using Intel Software Guard Extensions (SGX), which is a set of new instructions available on certain Intel CPUs to protect user-level codes from attacks by hardware and other processes on the same host machine. In the permissionless network, the clients supply their workloads in the form of tasks that can be run in an SGX-protected enclave (i.e., protected address space). The study in [100] exploits the truthfulness-guaranteeing feature of the Intel attestation service [108] in the SGX-protected platform to verify and measure the software running in an enclave. With the designed puzzle, the work of each consensus node is metered on a per-instruction basis, and the SGX enclave randomly determines whether the work results in a valid block proof by treating each instruction as a Bernoulli trial. Based on the TEE, each executed useful-work instruction is analogous to one hash query in the primitive PoW, and the enclave module works as a trusted random oracle.

Apart from delegation of useful computation, PoX can also be designed to incentivize distributed storage provision. For example, Permacoin [109] proposes a scheme of Proof of Retrievability (PoR) in order to distributively store an extremely large size of data provided by an authoritative file dealer. The file dealer divides the data into a number of sequential segments and publishes the corresponding Merkle root using the segments as the leaves. A consensus node uses its public key and the hash function to select a random group of segment indices for local storage. For each locally stored segment, the node also stores the corresponding Merkle proof derived from querying the Merkle tree. The challenge-proof pair is generated based on a subset of the locally stored segments and the corresponding Merkle proof. To ensure the non-interactiveness and freshness of the puzzle (cf. interactive PoR in [110]), the node needs a publicly known and non-precomputable puzzle ID to seed the process of segment selection called “scratch-off”. To help the readers understand the puzzle generation process, we present a simplified execution stage of PoR as follows (see also [109, Figure 1]):

- The execution stage of PoR: suppose a node is given

<sup>11</sup>The Fiat-Shamir scheme takes a similar form to the process of digital signature verification, see [107] for the definition.

<sup>12</sup>These problems should be worst-case hard for some time bound and can be represented by low-degree polynomials.

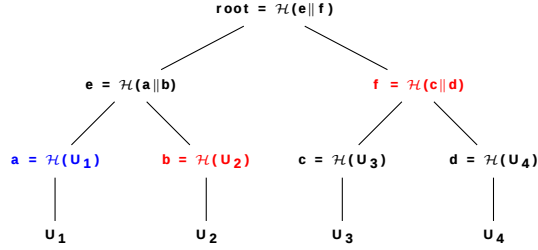


Figure 8. Illustration of Merkle proof: for segment  $U_1$ , the Merkle proof is obtained by climbing up the tree until the root (as the nodes in red).

the key pair  $(sk, pk)$ , the puzzle ID  $id_{puz}$ , the vector of locally stored segment indices  $\mathbf{v}$ , the required number of Merkle proofs  $k$ , the vectors of all the file segments  $\mathbf{U}$  and the corresponding Merkle proof vector  $\pi$ . The random IDs of the local segments for challenge generation can be determined by:

$$\forall 1 \leq j \leq k : r_j = \mathbf{v}(\mathcal{H}(id_{puz} || pk || j || nonce) \bmod |\mathbf{v}|), \quad (9)$$

where  $nonce$  is a random value chosen by the node. For each segment  $\mathbf{U}(\mathbf{v}(r_j))$  in the challenge, the proof is in the form of  $(pk_i, nonce, \mathbf{U}(\mathbf{v}(r_j)), \pi(\mathbf{v}(r_j)))$ .

The execution stage of PoR in [109] is composed of a fixed number of queries to the random oracle  $\mathcal{H}$ . Thereby, although PoR satisfies the principle properties of non-interactive ZKP, it does not simulate the random leader election process. In this sense, the proposed PoR scheme may not be able to achieve the claimed goal of “repurposing PoW” in [109]. Instead, it is more similar to the existing systems such as Stoj [111], Sia [112] and TorCoin [99], where PoX is only used to audit the execution of the smart contracts or script-based transactions instead of facilitating the consensus mechanism.

Further improvement to PoR can be found in the proposals of KopperCoin [70] and Filecoin [36]. In [70], KopperCoin adopts the same framework of distributed storage for a single file as in Permacoin [109]. Compared with Permacoin, the main improvement of the puzzle design in KopperCoin is to simulate the random leader election process for block proposal. KopperCoin introduces a bitwise XOR-based distance metric between the index of a locally stored data segment and a random, publicly known challenge  $c$ . A node needs to provide the valid Merkle proof (PoR) of a segment, of which the index (denoted by  $j$ ) should satisfy the following condition:

$$\mathcal{H}(x) \cdot 2^{|\mathbf{j} \oplus \mathbf{c}|} \leq D(h), \quad (10)$$

where the block payload  $x$  and the difficulty threshold  $D(h)$  are defined in the same way as in Definition 1. Compared with (1), the solution searching for (10) is now performed within the range of the locally-stored segment indices. The more segments a node offers to store, the better chance the node has to find a solution to (10). Again, the generation of the public, unpredictable random challenge  $c$  can be derived based on hashing the header of the most recent block. This approach presents another example of applying the Fiat-Shamir transformation to realize non-interactiveness [107].

In the Filecoin network [36], the concept of “spacetime” is introduced to allow metering the data stored in the network with an expiry time. Filecoin aims to provide the functionality

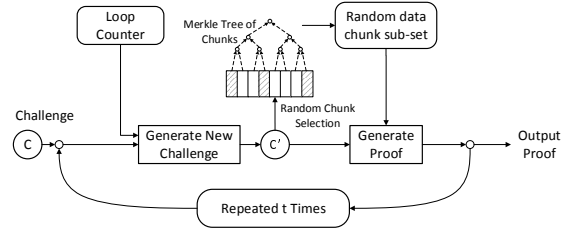


Figure 9. Illustration of the PoST scheme based on iterative PoR over time.

of recycling and re-allocating the storage on the provider (miner) side as well as easing the files retrieval process on the client side. Like in the proof-of-exercise scheme, Filecoin designs the market for storage and retrieval of multiple files based on smart contracts. A new puzzle, i.e., Proof of SpaceTime (PoST) [83], is adopted based on the intuition of generating a PoR sequence during a certain period to prove the holding time of useful storage. As illustrated by Figure 9, the major difference of PoST from PoR lies in the repeated execution phases for challenge updating without rerunning the initialization stage. Namely, a consensus node is required by the Filecoin network to submit PoR (e.g., in a similar way to Permacoin [109]) every time when the blockchain is extended by a certain number of blocks. Instead of simulating random leader election based on adjustable difficulty [70], the Filecoin network uses the following mechanism to determine whether a node  $i$  is elected for block proposal:

$$\frac{1}{2^L} \mathcal{H}(t | \text{rand}(t)) \leq \frac{w_i}{\sum_{j \in \mathcal{N}} w_j}, \quad (11)$$

where  $t$  is the index of consensus round (i.e., block index),  $L$  is the output string length of the hash function (see (1)),  $\text{rand}(\cdot)$  is an assumed random oracle, and  $w_i$  represents the storage power of node  $i$  (see also (2)). It is worth noting that the evaluation of  $w_i$  in (11) can only be done through PoST. Thus, the Filecoin network admits a double-challenge scheme, where the leader election is performed based on a second challenge, i.e., (11). The nodes with the better quality of PoST proofs (storage power) are more likely to win the second challenge. Under the framework of double challenges, a similar approach of puzzle design can also be found in the proof of space-based cryptocurrency proposal known as SpaceMint [83], [101].

#### D. Proof of Concepts for Performance Improvement

Alternative PoX schemes have also been designed with the emphasis on improving the performance of PoW in the aspects such as security, fairness and sustainability. To alleviate the problem of computation power centralization due to the massive adoption of ASICs, memory-hard PoW, also known as the Proof of Memory (PoM), is adopted by ZCash [44] and Ethereum [35] networks. In the ZCash network, the Equihash scheme [81] is adopted based on the generalized birthday problem [113]. The study in [81] has pointed out that any identified NP-complete problem can be the natural candidate for the PoX puzzle due to their proved hardness, as long as the solution verification can be completed in polynomial time. However, a puzzle design only satisfying the hardness requirement may not be able to combat the botnet or ASIC-based manipulation of hashrate. Thus, a suitable PoX is expected to

be “optimization-free” and “parallelism-constraint”. Namely, the solution searching process cannot be sped up by using alternative algorithms or through parallelization.

An ideal approach of imposing parallelism constraint is to ensure that the PoW scheme is inherently sequential. However, an inherently sequential NP problem that is known to be verified in short time is yet to be found [81]. Therefore, the study in [81] adopts an alternative approach by imposing enormous memory bandwidth to the parallel solution of the puzzle. According to [113], the generalized  $k$ -dimensional birthday problem is to find  $k$  strings of  $n$  bits from  $k$  sets of strings, such that their XOR operation leads to zero. Equihash employs the hash function  $\mathcal{H}(\cdot)$  to randomly generate the strings using the block payload data  $x$  and a nonce (as in (9)), such that both the XOR-based birthday problem solution and a PoW preimage of a given difficulty are found. It is shown in [113] that the best solution algorithm to this problem presents  $O(2^{n/k})$  complexity in both time and space and thus is memory-intensive. More importantly, for a  $k$ -dimensional problem, a discounting factor  $1/q$  in memory usage leads to  $O(q^{k/2})$  times more queries to the hash function. Due to the physical memory bandwidth limit, the computation advantage of parallelization is limited. These properties guarantee the ASIC-resistance of Equihash.

With the same purpose of preventing the “super-linear” profit through hashrate accumulation, Ethereum currently adopts a different puzzle design known as Ethash for ASIC resistance [114]. Ethash requires the consensus nodes to search for the PoW puzzle solution based on a big pseudorandom dataset, which increases linearly over time. The dataset is organized as the adjacency matrix of a DAG, where each vertex represents a randomly generated data field of 128 bits. In the execution stage of Ethash, the node starts a one-time search of the solution with a hash query, and uses the concatenation of the block payload and a nonce to seed the hash function for locating a random vertex in the DAG. Then, the search is completed in a fixed-iteration loop of queries to the hash function, for which the output of the last iteration, i.e., the data field of the last vertex in the path is used as the input to determine the position of the next vertex in the DAG. The final output of the loop is used to check against the preimage condition as in (1). As illustrated in Figure 10, the designed puzzle of Ethash makes the searching algorithm inherently sequential. With Ethash, the rate of data field fetching from the DAG is limited by the memory bandwidth. Then, paralleling the hash queries with ASICs cannot lead to much performance improvement in a single search of the puzzle solution.

Ethash [114] only makes the puzzle solution partially sequential within a single attempt of preimage search. Therefore, Ethash still faces the problem of PoW outsourcing since a consensus node can divide the puzzle solution search into multiple sub-problems and outsource them to different “mining workers” (i.e., puzzle solvers). Such a problem is also known as the formation of mining coalition (pool) [61] and may result in a serious problem of consensus manipulation by a handful of full nodes [4]. In [82], a nonoutsourcable “scratch-off puzzle” is proposed to disincentivize the tendency of mining task outsourcing. Intuitively, when a node effectively outsources

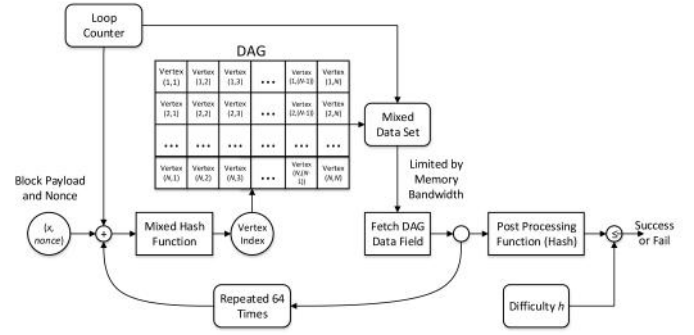


Figure 10. One query to the random oracle in Ethash for a given **nonce** based on the iterative mixed hash operation for vertex searching.

its puzzle-solving work to some mining machines, we call the puzzle nonoutsourcable if these miners can steal the block proposal reward of that node without producing any evidence to implicate themselves. The study in [82] employs Merkle proofs for puzzle design, which can be considered as a generalization of the PoR [109]. In [82], a Merkle tree is created based on a number of random strings. To generate a fresh puzzle, a node queries the hash function for the first time with a random nonce and the constructed Merkle root. The output of this query is used to select a random subset of distinct leaves on the Merkle tree. Then, the concatenation of the Merkle proofs for each leaf in subset and the same nonce is used as the input to the second query of the hash function. The output is used to compare with the preimage condition as given in (1). If a solution (nonce) is found, the payload of the proposed block is used as the input of the third query to the hash function, and the output is used to select another subset of random leaves on the Merkle tree. The corresponding Merkle proofs are treated as the “signature” of the payload of the proposed block. With such puzzle design, mining workers only need to know a sufficiently large fraction of the Merkle tree leaves to “steal” the reward by replacing the Merkle proof-based signature with their own proofs.

It is worth noting that the nonoutsourcable puzzle in [82] is generated in such a way to make the preimage search for (1) independent of the payload of the proposed block, i.e., using the randomly generated Merkle tree. Then, a mining worker is able to replace the original payload including the public keys from the outsourcer by its own payload without being detected. A similar proposal of nonoutsourcable puzzle can be found in [115], where a nonoutsourcable puzzle is designed based on two-fold puzzle. Namely, an inner puzzle is solved as a typical PoW puzzle, whose solution is used as the input of an additional PoW puzzle known as the outer puzzle. To prevent outsourcing the work load, a mining worker’s signature is required for the inner puzzle solution to be used by the outer puzzle. However, it is pointed out in [115] that such design can only be considered heuristic and is not guaranteed to have the formal properties of weak outsourcability [82].

Apart from the manipulation-resistant puzzles, other puzzles are proposed in [101], [102] with the emphasis on eco-friendliness. Therein, the major goal is to reduce/remove the repeated hash queries to curb energy consumption due to hash queries. In [101], the SpaceMint network is proposed based on Proof of Space (PoSp) [116]. Similar to PoR [109], PoSp

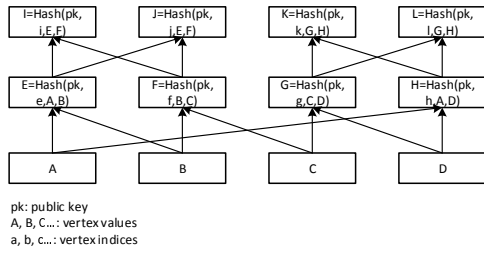


Figure 11. An example of DAG formation based on the hash of the parent vertices: for miner  $i$  adopting a public key  $pk_i$ , the value  $v_j$  of the  $j$ 's vertex in its DAG with  $m$  parent vertices  $\{p_1, \dots, p_m\}$  is obtained as  $v_j = H(pk_i, j, v_{p_1}, \dots, v_{p_m})$ .

requires the consensus nodes to provide non-interactive proofs of storage dedication during puzzle solution searching. The major difference from PoR lies in that PoSP does not need the prover to store useful data (from the verifiers), and the proof is based on a large volume of random data stored on the provers' hard drive. As in Ethash [114], the committed space is also organized as a DAG, where the value of each vertex is determined based on the hash of its parent vertices (see Figure 11). A consensus node is required to use the hash of an earlier block as the seed to sample a random set of vertex values. The set of the vertex values forms the challenge of the node's local PoSP puzzle. If the node is able to provide the Merkle proofs for all the vertices in the challenge set, namely, the sibling vertices that lie on the path between each challenge vertex and the end vertex in the DAG with no outgoing edge, the proposed block is considered a valid block candidate. SpaceMint also proposes to measure the quality of a set of Merkle proofs based on the hash value of the concatenated vertex in a Merkle tree. Then, the blockchain network is able to select the block with the best quality of proof from the candidate blocks when a fork occurs.

The study in [102] proposes to introduce a human-in-the-loop puzzle, i.e., the Proof of Human-work (PoH) into the Nakamoto protocol. The designing goal of PoH is to guarantee the properties of eco-friendliness, usefulness and centralization-resistance at the same time. It is proposed in [102] that PoH should be able to provide non-interactive, computer-generated puzzles which are moderately hard for a human but hard for a computer to solve, even for the computer that generates the puzzles. PoH is inspired by the widely-adopted systems of Completely Automated Public Turing-Test to tell Computers and Humans Apart (CAPTCHA) [117]. Traditional CAPTCHA systems usually take human-efficient input (e.g., images) with a known solution and generate the puzzle based on distortion to the solution. For PoH, a universal sampler [118] is assumed to be available to generate a random CAPTCHA instance for the consensus node such that the puzzle-generating machine is not able to directly obtain the puzzle solution. Then, the node (i.e., miner) needs human work to obtain the corresponding solution of the CAPTCHA puzzle. A two-challenge puzzle design is adopted and the solution of the CAPTCHA puzzle is used as the input of a small PoW puzzle as defined in (1). A complete PoH solution includes a CAPTCHA solution and a nonce such that they together satisfy the preimage condition in (1). PoH implicitly assumes that some Artificial Intelligence (AI) problems (e.g., recognition of

distorted audios or images) are human-efficient but difficult for machines. Then, by selecting a proper underlying CAPTCHA scheme, it is possible to extend the PoH with a variety of meaningful human activities ranging from that educational purposes to a number of socially beneficial programs [118].

For a progressive summary, we summarize in Table III the major properties of the PoX schemes reviewed in this section.

#### IV. STRATEGIES OF RATIONAL NODES IN THE FRAMEWORK OF NAKAMOTO CONSENSUS PROTOCOLS

In this section, we review the studies on the incentive compatibility of the Nakamoto consensus protocols. By adopting the basic assumption on rationality of the consensus nodes (i.e., block miners), we provide a comprehensive survey on the node strategies in the consensus process for block mining. It is worth noting that most of the analysis in the literature about the consensus nodes' mining strategies are presented in the context of the PoW-based Bitcoin network. Nevertheless, they can be readily extended to other PoX schemes under the framework of Nakamoto protocols. In particular, we focus on the game theoretic formulation of resource allocation during the mining process, and then explore how miners can exploit the vulnerability of the incentive mechanism of the Nakamoto protocols in permissionless blockchain networks.

##### A. Incentive Compatibility of Nakamoto Protocols

For Nakamoto protocols, monetary incentive plays the key role to ensure that most of the consensus nodes/miners follow the rules of blockchain state transition during the puzzle solution competition. In permissionless blockchain networks, the incentive mechanism is built upon the embedded digital token issuing and transferring schemes. In a typical PoW-based blockchain network, the leader/winner in the block proposal competition not only collects transaction fees from the approved transactions in the new block, but also gets token issuing reward, e.g., the "coinbase reward" in Bitcoin, for expanding the blockchain with the new block. For this reason, the puzzle competition process is compared to the process of "gold mining", since by casting resources into the competition, the nodes expect to receive monetary rewards carried by the tokens. As a result, the consensus participant nodes are better known as block "miners" to the public.

