Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership and Structure

Christopher Natoli, Jiangshan Yu* , Vincent Gramoli , and Paulo Esteves-Verissimo

***Abstract*—It is no exaggeration to say that since the introduction of Bitcoin, blockchains have become a disruptive technology that has shaken the world. However, the rising popularity of the paradigm has led to a flurry of proposals addressing variations and/or trying to solve problems stemming from the initial specification. This added considerable complexity to the current blockchain ecosystems, amplified by the absence of detail in many accompanying blockchain whitepapers.**

Through this paper, we set out to explain blockchains in a simple way, taming that complexity through the deconstruction of the blockchain into three simple, critical components common to all known systems: *membership selection*, *consensus mechanism* and *structure*. We propose an evaluation framework with insight into system models, desired properties and analysis criteria, using the decoupled components as criteria. We use this framework to provide clear and intuitive overviews of the design principles behind the analyzed systems and the properties achieved. We hope our effort will help clarifying the current state of blockchain proposals and provide directions to the analysis of future proposals.

I. Introduction

In 2008 Satoshi Nakamoto released Bitcoin [150], highlighting the power and importance of distributed systems. The use of distributed systems in our lives has been transparently controlled by significant actors. The introduction of Bitcoin provided the public with insight into ungoverned distributed systems and sparked a movement towards decentralization. Initially, the blockchain technology went vastly unnoticed and was heavily integrated with the deep web due to its pseudonymity properties [55, 217]. However, the concept of the blockchain quickly gained interest and grew to what it is today. The growing interest promoted the introduction of new chains and the growth of distributed ledger technologies [4, 10, 14, 78, 137, 141, 170]. However, the rising popularity of the paradigm has led to a flurry of proposals addressing variations and/or trying to solve problems arising from the initial specification.

The complexity introduced through popularity was amplified by the absence of detail in the proposals. As an example, the vast majority were presented through white papers [53, 123, 124, 150, 155, 168, 193, 196, 204], wiki documentations [116, 156, 180, 183] and websites [5, 11, 130, 133]. This is in contrast with the traditional academic research that conveys results through scientific publications typically peer-reviewed by specialists of particular domains, like distributed systems, cryptography, networking or game theory [63, 84, 94, 120]. The lack of detail has left room for interpretation that has sometimes led researchers to different conclusions [22, 75, 102].

Through this paper, we set out to explain blockchains in a simple way, taming that complexity through the deconstruction of the blockchain into three simple, critical components common to most known systems: *membership selection*, *consensus mechanism*, and *structure*. The *membership selection* component determines a committee of nodes that participate in the consensus; the *consensus mechanism* component is responsible for deciding on the next block, run by the selected committee of nodes; and the *structure* component represents how the data is organized in the blockchain. Thanks to our deconstruction, we are able to provide a clear and unique landscape of blockchains, as depicted in Figure 1. We then propose an evaluation framework with insights into different key system models, desired properties, and analysis criteria, using the decoupled components as parameters. We use this framework to provide overviews of the design principles behind the analyzed systems and the achieved properties.

There are notable works aggregating and analyzing blockchains, each providing unique evaluation but primarily focusing on one aspect and constructing specific frameworks to comparatively analyze. Cachin and Vukolić present a comparison of consensus protocols in permissioned blockchains [49]. Abraham and Malkhi also present the idea of deconstructing the blockchain into layers [18] but with a focus on relating Nakamoto's consensus versus Byzantine fault tolerant (BFT) protocols. Similarly, Vukolić [212] explores the contrast between blockchains and BFT replicated state machines, pointing to scalability problems faced by both and provide insight into upcoming proposals. Gramoli [97] investigates mainstream blockchains and discusses classical Byzantine consensus in the context of the blockchain and discusses some of the dangers of misunderstanding guarantees, problems that we address and systematize in our paper. Wang et al. [213] approach the blockchain consensus mechanisms in the perspective of game theory and the strategies of adoption of nodes. Bano et al. [26] provide an overview on how different blockchain consensus work, and compare existing Proof-of-* against other proposed

* Corresponding author.
Christopher Natoli is with University of Sydney, Australia.
Vincent Gramoli is with University of Sydney and CSIRO Data61, Australia.
Jiangshan Yu is with Monash University, Australia. E-mail: J.Yu.Research@gmail.com
Paulo Esteves-Verissimo is with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg.

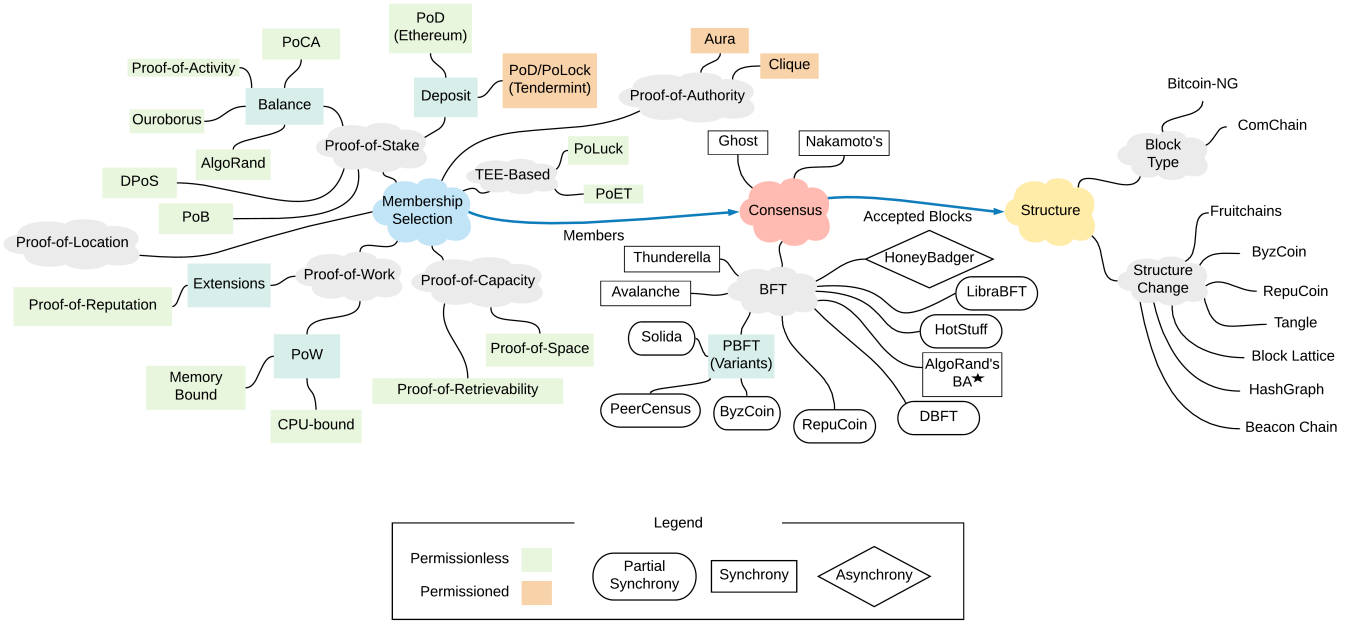

Figure 1. The blockchain landscape.

consensus mechanisms based on their properties. However, no other work has deconstructed the blockchain into these unique components and analysed all three.

While existing work provides good analysis of specific sets of the properties of blockchains, it remained challenging, even for the educated but non-expert readers to get a thorough and comparative picture of the design principles of different complicated systems. To our knowledge, we are the first to classify blockchain based on a decomposition into membership selection, consensus mechanism and structure, categorizing and providing an analysis of leading proposals. As an addition to our analysis, we provide an overview of the attacks and threats related to membership selection, consensus goals, and structural assumptions. We hope that our analysis will clarify future design directions and inspire new designs based on innovative coherent combinations revealed by our decomposition and categorization.

Thus, this paper presents the following contributions:

- It innovatively deconstructs the blockchain into three simple, critical components: membership selection, consensus mechanism and structure.
- It provides an evaluation framework with insight into system models, desired properties and analysis criteria, using the decoupled components as parameters.
- It provides a clear and intuitive overview of the design principles behind analyzed systems and their properties, in terms of the parameters above.
- It proposes, based on the analyzed state of the art, a categorization of membership selection and consensus approaches. In addition, it provides generic charts representing the design principles for each category of membership selection, to simplify future designs.

The remainder of the paper is as follows. Section II provides background information about blockchains and the core properties relating to our decomposition. Section III outlines the criteria used for comparison and analysis. Section IV discusses and analyzes the membership selection and proposals. Section V provides an analysis of consensus mechanisms. Section VI overviews structure proposals for the blockchain. Section VII highlights proposed attacks relating to the membership selection and consensus. Section VIII discusses other insights and aspects. We conclude with a summary of our analysis and discuss future directions.

II. Background

In this section, we explore the foundational concepts of the blockchain and discuss the initial specification of Bitcoin. We then discuss the applicability of blockchains and conclude by highlighting the known impediments.

A. Blockchain

Blockchain is an append-only distributed ledger of transactions. First introduced with Bitcoin [150], the blockchain originated as a decentralized electronic payment system, which removed the need for any third-party involvement for payment transfers. The original Bitcoin blockchain organized data as a chain of blocks directed by utilizing the block hash values, hence the name “blockchain”. Later proposals expanded from a single chain to parallel chains [120, 209, 221] and graphs [25, 128, 168]. This has led to the term “blockchain” becoming a misnamed concept for distributed ledgers.

The decentralized nature of blockchains means that all nodes verify and store the transactions that have taken place in the system, and propose new blocks to append to the chain. The blockchain structure can be seen as a linearly increasing linked list of transactions batched into blocks.

The chain begins with a *genesis* block at index 0 and each block appended links to its direct predecessor forming the chain. This, however, is the combination of pre-existing ideas constructed together to form what the blockchain is today. Haber et al. [98] first introduced the idea of timestamping to digitally verify a document, which is paralleled by the timestamped block being appended on the chain. Similarly, the distributed ledger of cryptographically-linked blocks can be represented as a fully replicated state machine [192], where the state machine is a hashchain [125].

From the initial specification detailed by Bitcoin, various blockchains have been proposed; improving and extending the original work by adding new features and functionality on top of the decentralized payment. An example is Ethereum [216], which introduced the ability to execute Turing complete code on the blockchain and perform conditional payments based on actions through *smart contracts*. A number of new blockchains [3, 4, 9, 13, 14, 137], and distributed ledger technologies [10, 170], have been created to fulfill new purposes and help integrate the blockchain into critical infrastructure today.

Such extensions have provided ways for which blockchains can be applied to existing systems. Transport [56, 66], Healthcare [139, 165, 223], and Finance [10, 106, 170] have shown potential for the use of blockchain technology and a number of applications are being used in systems today [107, 108, 135, 144].

B. Bitcoin

The seminal paper on Bitcoin [150] was the first to introduce blockchain as the completely decentralized electronic cash system on a peer-to-peer network. It facilitates pseudonymous payment between two parties without the requirement of a third party.

Each account on the Bitcoin blockchain is composed of a public and private key pair. The hash of a public key identifies the account of the key owner by forming an *address*. The public key address is used to accept coins, and the private key of an account is used to authorize a spending. Payment occurs with a "payer" (or "payers") signing a transaction using the private key, transferring assets to the address of the "payee" (or "payees"). Once signed, the "payer" broadcasts the transaction to the network, where the transaction is then mined into a block. The "payee(s)" can use its private key of the address to claim its ownership.