In [65] the consensus in blockchain networks is divided into three folds, namely, the consensus about the rules, e.g., about transaction dissemination and validation, the universality of the blockchain state and financial value that the digital token carries. Then, the studies on the Nakamoto protocol's incentive compatibility can also be categorized according to these three aspects. Since the introduction of ASIC devices and pool mining for PoW-based blockchain networks, concerns have been raised about the nodes' incentive to fully abide by the protocol [60], [61], [65], [119]. Due to the explosion of network-level hashrates (see Figure 7(a)), most of the practical blockchain networks, i.e., cryptocurrency networks, are nowadays dominated by the proxies of mining pools [66] (see Figure 12). An individual node in a mining pool is known

Table III  
COMPARISON OF DIFFERENT POX SCHEMES FOR PERMISSIONLESS BLOCKCHAINS

| Puzzle Name                             | Origin of Hardness (One-way Function)                                | Designing Goal                    | Implementation Description                                                  | ZKP Properties | Simulation of Random Function | Features of Puzzle Design                               | Network Realization              |
|-----------------------------------------|----------------------------------------------------------------------|-----------------------------------|-----------------------------------------------------------------------------|----------------|-------------------------------|---------------------------------------------------------|----------------------------------|
| Primitive proof of work [23], [86]      | Partial preimage search via exhaustive queries to the random oracle  | Sybil-proof                       | Repeated queries to cryptographic hash function                             | Yes            | Yes                           | Single challenge                                        | Bitcoin [1], Litecoin [92]       |
| Proof of exercise [105]                 | Matrix product                                                       | Computation delegation            | Probabilistic verification                                                  | N/A            | No                            | Single challenge                                        | N/A                              |
| Useful proof of work [84]               | $\mathbf{K}$ -orthogonal vector, 3SUM, all-pairs shortest path, etc. | Computation delegation            | Non-interactiveness via Fiat-Shamir transformation                          | Yes            | Yes                           | Single challenge with sequential hash queries           | N/A                              |
| Resource-efficient mining [100]         | N/A                                                                  | Computation delegation            | Guaranteed by TEE                                                           | Yes            | Yes                           | Trusted random oracle implemented by dedicated hardware | N/A                              |
| Proof of retrievability [110]           | Merkle proofs of file fragments in the Merkle tree                   | Distributed storage               | Non-interactiveness via Fiat-Shamir transformation and random Merkle proofs | Yes            | Conditional                   | Two-stage challenge                                     | Permacoin [109], KopperCoin [70] |
| Proof of space-time [36]                | The repeated proof of retrievability over time                       | Decentralized storage market      | Repeated PoR                                                                | Yes            | Conditional                   | Two-stage challenge and repeated PoR over time          | Filecoin [36]                    |
| Equihash [81]                           | The generalized birthday problem                                     | ASIC resistance                   | Time-space complexity trade-off in proof generation [81]                    | Yes            | Yes                           | Memory-hard                                             | ZCash [44]                       |
| Ethash [114]                            | Random path searching a random DAG                                   | ASIC resistance                   | Repeated queries to cryptographic hash function                             | Yes            | Yes                           | Sequential, memory-hard puzzle                          | Ethereum [35]                    |
| Nonoutsourcable scratch-off puzzle [82] | Generalization of proof of retrievability                            | Centralization resistance         | Random Merkle proof                                                         | Yes            | Yes                           | Two-stage challenge                                     | N/A                              |
| Proof of space [116]                    | Merkle proofs of a vertex subset in a random DAG                     | Energy efficiency                 | Random Merkle proof                                                         | Yes            | Yes                           | Two-stage challenge and measurement of proof quality    | SpaceMint [116]                  |
| Proof of human work [102]               | Random CAPTCHA puzzle requiring human effort                         | Useful work and energy efficiency | CAPTCHA and PoW                                                             | Yes            | Yes                           | Human in the loop                                       | N/A                              |

as a mining worker, since it no longer performs the tasks of transaction validation or propagation and does not even keep any blockchain data. On the contrary, only the proxy of the pool, i.e., the pool server/task operator maintains the replica of the blockchain. The pool server divides the exhaustive preimage search for PoW solution into a number of sub-tasks and outsources them to the mining workers<sup>13</sup>. In this sense, only the pool server can be considered as a node in the blockchain network. Studies have shown that joining a mining pool has become the more plausible strategy than working as an individual consensus node, since such a strategy reduces the income variance and secures stable profits [4], [61]. However, this leads to the formation of mining-pool Cartel [61] and is against the design goal of Nakamoto consensus in [1], that “the network is robust in its unstructured simplicity”.

A further study in [58] reveals that under the current framework of Nakamoto protocols, no incentive is provided for nodes to propagate the transactions that they are aware of. The study considers the situation when transaction fees dominate the block rewards [121]. The analysis in [58] models the paths of transaction dissemination as a forest of  $d$ -ary directed trees, where each transaction issuer considers its peer nodes

<sup>13</sup>According to the Stratum mining protocol [120], the pool server only needs to send a miner the Merkle root of the transactions in the block (see Figure 2) and a difficulty level to complete the puzzle solving sub-task.

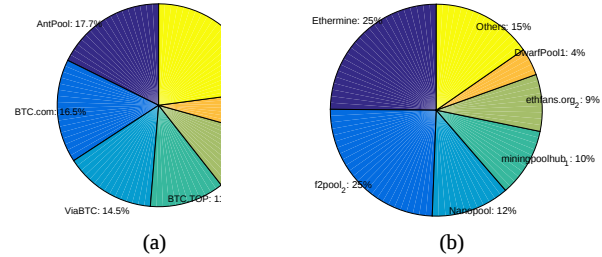


Figure 12. Hash rates controlled by mining pools in (a) Bitcoin (data source: <https://blockchain.info>) and (b) Ethereum (data source: <https://etherscan.io>).

as the tree roots and the nodes on the far end of the network as the leaves. During transaction dissemination, a consensus node can add any number of pseudo-identities (a.k.a., fake identities) before selectively relaying the transaction to any of its neighbors. It is shown that a consensus node tends to not broadcast any transaction that offers a fee. By doing so, it reduces the number of nodes that are aware of the transaction and hence the competition of mining that transaction. An improved protocol is proposed in [58] by introducing a broadcasting incentive mechanism. More specifically, the proposed mechanism requires that each relaying node in the path of transaction propagation shares a uniform portion of reward with the root (i.e., mining) node, when the height of the relaying node is small than a predetermined threshold in the directed tree. The analysis of the new protocol is based

on the formulation of a normal-form game [122], and thus the equilibrium strategy of each node can be obtained through iterative removal of dominated strategies. The designed incentive mechanism is shown to guarantee that only the non-Sybil and information propagating strategies survive in the iterated removal of weakly dominated strategies, as long as the miners are connected to sufficient many peers.

Similar studies to enforce honest block/transaction propagation can also be found in [62], [123]. The study in [62] casts the problem of incentivizing block propagation into the framework of routing in  $k$ -connected networks, where each rational node can freely choose between relaying and mining (or both). A protocol of transaction fee-sharing is designed therein to guarantee that the rational strategy of honest nodes in the network is to propagate the received transactions. It is required that a mining node shares the reward of a new transaction with the relaying nodes in one path between itself and the client which issues that transaction. According to [58], creating pseudo-identities does not increase the connectivity of a node. From such an observation, it is proved in [62] that assigning the propagation reward of each relaying node as a decreasing function of the hop count guarantees transaction propagation, as long as the computing power (or other resources for mining) controlled by each node does not dominate the network. Comparatively, the study in [123] ensures that the payment made to the transaction-relaying nodes cannot be denied by the miners of the new blocks. With the proposed propagation protocol in [123], each intermediate hop adds its own signature to the transaction before sending it to the next hop. While working on their own PoW-puzzle solution, the relaying nodes freely charge their descendants at least a minimum fee for propagation. The miner whose block finally gets confirmed by the blockchain will pay for the propagation fees to one selected path of nodes. As in [58] the process of transaction propagation and relaying price competition is modeled as a non-cooperative game in [123]. It is proved that with the proposed propagation protocol based on the chain of signatures, a rational miner's equilibrium strategy is to always choose the shortest path, and a rational intermediate node's equilibrium strategy is to always charge its descendants the minimum fees for relaying transactions.

When block creation reward dominates the mining reward, incentive incompatibility may appear in different forms. Intuitively, it is plausible for a rational miner to pack up a proper number of transactions with decent fees in the new block for profit maximization. However, empty blocks with only coinbase transaction or blocks with a tiny number of transactions can be frequently observed in the practical blockchain networks<sup>14</sup>. An informal game theoretic analysis in [124] indicates that the consensus nodes tend to ignore the received blocks of large size in a flat network and relay the smaller competing blocks instead. The reason is that large blocks result in longer delay due to transaction validation, hence increasing the probability of orphaning any blocks that are mined based on them. Although mining empty block does not violate the current Nakamoto protocol, it results in the same situation as a

Distributed Denial of Service (DDoS) attack [125] by blocking the confirmation of normal transactions.

Furthermore, the statistical studies in [126], [127] have shown that the consensus nodes behave rationally and are prone to prioritize the transactions with higher transaction fees during block packing. However, when the coinbase reward dominates the block mining reward, the miners are yet not incentivized to enforce strictly positive fees [127]. In the case study of Bitcoin network, extra delays for the small-value transactions are identified ranging from 20 minutes [127] to as long as 30 days [126]. Also, it is observed in [127] that most of the lightweight nodes still set an arbitrary transaction fee in the real-world scenarios. It is unclear whether the miners or the transaction issuers adopt best-response strategies systematically. The study in [128] simplifies the consensus process as a supply game subject to the trade of a specific type of physical goods. In the considered scenario, the miners essentially become the follower players in a two-level hierarchical/Stackelberg game<sup>15</sup> led by the blockchain network, which is assumed to be able to set the transaction prices. Then, they are expected to have an incentive for including all transactions if there exists no block-size limit. On the other hand, it is pointed out in [98] that, since the block orphaning probability exponentially grows with the block size, a healthy transaction fee market does not exist for unlimited block size due to the physical constraint of link capacity in the network.

Finally, it is worth noting that most of the existing studies are based on the presumption that the tokens carried by a blockchain have monetary value and their exchange rate volatility is small. An optimistic prediction is provided in [59] based on an assumption excluding any state variables on the user side except the belief in "proper functioning of a cryptocurrency". In the absence of investors and when the blockchain is used only for the purpose of remittance, it is shown in [59] that the tokens of a blockchain network admit a unique equilibrium exchange rate in each period of the belief evolution. Conditioned on the survival of a cryptocurrency, the equilibrium state depends on the excess in users' valuation of the blockchain over the other payment options as well as the supply of the tokens in the market. Together with the Stackelberg game-based interpretation in [128], it is reasonable to consider that the equilibrium price of a blockchain token is determined by the demand-supply relation in the market. It is worth noting that the data security is only guaranteed by sufficient PoW computation power in the blockchain network. Currently, except for a few studies such as [129], it is generally unclear how the impact of security issues is reflected in the users' valuation of the blockchain. As a result, whether the security requirement of the Nakamoto protocol is compatible with the market clearing price remains an open question.

## B. Resource Investment and Transaction Selection for Mining under Nakamoto Protocols

According to (2), an honest consensus node has to invest in the mining resources, e.g., hashrates, disk space, etc, to win

<sup>14</sup>See Blocks #492972 in Bitcoin and #3908809 in Ethereum for examples.

<sup>15</sup>A Stackelberg game is characterized by the sequential play of leaders and followers, where the leaders may expect better equilibrium payoffs [122].

the puzzle solution competition under Nakamoto consensus protocols. Intuitively, the more resources a miner casts into the network, the higher chance the miner has to win the puzzle competition and obtain the mining reward. However, the success is not guaranteed because this also depends on the mining resources of other miners. Since mining resources are usually expensive, how to properly invest in the mining resources to maximize the profit is a big concern of the miners.

The study in [130] abstracts the mining investment in the Bitcoin network as the energy consumption cost. It is assumed that  $N$  active miners in the network are competing in the “all-pay contest” for block-mining rewards. The cost of presenting a unit mining resource by each miner may be different, e.g., with different electricity prices in different areas. The miners determine how much to invest in mining resources (hashrates) such that the expected profit is maximized. This forms a non-cooperative game among the miners. Analysis of the game’s unique Nash equilibrium in [130] shows that the decision of a miner to participate in the mining process or not solely depends on its individual mining cost, as long as the block reward is positive. Meanwhile, the structure of the formulated mining game prevents the emergence of a monopolistic mining activity. Namely, it is guaranteed that at least two miners will remain active in the game with positive expected profits.

By (6) and (7), even if a miner succeeds in the puzzle solution competition, it is still possible for the proposed block to get orphaned due to the propagation delay. For ease of exposition, we can assume that all transactions in a block set the same amount of transactions fee  $F$ . Let  $R$  denote the fixed reward for block generation and  $m$  denote the number of transactions in the block. Then, the revenue to mine this block is  $R + mF$ . Apparently, a rational miner expects to include as many as possible transactions in a block to maximize the received reward. However, due to the risk of block orphaning, a miner also has to carefully balance the tradeoff between the mining reward and the risk of block orphaning. In [98], the author proposes a mining profit model by assuming the propagation delay of a block to follow a Poisson distribution. Thus, the orphaning probability can be approximated by (7). Let  $\eta$  denote the monetary cost per hash query and  $\psi$  denote the probability for the miner being the leader (see also (3)). Then, for an average block arrival duration  $T$  and block propagation time  $\tau$ , a miner’s profit can be modeled as follows:

$$U = (R + F)\psi e^{-\frac{\tau}{T}} - \eta h T. \quad (12)$$

The profit model in (12) is capable of reflecting the impact of miners’ strategies in both resource investment and transaction selection. Therefore, this model is especially appropriate for game-theoretic formulation of mining resource management problems. Recently, (12) and its variation have been adopted to construct the payoff function of miners by a series of studies, which propose to use different game-based models, e.g., evolutionary game [97], hierarchical game [131] and auctions [132], to capture the rational behaviors of individual miners in different network setups.

In [133], an alternative model of winning probability is proposed to explicitly capture the influence of the adversary

miners’ strategy of block-size selection. We denote  $s_i$  as block size of miner  $i$  in a blockchain network and  $w_i$  as its computational power. Then, the block winning probability of miner  $i$  can be expressed by [133]:

$$Pr_i^{\text{win}} = \frac{w_i}{T} \prod_{j \neq i} e^{-\frac{w_j(t + \tau(s_i) - \tau(s_j))}{T}}, \quad (13)$$

where  $t$  is the time when all miners start mining a new block and  $\tau(s_i)$  is the time needed for a block with size  $s_i$  to reach consensus. In (13), the first and second terms represent the probability for miner  $i$  to first solve the puzzle based on its block, for this block to be the first one reaching the consensus across the network, respectively. (13) implies that the strategy of mining a large block may have positive externalities to other miners in the network. By analyzing the Nash equilibrium of the non-cooperative mining game with two miners, the author of [133] shows an interesting result, namely, the miner with higher computational power will prefer blocks of larger sizes. Meanwhile, the author also discusses the scenarios in which the Nash equilibrium is a breaking point, i.e., miners adopt the strategy of including no transaction in their proposed blocks.

The studies in [98] and [133] essentially assume that the mining process is synchronized and all miners honestly follow the rules of block/transaction propagation in Nakamoto protocols. However, such assumptions may not be met in practical scenarios. Thus, related strategies may not be the miners’ best response and further investigation is needed on this topic.

### C. Rational Mining and Exploitation of Nakamoto Protocols

The discussions on the incentive compatibility of Nakamoto protocols and the strategies of resource investment lead to the following question: is it possible for a rational miner to exploit the vulnerability of Nakamoto Protocols and find a strategy leading to the reward more than that in proportion to the devoted resources? In this section, we will further devote our survey on the existing analysis of this problem.

1) *Selfish Mining Strategy*: The study in [61] shows that selfish miners may get higher payoffs by violating the information propagation protocols and postponing their mined blocks. Specifically, a selfish miner may hold its newly discovered block and continue mining on this block secretly. Thereby, the selfish miner exploits the inherent block forking phenomenon of Nakamoto protocols. In this case, honest miners in the network continue their mining based on the publicly known view of the blockchain, while the selfish miners mine on their private branches. If a selfish miner discovers more blocks in the same time interval, it will develop a private longer branch of the blockchain. When the length of the public chain known by honest miners approaches that of the selfish miner’s private chain, the selfish miner will reveal its private chain to the network. According to the longest-chain rule, the honest nodes will discard the public chain immediately when they learn the longer view of the chain from the selfish miner. Such a strategy of intentionally forking results in the situation of wasted computation by the honest miners, while the revenue of the selfish miner can be significantly higher than strictly following the block revealing protocol. More seriously, if selfish miners

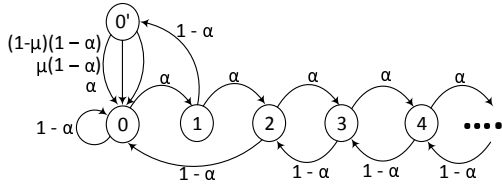


Figure 13. Blockchain state transition in the presence of a selfish pool (adapted from [61]).

collude and form a selfish mining pool with a sufficiently large amount of computational power, other rational miners will be forced to join the selfish mining pool, which can devastate the blockchain network [61].

In [61], the authors introduce an approach based on the Markov chain model to analyze the behavior as well as performance of a selfish mining pool. Figure 13 illustrates the progress of the blockchain as a state machine. The states of the system, i.e., the numbers in the circles represent the lead of the selfish pool in terms of the difference in block number between the private branch and the public branch. In Figure 13, state **0** is the original state when the selfish pool has the same view as the public chain. State **0'** indicates that two branches of the same length are published in the network by the selfish pool and the honest miners, respectively. The transitions in Figure 13 correspond to the mining event, i.e., a new block is mined either by the selfish pool or the honest miners.  $\alpha$  in Figure 13 represents the computational power of the selfish mining pool. Note that the transition from state **0** to state **0'** depends on not only the computational power of the selfish pool, but also the fraction, i.e.,  $\mu$  of honest miners that mine on the selfish pool's branch. In [61], the analysis on the steady state probability of the Markov chain leads to the following two important observations:

- For a given  $\mu$ , a selfish pool of size  $\alpha$  obtains a revenue larger than its relative size in the range of  $\frac{1-\mu}{3-2\mu} < \alpha < \frac{1}{2}$ .
- A threshold on the selfish-pool size exists such that each pool member's revenue increases with the pool size.

Extended from [61], the study in [134] introduces a new mining strategy known as the stubborn mining strategy, which is supposed to outperform the typical selfish mining strategy. The key idea behind the stubborn mining strategy is that the selfish miner is stubborn and may only publish part of the private blocks even when it loses the lead to the honest nodes. As shown in Figure 14, the major difference between the two selfish strategies lies in how the selfish miner publishes the private blocks. For example, at state 2, the typical selfish miner will immediately publish all the private blocks once the lead to the honest miners decreases by one block (see Figure 13). Then, the system transits to state 0. In contrast, every time when the honest miners mine a new block, the stubborn miner will stubbornly reveal one block of the private chain, even by doing so it will lose the lead. Simulations in [61] show that stubborn mining achieves up to 13.94% higher gains than selfish mining strategy.

Furthermore, the study in [134] also introduces another two extensions of the stubborn mining strategy, namely, the Equal-Fork Stubborn (EFS) and the Trail Stubborn (TS) mining strategies (see Figure 15). In Figure 15, state -1 indicates that

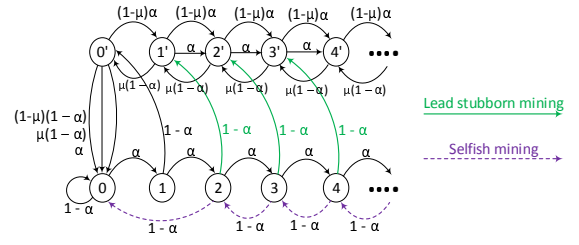


Figure 14. Lead-stubborn mining. The black and purple transitions together define the selfish mining state machine. The black and green transitions define the stage machine of lead-stubborn mining (adapted from [134]).

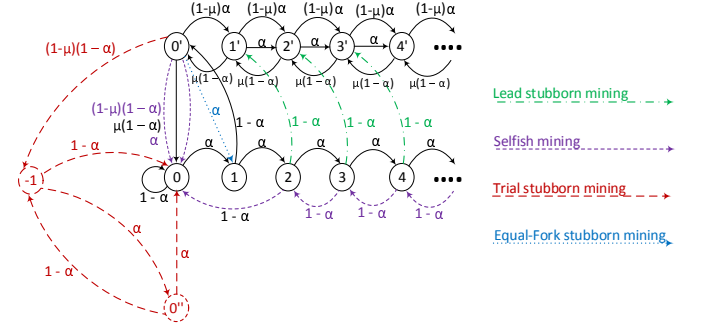


Figure 15. Lead, Equal-Fork, and Trail Stubborn mining. Black and purple transitions denote selfish mining. Black and green transitions denote lead-stubborn mining. Black and blue transitions denote Equal-Fork stubborn mining. Black and brown transitions denote Trail-stubborn mining (adapted from [134]).

the public chain is one block longer than the private chain. As indicated by the transitions from other states to state -1, the TS miner is more stubborn and keeps mining on the secret branch even when it is one block behind the public chain. From state -1, when the TS miner finds one new block ahead of the honest miners, the system will transit to state **0'**. Namely, the private chain catches up with the public chain and the block numbers on both chains are equal. In contrast, if the honest miners find a new block ahead of the ST miner, the system transits to state **0**. Namely, the ST miner starts to mine new blocks based on the public chain. Here, the difference between state **0''** and state **0'** lies in that only the ST miner knows the existence of the private chain in state **0''**, while in state **0'** the honest miners can freely choose to mine on one of the two chains. The comparisons between the three stubborn mining strategies are given in Figure 15. Simulations in [134] show that stubborn mining strategies can improve the profit by up to 25% than the original selfish mining strategy proposed in [61].

The author in [135] studies the impact of transaction fees on selfish mining strategies in the Bitcoin network. Note that due to the inherent design of the token issuing scheme in Bitcoin, the constant mining reward of each block halves every time when a fixed interval of blocks, i.e., every 210,000 blocks, is generated. Then, it is natural to increase the transaction fee to compensate for the mining cost of the consensus nodes. The arbitrary levels of transaction fees lead to a situation where some hidden blocks may have very high values. As a result, selfish miners want to publish it immediately due to the risk of orphaning. Hence, in the revised Markov chain model for selfish mining in Figure 16, the author introduces a new state **0''**. State **0''** is almost identical to state **0**, except

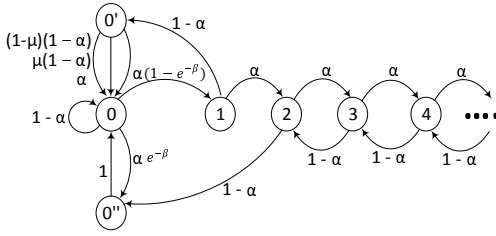


Figure 16. Improved Markov model for selfish mining with transaction fees (adapted from [135]).

that, if the selfish miner mines on the next block in state  $0''$ , it will immediately publish that block instead of holding it. Compared with the original selfish mining model in Figure 13, state  $0$  transits to state  $1$  with probability  $\alpha(1 - e^{-\beta})$  and to state  $0''$  with probability  $\alpha e^{-\beta}$ , where  $\beta$  is the size of the mining block. The new factor  $\beta$  is introduced to model the impact of transaction fees on the miner's decisions. With the revised transition probability, if the selfish miner finds a block of high value in state  $0$ , it may publish the block (i.e., transiting to state  $0''$ ) instead of holding it (i.e., transiting to state  $1$ ). The analysis in [135] shows that this improved selfish mining strategy leads to positive profit for all miners regardless of their hashrates.

From the aforementioned Markov models, we note that the selfish miner may adopt various policies by choosing to release an arbitrary number of block in each state. In [136]–[138], a Markov Decision Process (MDP) model is proposed to generalize such a process of policy derivation. As an example, the study in [136] considers the honest miners as non-adaptive players following the Nakamoto protocol. Then, the problem of searching optimal selfish-mining strategy can be modeled as a single-player MDP. Four actions are considered to control the state transitions in the MDP:

- *Adopt*: the selfish miner accepts the honest network's chain and all private blocks are discarded;
- *Override*: when taking the lead, the selfish miner publishes its private blocks such that the honest network discards its current view;
- *Match*: the selfish miner publishes a conflicting branch of the same height. A fraction of the honest network will fork on this branch;
- *Wait*: the selfish miner does not publish new blocks and keeps working on its private branch.

The state the MDP is defined by the difference in block lengths between the selfish miner and the honest network as well as the situation of computation forking among the honest miners. By controlling the maximum difference in block lengths, it is possible to obtain a finite-state MDP. Using standard MDP solution techniques, an  $\epsilon$ -optimal policy for selfish mining can be obtained based on such a truncated-state MDP.

In [139], the authors consider a similar mining competition between a selfish mining pool and the honest nodes. The study in [139] extends the model of selfish mining by considering the propagation delay between the selfish mining pool and the honest community. The delay is assumed to be exponentially distributed with rate  $\mu$ . The block-mining Markov model in [139] adopts a 2-dimensional state of  $(k, l)$ , which denotes the length of blocks built by the pool and the community upon

the common prefix blocks, respectively. Let  $\lambda_1$  and  $\lambda_2$  denote the block-arrival rate for the pool and the community. The authors then derive the following transition rates of the block mining system:

$$\begin{aligned} q_{\square}(k, l), (k+1, l)_{\square} &= \lambda_1, & k \geq 0, l \geq 0, \\ q_{\square}(k, l), (k, l+1)_{\square} &= \lambda_2, & k \geq 0, l \geq 0, \\ q_{\square}(k, l), (0, 0)_{\square} &= \mu, & k < l, \\ q_{\square}(k, k-1), (0, 0)_{\square} &= \mu, & k \geq 2, \\ q_{\square}(k, l), (k', l')_{\square} &= 0, & \text{otherwise.} \end{aligned} \quad (14)$$

Based on this transition map, the authors in [139] propose to detect selfish mining behaviors by monitoring the proportion of orphaned blocks. Specifically, if there is a significant increase in the fraction of orphaned blocks, it is highly possible that selfish mining exists in the network.

In [140], the authors adopt a more general assumption of multiple selfish miners in a Bayesian game-based formulation<sup>16</sup>. In the considered game, miners decide on whether to report a new block (R), i.e., to mine honestly, or not (NR), i.e., to mine selfishly. When a miner makes a decision, it does not know whether it is the real leader of the mining competition, or whether some other miners have secretly started mining on their private blocks. To ease the analysis of this mining game with incomplete information, the authors assume that a miner always reports when it finds two successive blocks. With this extra assumption, a decision tree can be constructed (see Figure 17), and the backward induction approach is adopted to find the miners' equilibrium strategies. Figure 17 presents the decision tree in a case of three miners. In the presented subgame, miner 1 believes that it is the real leader of the mining competition. Here, let  $h_i$  denote the normalized computational power of miner  $i$ , and  $\mu_i(h_i)$  denote miner  $i$ 's belief of being the leader of the puzzle solution competition. From the decision tree and following the Bayesian rule, we can obtain the information about the states, transition probabilities, and expected payoffs after miner 1 takes the action of NR. The authors provide the condition on the fraction of computational power for action NR to become the optimal mining strategy.

2) *Block Withholding in Pool-Based Mining*: Block withholding (BWH) is a mining strategy used by selfish miners to increase their revenues through diminishing the winning probability of honest miners in mining pools [142], [143]. In [143], the authors study the impact of BWH on the Bitcoin network. It is assumed that a selfish miner is able to split the computational power into different mining pools. It may spend most of its computational power to honestly mine on one pool, and use the rest computational power to perform BWH on the other pools. The mining pools are supposed to adopt the pay-per-share protocol [66, Section 2.2]. In the victim mining pools, the selfish miner submits all shares<sup>17</sup> to the pool operators except the valid puzzle solutions. Although this

<sup>16</sup>A Bayesian game [141, Chapter 4] describes the situation when players are of incomplete information. The players' payoffs are determined not only by their strategies but also by their types, which they may not be fully aware during the play.

<sup>17</sup>A share is a preimage solution for a block that meets the relaxed (i.e., approximated) difficulty requirement set by the pool. A miner receives its reward in proportion to the number of shares that it submits to the pool.

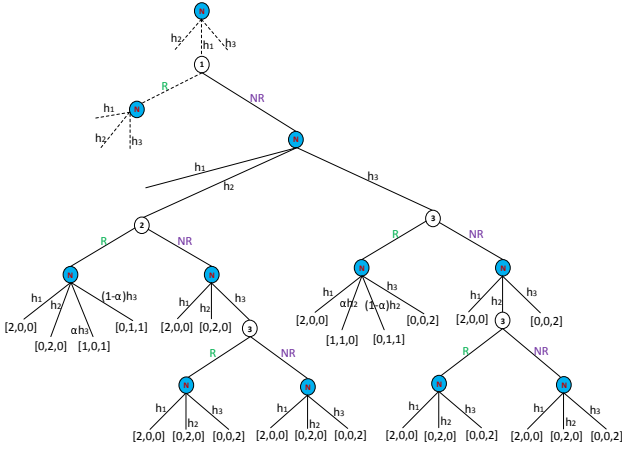


Figure 17. An illustration of the Bayesian mining game (adapted from [140]). Miner 1 believes that its is the real leader of the puzzle solving competition and decides to take action NR. Here  $\alpha$  is the probability for miners to mine on the first block when they receive two blocks in a short time.

mining strategy reduces the attacker's revenue in the attacked pools, it will increase the attacker's revenue in the pool that it chooses to mine honestly. A computational power splitting game with multiple players is formulated in [143]. In the game, one selfish miner adopts BWH and all the other miners mine honestly. The selfish miner chooses which pools to attack and how much computational power to allocate in the targeted pools. It is shown that the attacker always gains positive reward by mining dishonestly regardless of its mining power. This finding implies a risk for big mining pools to dominate the network through BWH attacks on smaller mining pools.

The study in [144] considers a more complicated case where mining pools attack each other with BWH. The author of [144] considers a scenario of two mining pools which attempt to send their miners to each other to diminish their opponents. As illustrated in Figure 18, pool  $P_1$  uses  $x_{12}$  out of the  $m_1$  computational power to attack pool  $P_2$ . Meanwhile, pool  $P_2$  uses  $x_{21}$  out of the  $m_2$  computational power to attack pool  $P_1$ . Then, the revenue of each pool can be derived as follows:

$$\begin{aligned} R_1 &= \frac{m_1 - x_{12}}{m - x_{12} - x_{21}}, \\ R_2 &= \frac{m_2 - x_{21}}{m - x_{12} - x_{21}}, \end{aligned} \quad (15)$$

where  $m$  is the total mining power in the blockchain network. By [144], the revenues of the pools can be expressed as the functions of  $x_{12}$  and  $x_{21}$ :

$$\begin{aligned} r_1(x_{12}, x_{21}) &= \frac{m_2 R_1 + x_{12}(R_1 + R_2)}{m_1 m_2 + m_1 x_{12} + m_2 x_{21}}, \\ r_2(x_{21}, x_{12}) &= \frac{m_1 R_2 + x_{21}(R_1 + R_2)}{m_1 m_2 + m_1 x_{12} + m_2 x_{21}}. \end{aligned} \quad (16)$$

Thus, by observing the attack rate of its opponent, a mining pool can adjust its attack rate in the next round to maximize its long-term revenue through repeated plays. The analysis of this repeated game reveals that the game admits a unique equilibrium, and the pool size will be the main factor that determines the attacking rates of each pool. A similar conclusion about the impact of the pool size on BWH attacks between two pools can also be found in [121].

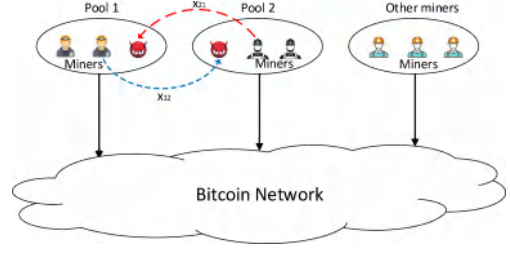


Figure 18. Block withholding attacks between two miners.

Extended from the studies in [143], [144], it is found out in [145] that when a mining pool performs a BWH attack to a victim mining pool, the other mining pools will benefit from this attack even if they do not adopt BWH. Thus, the other pools are interested in sponsoring the attacker to launch the BWH attack to the victim pool. Consequently, the expected gain of the attacker will be greater than the case in [143]. This implies that miners have more incentives to perform BWH attacks with the Nakamoto consensus protocols.

To alleviate the impact of BWH attacks, modifications to the Nakamoto protocol and the pool-mining protocols are suggested in the literature. The author in [66] proposes that the pool operator should insert mining tasks for which the solutions are known in advance, and tag the miners that do not submit the results. Since it is difficult to find puzzles with expected solutions, the author suggests that some new data fields should be added to the conventional block data structure (see Figure 2). These fields enable the pool operator to allocate mining tasks to its miners, but the miners are unable to know the exact puzzle solutions. Alternatively, in [146], the authors propose to give an extra reward to the miners that find the valid blocks, hence reducing the revenue of selfish miners and discouraging BWH attacks.

3) *Lie-in-Wait Mining in Pools*: Lie-in-wait (LIW) is a strategic attack where a selfish miner postpones submitting the block that it finds to a mining pool, and uses all of its computational power resources to mine on that pool [66]. In this case, an attacker is assumed to first split its computational power to mine in different pools. Then, if it finds a block in a pool, instead of submitting the block to get the reward from the pool, the attacker holds the block, and concentrates all of its computational power in other pools to mine on the pool where it finds the block. However, the attacker may take a risk by not releasing the block immediately and concentrating all the computational resources on the target pool. The reason is that if one of other pools finds a new block before this block is published, the selfish miner will lose its reward as well as suffer from the cost of mining in the target pool. It is shown in [66] that the success of attacks follows an exponential distribution, and the maximum expected gain of the LIW attacker is solely determined by the pool numbers and block interval in the network.

4) *Pool Hopping Strategy*: With the strategy of pool hopping, the miners exploit the vulnerability of the payment mechanism of mining pools to increase their own profits. With the pay-per-share protocol, the number of submitted shares in one block competition round follows a geometric distribution with success parameter  $\delta$  and mean  $D$  [66]. For  $I$  shares

Table IV  
SUMMARY OF SELFISH MINING STRATEGIES AND THEIR INCURRED RISKS IN BLOCKCHAIN NETWORKS

| Attacks                | Selfish mining                                                                                                           | Block withholding                                                                                                                 | Lie-in-wait                                                                                                                         | Pool hopping                                                                                                                |
|------------------------|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| References             | [61], [134]–[136], [139], [140]                                                                                          | [121], [143]–[146]                                                                                                                | [66]                                                                                                                                | [66]                                                                                                                        |
| Concept                | After finding a new block, the attacker hides the block and continues mining on the mined block secretly.                | After finding a new block in the victim pool, the attacker discards that block and continues mining on its block in another pool. | After finding a new block in a mining pool, the attacker holds the block and uses all the computational power to mine on that pool. | The attacker moves to another pool or start mining by himself when the mining time at its current pool reaches a threshold. |
| Risks of attackers     | A new attacker's found block can be discarded if one of other miners finds a new block before it finds a next new block. | The attacker loses its reward at the victim pool if it finds a new block in this pool.                                            | The attacker can lose its reward for its mined block and all computational power at the pool it found the block.                    | There is no risk and loss for the attacker if its mining pools use pay-per-share protocol.                                  |
| Risks of honest miners | Lose their rewards for their mined blocks.                                                                               | Lose their rewards for blocks found by attackers.                                                                                 | Can lose their rewards if the block found by the attacker in their mining pool is discarded from the network.                       | Their profits will be reduced if they are in mining pools using pay-per-share protocol.                                     |
| Suggested solutions    | Modification to the mining protocol, e.g., blockchain propagation method and blockchain update rule.                     | Modification to the task assignment protocol in pools such that miners do not know real results of their mining tasks.            | Modification to the task assignment protocol in pools such that miners do not know real results of their mining tasks.              | Change the payment method for mining pools.                                                                                 |

submitted to a pool, the pool still needs  $D$  more shares on average to mine the block. When ignoring the transaction fees, the more shares submitted to a pool in a round, the less each share is worth. Since a miner immediately receives the payment for the submitted share, this implies that a share submitted early may have a higher reward. Therefore, a selfish miner can benefit by mining only at the early stage of a round, and then hop to other pools to increase his revenue. The study in [66] shows that there exists a critical point measured in the number of submitted shares. The best strategy of a selfish miner is to mine on a pool until this point is reached, then hop to another pool or mine by himself.

One straightforward way to address the block hopping problem in pay-per-share mining pools is to increase the value of shares at the end of each round. The pool operator may score the shares according to the elapsed time since the beginning of each round. A share can be scored by an exponential score function  $s(t) = e^{t/\delta}$ , where  $t$  is the time stamp of the submitted share and  $\delta$  is a parameter controlling the scoring rate of shares. With the help of share scoring, we can handle pool hopping attacks in mining pools by decreasing the score of shares at the beginning and increasing the score of shares later. Such score-based method is also known as Slush's method and has been implemented in the mining pools such as Slushpool [147]. In [66], other incentive mechanisms such as pay-per-last- $N$ -shares and payment-contract-based methods are also sketched. However, analytical studies on these mechanisms are missing and their effectiveness in preventing pool hopping attacks still remain an open issue.

## V. VIRTUAL BLOCK MINING AND HYBRID CONSENSUS MECHANISMS BEYOND PROOF OF CONCEPTS

With the consensus protocols and the related issues reviewed in Sections III and IV, a natural question arises regarding whether it is possible to simulate the random leader-election process among permissionless nodes in an approach other than under the framework of Nakamoto-like protocols. To answer this question, we focus on the designing methodology of the virtual-mining protocols in this section. Then, we further introduce a category of protocol design aiming at

performance improvement by combining the properties of both the permissionless protocols and the classical BFT protocols.

### A. Proof of Stake and Virtual Mining

The concept of PoS was first proposed by Peercoin [76] as a modified PoW scheme to reduce the energy depletion due to exhaustive hash queries. Peercoin proposes a metric of “coin age” to measure the miner's stake as the product between the held tokens and the holding time for them. Miner  $i$  solves a PoW puzzle as in (1) with an individual difficulty  $D(h_i)$ . The Peercoin kernel protocol allows a miner to consume its “coin ages” to reduce the difficulty i.e.,  $h_i$ , for puzzle solution. The public verification of the “coin ages” is done through empirically estimating the holding time of the miner's Unspent Transaction Output<sup>18</sup> (UTXO) based on the latest block on the public chain.

By completely removing the structure of PoW-based leader election, the protocols of pure PoS are proposed in [33], [77], [78], [148]. To simulate a verifiable random function following the stake distribution (see also (2)), an algorithm, follow-the-coin (a.k.a., follow-the-satoshi), has been proposed by [78] and widely adopted by these works<sup>19</sup>. Here, the terms “coin” or “satoshi” are used to indicate the minimum unit of the digital tokens carried by the blockchain. Briefly, all the tokens in circulation are indexed, for example, between 0 and the total number of available coins in the blockchain network. A simplified PoS protocol can use the header of block  $t - 1$  to seed the follow-the-coin algorithm and determine the random mining leader for block  $t$ . Specifically, the hash function  $\mathcal{H}(\cdot)$  is queried with the header of block  $t - 1$ , and the output is used as the random token index to initialize the searching algorithm. The algorithm traces back to the minting block (i.e., the first coinbase transaction [33]) for that token or the UTXO account that currently stores it [78]. Then, the creator or the holder of the token is designated as the leader for generating block  $t$ . To enable public verification of the block, the valid leader is

<sup>18</sup>A UTXO is a transaction output whose value has not been spent by the receiver. It can be used as the input of a new transaction. Bitcoin-like networks sum up all the existing UTXOs of an account to recover its balance state.

<sup>19</sup>A reference implementation in Python (see also [78]) can be found at <http://www.cs.technion.ac.il/~iddo/test-fts.py>.

required to insert in the new block its signature, which replaces the data field “nonce” for PoW-based blockchains.

It is worth emphasizing that the pure PoS protocols do not rely on a Poisson process-based puzzle solution competition to simulate the random generator of the block leader. Therefore, the ZK puzzle-solving process can be simply replaced by the process of asymmetric key-based signing and verification, and the proof of resource is no longer needed. For this reason, PoS is also known as a process of “virtual mining” [4] since the block miners do not consume any resources. In the literature, a number of protocol proposals are claimed to be able to (partially) achieve the same purpose. However, these protocols either need special hardware support, e.g., Intel SGX-enabled TEEs for proof of luck/elapsed-time/ownership [79], [149], or are still under the framework of PoW, e.g., Proof of Burn (PoB) [150], Proof of Stake-Velocity (PoSV) [151] and “PoS” using coin age [76]. Strictly speaking, they cannot be considered as the real virtual mining schemes in permissionless blockchain networks.