A transaction contains three data fields, namely metadata, inputs, and outputs. The metadata field records the unique ID of the transaction (i.e. the hash of the entire transaction), the size of the transaction, the number of inputs, the number of outputs, and a `lock_time` field defining the time period that one have to wait to validate this transaction. An input specifies a previous transaction using its hash value. The input also contains the index of the previous transaction's outputs that is being claimed (as there may be more than one output). A valid signature for the current transaction is also required to prove the ownership of the claimed output (of the previous transaction), by using the signing key associated to the address of the claimed output. An output has two fields, namely value and address. They define the value to be transferred to the address. Bitcoins are just transaction outputs with an arbitrary value with 8 decimal places of precision. The smallest possible value is 10^{-8} BTC which is called 1 *Satoshi*. For the transaction to be valid, every input must be an unspent output (i.e. coin) of a previous transaction. Every input must be digitally signed. The total value in the input field must be no smaller than the total value in the output field. If the total value in the input is greater than the total value in the output, then the difference between the two total values is called a transaction fee.

Transactions are broadcast in the network. If a node receives the same transaction multiple times, it only broadcasts it once. So that transactions will not be broadcast in the network forever. For a received valid transaction, a node will use it as a part of the input in its "mining" process, and the transaction is accepted if it is included in the blockchain. All nodes in the network have the option to participate in the mining process, computing a valid hash as the Proof-of-Work (PoW). We will detail different Proof-of-Work systems in § IV-A. The block containing a set of verified transactions is then propagated through the network, and if valid and has been decided as the correct extension of the chain, the block containing transactions will be accepted.

The creator of a block that is included in the chain obtains bitcoins as reward of its work. A reward has two parts, namely a pre-defined amount of coins as mining reward, and transaction fees of contained transactions.

The seminal work introduced by Bitcoin was a scalable membership selection and consensus, dubbed Nakamoto's Consensus, discussed in § V-A, as well as the application of a distributed timestamping service for decentralized validity on a ledger. This assembly of ideas presented new technology, causing a paradigm shift in digital payments and providing a number of new possibilities for this technology to evolve.

C. Smart Contracts

As the blockchain continues to evolve, new functionalities emerge, allowing them to be applied in numerous use cases and further integrated into daily use. One major evolutionary impact was the introduction of *Smart Contracts* in Ethereum [78, 216]. The Ethereum blockchain runs a virtual machine in the lower layer, known as the EVM, holding the blockchain state and executing all blockchain commands. Through this, Ethereum introduced the functionality to run *bytecode* on the EVM of all nodes and interact directly with the blockchain. This provides mechanisms not only for conditional payment, but programmable applications that interact directly with the blockchain verified by all nodes.

A smart contract is typically written in a high-level language, such as Solidity [77], and compiled down to bytecode. A transaction is then composed to deploy the contract on the blockchain. The transaction data contains the contract code and any constructor arguments to instantiate the contract. Once deployed, the contract address is returned as part of the transaction receipt and can be interacted with. If a contract

function modifies the blockchain state, it must be invoked through a transaction, which does not require any asset transfer but must have enough "Gas"[1]. However, any functions that do not require state modification, or, viewing any public variables can be invoked without a required transaction.

Since the blockchain is seen as "immutable", once a contract has been deployed, it cannot be modified. This has been revealed as a severe weakness for many applications, as any vulnerable code is stuck on the blockchain and cannot be updated or patched. Current techniques often require "proxy" contracts which allow for contacts to be "upgraded" with minimal impact[2].

Most blockchain applications, known as Distributed Applications, or DApps, interact with contracts through libraries, written in languages such as JavaScript, Python, Go or Java. These are often seen as mobile or web applications and are similar to those used commonly today.

Other blockchain systems, such as HyperLedger [105] and EoS [6] also provide smart contract capabilities. However, blockchains such as Bitcoin allow for "scripts" to be written into the data field of their transactions and executed on the blockchain directly. Overall, smart contracts allow for the blockchain to be integrated with business logic and allow conditional payment through contract invocation.

D. Known Causes of Blockchain Varieties

The original Bitcoin blockchain provided the foundation for blockchain systems today by introducing the decentralized peer-to-peer transaction system. However, aspects of Bitcoin served as focal points of evolution to new blockchain varieties. We list below the main triggers for these varieties.

1) Transient Forks

With multiple nodes working to propose new blocks, there is a high chance of concurrent proposals. With structures that follow a single, canonical chain, the concurrent proposals create a transient branch in the chain, known as a *fork*, in which the proposed blocks share a common predecessor and can continue on their own separated path. At this point, the nodes of the network must decide which is the correct chain to extend through consensus. We defer the explanation of the different consensus approaches to Section V. The occurrence of forks impact transaction finality and can lead to a case of double-spending, where the same coin is spent multiple times on different branches. However, forks can also be used for protocol upgrades, known as *hard forks*, where the system diverges from a given block height and all nodes running the latest version will be on the same branch.

2) Storage Size

The blockchain is designed as an append only ledger, resulting in an linearly increasing structure. This requires storage space that is growing at dynamic rates. Bitcoin, currently 182.206GB[3] and Ethereum is 94GB (with Geth's Fast Sync)[4] and $600+$GB (full node)[5] reveal the size requirements for a node on the blockchain. The growing storage requirements present limitations for certain devices to participate, such as mobile or IoT, as well as long-term problems in maintaining a node. Current efforts, such as pruning and light clients, focus on this problem and aim to provide low space requirements for nodes.

3) Scalability

Blockchains provide a seemingly scalable service to growing numbers of users and nodes. However, this can be contrasted by the increasing energy consumption of Proof-of-Work blockchains and, in some cases, increased transaction latency. Scalability of decentralized distributed systems is a major problem area for a number of systems with a numerous proposals for improvements. Another important aspect of scalability is the system throughput, defined as the number of transactions per second included into a block, which is one of the major hindrances to blockchain deployment. When compared to current centralized systems, blockchain systems fail to meet throughput requirements to handle daily spending. The cost of true decentralization is observed through the lower throughput. However, many efforts are being made to improve the blockchain throughput while still offering the same guarantees [212]. We divert further discussions of scalability to Section VIII-D.

4) Clique Formation and Centralization

Bitcoin's primary attraction was the decentralized distributed system, without a required presence of a controlling party. Although the ideal blockchain would have homogeneous nodes distributed and owned by unique users, the incentive of block reward proved to be more profitable if miners formed cliques to produce blocks. Due to these cliques, seen as mining pools in current PoW blockchains, the decentralization of the system has been heavily impacted. For example, in Bitcoin's blockchain, as of February 2019, 4 pools can be seen to control larger than 50%, and 15 pools control more than 80% of the total reported hashrate [37].

III. Analysis Criteria

In this section, we introduce the criteria used in evaluation and comparison of the different membership selection and consensus mechanisms. We begin by describing the common assumptions made and follow by describing the properties of membership selection and consensus.

A. Common Assumptions

Both membership selection and consensus require assumptions to achieve their goals, often encapsulating system behavior (e.g. network delay, node numbers, etc) as well as a threat model (e.g. threshold of faults, adversary behavior).

[1]Ethereum employs the concept of Gas to prevent indefinite code execution. Each instruction on the Ethereum Virtual Machine has an associated cost, so function invocations would have a cost to run.

[2]A proxy contract provides an address which will always point to the "current" version of a contract, so if a new upgrade is deployed on the chain, people will still use the same contract address to access the upgraded contract.

[3]https://www.blockchain.com/charts/blocks-size. Accessed 09 Sep 2018.

[4]https://etherscan.io/chart2/chaindatasizefast. Accessed 13 Sep 2018.

[5]https://bitinfocharts.com/ethereum/ Accessed on 13 Sep 2018.

1) Network

The network assumptions detail the bound of time for messages to be delivered and can be categorized into three distinct classifications:

- *Synchronous*: there is a known, fixed upper bound on the time required for a message to be sent from one processor to another.
- *Asynchronous*: there is no upper bound on the time taken for a message to be delivered. It is proven by Fischer et al. [87] that it is impossible to have deterministic consensus schemes in an asynchronous model.
- *Partially Synchronous*: the upper bound for message delivery exists but is unknown a-priori [71].

2) Online Presence

The online presence considers whether all nodes are online at any given point in time, or only a subset of nodes.

- *All Online*: The model in which all nodes are online and have network connections to other nodes.
- *Sleepy*: The sleepy model considers crash fault tolerance in the BFT context. This concept was initially introduced for classical state machine replication [132], and later introduced for blockchains [29, 163]. Nodes are classified as "alert" or "sleepy". An alert node is one that is currently online and has predictable network connections to all other alert nodes. A sleepy node is one that is currently offline, but may become alert later, and vice versa.

3) Adversary Threat Model

The adversary threat model captures the threshold of Byzantine behavior that a system can tolerate. These threat models have two distinct properties, type and threshold, where the type defines what adversary is referenced, and the threshold is the upper bound on Byzantine power specific to the type [18]. Traditionally, Byzantine Fault Tolerant (BFT) systems put an assumption on the maximum number of committee members and clients that an attacker can control. This is often referred to as the maximum number f of Byzantine nodes. In our generic analysis, in order to capture all situations, we define N as the total voting power of the system, and f as the maximum voting power controlled by all attacker nodes (as a fraction of N). The $f - to - N$ relation becomes then a fraction defining the maximum percentage voting power that can be controlled by the adversary without the system failing to reach its goals.

- *Traditional*: The number of votes a Byzantine actor can control, often represented as the number of Byzantine votes f as a proportion of N, the total number of votes.
- *Computational*: The adversary is defined by the amount of computing power they control. The adversary is expressed as the amount of computing power f with respect to the total computing power of the system, N.
- *Stake*: For stake-based systems, the adversary is defined by the amount of asset the adversary has to stake. This is defined as the amount of a resource in possession of the adversary f in regards to total resources possessed by all users in the system N.
- *Space*: The amount of storage space readily available to an adversary with low-latency reads. This adversary, similar to the computational adversary, controls a percentage of the total storage space in the system, represented respectively as f and N.

4) Trust

The trust considers whether the proposal assumes a trusted presence to interact and participate.

- *Hardware*: The proposed mechanism assumes that trusted hardware components will be used in the operation and all output is taken as correct.
- *Participants*: This assumes that trusted authorities or participants, through policy or otherwise, are operational in this mechanism. This can be seen as an oracle or trusted entities in the system.
- *None*: This means there is no assumption on the trust of the hardware or participants. All aspects assume that there is no trust required in the system to operate.

B. Membership Properties

The membership selection provides the subset of nodes that are then admitted to participate in the consensus or block proposal. Each mechanism exhibits properties that allow for classification and categorization.

1) Openness

The openness represents the rules on how nodes participate in the system and how committees are formed. This can be seen as a fundamental aspect in the design of blockchain algorithms and depicts its suitability for certain environments [47, 212].

- *Permissionless*: A permissionless, or public, system has no restriction on the set of candidates and their behavior. That is, the size of the consensus committee is not predefined nor fixed, and a node is able to participate or leave at any given time.
- *Permissioned*: A permissioned, or private, system has complete restriction on the nodes participating. A node must join the system by being accepted in and as a known member. The consensus committee is often predefined, or, operates under the assumption that all nodes are known to all.
- *Partially-Permissioned*: A partially-permissioned system is often seen in consortium scenarios, where there are certain restrictions present on either the consensus members or the node participation. Some systems allow nodes to freely join, but only a pre-defined subset of nodes form the consensus; other systems require nodes to be accepted in the system but allow for any node to become a member of the consensus.

2) Selection Approach

The selection approach details the process used to rank and/or select the nodes which will form the consensus group. These

approaches are used to differentiate the consensus nodes from other nodes according to the way they prove to have met the pre-defined criteria for some specific parameter(s). In order to increase the probability of being selected, a node must show that it is better than other nodes according to the selection criteria. For example:

- *Work*: This work parameter measures the "amount of valid work" a node has produced and can be directly related to the amount of computational power used to produce the work.
- *Stake*: The amount of resources the node has "staked". This can be seen as the node's balance, or, the amount of asset the node has decided to place as stake. For a node to increase probability, they would have to increase the amount of stake.
- *Resource Count*: In some mechanisms each individual resource, for example CPUs, counts towards the probability of selection. For a node to increase the probability of selection, they must possess more of the elected resource.

3) Incentive

To allow for scalable and robust consensus, some schemes provide a mechanism to incentivize honest behavior through reward or punishment. We consider the quality of incentives a major factor for the stability and effectiveness of the selection mechanisms and consensus. When effective, they introduce *rationality* and thus influence node participation and heavily constrain Byzantine behavior.

- *Reward*: measures the benefit that one will get as a result of its contribution to the system.
- *Punishment*: measures the cost that one needs to pay if it behaves incorrectly. It is used to discourage any malicious behavior, and plays an important part to the efficiency of the system and may affect the overall performance.

C. Consensus Properties

Consensus forms the core property of the blockchain where the valid blocks are agreed upon to be appended to the chain. The *blockchain consensus* is composed of consensus instances, where each consensus instance is the decision for the block decision at each index of the chain. The valid blocks to be decided contain valid transactions, defined as the following.

Definition 1 (Transaction)**.** *A command that changes the state of the Blockchain.*

Definition 2 (Valid Transaction)**.** *A transaction that is consistent with the current blockchain state and has correct signatures and structure.*

1) Generic Blockchain Consensus Properties

The consensus instances exhibit properties that define how the decision is made and what guarantees are in place.

- *Agreement*: For a given consensus instance i, if a correct node decides block B, then all correct nodes decide B.
- *Termination*: All correct nodes eventually decide [136, 211].

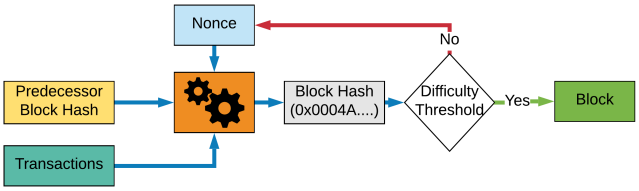

Figure 2. Proof-of-Work flow.

- *Validity*: If all correct nodes propose a valid transaction before starting a consensus instance i, then the block decided in i is not empty.

2) Blockchain State Properties

The blockchain state provides the foundation for the way transaction finality is interpreted and how state transitions are effected. However, both properties may not be observed at the same time.

- *Total Order Prefix*: A blockchain exhibits a total order prefix when consensus instances have terminated with agreement on a common prefix of states $\sigma_0, \sigma_1, \ldots \sigma_i$, but have not yet terminated for the states starting from σ_{i+1}. This is commonly observed with a *fork* [89, 90, 160].
- *Total Order*: A blockchain exhibits total order when all consensus instances have terminated and agreement has been reached for all states.

IV. Membership Selection

In this section, we present an overview of the membership selection algorithms deployed in current blockchain systems. The primary purpose of membership selection is to determine a committee of nodes to participate in the consensus, whose primary goal is to propose, or validate, blocks and provide a decision for the correct block at the next index of the chain. A summary of our overview is presented in Table I.

A. Proof-of-Work (PoW)

Proof-of-Work (PoW) is the mechanism used by Bitcoin and most mainstream blockchain systems today. It was proposed in this context to address the Sybil Attacks [69], where an attacker creates multiple entities and dominates the consensus committee. In particular, PoW addresses the Sybil attack by requiring nodes to prove that they have performed some work at a non-negligible cost, such as consumed electricity, time and hardware. The "work" is to produce a solution to a *puzzle*, where the solution is relatively difficult to find, but can be easily verified by others. This, in turn, acts as a disincentive for attackers due to the involved cost. In Bitcoin and its variants, the process of working to obtain a solution is called *mining*, where the nodes performing the work are called *miners*.

Proof-of-Work in Bitcoin was heavily derived from the initial proposal by Dwork et al. to combat junk mail [72], which was later adopted by HashCash [36]. The technique has two variants of implementations, the CPU-bound approach, as seen in Bitcoin [150], and the memory-bound approach [17, 70].

Table I
MEMBERSHIP SELECTION OVERVIEW

		PoW	**PoR**	**PoCA**	**PoD**	**PoLock**	**PoActivity**	**Ouroboros**	**DPoS**	**AlgoRand VRF**	**PoB**	**PoA**	**PoC**	**PoL/PoET**	**PoLocation**
Chain	Seminal Chain	Bitcoin [150]	Repucoin [221]	PPCoin [119]	Ethereum [79, 225]	Tendermint [40, 123]	– [28]	Ouroboros [118]	BitShares [48]	AlgoRand [94, 140]	Slimcoin [196]	Ethereum [116, 156]	Permacoin [142]	Intel Ledger* [112]	Platin [182]
Assumptions	Trust	None	None	None	None	None	None	None	None	None	None	Participants (Authorities)	None	Hardware	None
	Adversary	Computational	Computational	Stake	Stake	Stake	Stake / Computational	Stake	Stake	Stake	Stake	Traditional	Space	Traditional	Traditional
Properties	Openness	Permissionless	Permissionless	Permissionless	Permissionless	Permissioned	Permissionless	Permissionless	Permissionless	Permissionless	Permissionless	Permissioned	Permissionless	Permissionless	Permissionless
	Selection Approach	Memory / Hash Power	Past Contribution	Coin Age	Coins Deposited	Coins Locked	Coins Held, Online	Balance	Votes	Balance	Coins Burnt	Elected Authority	Storage	Number of CPUs	GPS Location, Nearby Peers
	Incentive Reward	✓	✓	✓	✓	✗	✓	✓	✓	✗ †	✓	✗	✓	✓	✓
	Incentive Punish-ment	✗	✓	✗	✓	✓	✓	✗	✓‡	✓	✗	✗ #	✓	✗	✓

* Intel's Ledger was later HyperLedger Sawtooth [111]
† AlgoRand reward based incentive mentioned as future work in the paper [94].
‡ DPoS punishment is that the malicious node will lose trust in the community and be voted out.
# There is no incentive mechanism paired with PoA, but an intrinsic reputation to uphold.

Proof-of-Work's strength lies in its ability to provide a mechanism for validity to a seemingly unlimited, unknown set of nodes which is necessary for this blockchain context. This quality has seen Proof-of-Work integrated as the backbone for other hybrid blockchains [19, 120, 164, 221].

Key Concept: The core concept of Proof-of-Work is proving validity through work. This allows for a distributed system to provide a validity check, as an aspect of scalability, to a large number of nodes whilst also adding resilience against Sybil attacks. In the blockchain context, a valid block proposal from a node must encapsulate the proof showing they have worked for some cost.

Proof-of-Work utilizes the process of solving the puzzle to express membership selection. If a node wishes to participate in the proposal of a block in the consensus, it must find a solution prior to learning of any other blocks that have been proposed for that given index. The membership selection accepts any number of nodes as long as they have provided a correct solution.

To produce a valid block with Proof-of-Work, as depicted in Figure 2, a miner repeatedly submits a chosen nonce along with the predecessor block hash, the transaction root hash, and other metadata to a hashing function. The output produced is then compared against the difficulty threshold to determine its validity. The difficulty threshold, dependent upon the blockchain, is directly proportional to the number of hashes in expectation to be performed to find a valid solution, as well as acting as a delay between proposals.

Weaknesses: Although Proof-of-Work achieves its goal for blockchain membership selection, it is hindered by a number of limitations. The process of mining required to solve the Proof-of-Work puzzle requires large amounts of computation, resulting in high resource and energy consumption. The rising difficulty to maintain a constant block proposal time results in increasing costs for running a miner, heavily impacting the scalability and future operation of the blockchain.

The cost of computation is directly proportional to the increasing difficulty. Finding a valid solution to the Proof-of-Work puzzles is a lengthy process, both allowing for fairness in computational cost as well as giving the network the chance to disseminate the information of the latest block before a forked block is found. The inherent delays in proposals relate directly to a weakness of Proof-of-Work in mainstream blockchains, as it must cater for a seemingly unknown network size as well as large amounts of hashing power.

Proof-of-Work assumes the distribution of the hashing power, where the honest nodes control the majority. However, there are many plausible attacks proposed against Proof-of-Work where an adversary can gain control of hashing power, or manipulate the node's connections to perform double spending.

Goals and Assumptions: Proof-of-Work strives to provide membership selection in an open, permissionless environment, where all nodes are freely able to join or participate. Proof-of-Work ties the probability of membership selection to the hash power of the node, assuming that an increase in hash power requires an increase of cost. The computational adversary assumptions made by Proof-of-Work follow this, where the majority of the computational power of the network will be owned by honest nodes, where the probability of selecting an adversary multiple times in a row is low.

For nodes to be incentivized, Proof-of-Work is often accompanied by a reward structure that rewards the nodes for their work, as seen in Bitcoin [150], or Ethereum [78] with the block rewards.

1) Proof-of-Work Variants

1.1) CPU-bound Proof-of-Work

Bitcoin [150] was the first mainstream blockchain, which introduced the Proof-of-Work algorithm for cryptocurrency. The design in bitcoin was heavily influenced by HashCash [36] together with Dai's B-Money [61]. The HashCash cryptographic puzzle requires a node to find the solution by repeatedly putting a *nonce* through a pseudorandom function with a SHA-1 hash. The Bitcoin Proof-of-Work deviates by requiring the pseudorandom function to produce a SHA-256 hash, the input is the combination of the nonce and the new block hash. The CPU-bound functions are directly related to the calculation speed of the hardware. Modern "Application-Specific Integrated Circuit" (ASIC) devices, such as the *Antminer R4* [1] were designed to surpass ordinary computer hashes per unit of money. This provides a major advantage to the owners that performed the Proof-of-Work CPU-bound mining. The introduction of the ASIC miners in the Bitcoin blockchain have heavily influenced the market, and since have dominated the mining due to the advantages provided. From the introduction of CPU-bound Proof-of-Work in Bitcoin, it has been implemented in most major mainstream blockchains, such as Ethereum and Litecoin.

To create a solution for the CPU-bound Proof-of-work, the miner batches a number of transactions and creates a Merkle tree [138] from the transaction hashes. The miner knows the global *threshold*, or difficulty, the latest block and the transactions. They then select a nonce and apply a pseudo-random function to the new block of transactions. The higher the difficulty is, the more leading zero is required in the hash value of a valid block. Table II depicts the relationship between difficulty, target and the number of expected hashes. The target is the number that the block hash must be lower than, and as shown the number of leading zeroes increases with increasing difficulty, which requires more expected effort to solve. We refer readers to [33] for a detailed explanation.