Compared with the PoX-based protocols, PoS keeps the longest-chain rule but adopts an alternative approach for simulating the verifiable random function of block-leader generation. For this reason, the same framework for analyzing the properties of Byzantine agreements in PoW-based blockchain networks [23] can be readily used for the quantitative analysis of PoS protocols. For example, the investigations in [77], [152] mathematically evaluate the properties of common prefix, chain quality and chain growth based on the same definition in Table II. The authors propose in [77] the “Ouroboros” protocol, and consider that the stakes are distributed at the genesis block by an ideal distribution functionality. By assuming an uncorrupted ideal sampling functionality, Ouroboros guarantees that a unique leader is elected in each block generation round following the stake distribution among the stakeholders (see also (2)). With Ouroboros, forking no longer occurs when all the nodes are honest. However, when adversary exists, forking may be caused by the adversarial leader through broadcasting multiple blocks in a single round. The study in [77] shows that the probability for honest nodes to fork the blockchain with a divergence of  $k$  blocks in  $m$  rounds is no more than  $\exp(-\Omega(k) + \ln(m))$  under the condition of honest majority. It is further shown that the properties of chain growth and chain quality are also guaranteed with negligible probability of being violated.

The studies in [78], [152] introduce the mechanism of epoch-based committee selection, which dynamically selects a committee of consensus nodes for block generation/validation during an epoch (i.e., a number of rounds). Compared with the single-leader PoS protocol, i.e., Ouroboros [77] and its asynchronous variation [153], the committee-based PoS gears the protocol design toward the leader-verifier framework of traditional BFT protocols (see also Figure 6). In [78], the scheme of Proof of Activity (PoA) is proposed with the emphasis that only the active stake-holding nodes get rewarded. The PoA is featured by the design that the leader is still elected through a standard PoW-based puzzle competition, and is only responsible for publishing an empty block. Using the header of this block to seed the follow-the-coin algorithm, a

committee of  $N$  ordered stakeholders is elected and guaranteed to be publicly verifiable. The first  $N - 1$  stakeholders work as the endorsers of the new empty block by signing it with their private keys. The  $N$ -th stakeholder is responsible for including the transactions into that block. The transaction fees are shared among the committee members and the block miner. In this sense, PoA can be categorized as a hybrid protocol that integrates both PoW and PoS schemes.

In [152], the authors propose a protocol called “Snow White”, which uses a similar scheme to select a committee of nodes as in [78]. However, only the selected committee members are eligible for running for the election of the block generation leader. Under the Snow White protocol, the leader of an epoch is elected through a competition based on repeated preimage search with the hash function. At this stage, the difference of Snow White from the standard PoW puzzle in (1) is that the hash function is seeded with the time stamp instead of an arbitrary nonce. Like PoA, Snow White also pertains the characteristics of a hybrid protocol. The analysis in [152] shows that the proposed protocol supports frequent committee reconfigurations and is able to tolerate nodes that are corrupted or offline in the committee.

The recent proposal by Ethereum, Casper [154] provides an alternative design of PoS that is more similar to traditional BFT protocols. The current proposal of Casper does not aim to be an independent blockchain consensus protocol, since it provides no approach of leader election for block proposal. Instead, the stakeholders join the set of validators and work as the peer nodes in a BFT protocol. The validators can broadcast a vote message specifying which block in the blockchain is to be finalized. The validator’s vote is not associated with its identity, but with the stake that it holds. According to [154], Casper provides plausible liveness (instead of probabilistic liveness with PoW) and accountable safety, which tolerate up to  $1/3$  of the overall voting power (weighted by stake) that is controlled by the Byzantine nodes.

## B. Issues of Incentive Compatibility in PoS

Regarding the incentive compatibility of PoS, an informal analysis in [77] shows that being honest is a  $\delta$ -Nash equilibrium<sup>20</sup> strategy when the stakes of the malicious nodes are less than a certain threshold and the endorsers are insensitive to transaction validation cost. However, a number of vulnerabilities are also identified in PoS. In [155], the nothing-at-stake attack is considered. In order to maximize the profits, a block leader could generate conflicting blocks on all possible forks with “nothing at stake”, since generating a PoS block consumes no more resource than generating a signature. A dedicated digital signature scheme is proposed to enable any node to reveal the identity of the block leader if conflicting blocks at the same height are found. Alternatively, a rule of “three strikes” is proposed in [33] to blacklist the stakeholder who is eligible for block creation but fails to properly do so for three consecutive times. In addition, an elected mining leader is also required to sign an auxiliary output to prove that it

<sup>20</sup>At a  $\delta$ -NE, the payoff of each player is within a distance of  $\delta > 0$  from the equilibrium payoff.

provides some extra amount tokens as the “deposit”. In case that this node is malicious and broadcasts more than one block, any miner among the consecutive block creation leaders can include this output as an evidence in their block to confiscate the attacker’s deposit. Such a scheme is specifically designed to disincentivize block forking by the round leader.

Grinding attack is another type of attacks targeting PoS [77]. With PoS, the committee or the leader is usually determined before a round of mining starts. Then, the attacker has incentive to influence the leader/committee election process in an epoch to improve its chances of being selected in the future. When the verifiable random generator takes as input the header of the most recent block for leader/committee election, the attacker may test several possible block headers with different content to improve the chance of being selected in the future (e.g., [77], [78]). It is expected to use an unbiased, unpredictable random generator to neutralize such a risk [77]. In practice, the protocol usually selects an existing block that is a certain number of blocks deep to seed the random function instead of using the current one [78], [152].

With all the aforementioned studies, a significant limit of the existing analyses about PoS-based protocols lies in the simplified assumption that ignores the stake trade outside the blockchain network (e.g., at an exchange market) [156]. A study in [157] provides a counterexample for the persistence of PoS in such a situation. The study in [157] assumes no liquidity constraint in a blockchain network, where nodes own the same stake at the beginning stage. The author of [157] considers a situation where a determined, powerful attacker attempts to destroy the value of the blockchain by repeatedly buying the stake from each of the other nodes at a fixed price. After taking into account the belief of the nodes that the attacker will buy more tokens, the interaction between the attackers and the stakeholders is modeled as a Bayesian repeated game. The study concludes that the success of the attack depends on two factors, namely, the attacker’s valuation of the event “destroying the blockchain” and the profit (e.g., monetary interest) that the nodes can obtain from holding the stake. When the former factor is large and the latter is small, the nodes in the network will end up in a competition to sell their stakes to the attackers. As a result, the blockchain can be destroyed at no cost.

### C. Hybrid Consensus Protocols

Despite the unique characteristics of permissionless consensus protocols, public blockchain networks are known to be limited in performance (e.g., transaction throughput) due to the scalability-performance tradeoff [18]. To boost permissionless consensus without undermining the inherent features such as scalability, a plausible approach is to combine a permissionless consensus mechanism (e.g., Nakamoto protocol) with a fast permissioned consensus protocol (e.g., BFT). Following our previous discussion (cf. PoA [78] and Casper [154]), we study in this subsection how a standard permissionless consensus protocol can be improved by incorporating (part of) another consensus protocol in the blockchain networks.

In [158], the protocol “Bitcoin-NG” is proposed to extend the PoW-based Nakamoto protocols. The prominent feature

of Bitcoin-NG is to decouple the consensus process in a blockchain network (e.g., Bitcoin network) into two planes: leader election and transaction serialization. To bootstrap the transaction throughput, the protocol introduces two types of blocks, namely, the key blocks that require a PoW puzzle solution for leader election and the microblocks that require no puzzle solution and are used for transaction serialization. The time interval between two key blocks is known as an epoch. In an epoch, the same leader is allowed to publish microblocks with the limited rate and block size. Although operation decoupling in Bitcoin-NG does not ensure strong consistency, it paves the way for incorporating additional mechanisms on the basis of standard Nakamoto protocols.

Following the methodology of [158], hybrid consensus mechanisms atop Nakamoto protocols are proposed in [159], [160] with the goal of providing strong consistency and immediate finality. In [159], the “PeerCensus” protocol is proposed by decoupling block creation and transaction committing/confirmation. PeerCensus consists of two core components, namely, a PoW scheme named as BlockChain (BC) and a BFT-based scheme named as Chain Agreement (CA). With the proposed BC protocol, nodes acquire the voting right of the CA protocol when they propose new blocks through PoW and are approved by the committee of CA. The CA protocol is adapted from BFT protocols such as PBFT [17] and the Secure Group Membership Protocol (SGMP) [161]. Through the four stages of propose, pre-prepare, prepare, and commit of BFT protocols (cf. Figure 6), CA designates the miner of the newest block in the chain as the leader for the next block proposal. The leader proposes one from the multiple candidate blocks obtained in BC. The peer nodes in the committee extend the pre-prepare stage with an operation of block validation. The design of PeerCensus ensures that committing transactions (i.e., CA) is independent of block generation (i.e., BC). Therefore, no forking occurs in the condition of honest majority and strong consistency is guaranteed.

In [160], a hybrid consensus protocol is proposed by combining the data framework of two-type blocks in Bitcoin-NG and the hybrid PoW-BFT design in PeerCensus. As in PeerCensus, the Nakamoto protocol is used to construct a “snailchain”, which is allowed to commit transactions from a specific mempool of outstanding transactions known as the “snailpool”. Following the quantitative analysis of the common prefix blocks in a chain in [23], only a fixed number of miners whose recently minted blocks are a certain number of blocks deep in the chain can be used to form the committee for the BFT protocol. In contrast to PeerCensus, the BFT committee of miners in the proposed protocol has no influence on how the next block on the snailchain is determined. Instead, it is responsible for committing transactions from an independent mempool known as the “txpool”. For this reason, the transactions approved by the BFT protocol are committed off the snailchain without relying on any mining mechanism. In this sense, these transactions can be considered similar to those in the microblocks of Bitcoin-NG. The hybrid consensus protocol in [160] explicitly addresses the problem of BFT-committee scalability in PeerCensus and provides a secured (with theoretical proof) consensus property of imme-

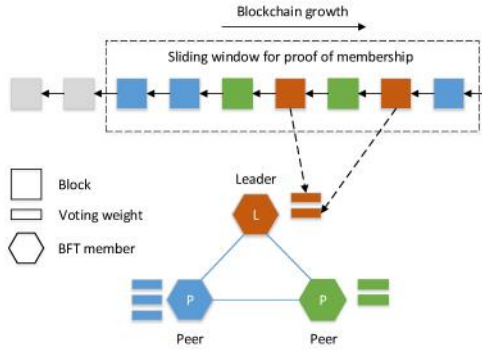


Figure 19. Illustration of BFT-committee formation with weighted voting power. Valid weights are only credited to the miners of the blocks in the sliding window (adapted from [162]).

diate finality. Namely, the transaction confirmation time for the txpool only depends on the network’s actual propagation delay. The method of using Nakamoto protocols to select nodes into a BFT committee is also known as the proof of membership mechanism [162]. A sliding-window mechanism is proposed in [162] to generalize the mechanisms of dynamic BFT-committee selection in [159], [160]. As illustrated in Figure 19, the BFT committee is maintained by a fixed-size sliding window over the PoW-based blockchain. The sliding window moves forward along the blockchain as new blocks are appended/confirmed. Consensus nodes minting multiple blocks in the window are allowed to create the same number of pseudo-identities in the BFT consensus process to gain the proportional voting power.

For hybrid consensus using BFT protocols to guarantee strong consistency, a natural thinking is to replace the Nakamoto protocols with virtual mining (e.g., PoS) for selecting the leader or committee in BFT-consensus processes. A typical example for such an approach can be found in the “Tendermint” protocol [163], where a node joins the BFT committee of block validators by posting a bond-deposit transaction. The validator no longer needs to prove its membership by competing for the PoW-puzzle solution. Alternatively, its voting power is equal to the amount of stake measured in bonded tokens. Meanwhile, instead of randomly electing the leader of block proposal in the committee (cf. [158]), Tendermint adopts a round-robin scheme to designate the leader in the committee. The similar design can be found in a number of recent proposals such as Proof of Authority (PoAu) [164] and Delegated Proof of Stake (DPoS) [165]. To generalize the mechanisms of BFT-committee selection based on virtual mining, the authors in [166] further propose a consensus protocol called “Algorand”. Like the other hybrid protocols, Algorand relies on BFT algorithms for committing transactions. It assumes a verifiable random function to generate a publicly verifiable BFT-committee of random nodes, just as in [78]. The probability for a node to be selected in the committee is in proportion to the ratio between its own stake and the overall tokens in the network. For leader election, Algorand allows multiple nodes to propose new blocks. Subsequently, an order of the block proposals is obtained through hashing the random function output with the nodes’ identities specified by their stake. Only the proposal with the highest priority will be propagated across the network.

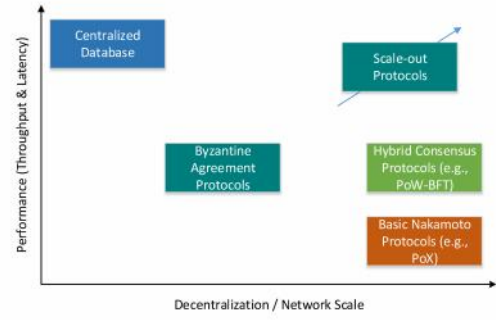


Figure 20. Illustration of performance and scalability of different consensus protocol families (see also the discussion in [18]).

In Table V, we provide a summary of the virtual-mining mechanisms and the hybrid consensus protocols discussed in this section.

## VI. RELAXED AND PARALLEL CONSENSUS PROTOCOLS FOR PERFORMANCE SCALABILITY

So far, we have surveyed the design methodologies of various consensus protocols, especially for permissionless blockchains. As our discussion indicates, the BFT-based consensus mechanisms achieve high transaction throughput with immediate finality at the cost of high message complexity. Thus, they are restricted to small numbers of replicas and offer limited network scalability in terms of the number of consensus nodes. In contrast, the permissionless protocols surveyed in Sections III and V provide good network scalability with low message complexity. However, most of the Nakamoto-like protocols (except the hybrid protocols guaranteeing immediate finality [159], [160]) provide only probabilistic consensus finality. As a result, consistency of replicas across the entire network (cf. the consistency condition for the PoW-based protocol in (8)) is maintained at the cost of low transaction throughput and high latency. Figure 20 provides a descriptive illustration of the scalability levels of different protocol families with respect to both performance and network size. For the protocols surveyed in our previous sections, network scalability and transaction throughput are generally considered as two performance indices that can only be attained at the cost of each other. In this section, we aim to review the solutions that scale out the throughput of a permissionless blockchain as the size of the network increases.

### A. Off-chain and Side-chain Techniques

For cryptocurrencies, one popular and straightforward approach to throughput enhancement is to adjust the parameters, e.g., the block size and confirmation time in Nakamoto-like protocols. A typical example of this approach can be found in the Segregated Witness proposal (SegWit) [167] for Bitcoin soft fork, which lifted the block-size limit from 1MB to 4MB. However, the study in [95] points out that such a reparameterization approach is constrained by the network’s bandwidth (e.g., for block size) as well as the blockchain’s security requirement (e.g., confirmation time). Thus, such an approach does not really scale out the throughput as the network size increases. With the emphasis on compatibility to the existing consensus protocol or network realization, alternative approaches, e.g., the Lightning network [168], that aim to lower

Table V  
SUMMARY OF VIRTUAL MINING AND HYBRID CONSENSUS PROTOCOLS FOR PERMISSIONLESS BLOCKCHAINS

| Protocol Name                                                                 | Virtual Mining | Hybrid Consensus | Simulating Leader Election with                                    | Rule of Longest Chain | Decoupling Block Proposal from Transaction Commitment                             | Featured Consensus Properties                                              |
|-------------------------------------------------------------------------------|----------------|------------------|--------------------------------------------------------------------|-----------------------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| Proof of stake [33], [77], [148]                                              | Yes            | No               | Verifiable random function, e.g., follow-the-coin                  | Yes                   | N/A                                                                               | No resource consumption                                                    |
| Proof of luck, elapsed-time and ownership [79], [149]                         | Yes            | No               | Trusted random function implemented by Intel-SGX-protected enclave | Yes                   | N/A                                                                               | No resource consumption. Special hardware support is needed                |
| Proof of burn [150]                                                           | Partially      | No               | PoW puzzle competition                                             | Yes                   | N/A                                                                               | Reduced resource consumption                                               |
| Proof of stake-velocity [151]                                                 | Partially      | No               | PoW puzzle competition                                             | Yes                   | N/A                                                                               | Reduced resource consumption                                               |
| Snow White [152]                                                              | Partially      | PoS-PoW          | Modified preimage search with the hash function                    | Yes                   | N/A                                                                               | Robust consensus through reconfigurable PoS committee                      |
| Proof of activity [78]                                                        | Partially      | PoW-PoS          | PoW puzzle competition for empty block proposal                    | Yes                   | Transactions are committed by a random group of stakeholders                      | Higher cost for attackers to compromise the network consensus than PoW/PoS |
| Casper [154]                                                                  | No             | PoW-PoS          | PoW puzzle competition                                             | Yes                   | N/A                                                                               | Validators use BFT protocols to anchor checkpoint blocks in the block tree |
| Bitcoin-NG [158]                                                              | No             | Partially        | PoW puzzle competition                                             | Yes                   | Proposals of microblocks do not need PoW solutions                                | Leader election is only performed at key blocks                            |
| PeerCensus [159]                                                              | No             | PoW-BFT          | PoW puzzle competition                                             | N/A                   | Yes, Blocks are committed by BFT committees                                       | Strong consistency without blockchain forking                              |
| Hybrid consensus protocol [160]                                               | No             | PoW-BFT          | PoW-puzzle competition in the snailchain                           | Yes                   | Partially, only the transactions in txpools are committed following BFT protocols | Immediate finality                                                         |
| Tendermint [163], Proof of authority [164] and delegated proof of stake [165] | Yes            | PoS-BFT          | Verifiable random function or deterministic mechanism              | N/A                   | Yes, following typical BFT protocols                                              | Deterministic consensus properties                                         |
| Algorand [166]                                                                | Yes            | PoS-BFT          | Verifiable random function                                         | N/A                   | Yes, following typical BFT protocols                                              | Safety and liveness are guaranteed under strong synchrony                  |

the frequency of global block validation/synchronization, are proposed by the development communities, specifically for value transfer networks.

The Lightning network [168] and its variations such as Blind Off-chain Lightweight Transactions (Bolt) [169] and the TEE-based Teechain [170] introduce the concept of (bidirectional) micro-payment channels between two nodes via untrusted intermediary relays. Specifically, the payment channels are realized as logical channels overlaying on the existing blockchains (e.g., on Bitcoin [168] or on ZCash [169]) and therefore do not modify the underlying consensus protocols. The value transfer between the two end nodes on each channel is kept “off-chain” as a local sequence of mutually-agreed balance-state updates, also known as commitment transactions [168]. In other words, the sequence of transactions on an established channel are not broadcast to the entire network and kept locally between the two end nodes as well as the intermediaries when needed. Then, transactions of value transfer over a channel are not confirmed as normal transactions and cannot be spend until the “closure” of the channel. When closing the channel, only the most recent commitment transaction is broadcast and needs to be mined by the blockchain network. By doing so, the requirement of validating/synchronizing every transaction across the network is relaxed and the number of transactions to be mined is greatly reduced, hence making the underlying blockchain network more throughput-scalable.