1.2) Memory-bound Proof-of-Work

In 2005, Dwork et al. [70] further studied the concept of memory-bound proof of work for preventing email spam. To cope with the influx of ASIC miners that dominate the node

Table II
THE RELATION BETWEEN DIFFICULTY AND THE EXPECTED NUMBER OF HASHES. AS THE DIFFICULTY INCREASES, THERE IS AN OBSERVABLE INCREASE IN THE NUMBER OF LEADING 0'S IN THE TARGET, SO A VALID BLOCK MUST HAVE A VALUE LOWER THAN THE TARGET.

Difficulty	Expected Number of Hashes	Approximate Target
1	4.295e+09	0x00000000FFFF0000000000000000000000...
100	4.295e+11	0x000000000028F5999999999A000000000...
1000	4.295e+12	0x00000000004188F5C28F5C2800000000...
10000	4.295e+13	0x0000000000068DB22D0E560400000000...
1000000	4.295e+15	0x00000000000010C6E6D9BE4CD7000000...
100000000	4.295e+17	0x000000000000002AF2F2D14354120000...
1000000000000	4.295e+21	0x0000000000000000000119787E99468E30...

hash power, memory-bound Proof-of-Work algorithms [17, 31, 70] are adapted by blockchains. In particular, the memory-bound Proof-of-Work, also known as *egalitarian Proof-of-Work*, relies on random access to slow memory rather than computational hashing power, emphasizes a dependency on latency. This makes the performance bound by memory-access speed rather than hashing power. Therefore, this provides the system with resilience to ASIC miners and fast memory-on-chip devices.

Following the success of Proof-of-Work in cryptocurrencies, but seeing the disadvantage of ASIC miners, the Equi-Hash [31] algorithm was adopted to memory-bound Proof-of-Work. The CryptoNote [204] as well as ZeroCash [190] both adopt the memory bound proof-of-work in the concept of a cryptocurrency.

2) Proof-of-Reputation

Proof-of-Reputation [221] was proposed to extend Proof-of-Work to mitigate the risks deriving from the ability to quickly gain computational power, which are at the root of bribery, or flash attacks [38]. Rather than using instantaneous mining power to select members, Proof-of-Reputation considers node's *integrated power*. In fact, it is calculated using the total amount of valid *work* a miner has contributed to the system, over the regularity of that work over the entire period of *time* during which the system has been active. Hence the name, and the physical bound on power growth rate. When a miner deviates from the system specifications, the miner's reputation, and hence its voting power, will lower in consequence of this negative contribution. This prevents a powerful malicious miner from attacking the system repeatedly without significant consequences, as can occur in the PoW-based systems.

In this way, an adversary can only be successful after gaining sufficient reputation, requiring costly investment over time. Thus, it can tolerate an attacker who controls a majority of mining power for a short time period. In addition, the node's reputation drops to zero as soon as an attack is detected, meaning an adversary can only launch this attack at one time before having to rebuild reputation. Therefore, any bribes or rented computational power require costly time investments, lowering the economical incentive for the adversary to launch the attack, further mitigating the risk of this attack occurring.

B. Proof-of-Stake (PoS)

Proof-of-Work's major impediments include the extremely high resource consumption to produce a block as well as the low throughput of transactions per second [49, 93]. Efforts have been made to pursue new mechanisms that can provide similar guarantees for a permissionless environment whilst improving on the fall-backs of Proof-of-Work. The first noted proposal on a stake-based voting system was through Wei Dai's b-money [61] in 1998, where it mentions users' voting on included transactions, staking money for dispute resolution. The concept of Proof-of-Stake in blockchain was first proposed in a Bitcoin community forum [169] in 2011 to provide quicker and more definite transaction confirmation through a virtual mining mechanism. The main idea of Proof-of-Stake is that the consensus participants are required to deposit something of value at stake, and this deposit will be taken away if the node is found to be acting incorrectly.

The virtual mining not only provides an environmentally friendlier blockchain system, by saving the overall energy consumption when compared to PoW, but also significantly improves the throughput as blocks can be appended and committed to the chain faster.

The concept of Proof-of-Stake provides the flexibility to be implemented in a variety of ways. Each implementation calculates the user's stake differently and imposes different incentive mechanisms to promote node behavior.

Key Concept: The primary motivation behind Proof-of-Stake is to mitigate the energy waste caused by the block production process in Proof-of-Work. The core concept lies in a node proving validity by staking assets, replacing the hashing to solve a cryptographic puzzle with a stake-based selection, while still preserving the permissionless nature of the blockchain. The stake is either taken from their current balance, or, is locked or deposited by the shareholder. The voting power of the node can be proportionally mapped to the stake they have issued. Figure 3 presents the Proof-of-Stake calculations.

"Nothing at Stake" is a well known attack on Proof-of-Stake systems, where stakeholders vote for all possible proposals by using the same stake. Since the voting will lead to an eventual block decision, the stakeholders will not lose any assets but gain profit, acting as an incentive to behave in this way. We will discuss more about such attacks in Section VII. To prevent

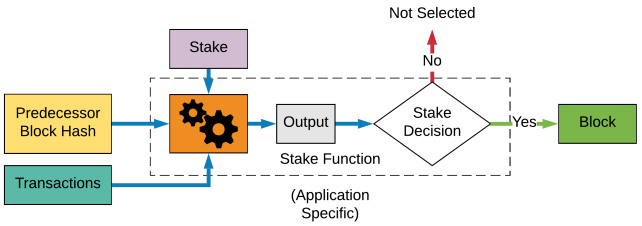

Figure 3. Proof-of-Stake flow.

such an attack, designs include punishments that result in asset loss upon detection of a stakeholder's malicious behavior.

The Proof-of-Stake core concept has been adapted into a number of different membership selection attributes. The goal of the membership selection is to filter the nodes to form a committee to participate in consensus, and can assign a weight to the node's vote proportional to the amount of stake held.

Weaknesses: Whilst Proof-of-Stake mitigates the energy usage tied to Proof-of-Work, it is prone to a number of inherent weaknesses. The outstanding issue with Proof-of-Stake is the increased potential of centralization and the concerns of governance. The distribution of assets, as well as people's willingness to stake their assets, greatly influence the membership selection and could lead to a small subset of nodes being in a position of power for the entire system operation. To solve this problem, some Proof-of-Stake implementations utilize frequent membership changes and integrate randomness so the committee changes and a larger pool of nodes is used. However, this is still greatly impacted by the use of incentive mechanisms [83].

Proof-of-Stake, while satisfying the requirements of membership selection, still suffers from a number of attacks. We defer the discussion of attacks to Section VII.

Goals and Assumptions: All Proof-of-Stake implementations share some common goals and assumptions. Members are selected through stake calculations using balances, deposits or votes. To increase the probability of selection, nodes obtain or deposit more stake. However, in some cases such as Delegated Proof-of-Stake [48], the nodes are selected by being voted in, where the votes are weighted proportionally to balance. Proof-of-Stake assumes that there is a limited amount of resources for a node to stake, and gaining the resource requires time or significant cost.

The various implementations provide incentives by rewarding, or punishing, behavior and participation. Nodes that are selected are often rewarded through transaction fees or block rewards. However, Proof-of-Stake is often accompanied by punishment; when a node is found guilty of misbehavior it loses the stake that it has deposited, or owns.

Similar to Proof-of-Work, Proof-of-Stake assumes that an adversary does not control large portions of the stake, that the stake follows a distribution that allows for the honest nodes to be selected with higher probability than the adversary.

1) Deposit based

Deposit Based implementations of Proof-of-Stake allow any node to deposit stake to have the chance to be selected as a member. The nodes have a choice to increase their probability of selection by depositing more stake, and vise-versa. Existing deposit based implementations include proof-of-lock and proof-of-deposit, as follows.

1.1) Proof-of-Lock

Tendermint's [123] implementation of Proof-of-Stake requires nodes to "lock" or "freeze" coins as stake and if found to act maliciously, the frozen assets are removed. The membership selection selects the nodes that have frozen coins to be part of the consensus group, where their voting weight is proportional to the amount of coins they have frozen as a ratio to total amount of frozen coins. This is effective against the "nothing at stake" attack, as the malicious behavior will result in direct loss of deposited coins.

1.2) Proof-of-Deposit

Similar to Proof-of-Lock, the Proof-of-Deposit as implemented in Ethereum's original Proof-of-Stake [79] proposal, requires nodes to deposit a selected amount of coins. The blockchain keeps track of a set of nodes that have deposited their coins and randomly selects members from this set to form groups for the consensus. Dependent upon the implementation [225], the number of selected nodes is different, i.e., a single node is selected if the chain-based implementation is chosen, or, several nodes are selected if the BFT-style implementation is used.

2) Balance Based

Alternative to the deposit-based implementations, balance-based implementations calculate stake using the node's current account balance, giving no choice in the amount of asset staked by a node. However, some implementations allow nodes to present their stake when they wish to be selected, whereas others assume the entire set of nodes are willing to participate in the membership selection.

2.1) Proof-of-Coin-Age

Proof-of-Stake allows for anyone to quickly gain assets and utilize them as stake, which could have adverse effects on the system stability. To prevent instantaneous stake gain, PeerCoin, or PPCoin [119], implements Proof-of-Stake through Proof-of-Coin-Age, which calculates the stake based on balance multiplied by the length of *time* a node has held that coin for. If the coin is ever traded, the elapsed time is reset to 0 and the stake is lost. This provides a disincentive for adversarial behavior as there is now a required time investment, proving to be costly.

2.2) Proof-of-Activity

Although instantaneous stake gain is a challenge faced by Proof-of-Stake, many implementations do not require explicit safeguards to provide resilience against powerful actors. Proof-of-Activity [28] selects "lucky" nodes which must be online to participate and get rewarded. This approach utilizes Proof-of-Work where a miner creates an empty block header to randomly derive N *Satoshi's*[6] amongst all coins in existence. The Proof-of-Activity's *Follow-the-Satoshi* is then invoked to select the owners of the selected Satoshi. The owner of this Satoshi is a stakeholder chosen to propose the next block. All stakeholders verify the empty block header to verify if they are the chosen stakeholder and sign the empty block. Once a stakeholder can verify that $N-1$ signatures have signed the empty block header, the stakeholder creates a wrapped block that extends the empty block header, adding the transactions and N valid signatures. Once the block has been accepted, the transaction fees are shared between the original miner of the empty block and the N stakeholders that signed.

2.3) AlgoRand's VRF

Previous approaches to Proof-of-Stake may be vulnerable to an adversary determining consensus members prior to the group creation, which could lead to a Denial-of-Service attack against the consensus and prevent block signing. To prevent this, AlgoRand's [94, 140] Proof-of-Stake implementation randomizes a new subset of members for each step of block commitment using a Verifiable Random Function (VRF). This implementation provides a unique feature, where only the nodes who are selected for the next consensus step know that they are selected, while other nodes can only verify this membership selection result afterwards. The use of VRF prevents an adversary from predicting the set of members chosen for the next step of committing a block. AlgoRand measures stake based on the amount of currency held by an entity. Each member is randomly selected, with voting power proportional to the amount of stake owned.

2.4) Ouroboros

Similar to AlgoRand, Ouroboros [118][7] applies a lottery-like method for membership selection. In particular, the set of current members for the consensus run a multi-party coin-flip to generate a uniformly random string. This string is then used to randomly derive the leader for the current epoch as well as the set of members for the next epoch.

3) Delegated Proof-of-Stake

Delegated Proof-of-Stake [2, 34, 35, 129] (DPoS) provides a democratic, vote-based membership selection. The *Stakeholders*, nodes that hold coins, vote on nodes to become

[6]A "Satoshi" is the lowest value of currency in the Bitcoin blockchain.

[7]A version of Ouroboros is presented in [63] which applies the "semi-synchronous" network model.

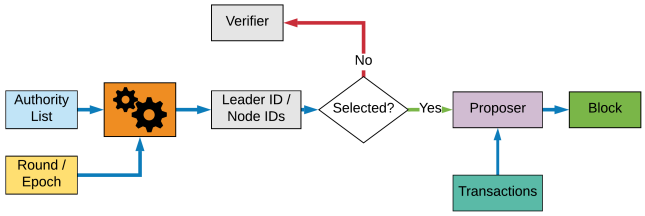

Figure 4. Proof-of-Authority flow.

Delegates for specified intervals. In some implementations, such as Lisk [129], the nodes are assigned a voting weight proportional to the amount of coins held. The Delegates are responsible for block production in the entire system, forming the consensus committee and signing blocks to be appended to the chain. To incentivize participation, Delegates are rewarded with block rewards or transaction fees.

BitShares [2] was the original Delegated Proof-of-Stake system to be implemented. The concept of "voting through transactions" was first discussed in the forums [48] and was later developed. Since BitShares, a number of DPoS systems have been implemented; Lisk [8], Steem [11], and EOS [6] are major examples.

4) Proof-of-Burn (PoB)

Proof-of-Burn [198] acts as an alternative to Proof-of-Work and is built upon the primitives of Proof-of-Stake. Initially discussed in the BitcoinTalk forum [199], the concept was proposed to rival Proof-of-Stake to provide something "difficult" for all to do. At the time it was proposed, "staking" in Proof-of-Stake only locked currency, in which a node could withdraw and their currency would still circulate. To prevent "burning coins" being effected by potential chain reorganization, the Proof-of-Burn specification outlines that the burning of coins should happen a number of blocks prior. The initial concept of burning coins was proposed as a way to transition between cryptocurrencies [176, 177].

Slimcoin [196], influenced by Bitcoin [150] and PPCoin [119], implemented Proof-of-Burn alongside Proof-of-Work as a mechanism to mitigate some of the requirements of powerful hardware. Proof-of-Work blocks are produced as in Bitcoin, whereas the Proof-of-Burn blocks are created by burning a threshold of coins, where the threshold is a function of the difficulty of the Proof-of-Work block as well as the number of coins burnt by the network. Proof-of-Burn blocks are only able to appear after a Proof-of-Work block has been mined. This makes Proof-of-Burn applicable to permissionless systems, as it inherits the same guarantees and properties as Proof-of-Work with the added benefits of combined Proof-of-Stake and Proof-of-Burn.

C. Proof-of-Authority (PoA)

Proof-of-Authority was initially proposed as an addition to the Ethereum blockchain [116, 156] for testnet[8] usage, which was a response to the risks with Proof-of-Work in this context. The core of Proof-of-Authority is that only chosen, trusted nodes are able to mint new blocks [20]. Although this centralizes the overall membership of the blockchain, the usage is suitable for small networks and testnets.

Key Concept: Proof-of-Authority was proposed to maintain a blockchain with minimal waste of energy and primarily focused for the testing network context. The core require Authorities that are chosen to verify and produce blocks for the blockchain. For each round, or epoch, a leader or subset of the authorities are chosen to produce the blocks that are proposed to extend the chain. The authority nodes are agreed upon in the configuration of the blockchain. A depiction of this can be found in Figure 4.

Weaknesses: Although Proof-of-Authority requires Authorities to be chosen as part of the blockchain configuration, there is a selected leader for each round. If the leader exhibits malicious behavior, there are cases where they are able to censor and produce invalid blocks. However, in some implementations the leader can be effectively removed as an authority if this behavior is observed and voted upon by the other authorities.

Goals and Assumptions: To provide membership selection in this environment, Proof-of-Authority variants require preselected nodes to form the authority committee, and thus, the consensus committee. From this committee, a leader is selected based on calculations of time [156] or blocks [116]. The network is therefore placing trust in the authorities to create valid blocks and progress the blockchain. Although this membership selection has a committee of "chosen" members, there is no explicit incentive reward or punishment, rather an implied reputation for the nodes running as the authorities to maintain their authority position as well as their status in the community.

1) Aura

Aura [156] is the Proof-of-Authority implemented by the Parity Ethereum client [181] and progresses in rounds. Out of the trusted entities, one is the chosen leader of the round who proposes, or "seals" by signing, blocks to the other members. Each round a leader is expected to propose a block, if there is no transaction available then an empty block is proposed. In the case that the block proposed is invalid, the trusted entities form a vote to kick the malicious leader.

[8]A testnet is a blockchain network that runs the blockchain but is used primarily for testing and has no financial value. It reflects the same characteristics as the main blockchain network, but can often be seen with changed consensus, smaller difficulty or quicker block times.

2) Clique

Alternatively, Clique [116] is the Proof-of-Authority implemented by the Go Ethereum client [178] (Geth). Similar to Aura, Clique progresses in epochs and has a leader for an epoch. However, to minimize the threat of a malicious leader, Clique PoA allows more than one Authority to propose a block, specifically $A - (A/2 + 1)$ where A is the number of Authorities. Each epoch elects a new leader, and each Authority is limited to proposing every $A/2 + 1$ blocks.

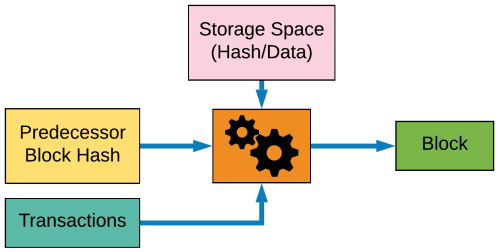

Figure 5. Proof-of-Capacity flow.

D. Proof-of-Capacity (PoC)

Similar to the motivation behind Proof-of-Stake, Proof-of-Capacity aims to reduce the amount of wasted resources on the blockchain by utilizing the resources that would be used for block production.

As mentioned previously, Proof-of-Work has resulted in an observable increase in the investment of computing hardware, through the renting of cloud computing, or, the purchasing of specialized hardware for mining. As a consequence, a drastic change to the mining process may cause the hardware to rapidly depreciate in value. The repercussions of such depreciation could see the loss of adoption in the new membership selection.

Proof-of-Capacity, otherwise known as Proof-of-Retrievability [114, 142] or Proof-of-Space [73, 159], provides a use for the hardware running the blockchain so that the block production process is not wasted. The process for block production utilizes hardware storage which is then used to maintain the blockchain, rather than wasting resources on useless computing work. The proposed membership selection requires a node to prove validity through storing and retrieving shards of a given file. Once the node has successfully accessed the storage and delivered the file, the node has not only proven validity, but also aided in the operation of the blockchain as the stored files are part of the blockchain itself. This process allows for the blockchain to be stored and sharded amongst the entire network, reducing the overall replication required, but maintaining enough redundancy to be fully decentralized. By doing so, information can be accessed at any given time while minimizing the storage costs for the blockchain. The implementation has been adapted to a number of blockchains, such as SpaceMint [159], Permacoin [142] and BurstCoin [91].

However, a major issue with proving validity through capacity is the impact of cloud infrastructure and the ease of providing largely scalable storage. To oppose this, implementations were designed to require fast access to storage, where the cost of outsourcing impacts the node's ability to participate. This mitigates the risk of an adversary purchasing large amounts of cloud storage to overcome the network for a short period of time.

Key Concept: The core focus of Proof-of-Capacity was to re-purpose hardware that was used for Proof-of-Work, avoiding the waste of computational and electrical resources. The main concept for this membership selection is that a node proves validity by storing important information and showing they can retrieve it at a later point in time. This provides motivation for a number of implementations that use the block creation process to aid the operation of the blockchain. By sharding information amongst nodes successfully, the blockchain storage can be decentralized whilst minimizing the number of full replications. A figure presentation of Proof-of-Capability is shown in Figure 5. Similar to the computation of Proof-of-Work, storage space is used to construct a proof of validity allowing a node to propose a block to the consensus.

Weaknesses: The variants of Proof-of-Capacity suffer from the advances of technology. The foundation is that space is utilized as an aspect of stake, a node provides storage space for the power to create blocks. However, the advances in cloud providers and large companies reveal an increasing possibility of centralization and monopolization in the mining process.

Goals and Assumptions: To provide membership selection in a permissionless environment, the variants of Proof-of-Capacity utilize storage space which also acts as a proof that the node is valid. For a node to be selected, it must perform a function on it's storage space, committing space or retrieving information, which generates the proof similar to that of Proof-of-Work. To increase the probability of selection, a node can increase its storage space which has an assumed cost. However, to act as a disincentive for cloud-based solutions, the proofs employ mechanisms to ensure nodes can only have local storage. This is enforced by taking data access time into consideration, and nodes access to remote cloud storage will lose its advantage in the mining process. Nodes are rewarded for successful selection and proposal with freshly minted coins, which also acts as an incentive to continue to commit storage space. In SpaceMint [159], "punishment transactions" allow misbehaving nodes to be punished and their rewarded coins revoked.

1) Proof-of-Retrievability

Proof-of-Retrievability was first proposed for storing large files in a distributed archive in 2007 [114], which was later adapted into the blockchain context through Permacoin [142] in 2014. Generating a proof of validity requires a node to prove it has access to selected files. A valid proof provided by a node shows that the node has a copy of the associated file.

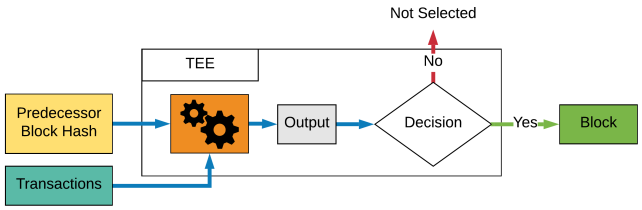

Figure 6. TEE-based flow.

To achieve a successful distributed archive, the system needs to incentivize storing files locally, away from cloud-based solutions. Proof-of-Retrievability employs this by requiring the consensus member to create a block through local random access to segments of files. In PermaCoin, the payment private key is integrated into the puzzle solution, so the solution must be created with access to this file. In addition to this, and as a further disincentive for cloud-based storage, the solution requires frequent random and non-precomputable access to storage. Thus, in comparison to storing files locally, a node outsourcing storage will have a delay in creating a valid proof.

2) Proof-of-Space

Similarly, nodes in Proof-of-Space [73] prove that they have committed dedicated storage space to the system. This concept was later adapted into SpaceMint [159]. To create a block and to be selected in the SpaceMint, each node creates a transaction committing dedicated storage space, and ties the commitment to its public key. Once the transaction is formed, the node computes a proof using the Proof-of-Space algorithm by taking the hash value of the last block as input and outputting not only the proof of space, but also a "quality" of the proof. The quality of the proof determines whether the node will be selected to propose a new block.