Due to the lack of trust, simply relaxing the consensus re-

quirement and keeping transactions in local payment channels will incur the risk of double spending. To address this problem, the technique of 2-of-2 multisignature<sup>21</sup> is enforced in the Lightning networks and a number of specifically designed smart contracts (i.e., scripts in Bitcoin) are introduced. To establish a channel, a funding transaction has to be created jointly by the end parties and broadcast to the network in order to lock their submitted tokens in escrow. An order of broadcast is defined by creating for each party a different version of every subsequent commitment transaction, i.e., in the form of a half-signed transaction containing only the signature of the counterparty, with the same balance outputs. An accompanying revocable transaction<sup>22</sup> is also created to enable updating the balance changes. It also provide a means of revoking transactions in case a violation occurs or a waiting time limit is reached. In normal scenarios, only the latest commitment transaction is broadcast to close the channel. Otherwise, by broadcasting the right version of revocable transactions, one end node is able to provide the publicly verifiable proof of recognizing a malicious behavior by the counterparty, and claim all of its deposit in the funding

<sup>21</sup> An **m-of-n** “multisig” transaction requires the verification of a tuple of at least **m** signatures for the same text from **n** corresponding public keys [171].

<sup>22</sup> A revocable transaction has two payout paths. If both parties of it agree, its output can be spent immediately. Otherwise if after a certain waiting time one or both parties do not broadcast, the fund can be redeemed. It is revoked only when both parties agree to update with a superseding transaction.

transaction as a punishment.

Other than the off-chain schemes that aim to reduce the amount of transactions over the network, an alternative design is to extend an existing blockchain-based value transfer network with multiple “side-chains” [172]. A side-chain is an independent blockchain network that validates a subset of transactions and keeps track of the corresponding assets. Such a design introduces parallelism into the existing network and each side-chain is only responsible for validating a fraction of the total amount of transactions in the network. Therefore, it is able to increase the transaction throughput by adding more side-chains. As in the off-chain techniques, side-chains do not modify existing consensus protocols. Instead, the fundamental goal is to enable bidirectional atomic value transfer between side-chains. More specifically, any value transaction between side-chains is either completely confirmed by both side-chains or not at all. Meanwhile, the value carried by the transaction can be imported from and returned to a side-chain with no risk of double spending. To achieve such a goal (also known as “two-way peg” in [172]), special proofs of value locking and redeeming are needed whenever inter-chain transfer happens. Especially, since the consensus nodes of the receiving side-chain usually do not track the state changes of the sending side-chain, providing a compact, non-interactive proof of events occurring on side-chains becomes the utmost concern of the network designers.

In [172], the Simplified Payment Verification (SPV) proof is adopted from [1] based on the proof-of-inclusion path in Merkle trees to provide compact proofs of value locking for atomic transfer (cf. Figure 8). Further enhancement of the proof is also proposed in [172] by introducing a trusted cross-chain federation of mutually distrusting functionaries (i.e., approving nodes). Out of the federation, the majority vote in the form of an  $m$ -of- $n$  multisignature is used to replace the SPV proof for locking/redeeming a cross-chain pay-to-contract transaction. Furthermore, an SPV proof is accompanied by an array of block headers, whose parent is the block containing the SPV-locked transaction on the sending side-chain. This can be informally considered as a “proof of PoW” shown to the receiving side-chain that the transaction in concern is sufficiently deep in the sending side-chain and thus safely locked (see also our discussion about (8)). In [173], a formal primitive called Non-Interactive-Proofs-of-Proof-of-Work (NIPoPoW) is proposed to fill the gaps of compactness and non-interactiveness in the proposal of [172] for PoW-based side-chain networks. To avoid tracking/validating every block on the sending side-chain, the study in [173] proposes to replace the linear list-based blockchains with a skiplist-like data structure called interlink (see Figure 21 and also [174]). As with SPV, a valid NIPoPoW of transaction confirmation also contains an array of blocks (i.e., suffix proof) preceded by the block in concern as a stability proof of that block in the chain. Instead of validating the entire source side-chain, NIPoPoW only has to include  $2m$  blocks in expectation from each level of the hierarchical blockchain in the proof. Here,  $m$  is a system-determined security parameter to ensure that for every level  $\mu$ , the proof only needs to include a number of blocks from the tail of level  $\mu$  to span the last  $m$ -size suffix of

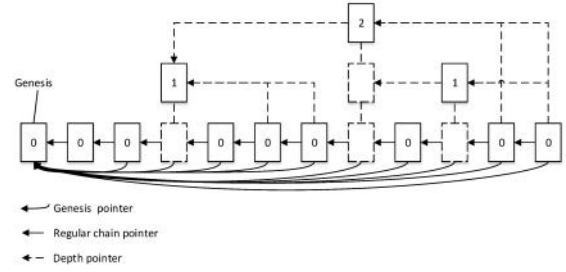


Figure 21. A graphical example of the hierarchical blockchain with levels 0, 1 and 2. A block with header  $\mathbf{bh}$  is of level  $\mu$  if  $\mathbf{bh} < \mathbf{D}(\mathbf{h})/2^\mu$  (see also (1)). Besides the regular hash pointer to the previous block, a block of level  $\mu$  also maintains a list of hash pointers (interlinks) to the most recent preceding blocks in every level  $\mu'$  such that  $\mu' > \mu$ . The genesis block is defined to be of infinite level and hence every other block has to include a pointer to it.

blocks in the higher level  $\mu + 1$ . Compared with a secured SPV proof for inter-chain transaction, with NIPoPoW the number of source-chain blocks tracked by the receiving side-chain is only a polylogarithmic function of the source side-chain’s length.

### B. Sharding for Scale-out Throughput

Inspired by the infrastructures of distributed database and cloud, the concept of “sharding” [95] is also applied to the blockchain networks. As in side-chain networks, the approach of sharding partitions the global blockchain state into parallel subsets (i.e., shards), and each shard is maintained by a subgroup (i.e., committee) of nodes instead of the entire network. To improve the transaction throughput as well as retain the open-membership nature of permissionless blockchains, multiple BFT committees can be constructed following a similar procedure of the hybrid protocols (cf. Section V-C). As a result, the sharding protocols generally face the same challenges as in side-chain networks and hybrid consensus protocols, i.e., in providing secured shard formation to guarantee permissionless decentralization and in providing cross-shard synchronization to guarantee atomic transactions.

The study in [175] adopts the UTXO structure from Bitcoin and proposes the “spontaneous sharding” mechanism specifically for value transfer networks. Spontaneous sharding introduces a level of individual (spontaneous) chains for each node to maintain its own transactions of interest in a first-in-first-out fashion. It keeps a globally shared main chain, which only records the signed abstracts (i.e., header) of the blocks on each individual chain using a BFT-based consensus protocol. In this sense, spontaneous sharding is considered to be a transitional design from micro-payment channels to sharding, since it admits only the transaction-sharding process but not the validator-sharding process. The validity of the proposed mechanism is built upon the assumption that all nodes in the network are rational. Namely, a node is interested in inspecting a transaction only if it needs that transaction to validate a subsequent transaction output that it receives. Only the rational owner of an unspent transaction is responsible for providing the proof to the validators (i.e., receivers). However, due to the existence of sharded individual chain, the protocol in [175] faces an unresolved problem of lacking compact proof (cf. [173]), since for every proof, the validators have to trace back to the genesis block of each related individual chain.

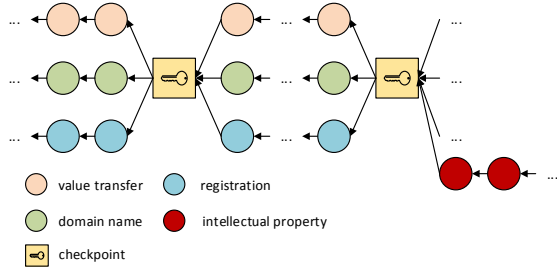


Figure 22. Service oriented sharding with multi-chain structure (adapted from [176]). PoW solution is required for generating a checkpoint. Users are able to propose new services by posting transactions to register the corresponding channels in a checkpoint block (see the sub-blockchain for the “intellectual property” service).

In [176], a different approach of transaction sharding is proposed under the name of “Aspen”. Instead of maintaining an individual chain for each node, Aspen organizes transactions into sub-blockchains (see Figure 22) based on the type of services related to each transaction. It introduces periodic checkpoint blocks for synchronizing sub-blockchains (cf. the anchor points in Casper [154]). Aspen is instantiated on Bitcoin-NG [158] and only requires the checkpoint blocks to be generated upon PoW-puzzle solution to determine the proposal leaders of micro-blocks in each service channel (i.e., sub-blockchain). To avoid designing complex proofs of cross-chain transactions (cf. [172], [175]), Aspen does not allow two-way transfer between channels and requires that each fund is only spendable in a specific channel.

In [177], a different sharding protocol named “Elastico” is proposed with the emphasis on the process of validator sharding through dynamically forming multiple BFT-committees. Elastico organizes the transaction approving process by epochs, and in each epoch a number of committees are formed in parallel based on the PoW-puzzle solution in a similar way to the proof of membership in [162]. The study in [177] proposes a mechanism of generating distributive epoch randomness by using one network-level BFT committee, which determines a subset of hash values randomly provided by its members. The committee can run any non-leader interactive consistency protocol, e.g., [178] to reach an agreement on such a single set to generate the public random number. In an epoch, the candidates of the committees have to solve the PoW puzzle based on the public random number. Elastico also uses the least-significant bits of the PoW solution (i.e., the hash value) to group the candidate nodes into different committees. Thus, this procedure guarantees that the committees are randomly formed and unpredictable. Meanwhile, to avoid designing complex proofs of cross-shard transactions (cf. [175]), Elastico relies on the network-level committee to merge the locally agreed values in each committee into a single chain. The network-level committee first checks whether the values received from each local committee are signed by their majority members. If so, it merges the received values into an ordered union and runs a similar BFT protocol to approve the final result with signatures by the committee majority. By limiting the burden of quadratic message complexity within shard committees of small size, Elastico is able to achieve roughly  $O(n)$  message complexity and provide almost linear throughput scalability in terms of the hash power in the

network. Also, compared with the aforementioned throughput-scalable protocols, e.g., [171], [172], [175], Elastico does not limit itself to value transfer networks and can be applied to generic data services with non-spendable transactions.

By enabling parallelization of both data storage and network consensus, protocols aiming at “full sharding” are proposed in [179], [180]. In [179], a protocol named “OmniLedger” is designed to provide “statistically representative” shards for permissionless transaction processing. As in [177], OmniLedger is built upon two levels of epoch-based Byzantine agreement processes, with the network level being responsible for epoch randomness generation and the shard level for intra-committee consensus. In the network level, a global identity blockchain is adopted and can only be extended by the network-level leaders. Any node that wants to join a committee has to register to this global blockchain through a Sybil-proof identity establishment mechanism. Especially, such a mechanism is not limited to PoW and can be replaced by other means, e.g., PoS. At the beginning of an epoch, all the nodes with established identities are required to run an interactive consistency protocol by sharing with each other a “ticket” based on a gossip protocol. The ticket is generated as the hash value of the node’s address and the header of the identity blockchain. The node that generates the smallest ticket will be elected as the network-level leader. The leader is expected to run a verifiable random function (e.g., RandHound [181]) and generate a global random string with a valid proof. Upon reception of this random string, other registered nodes are able to compute a permutation based on this string as well as their own identity, and then finish the assignment of shard committees by subdividing their results into equally-sized buckets. In addition, OmniLedger proposes to swap gradually in-and-out committee members per epoch. This design not only allows bootstrapping new nodes joining the network, but also avoids excessive message overhead and latency due to complete shard reconstruction (cf. Elastico). In the shard level, a committee can employ any leader-based BFT protocol (e.g., ByzCoin [162]) to provide intra-shard consensus.

In [180], another epoch-based, two-level-BFT protocol for full sharding is proposed under the name “RapidChain”. In the network level, RapidChain requires a reference BFT-committee to run a distributed randomness generation protocol similar to [177] and generate a public random string to initialize the formation of shard-level committees. As in [179], the shard-level committee reconfiguration in RapidChain only reorganizes a subset of committee members at each epoch to ensure operability during committee transition. At the bootstrapping stage in a network of  $n$  nodes, the established identity of a node is mapped to a random position in the range  $[0, 1)$  by using the hash function. Then, with some constant  $k$  (i.e., committee size), the range is partitioned into  $n/k$  regions, and the shard-level committees are consequently formed based on this region partition. At the reconfiguration stage, RapidChain defines the set of the first half shard-level committees with more active members as the “active committee set”. The network-level committee is responsible for assigning new nodes into the active shard-level committees uniformly at random. After that, it shuffles a constant number of members

from every existing committee and randomly reassign them to other committees. On the shard level, RapidChain requires the members of each BFT-committee to run also the distributed randomness generation protocol and generate a local random string. Then, the committee members compete for the leader election through solving the standard PoW puzzle based on the local random string. The members elect a node with the smallest PoW solution by gossiping their votes with signatures to each other. Then, the BFT protocol will be led by that node to reach the intra-shard consensus for transaction commitment.

As in [175], [176], full sharding also partitions the storage of the blockchain state into multiple shards (e.g., local ledgers). Then, the full sharding protocols [179], [180] are characterized by their ways of handling cross-shard transactions to guarantee atomic transaction commitment. In [179], OmniLedger uses UTXO to represent the client’s balance state. Therefore, a cross-shard transaction is always associated with at least an input shard as well as an output shard (see Figure 23(a)). OmniLedger adopts a lock-unlock-abort mechanism by requiring the input shard of a cross-shard transaction to “lock” the input first. Namely, the leader of the input shard has to provide a proof-of-acceptance in the form of Merkel proof before the corresponding transaction can be committed. If the transaction is found to be invalid, the input shard creates a proof-of-rejection in a similar form by using a designated bit to indicate an acceptance or rejection. Even with a proof-of-acceptance, the receiving client still cannot freely spend the UTXO. The receiver is required to send an unlock-to-commit transaction with that proof to the output-shard committee. Until the output shard validates this special transaction and includes it into the new block, the receiver is able to spend the UTXO of the original transaction.

In [180], RapidChain proposes a different approach of committing cross-shard transaction, which does not require a receiver to collect proofs from the input shards. Instead, for any input value of a transaction from a different shard, the output-shard leader is required to create a single-in-single-out transaction where the output is equal to the input of the original transaction. By doing so, the output committee tries to create a local record of the input and holds the input value in escrow. To confirm the escrow, the output-shard leader is responsible for sending this new transaction back to the input-shard committee for approval. After the input committee adds this transaction into its ledger, the output-shard leader will create a final transaction, with the UTXO of the escrow transaction being the input and the same outputs of the original transaction. After the output-shard committee adds the final transaction to its ledger, the transfer process is finished and the corresponding UTXO becomes spendable by the receivers. An illustrative comparison between the protocols of cross-shard transactions in OmniLedger and RapidChain is given by Figure 23.

### C. Nonlinear Block Organization

Another approach aimed at improving the network throughput focuses on the design of transaction data organization. As we briefly introduce in Section II-B, instead of organizing

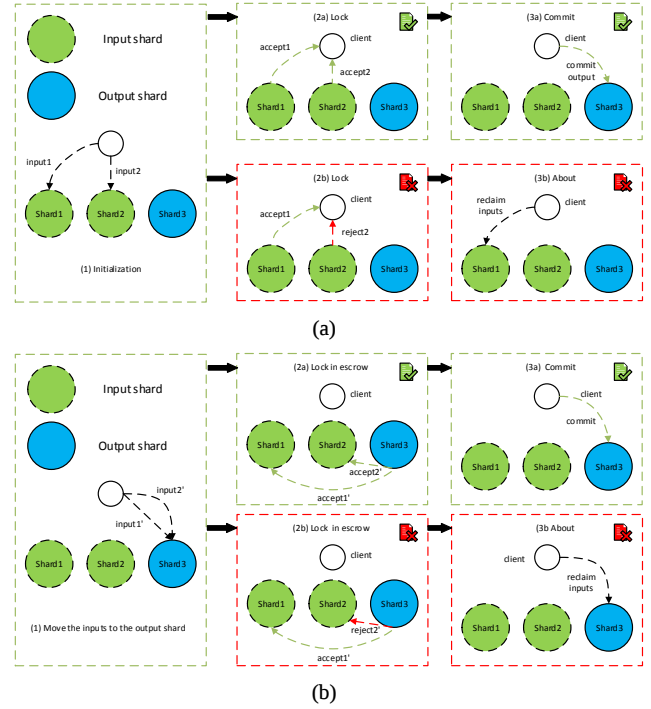


Figure 23. Atomic cross-shard transaction protocols in (a) OmniLedger [179] and (b) RapidChain [180]. In the two protocols, different parties are responsible for collecting input-shard approvals for committing transactions.

block in a linear list, the approaches of nonlinear block organization are able to (partially) address the scalability problem by changing the mechanism of transaction validation in the consensus layer. The earliest scheme of nonlinear block organization can be found in [25] as the protocol of Greedy Heaviest-Observed Sub-Tree (GHOST). In a GHOST-based network, nodes store all the locally observed valid blocks and consequently maintain a tree of their respective forks. As an alternative to the longest-chain rule, GHOST extends the canonical chain of PoW-generated blocks by the block with the heaviest subtree, i.e., the subtree with the largest number of tree-nodes (see Figure 24). In [38], a unified security description of GHOST and the Nakamoto protocol is established by slightly modifying the  $K$ -consistency property in [94] (see also Section III-B) into a new property of  $K$ -dominance, which measures the discrepancy in the weights between sibling subtrees. As pointed out in [25], the rate of main-chain growth of GHOST is lower than that of the longest-chain rule when the block generation rate and the network delay are the same. However, since GHOST relaxes the block-generation constraint for the same level of security requirement against 51% attacks, it is able to shorten safely the waiting time for block confirmation and thus has a limited ability of improving the network throughput.

A further step toward nonlinear block organization is proposed in [182], where blocks are ordered in a DAG and each block is allowed to have multiple predecessors (cf. single parent block in GHOST [25]). Namely, the header of each block may contain more than one pointer to the precedent blocks to indicate the pairwise order. The DAG-based protocol in [182] also selects a main chain (cf. GHOST) of linear order from the DAG. To form such a linear order on the blocks at the current view, a node runs for each block a postorder traversal

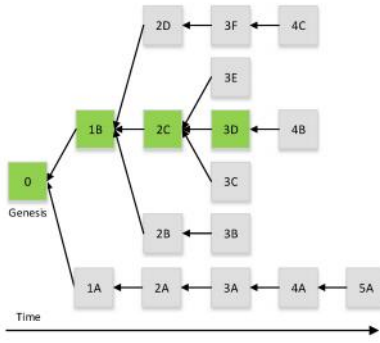


Figure 24. A tree of blocks. Instead of choosing the longest chain 1A to 5A), Block 1B with a subtree weight 11 is selected into the main chain. Consequently, Blocks 2C (with a subtree weight 5) and 3D (with a weight 2) are selected into the main chain of the current view.

algorithm on the DAG and checks if the transactions current block are consistent with the visited one. Con with the longest-chain rule or GHOST, the DAG-based chain expansion allows the non-conflicting, off-chain blocks to be selectively included into the ledger view. For example, from the perspective of a main-chain block, its off-chain descendant blocks can still be included into the ledger as long as they are not far away from the main chain as both predecessor and descendants. Then, by including the discarded (i.e., off-chain) blocks, the proposed protocol possesses a limited ability to increase the network throughput.

To further improve the network throughput, the protocol proposed in [182] is later extended to the protocol “SPECTRE” in [26]. SPECTRE relaxes the requirement on node synchronization, and allows blocks to concurrently grow on the ledger without specifying a main branch. To define the rule of ledger extension, SPECTRE introduces a virtual pairwise voting mechanism to determine the order of any pairwise blocks in the DAG. In brief, each block in the DAG contributes to the vote on the relative order of not only its preceding blocks but also its descendant blocks according to the topology of the DAG. Compared with the main chain-based rules, SPECTRE is shown to be robust to block-withholding attacks (cf. [143]). The reason is that with vote-based pairwise ordering, secret chains published by the attackers cannot win the votes by existing blocks from the honest nodes due to the lack of connections in the DAG (see Figure 25). Without undermining the network security, i.e., increasing the transaction reversal probability, SPECTRE admits faster commitment time as the block creation process is accelerated. By (4), the more nodes in the network, the higher the expected block generation rate is given a fixed PoW difficulty. As indicated by [26], for a target transaction-reversal probability, a known propagation delay and a fixed PoW-difficulty level, SPECTRE is able to increase the transaction throughput as the network size increases.