E. TEE-Based

With technological advancements in hardware, new alternatives were made possible to be used in blockchain systems. *Trusted Execution Environment* (TEE) is a secure space of a trusted hardware. It guarantees that code running inside will honestly follow the pre-defined policies and algorithms. The concepts of Trusted Execution Environments were originally heavily influenced by the ARM TrustZone [175]. From this, many major hardware manufacturers, such as AMD [174] and Intel [110] provide Trusted Execution Environments (TEEs).

TEEs allow for trust to be placed on the hardware running on a node, allowing algorithms that use randomness and waiting time to be utilized. This places stronger assumptions on the system, but allows for a range of different techniques to be explored, resulting in overall improved performance. However, although the technology provides great guarantees for running secure code, it is slowly being integrated with average consumer hardware. In addition, it requires users to trust the manufacturers of the trusted components.

Key Concept: Members can be selected by using a random that utilizes the trusted execution environments on hardware. Rather than wasting work or requiring nodes to sacrifice funds, TEE-based membership selection selects nodes with a probability proportional to the number of TEEs that node controls. This means a node is able to increase the probability of selection by purchasing more hardware devices. One critical aspect of TEE hardware is that each hardware device has a unique identification, meaning each node's hardware device can only be used once. This prevents a node from spoofing the number of hardware devices they are in control of.

Weaknesses: The prominent weakness in TEE-based membership selection is the reliance on specialized hardware. Although Trusted Execution Environments are manufactured to provide guarantees, other factors may influence the trust placed in the hardware. Trusting hardware requires placing trust in the vendors that supply the hardware, which inevitably can lead to centralization or monopolization through vendors.

However, even if monopolization or centralization provide negligible impact, the inherent trust in the hardware may prove fatal. TEEs, although manufactured and developed correctly, may suffer from a number of flaws or attacks [96, 113, 146, 194, 214], where an adversary can bias the operation of the blockchain.

Goals and Assumptions: TEE-based membership selection techniques, such as Proof-of-Elapsed-Time and Proof-of-Luck, place trust into hardware to aid with the selection of members. This allows for lottery-like algorithms to be used to randomly generate data, which is then used to determine whether a node is selected. The function to generate the randomization is run in the trusted environment, so the algorithm can assume that the output will be random. To increase the selection probability, the node will need to purchase more specialized hardware with TEEs. Figure 6 presents the abstract idea of this class of membership selection.

Proof-of-Elapsed-Time and Proof-of-Luck are often paired with reward-based incentive mechanisms, where winning proposals are rewarded with freshly minted coins or transaction fees.

1) Proof-of-Elapsed-Time (PoET)

Intel ltd. proposes Proof-of-Elapsed-Time [54, 112], which relies on secure instruction execution to achieve the features provided by PoW-based systems, while not consuming as much energy as PoW. This project was later implemented as a HyperLedger project, called "HyperLedger Sawtooth" [111].

Proof-of-Elapsed-Time is performed by each node in the blockchain network executing a '*sleep*' function for a random amount of time. The first node to wake from the sleep gets elected as the leader to propose the new block. The time to sleep is a random number generated using Intel's Software Guard Extensions (SGX) [110], which allows applications to run trusted code in a protected environment, this in turn ensures that the randomness of the sleep time generated for each machine is trusted randomness.

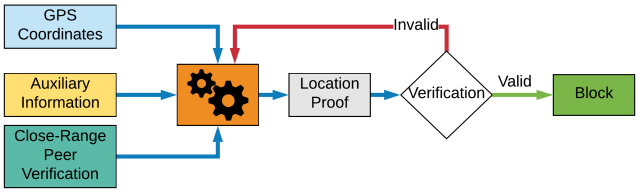

Figure 7. Proof-of-Location flow.

2) Proof-of-Luck

As an alternative method, Proof-of-Luck [145] proposes a similar concept in which nodes perform computation inside a Trusted Execution Environment (TEE), such as Intel's SGX, to generate a random winner for block proposal. The generation of the random winner occurs as all nodes execute the block proposal code inside the TEE producing a number that is compared against all other proposed numbers. The core element of this mechanism lies in the trust of the TEE to guarantee that all code run will produce valid expected output. The Proof-of-Luck also utilizes sub-proofs to prove that a node has provided the correct solution.

F. Proof-of-Location

The growing use of mobile and IoT devices has influenced the introduction of the blockchain to these platforms [67, 68, 103, 121, 184]. However, the major blockchains utilizing Proof-of-Work based membership requires computationally intensive tasks, impractical for the low-powered devices. The Proof-of-Location [39, 182] aims to utilize the dynamic location of mobile devices to create and verify proofs of a given device's location. This can be used for signing transactions, interacting with blockchain contracts or for pure verification of location.

Proof-of-Location [39] heavily relies on close peers to validate the location of another. Originally proposed for collusion resistance [228], the Proof-of-Location utilizes the geographic location of the node that is then signed by other nodes to verify that it is correct, as depicted in Figure 7. The location verification is required to stop geographic location spoofing, which is done through location requests as well as utilizing low-range communication channels.

Key Concept: The Proof-of-Location's core concept is to utilize the dynamic geographic location of mobile devices to integrate into the interaction with the blockchain. To mitigate the risk of spoofed location, some proposals for Proof-of-Location utilize a range of communication channels [39] for validation and verification of another node's location. Dependent upon implementation, the selection will work via verification of location, either a node is in a certain location [182], or, are able to verify their location with closest peers [39]. To ensure good behavior, the Proof-of-Location provides incentives to act good, through reward, as well as punishment if they are found to falsify the location.

Weaknesses: Although there are countermeasures to location spoofing, collusion amongst players may lead to spoofed locations being accepted as valid.

Goals and Assumptions: Proof-of-Location aims to provide a membership selection mechanism[9] based of a device's geographical location. To do so, the location must be verified by other nearby devices. This will enable the membership selection to be applied to a permissionless environment with no inherent trust required. The combination of both rewards and punishment act as incentives for the nodes to behave correctly.

V. Consensus

In this section we present an overview of the consensus mechanisms that have been applied to blockchain systems today. A summary of the consensus covered in this section can be found in Table III.

The consensus in the blockchain holds the responsibility of deciding the next chosen block to be appended to the chain. This step is vital to the operation of the blockchain as it produces the blocks that contain the transactions and state transfers. Byzantine Fault-Tolerant (BFT) protocols, also known as Byzantine Agreement (BA) protocols, were proposed to achieve consensus in the presence of distributed participants where some may be potentially malicious. This problem was originally detailed as the *Byzantine Generals Problem* [126] in distributed computing systems and has been thoroughly studied for over 40 years.

The Byzantine Generals Problem: The Byzantine Generals Problem depicts a situation where the Byzantine army is separated into different units, each commanded by a general. The army wishes to make an attack on the city, however, the attack will only succeed if the units are coordinated to attack at the same time. To prepare a coordinated attack, the generals must communicate through messengers and decide whether to attack or retreat. However, one or more generals may be traitors who will try to intentionally subvert the process so the loyal generals cannot arrive at a unified plan. The goal is to have every honest general agreeing on the same decision even in the presence of traitors.

To illustrate the problem, let the army have $2N+1$ generals in total, and one of them is a traitor. If half of the loyal generals want to attack, and half want to retreat, the traitor has the ability to misguide the set of generals to attack or retreat. This can lead to the situation where N generals attack and N retreat, fatal for the Byzantine army. This problem becomes worse as messengers are used for communication and the traitor may prevent messages from being delivered to a loyal general, or, by forging a falsified message.

A. Nakamoto's Consensus

Nakamoto's Consensus, introduced with Bitcoin, was the first consensus to be applied in the permissionless blockchain context, and has been adopted by many other blockchains [130, 190, 216]. The core aspect of Nakamoto's consensus is the *Longest Chain* rule, which is used in the event that two, or more, blocks had been proposed at the same height and the chains have progressed in a forked state. The longest chain states that nodes must select the chain with the highest block number as the single, canonical chain. The longest chain is believed to have the most work performed, adhering to the principle that the strict majority of the network, and therefore computational power, is owned by honest participants. In the case two chains have the same length, the nodes perform work on the first valid block they receive, while rejecting other conflicting blocks until one branch becomes the longest branch.

In the context of a permissionless blockchain, Nakamoto's consensus only guarantees eventual consistency [210], that is, if there are no new blocks proposed, all nodes will eventually see the latest block at the height of the chain. During the time the blockchain experiences a transient fork and consistency is not met, Nakamoto's consensus will provide the fork resolution eventually reaching a consistent state on all nodes.

Nakamoto's consensus provides security under the assumption that the strict majority of the network is honest, and when blocks are propagated through the network prior to any new proposals. With high block creation speed, more nodes will propose blocks based on stale information, as the information has not fully propagated before their proposal is created. This results in an increasing number of blocks not extending the "longest chain", which may have a number of implications for performance or potential attacks.

The rule of selecting the longest chain as a fork resolution protocol causes a number of proposals to be wasted. If paired with Proof-of-Work-based membership selection, then this can lead to a waste of mining power and, at large, energy, as all blocks that are not selected on the main branch are considered "not accepted" [65, 95, 153, 160].

One hindrance to Nakamoto's Consensus is the reduced security with frequent block proposals. As the voting power in the network increases, through node number or otherwise, the number of possible proposals increases. This resulted in a requirement in Nakamoto's consensus to have a limited number of proposals with satisfactory delay between proposals[10].

The increased frequency of block proposals presents a problem for Nakamoto's consensus, as it is heavily reliant on the dissemination of blocks prior to any mining. Once the block creation speed exceeds the propagation time, the probability of concurrent proposals increases and the honest nodes will have more conflicting blocks, wasting overall mining power. At this point, Nakamoto's consensus will select the longest chain and as a consequence a number of attacks, such as double spending, may have a higher success rate without the required majority voting power.

Goals and Assumptions: Nakamoto's consensus will operate correctly assuming a synchronous network, as it requires

[9]Proof-of-Location is used in a variety of ways. Some location proofs are utilized for interaction, others are utilized to share location in a decentralized context depending on the usage. [39, 67, 182]

[10]Typically, Nakamoto's consensus is coupled with Proof-of-Work, as in Bitcoin, and the delay of proposals are adjusted through a dynamically changed difficulty parameter.

all nodes to have knowledge of the latest blocks in order to correctly extend the main chain. Due to this, the consensus will only terminate given that all blocks for a given height are seen by all the nodes. In practice, it provides probabilistic consensus when its time bounds are exceeded as experienced by the Internet. The probabilistic consensus can be due to either termination or agreement not being reached as there is a possibility of unseen chains after sufficient time has elapsed. The Nakamoto Consensus model can be seen to provide partial order during time before agreement is met or before termination is achieved. Once the fork has been resolved, the common prefix increases. Nakamoto's consensus fails to ensure validity, where all correct nodes proposing a block with a transaction may still decide on an empty block upon termination of the consensus instance for the given height.

The adversary model employed by Nakamoto's Consensus places heavy assumptions on the block proposals, where the honest nodes produce new proposals quicker than the adversarial nodes, such that the honest nodes will always be proposing a longer, valid chain. This can be achieved through the majority of the voting power resource being owned by the honest nodes.

B. Ghost

Inherent impediments caused by the longest chain consensus prompted a proposal for a new consensus mechanism. The *Greedy Heaviest Observed Sub-Tree*, or GHOST, protocol [197] was proposed as a new consensus mechanism for fork resolution in the blockchain, attempting to alleviate the problems discovered in the longest chain approach. The conflict-resolution mechanism employed by GHOST utilizes blocks that are not included on the canonical chain, providing a mechanism to consider all efforts on the blocks. The chain selection algorithm iteratively selects the heaviest sub-tree, rooted at the genesis block, block 0, and building up the correct chain to the current block.

By including the number of proposed blocks that are not included in the canonical branch, as a calculation of a sub-tree's weight, GHOST provides some resilience to forks occurring in high throughput environments. The main advantage of GHOST is that it maintains the 50% threshold for attacks at higher throughput and larger block sizes. The key concept of GHOST is that the blocks not accepted on the canonical chain still impact the weight and therefore contribute to the fork-resolution protocol, providing resilience to some adversary activity in the case of high throughput.

Goals and Assumptions: GHOST, similar to Nakamoto's consensus, requires synchrony to achieve a consensus agreement. Under the conditions where synchrony is not achieved, it will fail to terminate or reach agreement, as nodes will be observing stale branches of the chain. During the point at which no branch is heaviest, GHOST provides a partial order, as the prefix of blocks is common to all forked branches until the fork has been resolved. However, GHOST does not achieve validity in proposals, as empty blocks can still be considered even though all valid nodes propose blocks with valid transactions being contained.

The adversary model follows Nakamoto's consensus, where the strict majority of the voting power belongs to honest nodes. This ensures that the heaviest sub-tree has the highest probability of being created by honest nodes.

Ethereum and GHOST: Computing the entire sub-tree requires a substantial amount of time and resources, impeding the progress of the blockchain. Ethereum utilizes a variant of GHOST, a simplified GHOST protocol for reward[11], where each block specifies an *uncle* block if it has competing predecessor blocks of the same height. The simplification comes with the depth, as the ancestors considered are only in the order of 7 generations, whereas the full version would go the full depth.

C. BFT Protocols

Traditional BFT protocols provide solutions to this problem through techniques such as the use of signatures, being able to detect the absence of messages, or otherwise. Given the context of the blockchain, where a number of nodes may be acting malicious or experiencing unusual network traffic, BFT protocols were a suitable selection to achieve consensus.

To utilize BFT protocols in a Bitcoin-like context, three main problems must be addressed. First, the original membership selection has been designed for Nakamoto/Ghost-like consensus, and it cannot be directly applied to the traditional BFT protocols. If not paired with a cost, this will allow an attacker to launch Sybil attacks [69] and disrupt the BFT consensus from reaching agreement or terminating. Secondly, the set of participants in the Blockchain eligible to participate in the consensus is generally not fixed, nor predefined. Thirdly, the participants may join or leave the system at arbitrary times, therefore the quorum size [134] required for agreement is not constant. These questions are generally addressed by applying a membership selection technique.

1) PBFT-based solutions

The Practical Byzantine Fault Tolerance (PBFT) [52] algorithm achieves consensus through leader-based communication with three phases. The first phase, *pre-prepare*, begins with the leader assigning a sequence number and multicasting to all other nodes. Once a node receives the pre-prepare, it enters the *prepare* phase and muilticasts to all other nodes. If a node receives $2f$ *prepare* messages that match the *pre-prepare* message, where f is the number of potentially faulty nodes, it enters the commit stage and multicasts a *commit* message. A message is then *committed* if a node had received enough commit messages from $2f + 1$ replicas that match the pre-prepare for the message. The *pre-prepare* and *prepare* phase are used to totally order the messages, whereas the prepare and commit phases are used to ensure requests that commit are totally ordered across all nodes.

[11]Details can be found at https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation

Table III
CONSENSUS MECHANISMS

		Nakamoto	Ghost	BA⋆	HoneyBadger	RepuCoin	DBFT	Tendermint	PeerCensus	Solida	ByzCoin	Thunderella	Avalanche	HotStuff	LibraBFT
Chain	Seminal Chain	Bitcoin [150]	- [197]	AlgoRand [94, 140]	- [143]	RepuCoin [221]	RedBelly [59, 60]	Tendermint [40, 123]	PeerCensus [64]	Solida [19]	ByzCoin [120]	Thunderella [164]	AVA [201]	- [219]	Libra [27, 179]
Assumptions	Network	Sync	Sync	Sync	Async	Partial-Sync	Partial-Sync	Partial-Sync	Partial-Sync	Sync	Partial-Sync	Partial-Sync	Sync[§]	Partial-Sync	Partial-Sync
	Online Presence	Online	Online	Online	Online	Online	Online	Online	Online	Online	Online	Sleepy	Online	Online	Online
	Adversary	$f' < \lfloor \frac{N'}{2} \rfloor$ [*]	$f' < \lfloor \frac{N'}{2} \rfloor$ [*]	$f' < \lfloor \frac{N'}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$	$f' \leq \lfloor \frac{N'}{3} \rfloor$, $f \leq \lfloor \frac{N}{3} \rfloor$ [⋆]	$f \leq \lfloor \frac{N}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$	$f < \frac{N}{5}$ [¶]	$f \leq \lfloor \frac{N}{3} \rfloor$	$f \leq \lfloor \frac{N}{3} \rfloor$
Goals	Agreement	Probabilistic [#]	Probabilistic [#]	Common Coin	Common Coin	Deterministic	Deterministic	Deterministic	Deterministic	Deterministic	Deterministic	Deterministic	Probabilistic [#]	Deterministic	Deterministic
	Termination	Probabilistic [#]	Probabilistic [#]	Deterministic	Probabilistic	Deterministic	Deterministic	Deterministic	Deterministic	Deterministic	Deterministic	Deterministic [†]	Probabilistic [#]	Deterministic	Deterministic
	Validity	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Total Order	✓/✗	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓	✓
Partition	Consistency	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗[‡]	✗	✓	✓
	Availability	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗

f Denotes the number of Byzantine nodes.
f' Denotes voting power of the adversary
N Denotes the total number of nodes.
N' Denotes Total voting power.
✓ The system provides the associated property.
✗ The system does not provide the associated property.
✓/✗ The system experiences forks resulting in states of total order prefix with tails of unconfirmed blocks.
§ The paper assumes a synchronous network, but conjecture that the results hold in a partially synchronous network. However, the proof is left to future work in the paper [201].
¶ The authors in [201] recommend that system designers select their value of Byzantine threhsold. The value here was an example provided in the paper.
* The adversary model follows that the majority of the voting power is owned by the honest majority. This can translate into "the honest majority produces blocks faster than the adversary".
# The termination OR agreement can be seen as probabilistic. Discussed in Section VIII.
⋆ With RepuCoin, only the top reputed miners in the committee have voting power, other participants have a reputation score, but have zero voting power before they become a top reputed miner.
† Under optimistic conditions with an honest leader. Under non optimistic conditions and a faulty leader, falls back to Proof-of-Work for the cooldown period.
‡ The Thunderella blockchain utilizes a Hybrid mechanism, which in non-optimal cases can fall back to availability in the presence of a partition.

In the case that the primary exhibits faulty behavior, or the replica "timers" expire, PBFT offers a *view-change* mechanism that elects a new primary. To perform the view-change, all replicas will send the `VIEW-CHANGE` message to all replicas, which contains the current state. The new primary then sends the `NEW-VIEW` message to all replicas to initiate the new view.

To provide BFT to the blockchain, systems such as PeerCensus [64], ByzCoin [120], and Solida [19] utilize variants of the PBFT algorithm tailored to their requirements.

Goals and Assumptions: The variants of PBFT explored in this survey, PeerCensus [64] and ByzCoin [120], exhibit similar goals and assumptions. Consensus can be reached in a partially-synchronous environment, where there are strictly less than a third of the nodes exhibiting Byzantine behavior. However, Solida [19] assumes a synchronous environment to ensure that a delta message delay is known a-priori. Agreement is reached, through the use of signed messages in a leader-based consensus protocol, however, ByzCoin provides a variant that uses a two-phase signing scheme to reach agreement. The termination, that of PBFT, is achieved when the nodes vote on the proposed block.

PeerCensus' Consensus: The consensus mechanism employed by the PeerCensus [64] blockchain is a variant of the PBFT leader-based consensus mechanism. The ability for a member to join and leave are managed through a secure group membership protocol (SGMP) [172], which allows for the members in the consensus committee to observe any changes in membership. This provides the knowledge of all members in the committee, allowing communication between known nodes and thus can perform the required communication.

To commit a transaction, the transaction must be sent to the current leader. The leader then performs the PBFT-based consensus with the current committee. Similarly, blocks are committed through the committee performing PBFT with the leader proposing any conflicting blocks and obtaining a decision from the members on which block is to be decided. This solves the second and third problems mentioned above.

ByzCoin's Consensus: ByzCoin [120] improves upon PeerCensus by altering the consensus and adding scalability through the use of a collective signing scheme, CoSi [200]. In particular, the consensus committee has a fixed size[12], and any newly elected members replace the oldest member. For each consensus execution, the committee executes the PBFT view-change protocol and elects a new leader.

The consensus protocol is a two-phase PBFT protocol which replaces the PBFT MAC authentication with the collective signing scheme presented in CoSi [200]. The signing scheme allows a large set of selected members to efficiently and collectively commit and produce a signature. This allows for larger committee sizes to run the consensus, thus improving scalability in respect to the committee size.

[12]The size of the consensus committee is implemented as the size of "window" in [120]

Solida's Consensus: In Solida [19][13], the consensus is a variant of PBFT, which is performed by a dynamically configured group of members. Similar to ByzCoin, newly selected members replace the oldest members in the committee, but also become the leader until a new leader is elected. A leader that is unable to make progress is replaced through a similar mechanism to the PBFT view-change.

Members are selected by ranking non-member miners based on their solution to the Proof-of-Work puzzle. If two solutions are found at the same time, the miners are ranked using the hash value of their solution. However, the selection progress has preference to select non-member miners.

2) RepuCoin

RepuCoin's consensus, integrated with the Proof-of-Reputation as discussed in § A.2, provides a weighted vote-based consensus. In particular, each member in the consensus committee is assigned a weight associated with that member's reputation. To reach agreement, RepuCoin requires both sufficient number of votes, and, majority of collective weight of the votes.

The main novelty of this weighted consensus comes from the decoupling of mining power and voting power. In particular, the voting power in RepuCoin is "integrated power" (i.e. reputation) which represents the total contribution of a node in the system, rather than the node's "instantaneous mining power" (e.g. hashing power in PoW) that can be gained quickly. So, a newly joined miner may have a lot of computing power, but it will not have any voting power before it achieves a contribution threshold that is relative to the contributions of other miners. In this way, RepuCoin is secure against an attacker who can even obtain a majority of mining power in a short time. Also, when detecting malicious behavior of a node, the reputation of this node will be set to 0. This not only reduces the incentive of a node attacking the system, but also prevents a malicious node from repeating the attacks without any consequence.

RepuCoin did not provide a detailed consensus algorithm, rather it provides ways to adapt existing secure BFT protocols, making the design more generic. In particular, one modification to make for adapting existing BFT protocols is the change of views and update of consensus committee. In RepuCoin, at the end of each epoch, the system enforces view-change and membership updates. Also, upon detecting crashed committee member, the system also updates the membership.

Goals and Assumptions: Currently, all proof-of-work based systems assume that no single attacker can control more than 50% of the mining power at any time, as otherwise the attacker can launch 51% attacks to double spend coins. For example, Ethereum classic has been attacked by 51% attack, and lost millions of dollars [57, 202].