Based on the aforementioned protocols, a number of DAG-based schemes have been proposed with a variety of emphasis on different performance indices. For example, Byteball [183] adopts the concept of main chain/tree (see also [25], [38], [182]) but uses authenticated witnessing nodes to determine the partial order of blocks at each user’s view. Another DAG-based protocol, i.e., Conflux [184] modifies GHOST by adding in each new block the reference pointers to all existing blocks without descendants at the current DAG view.

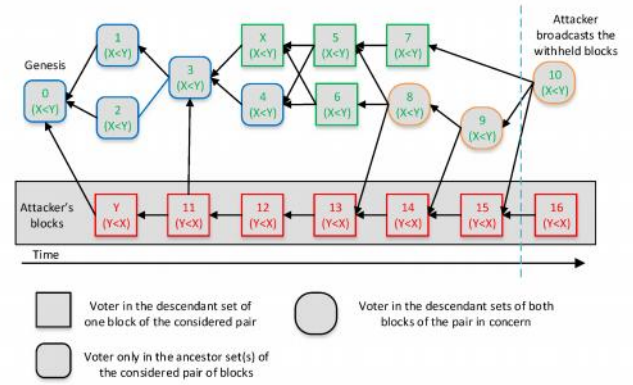


Figure 25. An example of the virtual voting procedure on the order of blocks  $X$  and  $Y$  in a DAG with block withholding attacks. Blocks (voters) in the descendant set of  $X$  will vote  $X < Y$  (i.e.,  $X$  preceding  $Y$ ) since they only see  $X$ . Blocks 0-4 will vote  $X < Y$  since they see more  $X < Y$  votes in their sets of descendant block. Blocks 8-10 which have both  $X$  and  $Y$  as the ancestors run a recursive query to their predecessor sets and use the majority voting results as their own votes.

Compared with [25], [182], Conflux is claimed to provide 100% utilization of the off-chain blocks and thus is able to improve the network scalability. Furthermore, a similar protocol to SPECTRE is proposed in [24], [185] as IoTA Tangle. The major difference of IoTA Tangle lies in that it discards the data structure of block as a package of transactions. Instead, it requires nodes to publish directly transactions onto the transaction DAG. A node is enforced by the protocol to approve/reference more than two transactions by linking their hash values in the header of its new transaction to expand the DAG. By doing so, the node expects to accumulate sufficient weight (cf. votes on the partial orders in SPECTRE) for this transaction from the future transactions<sup>23</sup> by other nodes to finally confirm it. So far, complete theoretical proof of the liveness property of IoTA Tangle is still an open issue [24], [185]. However, the study in [185] implies that, if self-interested nodes have the same capability of information acquisition and transaction generation as the other nodes, they will possibly reach an “almost symmetric” Nash equilibrium. Namely, they will be forced to cooperate with the network by choosing the default parent-selection strategy followed by the honest nodes.

## VII. EMERGING APPLICATIONS AND RESEARCH ISSUES OF BLOCKCHAINS WITH PUBLIC CONSENSUS

In the previous sections, we have provided an in-depth survey on three main categories of permissionless consensus protocols for blockchain networks, namely, the Nakamoto-like protocol based on PoX puzzles, the virtual mining and hybrid protocols and the emerging open-access protocols emphasizing the scale-out performance. On top of the consensus provided by these protocols, the blockchain is able to fully exert its functionalities such as smart contracts for a wide range of applications. In general, we can divide the studies on the emerging blockchain-based applications into two categories: the service provision atop the blockchain consensus layer and the consensus provision to existing blockchain frameworks.

<sup>23</sup>As in SPECTRE, an IoTA transaction (indirectly) approves/references an earlier transaction if it can reach that transaction via directed links.

The former category of studies usually exploit special characteristics of blockchain networks, e.g., self-organization and data security, to guarantee target features in their respective services. In contrast, the latter emphasizes the P2P or decentralized characteristics of blockchain networks. Hence, most of them focus on rational nodes' strategies or the overlaid incentive mechanism design of resources allocation in the consensus process. In this section, we provide an extensive review on the properties of blockchain networks and the applications exerting mutual influence on each other. Meanwhile, a series of open research issues are also identified.

### A. General-Purpose Data Storage

The Cambridge's 2017 annual blockchain benchmarking study identified that the majority of blockchains use cases are still dominated by the capital market sectors [186]. Nevertheless, significant effort has recently been put into the study of using blockchains for storage of generic data, which aims at preserving the properties of data immutability and trackability in a decentralized environment. A naive approach is to "piggy-back" arbitrary data (e.g., non-transferable metadata) onto transactions in established public blockchains [187]. For example, in the Bitcoin network, nodes can use the special script instruction `OP_RETURN` to indicate that the transaction output is unspendable and expected to be removed from the UTXO. Then, the transaction is allowed to carry a limited length of arbitrary data onto the chain. Typical examples of directly storing metadata onto blockchains can be found in asset ownership registration, e.g., Namecoin<sup>24</sup> [188] as a blockchain-based namespace system. Note that the direct on-chain storage is limited by the message length and naturally requires full replication of each data object over the network. Then, this solution needs to be improved to lift the data-length constraint and reduce the synchronization cost. In [189], where a naming system is constructed on top of Namecoin, the data storage is decoupled from the block serialization (i.e., name registration) process. In order to achieve this, the authors of [189] adopt a "virtualchain" to process registration/modification operations of names (e.g., domain names or IP addresses). Only the minimal metadata, i.e., the hashes of the name-payload pairs and state transitions are stored on the blockchain. The third party storage is connected by virtualchain to store the payload of arbitrary length with digital signatures from the data owner.

The same idea of decoupling the storage layer from the main blockchain can also be found in works such as [190]–[192]. The studies in [190], [191] focus on data storage and sharing for large-scale IoTs. Therein, two similar blockchain frameworks are proposed by introducing the off-chain storage. In brief, the data generated by IoT devices is stored in DHTs, and only the pointer to the DHT storage address needs to be published onto the blockchain. The DHT-based storage is provided by an off-chain layer of decentralized DHT nodes. Upon seeing that transactions of storing/accessing requests are confirmed by the blockchain, the DHT nodes are responsible for accordingly storing or sending the data from/to the

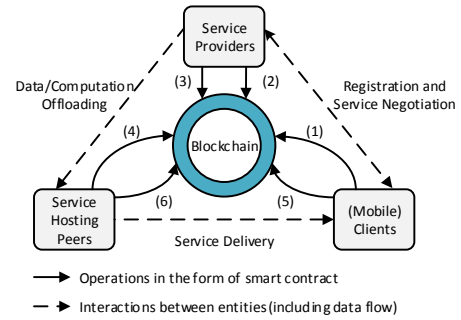


Figure 26. A generic framework of using blockchains as system integrators for self-organization. The operation flow is realized as a sequence of smart contracts: (1) service registration/requests by the clients, (2) access/certificate granting by the providers, (3) requesting service hosting (e.g., auction for computation/storage offloading) by the providers, (4) peers answering (e.g., bidding for) the hosting requests, (5) delivery negotiation between hosting peers and clients and (6) service completion with proofs of delivery.

legitimate IoT nodes. In [192], further discussion is provided regarding the issue of how to control the data replication factor in the network. Instead of using an off-chain storage layer, the design in [192] compromises the property of decentralization in exchange for a stronger control of replication synchronization. In the proposed framework of blockchain-like database, i.e., BigchainDB, P2P communication protocols are replaced by the built-in broadcasting protocol, and a committee (i.e., federation) of voting nodes are designated for block validation and ordering. Such a permissioned design shares a certain level of similarity with the framework of HyperLedger [39]. By doing so, it is possible for the federation nodes to control where to store a submitted transaction and flexibly determine the replication factor (i.e., the number of shards/replicas) per table in the underlying distributed database. Such design avoids the issue of full data replication over the network and makes it possible for constructing a large-scale, high-throughput database directly on a blockchain network.

### B. Access Control and Self-Organization

The most popular design approach sees blockchains as enabling technologies for implementing accountable and secure services in a decentralized fashion. In other words, blockchains are utilized as a decentralized intermediary for channeling/accounting services upon demands as well as for guaranteeing data security and confidentiality. In Figure 26, we describe a generic framework of decentralized service provision built upon blockchains. The most prominent feature of this framework lies in that the interactions between different entities in the system are all tunneled autonomously in the form of smart contracts. Such a framework has been adopted by a wide range of service provision systems including P2P file sharing based on InterPlanetary File System<sup>25</sup> (IPFS) [36], decentralized content delivery [193], [194], access control in telecommunication networks [37], [195] and various missions for access and permission management, e.g., in IoTs [196] and clouds [197]. For different task requirements, this application framework can be expanded by including additional entities, e.g., third-party auditors [198], as well as new operations, e.g., Hierarchical Identity Based Encryption (HIBE, see

<sup>24</sup><https://namecoin.org>.

<sup>25</sup><https://github.com/ipfs/ipfs>.

also [199]) [200]. To provide a better idea on how this emerging framework can be shaped in recent studies, we categorize the blockchain-based proposals for self-organization according to the areas or context that they are applied in.

1) *Access Control in Wireless Networks:* In [195], the authors propose to use blockchains for providing Identity and Credibility Service (ICS) in cloud-centric Cognitive Radio (CR) networks. The CR users utilize their pseudonymous identities on the blockchain to negotiate with the network operator, i.e., the spectrum owner, for granting opportunistic access and settling payment. According to [195], the ICS can be provided by either the blockchain itself or a third-party entity registered on-chain, and the network access negotiation is automated by smart contracts. Meanwhile, it is pointed out in [195] that the blockchain's consensus mechanism can be employed for coordinating spectrum sensing among the distributed CR users. However, it is not clear how the CR-user consensus can be achieved on top of the ledger consensus as with the classical methods [201] in CR networks.

In another study [202], the same authors propose to use a permissioned blockchain to handle the network access exchange, i.e., the spectrum handoffs. The CR users and their base station controller submit the information of spectrum and network utilization as metadata onto the blockchain. Then, the CR network responds by updating the smart contracts and publishing the new access prices and number of network access units allocated to each CR onto the blockchain for execution. A similar design with more technical details can be found in [203]. Therein, a blockchain based on the Nakamoto protocol with its embedded tokens and smart contract layer is adopted as a spectrum auction platform. More specifically, multiple primary users as providers sell their unused bands at a certain price with smart contracts and allocate them to responding CR users when the contracts are executed upon certain conditions. It is claimed in [203] that the blockchain-based spectrum allocation outperforms the conventional medium-access protocols such as Aloha. However, technical details are missing about how the issue of high transaction latency is addressed to satisfy the CR network's constraint due to the timescale of small-scale fading in wireless channels.

Blockchains are also introduced into vehicular ad-hoc networks (VANETs) to address the issues of network volatility due to high mobility. For Vehicles-to-Infrastructure (V2I) communication, the study in [204] uses the Nakamoto-based blockchain as a secure key-delivery channel to handle the access of a moving vehicle to groups of Road Side Units (RSUs) in different regions. By encapsulating the key information in a blockchain transaction, the security manager of one region is responsible for issuing the transactions to that of the new region as well as mining the new blocks onto the blockchain. Comparatively, the study in [205] focuses more on the ad-hoc nature of VANETs and employs the blockchain to collect the trustworthiness rating on messages sent to each other by the peer vehicles. The RSUs do not only work as the consensus nodes in the blockchain network but also work as the decentralized storage hosting peers of the trust rating data (cf. Figure 26). It is worth noting that in [205] the transactions carrying vehicle reports are essentially unspendable. The RSUs

employ weighted average to the rating scores to estimate the quality of the received reports. Then, they use the estimation results as the difficulty parameter for PoW-based mining in a similar manner of the Peercoin-like protocols (see also Section V-A).

In the existing studies on blockchains-based network access control, the study in [206] is among the few to explicitly address the issue of high signaling latency over the blockchain due to the adoption of Nakamoto protocols. In [206], the process of authentication transfer for User Equipments (UEs) in a 5G ultra dense network is handled by a blockchain in a similar way as in [202]. Instead of delegating the transaction/contract execution process to a dedicated overlay blockchain, it is proposed in [206] that the Access Points (APs) use the PBFT protocol within a dynamic consensus committee to handle the requests of authentication by UEs in the form of transactions or smart contracts. In order to implement the PBFT protocol, a local server center is introduced as the primary peer (i.e., leader) of the committee. Nevertheless, we note that any non-leader consistency protocol can be adopted in this framework to preserve the property of complete decentralization (see also Section VI-B). According to [206], the PBFT-based blockchain is able to keep the transaction delay around 100ms. Compared with the standard Nakamoto protocols, it is more practical to deploy network control mechanisms over PBFT-based blockchains for delay-critical tasks such as access hand-over. However, how to find a balance between the required levels of latency and decentralization (e.g., with hybrid consensus protocols) still remains an open question.

2) *Self-Organization and Security Enhancement under Various Network Architectures:* Apart from network access control, blockchains have also been applied to various scenarios as a decentralized platform for self-organization. As briefly shown in Section VII-B1, blockchains can also be used for security enhancement with its embedded cryptographic functionalities. Typical examples for the former applications can be found in proactive caching and Content Delivery Networks (CDNs) [193], [194], [207]. In [194], a decentralized CDN platform is established with the help of blockchains among the three parties of content providers, content serving peers and clients (cf. Figure 26). With smart contracts, the content providers offload the tasks of content delivery to multiple content serving peers. It is suggested in [194] that the content providers use smart contract prices to control the file placement on multiple serving peers according to the demand frequency and achievable QoS at the peers. Furthermore, the work in [193] mathematically formulates the pricing-response interaction between the providers and the serving peers as a potential game [141, Chapter 3.4]. Then, it designs a series of smart contracts for automatically matching the peers to the providers under the same CDN framework. A modified PoS protocol is subsequently proposed to incentivize the serving peers to work as the consensus nodes of the blockchain without consuming significant computational power.

In [207], the authors design a blockchain-based brokering platform for video delivery in a user-centric CDN ecosystem. The proposed platform is built upon three independent blockchains for content brokering, delivery monitoring

and delivery provisioning, respectively. The content broking blockchain handles the content requests and matches the clients' requests to the providers' offers in a series of smart contracts among the three parties. The delivery monitoring blockchain records proofs of delivery and finalizes the payment and refund between the providers and the clients. The delivery provisioning blockchain provides smart contracts for content dissemination between the providers and the serving peers. In such a framework, the decentralized entities in the CDN treat the blockchain as a ready-to-use service offered by a third party. Therefore, any form of blockchains (e.g., the permissioned HyperLedger) can be employed as long as the requirement of transaction throughput and latency is met.

In various applications of edge/fog/cloud computing, more and more attempts are also found to use blockchains for providing services such as trusted auditing and secured data delivery in addition to autonomous brokering. In [198], the blockchain is used as a tamper-proof provenance database on the cloud server side to record the history of the creation and operations performed on a cloud data object. By adopting a public blockchain, any node in the blockchain network is able to perform data auditing. By using pseudonymous identities on blockchains, the proposed auditing mechanism reduces the probability that auditors can correlate the real identity of a specific user with the operations. In other works such as [196], [208], the blockchain is introduced into the three-layer paradigm of edge-fog-cloud computing. In [196], the blockchain is used as a connector to provide encrypted channel by using the public key functionality for data delivery from the edge devices to the fog and cloud. More specifically, the study in [196] considers a smart video surveillance network, where the preprocessing tasks such as object tracking are handled at the edge devices, and the more sophisticated tasks of data aggregation and decision making are performed in the fog/cloud based on the data filtered at the edge. To prevent malicious modification on video frames in the untrusted fog layer, the cloud layer deploys smart contracts on the blockchain to provide an indexing service and generate unique index for every video frame with transactions published onto the blockchain. The work in [208] adopts the same data-processing flow from the edge to the cloud as in [196]. In contrast to [196], the blockchain is used to provide automatic matching between the data-service requests and the providers in the cloud's service provider pool. In this sense, the blockchain is again used to provide the broking service as in [193], [194].

3) *Trusted Broking Services in Cyber-Physical Systems*: In the context of crowdsourcing (e.g., crowdsourcing of mobile sensors, a.k.a., crowdsensing), permissionless blockchains are also found to be especially appropriate for providing non-manipulable brokering services between clients (i.e., task requesters) and service providers (i.e., crowdsourcing workers). In [209], a purely decentralized crowdsourcing system for general purpose is proposed following the paradigm described by Figure 26. In the proposed framework, the procedures of identity registration, task/receiving, reputation rating and reward assignment are all automated in the form of smart contracts. Following the approaches described in Section VII-A, the blockchain network delegates the data storage to an inde-

pendent storage layer and only keeps the metadata on-chain. Similar blockchain-based frameworks are also adopted for crowdsensing in recent studies such as [210], [211], where additional functionalities are adopted in the blockchain networks to address different performance requirement such as throughput scalability [210] and anonymity enhancement [211].

In the context of IoTs, blockchain-based infrastructure is also envisioned as a promising alternative of the centralized one for data management, trading automation and privacy protection. In [212], the authors introduce the micro-payment channels (see also Section VI-A) based on a Bitcoin-like blockchain to conduct energy trading in a decentralized smart grid without relying on trusted third parties. In [213], a P2P surplus-energy trading mesh of the plug-in hybrid electric vehicles is built on a Nakamoto protocol-based blockchain. In the proposed framework, a number of authorized nodes are responsible for processing and recording the transactions and an iterative double auction mechanism is deployed based on the transactions published on the blockchain. This framework of blockchains as a P2P trading mediator is also adopted in [214], [215], where the PBFT protocol is used to replace the Nakamoto protocol and form a consortium blockchain. Furthermore, the mathematical tool of contract theory (see [216] for more details) is adopted to determine the optimal prices and requested utility in the relevant smart contracts.

### C. Consensus Provision and Computation Offloading under Nakamoto Protocols

In contrast to the studies that we review in Sections VII-A and VII-B, another line of works focus on (decentralized) resource allocation for consensus provision in the Nakamoto-based blockchain networks. In other words, these studies view the consensus in blockchain networks of a given protocol as the goal to be achieved instead of a ready-to-use service. Recall that the Nakamoto protocols require consumption of certain resources in the PoW-like puzzle solution competition for new block proposing (see also Section III). With this property in mind, a plethora of works, e.g., [131], [132], [217]–[220], are devoted to the studies of resource allocation in the block mining process in exchange for monetary rewards (i.e., mining reward in tokens) offered by the blockchain. In [131], [218], [219], a scenario of deploying blockchains on the mobile edge devices is considered. Due to the intensive resource consumption for PoW solution, it is difficult to directly migrate blockchain networks to the mobile environment [218]. Therefore, the computation offloading schemes are proposed in these studies by either formulating the problems in a nonlinear/binary programming framework [219] or as a hierarchical (i.e., Stackelberg) game [131], [218].

We use [131] as an example to explain how the PoW-work offloading process can be formulated as a conventional optimization or game theoretic problem. To offload the tasks of PoW-solution searching from mobile devices to the edge/fog/cloud, a series of factors including transaction transmission delay and blockchain-forking probability need to be considered when constructing the utility model of the mobile node at the edge. Considering that the computation

providers at the edge/fog are able to control the price of the offered computational resource, the offloading process is modeled in [131] as a two-stage Stackelberg game. In brief, the cloud/fog providers act as the leader to set the resource price, and the edge devices act as the follower to determine the share of resource to purchase for offloading the mining tasks. According to the various assumptions about the offloading scenarios (e.g., multi-leader vs. single leaders), different approaches such as nonlinear optimization formulation or best response-based equilibrium searching are applied to each layer's subproblem in the manner of backward induction [141, Chapter 3.4.2]. Extending from the basic scenarios in [131], [219], various tools of mechanism design, e.g., auctions [132], [217], can be further applied into the similar offloading problems for resource allocation in the blockchain consensus process.

#### D. Some Open Issues and Potential Directions

In the existing literature on blockchains, a number of open issues have been discussed regarding the non-consensus layers in blockchains, e.g., the issues of security and privacy [20] and quantitative analysis of smart contract performance [221]. In the following, we discuss issues and emerging research directions that have not been covered in the surveyed works.

1) *Cost of Decentralization*: The properties of permissionless blockchains such as trustlessness and self-organization have been widely recognized as the advantage over the conventional ledger/brokering systems. However, decentralization with blockchain networks is not “at no cost”. As we have partly discussed in Section VI, even the scalable consensus protocols do not completely solve the problem of balancing between the requirement of security and resource efficiency. For instance, how to adaptively control the replication factor in shards still remains an open issue.

Furthermore, consider that historical data such as spent transactions become huge as the blockchain grows. With the current design of append-only chains, it seems inevitable for ordinary nodes to eventually run out of storage and for the blockchain network to be controlled by a few powerful nodes. Then, it is plausible to seek an approach for “pruning” the blockchain data without undermining its immutability. Although hard forks such as SegWit [167] can be considered a manual pruning process, it is better expected that the out-of-date blocks “have the right to be forgotten” [222]. Unfortunately, except a handful of experimental proposal [223], [224], the issues of data pruning, e.g., how to delete obsolete transactions and migrate UTXOs buried in the chain, also remains an open issue.

2) *Support for Secure Big-Data Computation*: In the existing research, privacy concerns for blockchains are mostly placed on the levels of identity registration and encrypted data delivery (see Section VII-B). With more and more demands for big-data processing in various fields [225], [226], the question arises regarding whether it is also possible to provide on- or off-blockchain support for secure Multi-Party Computation (MPC). For example, hospitals may want to learn patterns for diagnosis by using the private electronic medical records from the patients without seeing the raw data. In such a scenario,

the existing privacy policies offered by blockchains (e.g., access authentication) turn out to be insufficient. This issue is partially touched in [227] for mobile federated learning, where each node connected to the blockchain trains on the same structure of deep neural network with the local data. Then, they only exchange the locally trained model for global model aggregation [228]. Note that in [227] the blockchain is merely used to conduct a convoy of the locally trained parameters as in [196]. Following such design arises a natural question, namely, how can we directly offer general-purpose, privacy-preserving MPC on-chain (e.g., in blocking mining work) or off-chain with decentralized providers (cf. Figure 26)?

The question above generally remains unaddressed, and only a few works [229], [230] can be found in the literature with limited strength for specific-purpose MPC provision. These works are based on the framework of cryptographic MPC techniques and allow mutually trustless parties to compute a joint function directly on their encrypted inputs to obtain the right outcome. In [229], the multi-parties store their public-key-encrypted data on an off-chain storage plain as in [190], while in [230] the encrypted data is stored directly on a permissioned blockchain (e.g., HyperLedger). However, due to the quadratic message complexity of the existing MPC protocols [229], only a small number of computation parties can be supported on-chain [230]. Moreover, only a limited number of mathematical operations (e.g., polynomial functions) are supported by the protocols, and the MPC-based blockchain framework is still far from matured.

## VIII. CONCLUSIONS

In this paper, we have provided a comprehensive survey on the recent development of blockchain technologies, with a specific emphasis on the designing methodologies and related studies of permissionless, distributed consensus protocols. We have provided in the survey a succinct overview of the implementation stacks for blockchain networks, from where we started our in-depth investigation into the design of consensus protocols and their impact on the emerging applications of blockchain networks. We have examined the influence of the blockchain consensus protocols from the perspective of three different interested parties, namely, the deployers of blockchain networks, the consensus participants (i.e., the consensus nodes) in the blockchain networks and the users of blockchain networks.

We have provided a thorough review of the blockchain consensus protocols including BFT-based protocols, Nakamoto protocols, virtual mining and hybrid protocols, for which we highlighted the link of permissionless consensus protocols to the traditional Byzantine agreement protocols and their distinctive characteristics. We have also highlighted the necessity of incentive compatibility in the protocol design, especially for the permissionless blockchain networks. We have provided an extensive survey on the studies regarding the incentive mechanism embedded in the blockchain protocols. From a game-theoretic perspective, we have also investigated their influence on the strategy adoption of the consensus participants in the blockchain networks.

Based on our comprehensive survey of the protocol design and the consequent influence of the blockchain networks, we have provided an outlook on the emerging applications of blockchain networks in different areas. Our focus has been put upon how traditional problems, especially in the areas of telecommunication networks, can be reshaped with the introduction of blockchain networks. This survey is expected to serve as an efficient guideline for further understanding about blockchain consensus mechanisms and for exploring potential research directions that may lead to exciting outcomes in related areas.

## REFERENCES

- [1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Self-published Paper*, May 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [2] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.
- [3] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2084–2123, third quarter 2016.
- [4] 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 *2015 IEEE Symposium on Security and Privacy*, San Jose, CA, May 2015, pp. 104–121.
- [5] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, May 2016.
- [6] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, May 2016, pp. 839–858.
- [7] K. Yeow, A. Gani, R. W. Ahmad, J. J. P. C. Rodrigues, and K. Ko, "Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2018.
- [8] F. Glaser, "Pervasive decentralisation of digital infrastructures: A framework for blockchain enabled system and use case analysis," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, Waikoloa, HI, Jan. 2017.
- [9] N. Kshetri, "Can blockchain strengthen the internet of things?" *IT Professional*, vol. 19, no. 4, pp. 68–72, Aug. 2017.
- [10] N. Bozic, G. Pujolle, and S. Secci, "Securing virtual machine orchestration with blockchains," in *2017 1st Cyber Security in Networking Conference (CSNet)*, Rio de Janeiro, Brazil, Oct. 2017, pp. 1–8.
- [11] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology - CRYPTO '87: Conference on the Theory and Applications of Cryptographic Techniques*, C. Pomerance, Ed., Santa Barbara, CA, Aug. 1987, pp. 369–378.
- [12] A. Mohr, "A survey of zero-knowledge proofs with applications to cryptography," Southern Illinois University, Carbondale, Tech. Rep., 2007.
- [13] O. Goldreich, "Zero-knowledge twenty years after its invention," IACR Cryptology ePrint Archive, Report 2002/186, 2002, <https://eprint.iacr.org/2002/186>.
- [14] M. Raynal, *Communication and agreement abstractions for fault-tolerant asynchronous distributed systems*, ser. Synthesis Lectures on Distributed Computing Theory. Williston, VT: Morgan & Claypool Publishers, May 2010.
- [15] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, Dec. 1990.
- [16] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Sok: Consensus in the age of blockchains," *arXiv preprint arXiv:1711.03936*, 2017.
- [17] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, Nov. 2002.
- [18] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *Open Problems in Network Security: IFIP WG 11.4 International Workshop*, Zurich, Switzerland, Oct. 2015, pp. 112–125.
- [19] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," School of Data and Computer Science, Sun Yat-sen University, Tech. Rep., 2016.
- [20] M. Conti, S. K. E. C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys Tutorials*, pp. 1–1, May 2018, early access.
- [21] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software*, Uppsala, Sweden, Apr. 2017, pp. 164–186.
- [22] S. Ghosh, *Distributed systems: an algorithmic approach*. CRC press, 2014.
- [23] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*, Sofia, Bulgaria, Apr. 2015, pp. 281–310.
- [24] S. Popov, "The tangle version 1.4.3," IOTA Foundation, Tech. Rep., Apr. 2018.
- [25] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *19th International Conference on Financial Cryptography and Data Security*, San Juan, Puerto Rico, Jan. 2015, pp. 507–527.
- [26] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," IACR Cryptology ePrint Archive, Report 2016/1159, 2016, <https://eprint.iacr.org/2016/1159>.
- [27] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems: First International Workshop*, Cambridge, MA, Mar. 2002, pp. 53–65.
- [28] Ethereum Foundation, "Ethereum wire protocol v.5," <https://github.com/ethereum/wiki/wiki/Ethereum-Wire-Protocol>, accessed: 2017-11-15.
- [29] —, "Whisper protocol v.5," <https://github.com/ethereum/go-ethereum/wiki/Whisper>, accessed: 2017-11-15.
- [30] <https://github.com/telehash/telehash.github.io>, accessed: 2017-11-15.
- [31] JSON-RPC Working Group, "Json-rpc 2.0 specification," 2012.
- [32] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, 2014.
- [33] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *International Conference on Financial Cryptography and Data Security*, Christ Church, Barbados, Feb. 2016, pp. 142–157.
- [34] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*, Gothenburg, Sweden, Apr. 2017, pp. 243–252.
- [35] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," Ethereum Foundation, Tech. Rep., 2014. [Online]. Available: <https://github.com/ethereum/wiki/wiki/White-Paper>
- [36] Protocol Labs, "Filecoin: A decentralized storage network," Protocol Labs, Tech. Rep., Aug. 2017.
- [37] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Network*, vol. 32, no. 5, pp. 112–117, Sep. 2018.
- [38] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain," IACR Cryptology ePrint Archive, Report 2016/545, 2016, <https://eprint.iacr.org/2016/545>.
- [39] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL 2016)*, Chicago, IL, Jul. 2016.
- [40] F. Baldimtsi, A. Kiayias, T. Zacharias, and B. Zhang, "Indistinguishable proofs of work or knowledge," in *Advances in Cryptology - ASIACRYPT 2016*, Hanoi, Vietnam, 2016, pp. 902–933.
- [41] D. Chatzopoulos, M. Ahmadi, S. Kosta, and P. Hui, "Floppcoin: A cryptocurrency for computation offloading," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1062–1075, May 2018.
- [42] J. Backman, S. Yrjö, K. Valtanen, and O. Mmmel, "Blockchain network slice broker in 5g: Slice leasing in factory of the future use case," in *2017 Internet of Things Business Models, Users, and Networks*, Copenhagen, Denmark, Nov. 2017, pp. 1–8.
- [43] A. Mackenzie, S. Noether, and Monero Core Team, "Improving obfuscation in the cryptonote protocol," Monero Research Lab, Tech. Rep., Jan. 2015.

- [44] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," Zerocoin Electric Coin Company, Tech. Rep., Dec. 2017.
- [45] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proceedings of IEEE International Conference on Peer-to-Peer Computing*, Trento, Italy, Sep. 2013, pp. 1–10.
- [46] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonimisation of clients in bitcoin p2p network," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, Scottsdale, AZ, USA, 2014, pp. 15–29.
- [47] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," *arXiv preprint arXiv:1801.03998*, 2018.
- [48] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. O'Reilly Media, Inc., 2014.
- [49] J. R. Douceur, "The sybil attack," in *First International Workshop on Peer-to-Peer Systems*, Cambridge, MA, Mar. 2002, pp. 251–260.
- [50] V. Buterin, "Bitcoin network shaken by blockchain fork," *Bitcoin Magazine*, vol. 12, Mar. 2013. [Online]. Available: <https://bitcoinmagazine.com/articles/bitcoin-network-shaken-by-blockchain-fork-136144406>
- [51] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *Advances in Cryptology – CRYPTO 2001: 21st Annual International Cryptology Conference*. Santa Barbara, CA: Springer Berlin Heidelberg, Aug. 2001, pp. 524–541.
- [52] M. Correia, N. F. Neves, and P. Verssimo, "From consensus to atomic broadcast: Time-free byzantine-resistant protocols without signatures," *The Computer Journal*, vol. 49, no. 1, pp. 82–96, Jan. 2006.
- [53] C. Cachin and M. Vukolic, "Blockchain Consensus Protocols in the Wild (Keynote Talk)," in *31st International Symposium on Distributed Computing (DISC 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 91, Vienna, Austria, 2017, pp. 1:1–1:16.
- [54] A. Miller and J. J. LaViola Jr, "Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin," University of Central Florida, Computer Science, Tech. Rep., Apr. 2014. [Online]. Available: <http://nakamotoinstitute.org/research/anonymous-byzantine-consensus>
- [55] F. Sun and P. Duan, "Solving byzantine problems in synchronized systems using bitcoin," *Self-published Paper*, Sep. 2014. [Online]. Available: <https://allquantor.at/blockchainbib/pdf/sun2014solving.pdf>
- [56] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolic, "Xft: Practical fault tolerance beyond crashes," in *12th USENIX Symposium on Operating Systems Design and Implementation*, Savannah, GA, Nov. 2016, pp. 485–500.
- [57] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.
- [58] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On bitcoin and red balloons," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, ser. EC '12. New York, NY: ACM, Jun. 2012, pp. 56–73.
- [59] S. Athey, I. Parashkevov, V. Sarukkai, and J. Xia, "Bitcoin pricing, adoption, and usage: Theory and evidence," Stanford Institute for Economic Policy Research, Tech. Rep. Working Paper No. 3469, Aug. 2016.
- [60] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is bitcoin a decentralized currency?" *IEEE Security Privacy*, vol. 12, no. 3, pp. 54–60, May 2014.
- [61] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable (revised selected papers)," in *Financial Cryptography and Data Security: 18th International Conference*, Christ Church, Barbados, Mar. 2014, pp. 436–454.
- [62] O. Ersoy, Z. Ren, Z. Erkin, and R. L. Lagendijk, "Information propagation on permissionless blockchains," *arXiv preprint arXiv:1712.07564*, 2017.
- [63] J. Wu, S. Guo, H. Huang, W. Liu, and Y. Xiang, "Information and communications technologies for sustainable development goals: State-of-the-art, needs and perspectives," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2389–2406, thirdquarter 2018.
- [64] J. Debus, "Consensus methods in blockchain systems," Frankfurt School of Finance & Management, Blockchain Center, Tech. Rep., May 2017.
- [65] J. A. Kroll, I. C. Davey, and E. W. Felten, "The economics of bitcoin mining, or bitcoin in the presence of adversaries," in *Proceedings of The Workshop on the Economics of Information Security (WEIS)*, vol. 2013, Washington, D.C., Jun. 2013.
- [66] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *arXiv preprint arXiv:1112.4980*, 2011.
- [67] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.
- [68] C. Cachin, "Yet another visit to paxos," *IBM Research, Zurich, Switzerland, Tech. Rep. RZ3754*, 2009.
- [69] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Luxembourg City, Luxembourg, Jun. 2018, pp. 51–58.
- [70] H. Kopp, C. Bösch, and F. Kargl, "Koppercoin – a distributed file storage with financial incentives," in *12th International Conference on Information Security Practice and Experience*, Zhangjiajie, China, Nov. 2016, pp. 79–93.
- [71] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols (extended abstract)," in *Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99)*, Leuven, Belgium, Sep. 1999, pp. 258–272.
- [72] J. Aspnes, C. Jackson, and A. Krishnamurthy, "Exposing computationally challenged byzantine impostors," Yale University, Tech. Rep. 136144406/TR-1332, 2005.
- [73] J. Alwen and B. Tackmann, "Moderately hard functions: Definition, instantiations, and applications," in *Proceedings of the 15th International Conference on Theory of Cryptography: Part I*, Baltimore, MD, Nov. 2017, pp. 493–526.
- [74] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, New York City, NY, Oct. 1999, pp. 120–130.
- [75] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," in *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference*, Santa Barbara, CA, Aug. 2013, pp. 90–108.
- [76] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *Self-published Paper*, Aug. 2012. [Online]. Available: <https://peercoin.net/assets/paper/peercoin-paper.pdf>
- [77] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology – CRYPTO 2017: 37th Annual International Cryptology Conference*, Santa Barbara, CA, Aug. 2017, pp. 357–388.
- [78] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake (extended abstract)," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, Dec. 2014.
- [79] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proceedings of the 1st Workshop on System Software for Trusted Execution*, ser. SysTEX '16, Trento, Italy, Dec. 2016, pp. 2:1–2:6.
- [80] M. O. Rabin, "Transaction protection by beacons," *Journal of Computer and System Sciences*, vol. 27, no. 2, pp. 256 – 267, 1983.
- [81] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric proof-of-work based on the generalized birthday problem," *Ledger Journal*, vol. 2, pp. 1–30, Apr. 2017.
- [82] A. Miller, A. Kosba, J. Katz, and E. Shi, "Nonoutsourcable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. Denver, CO: ACM, Oct. 2015, pp. 680–691.
- [83] T. Moran and I. Orlov, "Rational proofs of space-time," Bar-Ilan University Cyber Center, Tech. Rep., 2017.
- [84] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work," IACR Cryptology ePrint Archive, Report 2017/203, 2017, <https://eprint.iacr.org/2017/203>.
- [85] S. Al-Kuwari, J. H. Davenport, and R. J. Bradford, "Cryptographic hash functions: Recent design trends and security notions," IACR Cryptology ePrint Archive, Report 2011/565, 2011, <https://eprint.iacr.org/2011/565>.
- [86] J. A. Garay, A. Kiayias, and G. Panagiotakos, "Proofs of work for blockchain protocols," IACR Cryptology ePrint Archive, Report 2017/775, Aug. 2017.
- [87] D. Kraft, "Difficulty control for blockchain-based consensus systems," *Peer-to-Peer Networking and Applications*, vol. 9, no. 2, pp. 397–413, Mar 2016.
- [88] K. Saito and H. Yamada, "What's so different about blockchain? – blockchain is a probabilistic state machine," in *2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Nara, Japan, Jun. 2016, pp. 168–175.
- [89] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol with chains of variable difficulty," in *Advances in Cryptology – CRYPTO 2017*, Santa Barbara, CA, Aug. 2017, pp. 291–323.

- [90] L. Fan and H.-S. Zhou, "A scalable proof-of-stake blockchain in the open setting (or, how to mimic nakamoto's design via proof-of-stake)," IACR Cryptology ePrint Archive, Report 2017/656, 2017, <https://eprint.iacr.org/2017/656>.
- [91] M. B. Taylor, "The evolution of bitcoin hardware," *Computer*, vol. 50, no. 9, pp. 58–66, 2017.
- [92] [www.coinmarketcap.com](https://coinmarketcap.com/coins/views/all/), <https://coinmarketcap.com/coins/views/all/>, accessed: 2017-11-15.
- [93] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols," IACR Cryptology ePrint Archive, Report 2015/1019, 2015, <https://eprint.iacr.org/2015/1019>.
- [94] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Paris, France, May 2017, pp. 643–673.
- [95] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security: International Workshops on BITCOIN, VOTING and WAHC*, Christ Church, Barbados, Feb. 2016, pp. 106–125.
- [96] P. R. Rizun, "Subchains: A technique to scale bitcoin and improve the user experience," *Ledger*, vol. 1, pp. 38–52, 2016.
- [97] X. Liu, W. Wang, D. Niyato, N. Zhao, and P. Wang, "Evolutionary game for mining pool selection in blockchain networks," *IEEE Wireless Communications Letters*, pp. 1–1, Mar. 2018, early access.
- [98] P. R. Rizun, "A transaction fee market exists without a block size limit," *Self-published Paper*, Aug. 2015.
- [99] M. Ghosh, M. Richardson, B. Ford, and R. Jansen, "A torpath to torcoin: proof-of-bandwidth altcoins for compensating relays," U.S. Naval Research Laboratory, Washington, DC, Tech. Rep., 2014.
- [100] F. Zhang, I. Eyal, R. Escrivá, A. Juels, and R. V. Renesse, "REM: Resource-efficient mining for blockchains," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1427–1444.
- [101] S. Park, K. Pietrzak, J. Alwen, G. Fuchsbauer, and P. Gazi, "Spacecoin: A cryptocurrency based on proofs of space," MIT, Tech. Rep., Jun. 2015.
- [102] J. Blocki and H.-S. Zhou, "Designing proof of human-work puzzles for cryptocurrency and beyond," in *14th International Conference on Theory of Cryptography*, Beijing, China, Oct. 2016, pp. 517–546.
- [103] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," *Self-published Paper*, Jul. 2013. [Online]. Available: <http://primecoin.io/bin/primecoin-paper.pdf>
- [104] J. Andersen and E. Weisstein, "Cunningham chain. from mathworld—a wolfram web resource," 2005. [Online]. Available: <http://mathworld.wolfram.com/CunninghamChain.html>
- [105] A. Shoker, "Sustainable blockchain through proof of exercise," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, Oct. 2017, pp. 1–9.
- [106] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Average-case fine-grained hardness," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2017, Montreal, Canada, 2017, pp. 483–496.
- [107] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proceedings on Advances in cryptography—CRYPTO '86*, Santa Barbara, California, USA, Aug. 1986, pp. 186–194.
- [108] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel® software guard extensions: Epid provisioning and attestation services," Intel, Tech. Rep., 2016.
- [109] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy*, San Jose, CA, May 2014, pp. 475–490.
- [110] A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, Oct. 2007, pp. 584–597.
- [111] S. Wilkinson, "Storj a peer-to-peer cloud storage network," Storj Labs Inc., Tech. Rep., Dec. 2014. [Online]. Available: <https://storj.io/storj.pdf>
- [112] D. Vorick and L. Champine, "Sia: Simple decentralized storage," Nebulous Inc., Tech. Rep., Nov. 2014. [Online]. Available: <https://coss.io/documents/white-papers/siacoin.pdf>
- [113] D. Wagner, "A generalized birthday problem," in *Advances in Cryptology — CRYPTO 2002*, Santa Barbara, California, Aug. 2002, pp. 288–304.
- [114] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger (eip-150 revision)," *Ethereum Project Yellow Paper*, vol. 151, 2017.
- [115] P. Daian, I. Eyal, A. Juels, and E. G. Sirer, "(short paper) piecework: Generalized outsourcing control for proofs of work," in *Financial Cryptography and Data Security: FC 2017 International Workshops on WAHC, BITCOIN, VOTING, WTSC, and TA*, Sliema, Malta, Apr. 2017, pp. 182–190.
- [116] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference*, Santa Barbara, CA, Aug. 2015, pp. 585–605.
- [117] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in Cryptology – EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques*, Warsaw, Poland, May 2003, pp. 294–311.
- [118] D. Hofheinz, T. Jager, D. Khurana, A. Sahai, B. Waters, and M. Zhandry, "How to generate and use universal samplers," in *Advances in Cryptology – ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security*, Hanoi, Vietnam, Dec. 2016, pp. 715–744.
- [119] N. T. Courtois, "On the longest chain rule and programmed self-destruction of crypto currencies," *arXiv preprint arXiv:1405.0534*, 2014.
- [120] R. Recabarren and B. Carbutar, "Hardening stratum, the bitcoin pool mining protocol," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 3, pp. 57–74, 2017.
- [121] A. Laszka, B. Johnson, and J. Grossklags, "When bitcoin mining pools run dry," in *Financial Cryptography and Data Security: FC 2015 International Workshops on BITCOIN, WAHC and Wearable*, San Juan, Puerto Rico, Jan. 2015, pp. 63–77.
- [122] M. Maschler, E. Solan, and S. Zamir, *Game Theory*. Cambridge University Press, 2013.
- [123] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus," *arXiv preprint arXiv:1612.02916*, 2016.
- [124] A. Stone, "An examination of single transaction blocks and their effect on network throughput and block size," *Self-published Paper*, Jun. 2015. [Online]. Available: <http://ensocoin.org/resources/1txn.pdf>
- [125] K. Baqer, D. Y. Huang, D. McCoy, and N. Weaver, "Stressing out: Bitcoin 'stress testing'," in *Financial Cryptography and Data Security: International Workshops on BITCOIN, VOTING and WAHC*, Christ Church, Barbados, Feb. 2016, pp. 3–18.
- [126] G. Pappalardo, T. Di Matteo, G. Caldarelli, and T. Aste, "Blockchain inefficiency in the bitcoin peers network," *arXiv preprint arXiv:1704.01414*, 2017.
- [127] M. Möser and R. Böhme, "Trends, tips, tolls: A longitudinal study of bitcoin transaction fees," in *Financial Cryptography and Data Security: FC 2015 International Workshops on BITCOIN, WAHC and Wearable*, San Juan, Puerto Rico, Jan. 2015, pp. 19–33.
- [128] N. Houy, "The economics of bitcoin transaction fees," GATE Groupe d'Analyse et de Théorie Économique Lyon-St Étienne, Tech. Rep., Feb. 2014.
- [129] S. Feng, W. Wang, Z. Xiong, D. Niyato, P. Wang, and S. Wang, "On cyber risk management of blockchain networks: A game theoretic approach," *IEEE Transactions on Services Computing*, pp. 1–1, Oct. 2018, early access.
- [130] N. Dimitri, "Bitcoin mining as a contest," *Ledger Journal*, vol. 2, pp. 31–37, Apr. 2017.
- [131] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet of Things Journal*, pp. 1–1, 2018, early access.
- [132] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, Kansas, May 2018.
- [133] N. Houy, "The bitcoin mining game," *Ledger Journal*, vol. 1, no. 13, pp. 53 – 68, 2016.
- [134] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, Saarbrücken, Germany, Mar. 2016, pp. 305–320.
- [135] M. Carlsen, "The impact of transaction fees on bitcoin mining strategies," Master's thesis, Princeton University, 2016.
- [136] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *Financial Cryptography and Data Security:*

- 20th International Conference, Revised Selected Papers, Christ Church, Barbados, Feb. 2017, pp. 515–532.
- [137] Y. Sompolinsky and A. Zohar, “Bitcoin’s security model revisited,” *arXiv preprint arXiv:1605.09193*, 2016.
- [138] 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*, ser. CCS ’16. Vienna, Austria: ACM, Oct. 2016, pp. 3–16.
- [139] J. Gbel, H. Keeler, A. Krzesinski, and P. Taylor, “Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay,” *Performance Evaluation*, vol. 104, no. Supplement C, pp. 23–41, 2016.
- [140] J. Beccuti and C. Jaag, “The bitcoin mining game: On the optimality of honesty in proof-of-work consensus mechanism,” Swiss Economics, Working Papers 0060, Aug. 2017. [Online]. Available: <https://ideas.repec.org/p/chc/wpaper/0060.html>
- [141] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge University Press, 2012.
- [142] D. K. Tosh, S. Shetty, X. Liang, C. A. Kamhoua, K. A. Kwiat, and L. Njilla, “Security implications of blockchain cloud with analysis of block withholding attack,” in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Madrid, Spain, May 2017, pp. 458–467.
- [143] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, “On power splitting games in distributed computation: The case of bitcoin pooled mining,” in *2015 IEEE 28th Computer Security Foundations Symposium*, Verona, Italy, Jul. 2015, pp. 397–411.
- [144] I. Eyal, “The miner’s dilemma,” in *2015 IEEE Symposium on Security and Privacy*, San Jose, CA, May 2015, pp. 89–103.
- [145] S. Bag, S. Ruj, and K. Sakurai, “Bitcoin block withholding attack: Analysis and mitigation,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1967–1978, Aug. 2017.
- [146] S. Bag and K. Sakurai, “Yet another note on block withholding attack on bitcoin mining pools,” in *19th International Conference on Information Security*, Honolulu, HI, Sep. 2016, pp. 167–180.
- [147] “Slushpool,” Dec. 2017. [Online]. Available: <https://slushpool.com/home/>
- [148] A. Kiayias, I. Konstantinou, A. Russell, B. David, and R. Oliynykov, “A provably secure proof-of-stake blockchain protocol,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 889, 2016.
- [149] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, “On security analysis of proof-of-elapsed-time (poet),” in *19th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, Boston, MA, Nov. 2017, pp. 282–297.
- [150] “Slimcoin: A peer-to-peer crypto-currency with proof-of-burn,” [www.slimcoin.org](http://www.slimcoin.org), Tech. Rep., May 2014. [Online]. Available: <https://github.com/slimcoin-project/slimcoin-project.github.io/blob/master/whitepaper/whitepaper.pdf>
- [151] L. Ren, “Proof of stake velocity: Building the social currency of the digital age,” *Self-published Paper*, Apr. 2014. [Online]. Available: <https://coss.io/documents/white-papers/reddcoin.pdf>
- [152] I. Bentov, R. Pass, and E. Shi, “Snow white: Provably secure proofs of stake,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 919, Sep. 2016.
- [153] B. David, P. Gaži, A. Kiayias, and A. Russell, “Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain,” in *Advances in Cryptology – EUROCRYPT 2018*, Tel Aviv, Israel, Apr. 2018, pp. 66–98.
- [154] V. Buterin and V. Griffith, “Casper the friendly finality gadget,” *arXiv preprint arXiv:1710.09437*, 2017.
- [155] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, “Securing proof-of-stake blockchain protocols,” in *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops*, Oslo, Norway, Sep. 2017, pp. 297–315.
- [156] A. Poelstra, “Distributed consensus from proof of stake is impossible,” *Self-published Paper*, May 2014. [Online]. Available: <https://download.wpsoftware.net/bitcoin/old-pos.pdf>
- [157] N. Houy, “It will cost you nothing to kill a proof-of-stake cryptocurrency,” GATE Groupe d’Analyse et de Théorie Économique Lyon-St Étienne, Tech. Rep., 2014.
- [158] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, “Bitcoinng: A scalable blockchain protocol,” in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 45–59.
- [159] C. Decker, J. Seidel, and R. Wattenhofer, “Bitcoin meets strong consistency,” in *Proceedings of the 17th International Conference on Distributed Computing and Networking*, ser. ICDCN ’16, Singapore, 2016, pp. 13:1–13:10.
- [160] R. Pass and E. Shi, “Hybrid Consensus: Efficient Consensus in the Permissionless Model,” in *31st International Symposium on Distributed Computing (DISC 2017)*, vol. 91, Vienna, Austria, Oct. 2017, pp. 39:1–39:16.
- [161] M. K. Reiter, “A secure group membership protocol,” *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 31–42, Jan. 1996.
- [162] 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,” in *25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, Aug. 2016, pp. 279–296.
- [163] J. Kwon, “Tendermint: Consensus without mining (draft),” *Self-published Paper*, fall 2014. [Online]. Available: <https://tendermint.com/static/docs/tendermint.pdf>
- [164] “Proof of authority chains,” Jan. 2018. [Online]. Available: <https://github.com/paritytech/parity>
- [165] D. Larimer, “Delegated proof-of-stake (dpos),” Bitshare whitepaper, Tech. Rep., 2014.
- [166] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP ’17)*. Shanghai, China: ACM, Oct. 2017, pp. 51–68.
- [167] E. Lombrozo, J. Lau, and P. Wuille, “Segregated witness (consensus layer),” Tech. Rep., Dec. 2015. [Online]. Available: <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>
- [168] J. Poon and T. Dryja, “The bitcoin lightning network: Scalable off-chain instant payments,” Lightning Labs, Tech. Rep., Nov. 2016.
- [169] M. Green and I. Miers, “Bolt: Anonymous payment channels for decentralized currencies,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’17. Dallas, Texas, USA: ACM, Oct. 2017, pp. 473–489.
- [170] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, and E. G. Sirer, “Teechain: Scalable blockchain payments using trusted execution environments,” *arXiv preprint arXiv:1707.05454*, 2017.
- [171] K. Okupski, “Bitcoin developer reference,” Technische Universiteit Eindhoven, Tech. Rep., Jul. 2016. [Online]. Available: <http://enetium.com/resources/Bitcoin.pdf>
- [172] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, “Enabling blockchain innovations with pegged sidechains,” Blockstream Inc., Tech. Rep. [Online]. Available: <http://kevinrignen.com/files/sidechains.pdf>
- [173] A. Kiayias, A. Miller, and D. Zindros, “Non-interactive proofs of proof-of-work,” *IACR Cryptology ePrint Archive*, Report 2017/963, 2017, <https://eprint.iacr.org/2017/963>.
- [174] A. Kiayias, N. Lamprou, and A.-P. Stouka, “Proofs of proofs of work with sublinear complexity,” in *International Conference on Financial Cryptography and Data Security*, Christ Church, Barbados, Feb. 2016, pp. 61–78.
- [175] Z. Ren and Z. Erkin, “A scale-out blockchain for value transfer with spontaneous sharding,” *arXiv preprint arXiv:1801.02531v2*, 2018.
- [176] A. E. Gencer, R. van Renesse, and E. G. Sirer, “Short paper: Service-oriented sharding for blockchains,” in *Financial Cryptography and Data Security*, Sliema, Malta, Apr. 2017, pp. 393–401.
- [177] 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*, ser. CCS ’16. Vienna, Austria: ACM, Oct. 2016, pp. 17–30.
- [178] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of faults,” *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, 1980.
- [179] E. Kokoris Kogias, P. S. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. A. Ford, “OmniLedger: A secure, scale-out, decentralized ledger via sharding,” in *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, May 2018, pp. 583–598.
- [180] M. Zamani, M. Movahedi, and M. Raykova, “Rapidchain: Scaling blockchain via full sharding,” *IACR Cryptology ePrint Archive*, Report 2018/460, 2018, <https://eprint.iacr.org/2018/460>.
- [181] 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 *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, May 2017, pp. 444–460.
- [182] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, “Inclusive block chain protocols,” in *Financial Cryptography and Data Security*, San Juan, Puerto Rico, Jan. 2015, pp. 528–547.
- [183] A. Churymov, “Byteball: a decentralized system for storage and transfer of value,” byteball.org, Tech. Rep., 2017.

- [184] C. Li, P. Li, W. Xu, F. Long, and A. C.-c. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018.
- [185] S. Popov, O. Saa, and P. Finardi, "Equilibria in the tangle," *arXiv preprint arXiv:1712.05385*, 2017.
- [186] G. Hileman and M. Rauchs, "2017 global blockchain benchmarking study," Cambridge Centre for Alternative Finance, Tech. Rep., 2017.
- [187] M. Bartoletti and L. Pompianu, "An analysis of bitcoin op\_return metadata," in *Financial Cryptography and Data Security*, Malta, Apr. 2017, pp. 218–230.
- [188] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of namecoin and lessons for decentralized namespace design," in *2015 Workshop on the Economics of Information Security (WEIS)*. Delft, Netherlands: TUDelft, Jun. 2015.
- [189] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. Denver, CO: USENIX Association, Jun. 2016, pp. 181–194.
- [190] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*, ser. CCSW '17. Dallas, Texas, USA: ACM, Nov. 2017, pp. 45–50.
- [191] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, pp. 1–1, Jul. 2018, early access.
- [192] 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," *BigChainDB White Paper*, 2016.
- [193] W. Wang, D. Niyato, P. Wang, and A. Leshem, "Decentralized caching for content delivery based on blockchain: A game theoretic perspective," in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, USA, May 2018, pp. 1–6.
- [194] P. Goyal, R. Netravali, M. Alizadeh, and H. Balakrishnan, "Secure incentivization for decentralized content delivery," *arXiv preprint arXiv:1808.00826*, 2018.
- [195] S. Raju, S. Boddepalli, S. Gampa, Q. Yan, and J. S. Deogun, "Identity management using blockchain for cognitive cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017, pp. 1–6.
- [196] S. Y. Nikouei, R. Xu, D. Nagothu, Y. Chen, A. Aved, and E. Blasch, "Real-time index authentication for event-oriented surveillance video query using blockchain," *arXiv preprint arXiv:1807.06179*, 2018.
- [197] C. Xu, K. Wang, and M. Guo, "Intelligent resource management in blockchain-based cloud datacenters," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 50–59, November 2017.
- [198] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Madrid, Spain, May 2017, pp. 468–477.
- [199] A. Lewko and B. Waters, "Unbounded hibe and attribute-based encryption," in *Advances in Cryptology – EUROCRYPT 2011*, Tallinn, Estonia, May 2011, pp. 547–567.
- [200] N. Fotiou and G. C. Polyzos, "Decentralized name-based security for content distribution using blockchains," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, San Francisco, CA, Apr. 2016, pp. 415–420.
- [201] F. R. Yu, M. Huang, and H. Tang, "Biologically inspired consensus-based spectrum sensing in mobile ad hoc networks with cognitive radios," *IEEE Network*, vol. 24, no. 3, pp. 26–30, May 2010.
- [202] S. Raju, S. Boddepalli, N. Choudhury, Q. Yan, and J. S. Deogun, "Design and analysis of elastic handoff in cognitive cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.
- [203] K. Kotobi and S. G. Bilen, "Secure blockchains for dynamic spectrum access: A decentralized database in moving cognitive radio networks enhances security and user access," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 32–39, Mar. 2018.
- [204] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.
- [205] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, pp. 1–1, May 2018, early access.
- [206] Z. Chen, S. Chen, H. Xu, and B. Hu, "A security authentication scheme of 5g ultra-dense network based on block chain," *IEEE Access*, pp. 1–1, Sep. 2018, early access.
- [207] N. Herbaut and N. Negru, "A model for collaborative blockchain-based video delivery relying on advanced network services chains," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 70–76, 2017.
- [208] P. K. Sharma, M. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for iot," *IEEE Access*, vol. 6, pp. 115–124, 2018.
- [209] M. Li, J. Weng, A. Yang, and W. Lu, "Crowdabc: A blockchain-based decentralized framework for crowdsourcing," Online, available at: <https://eprint.iacr.org/2017/444.pdf>.
- [210] S. Feng, W. Wang, D. Niyato, D. I. Kim, and P. Wang, "Competitive data trading in Wireless-Powered internet of things (IoT) crowdsensing systems with blockchain," in *2018 IEEE International Conference on Communication Systems*, Chengdu, China, Dec. 2018.
- [211] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, pp. 17 545–17 556, 2018.
- [212] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, Sep. 2018.
- [213] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2017.
- [214] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3690–3700, Aug 2018.
- [215] Z. Su, Y. Wang, Q. Xu, M. Fei, Y. Tian, and N. Zhang, "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet of Things Journal*, pp. 1–1, 2018, early access.
- [216] Y. Zhang, M. Pan, L. Song, Z. Dawy, and Z. Han, "A survey of contract theory-based incentive mechanism design in wireless networks," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 80–85, Jun. 2017.
- [217] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, Kansas, May 2018, pp. 1–6.
- [218] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, Aug. 2018.
- [219] M. Liu, R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2018, early access.
- [220] K. Suankaewmanee, D. T. Hoang, D. Niyato, S. Sawaditang, P. Wang, and Z. Han, "Performance analysis and application of mobile blockchain," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, Maui, Hawaii, Mar. 2018, pp. 642–646.
- [221] P. Tsankov, A. Dan, D. D. Cohen, A. Gervais, F. Bueznli, and M. Vechev, "Securify: Practical security analysis of smart contracts," *arXiv preprint arXiv:1806.01143*, 2018.
- [222] R. Dennis, G. Owenson, and B. Aziz, "A temporal blockchain: A formal analysis," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*, Orlando, FL, Oct. 2016, pp. 430–437.
- [223] J. Sidhu, "Syscoin: A peer-to-peer electronic cash system with blockchain-based services for e-business," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, Vancouver, BC, Jul. 2017, pp. 1–6.
- [224] J. Bruce, "The mini-blockchain scheme rev 3," *Self-published Paper*, Mar. 2017. [Online]. Available: <http://cryptonite.info/files/mbc-scheme-rev3.pdf>
- [225] J. Wu, S. Guo, J. Li, and D. Zeng, "Big data meet green challenges: Big data toward green applications," *IEEE Systems Journal*, vol. 10, no. 3, pp. 888–900, Sep. 2016.
- [226] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018, early access.
- [227] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," *arXiv preprint arXiv:1808.03949*, 2018.

- [228] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” *arXiv preprint arXiv:1610.02527*, 2016.
- [229] G. Zyskind, O. Nathan, and A. Pentland, “Enigma: Decentralized computation platform with guaranteed privacy,” *arXiv preprint arXiv:1506.03471*, 2015.
- [230] F. Benhamouda, S. Halevi, and T. Halevi, “Supporting private data on hyperledger fabric with secure multiparty computation,” in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, Orlando, FL, Apr. 2018, pp. 357–363.