The main goal of RepuCoin is to tolerate a malicious miner who may even control a majority of mining power for a time

[13]This work was previously known as "Solidus: An Incentive-compatible Cryptocurrency Based on Permissionless Byzantine Consensus", which was first available one arXiv.org at Dec 2016.

period. This goal is achieved by the separation of mining power and voting power, as previously mentioned. In addition, RepuCoin provides a higher throughput (about 10K TPS) by using a parallel chain structure, which we will detail in §VI.

RepuCoin's leader-based consensus operates using a selection of miners who hold the highest reputation, which is then translated into weighted votes in the consensus. The consensus operates under the assumption of a partially-synchronous network and can be achieved given that (a) no more than $\frac{1}{3}$ of selected nodes for running consensus are malicious; and (b) the honest nodes collectively hold more than $\frac{2}{3}$ of the voting power in the consensus committee.

3) Thunderella

Thunderella [164] provides a unique consensus for blockchains in two conditions, the optimistic conditions and the worst case conditions. In optimal conditions, the leader is honest and the adversary controls strictly less than $\frac{1}{4}$ of the online voting power in each round and can only corrupt a target node after a certain time delay. In non-optimal conditions, where the adversary controls strictly less than $\frac{1}{2}$ of the total voting power, Thunderella resorts to the underlying blockchain to satisfy the liveness and safety guarantees of their consensus model.

During optimistic conditions, the consensus is to simply collect $\frac{3}{4}$ votes from the consensus committee. This provides a fast, two-round communication consensus for a transaction to be committed. If the leader is seen as dishonest, and the optimistic conditions are not met, Thunderella falls back to Nakamoto's consensus. In particular, miners create a blockchain with uncommitted transactions. If the leader is honest, they should commit these transactions within a defined time period through the optimistic consensus. If this fails to occur, it provides evidence towards an incorrect leader and Thunderella falls back for a cool-down period. The Nakamoto's consensus ensures consistency of the miners' views of the blockchain.

Goals and Assumptions: Thunderella aims at eliminating the complex process of view-change in PBFT, by replacing it with the Nakamoto consensus. In this way, when the attacker is less powerful, Thunderella provides a high throughput; when the attacker is more powerful, then it falls back to the guarantees of normal Nakamoto consensus.

It assumes a partially-synchronous environment, and provides deterministic leader-based consensus in the optimal case. If there is a faulty leader, Thunderella falls back to Nakamoto's consensus. The optimistic case of Thunderella requires $\frac{3}{4}$ of the votes to be collected for a single block, in which only two communication rounds are needed to commit the block. Else, similar to the traditional blockchain consensus, $\frac{1}{2}$ of votes must be from honest members.

The consensus operates like a classical leader-based BFT consensus. The leader receives transactions and signs with a sequence number, which is then broadcast to the committee. Once the committee has received the transaction, they acknowledge and vote on the transaction, but only one per sequence number. Once the transaction has gained $\frac{3}{4}$ votes, it is considered committed. In the case it cannot be committed, the cool-down period will come into effect and Nakamoto's consensus will be utilized.

4) AlgoRand's BA⋆

AlgoRand's [94, 140] BA⋆ provides a vote-based Byzantine Agreement aimed at providing efficient probabilistic consensus. The members of the consensus committee are chosen through randomization provided by a blockchain-based common coin, which is also used in the consensus rounds. The selection occurs at each new block height, selecting both members and a leader for the committee.

The common coin is implemented by using the last block of the blockchain. In particular, leaders and consensus committees are selected based on the hash value of their signature on some commonly shared information, including the last block, the current consensus step number and consensus round number. The unique element in AlgoRand is that nodes are only able to see if they are selected and cannot see, or predict, other node's participation.

BA⋆ is expected to terminate within 6 rounds. Each BA⋆ execution consists of two phases. The first reduction phase, where a block decision is reduced to a binary value, and the second phase where voting on the binary value is run to reach consensus, or, reaches a decision on an empty block. Each phase consists of a number of steps to reach a final agreement. At each step the node performs local computation to produce a vote. The votes are then counted and steps progress only if the threshold of votes is met.

Goals and Assumptions: AlgoRand aims to provide a balance between scalability, decentralization, and security. In particular, it aims to prevent an attacker from predicting consensus committee members. This eliminates potential targeted attacks.

AlgoRand provides blockchain consensus in a synchronous network. The agreement of the BA⋆ algorithm utilizes a common coin paired with cryptographic signatures to reach agreement. The Byzantine Agreement is reached in 6 rounds, 13 worst case, but utilizes a common coin for randomization if a decision could not be made. The validity of the proposals requires the message to have a valid signature from the node in the current sortition, but also ensure that the block *seed*, a random seed for a round, is correct for the given round.

Although AlgoRand requires synchrony to reach agreement, it can achieve safety in periods of weak synchrony where the network is asynchronous for a bound period of time. The AlgoRand BA⋆ can provide consensus given that there are more than two thirds honest nodes.

5) Tendermint

Tendermint [40, 123] provides a leader-based BFT consensus for the blockchain. The selected nodes form a committee and take turns proposing a new block for the given height in rounds. Each round consists of a proposal of a valid block, voting to decide upon accepting the block, and finally, if more

than two-thirds of the committee vote to commit the block, the block gets appended to the blockchain and committed.

The Tendermint consensus requires a locking mechanism to ensure that a selected node does not double vote for different blocks at the same height. To ensure this, the selected node is seemingly locked if they had pre-voted and pre-committed for a specified block, however, they are able to unlock their vote if they have witnessed more than two-thirds of the votes for a block in a given round. The switching of a vote is permitted to protect liveness, but must be done in a way that it does not compromise safety, therefore only after two-thirds of the committee have voted to commit a block.

Goals and Assumptions: Tendermint provides consensus under the assumption of partial-synchrony, where the honest nodes control more than $\frac{2}{3}$ of the voting power. The consensus is deterministic, utilizing increasing timeouts for the proposal round, but allowing for asynchrony during vote stages. During the event that the network is partitioned, or more than a third of the nodes exhibit Byzantine behavior, the network may halt all-together.

6) HoneyBadger BFT

The HoneyBadger BFT [143], aims at providing blockchain consensus for permissioned blockchains, where the network is asynchronous but reliable channels exist between nodes that guarantee message delivery once a message is placed into the channel. The execution of the HoneyBadger BFT utilizes a reduction from multi-value consensus to a binary consensus, utilizing erasure codes to further improve the efficiency. To cope with the complete asynchrony, the implementation uses a common coin for randomization in the consensus.

For each consensus execution, each node of a pre-defined consensus committee (of size up to 104) randomly selects a set of uncommitted valid transactions, encrypts it through a threshold-based encryption scheme, and disseminates it to all nodes through a reliable broadcast protocol. Each node produces its part of the decryption share for each received message, broadcasts it, and perform the threshold-based decryption upon receiving $f + 1$ shares. The successfully decrypted transactions are considered to be accepted by the consensus committee, and are sorted in canonical order to place into the block. Following this, the adversarial model adopted is static faults, where an adversary is able to control up to f nodes, where $3f + 1 \leq N$.

Goals and Assumptions: HoneyBadger BFT provides blockchain consensus in a completely asynchronous network through the use of a common coin for randomization. Although the network is completely asynchronous, with no bound on the message delay, there is an assumption that there is a reliable channel between nodes to guarantee the message delivery. The common coin provides a high probability that agreement will be reached. Due to this, the termination is probabilistic, as the common coin can lead to disagreement. If less than a third of the members are exhibiting malicious behavior, then consensus can be achieved.

7) Democratic Byzantine Fault Tolerant (DBFT)

The consensus mechanism proposed by Crain et al. in [59] provides a deterministic algorithm for Blockchain consensus. The unique properties of this consensus is that it requires no leaders, signatures or randomization. However, by sacrificing liveness, it ensures safety even in the event of arbitrary delays. Another unique aspect of the Democratic Byzantine Fault Tolerant (DBFT) consensus is that it performs a union on the proposed values, allowing for the maximum set of values to be agreed upon and committed by the consensus committee.

Goals and Assumptions: The DBFT consensus provides blockchain consensus in a partial synchronous network. It can provide consensus given the Byzantine nodes are less than a third of the total consensus nodes, so a decision can be made with adequate votes. By removing the randomization, the agreement is deterministic based on the votes received by the nodes, in turn providing deterministic termination. DBFT provides validity by providing a union of proposals, such that if all honest nodes propose valid transactions, the block agreed in the consensus instance will not be empty.

8) Avalanche

Avalanche [201] introduces a class of leaderless Byzantine Fault Tolerant consensus protocols. The protocols utilize gossip communication and select subsets of nodes in the network to converge to a decision. A node uniformly selects nodes in the network, interacting with the subset of nodes and gossiping their decision. Once the node has interacted with the entire network, they can determine the decision based on the given threshold. Avalanche is built up from *Slush*, a non-Byzantine consensus core for metastability, *Snowflake*, a Byzantine Fault Tolerant adaptation of Slush, and *Snowball*, an adaptation of Snowflake with confidence for switching decisions. Inspired by Nakamoto Consensus [150], Avalanche provides probabilistic guarantees with probability $1-\epsilon$, where ϵ is a security parameter chosen by system. Avalanche utilizes a *DAG* structure for transactions and confidence.

Goals and Assumptions: Avalanche is modeled in a synchronous environment, but is conjectured to work in a partial synchronous environment. The leaderless consensus exhibits probabilistic guarantees, inspired by Nakamoto's consensus in Bitcoin. With these probabilistic guarantees, the consensus has a preference for availability over consistency, as it will allow applications to continue to be used and transactions committed once they reach a certain confidence threshold.

9) HotStuff

HotStuff [219] is a leader-based Byzantine Fault-Tolerant consensus mechanism that builds upon concepts from PBFT [52], Tendermint [40, 123] and Ethereum's Casper [46, 188, 225]. HotStuff presents a three phase consensus, with PREPARE, PRE-COMMIT and COMMIT phases. The three phases are required for liveness as it allows nodes to change their initial decision and not lock [195]. In each phase, the leader collects

messages, waiting for $N-f$ votes, where N is the number of nodes participating in the consensus and f is the number of Byzantine faults, before moving to the next phase. By moving to the next phase as soon as the leader collected enough messages, it allows for progress to follow the delay of the network rather than arbitrary waits. This is done by using *Quorum Certificates* (QC), a concept of signed data containing the $N-f$ votes to show proof of message reception as well as progression.

HotStuff also presents a linear ($O(N)$) view-change mechanism, used to elect new leaders [219]. The reduction in complexity allows the view-change to be integrated as part of the normal operation with minimal impact to the overall performance.

Chained HotStuff: Vitalik Buterin proposed a Casper [46] message reduction to improve efficiency by changing prepare and commit to simple votes. Similarly, HotStuff proposes an improvement by allowing a view-change to happen on *every* PREPARE phase, reducing the number of messages required for a single round (r), as well as pipelining the commit process. When collecting the votes for the PREPARE phase, the Quorum Certificate ($N-f$ *votes*) formed is passed to the next round ($r+1$) leader. The next leader initiates the PREPARE for their view, but also includes the PRE-COMMIT for the previous round $r-1$ as well as the COMMIT phase for round $r-2$.

Goals and Assumptions: HotStuff presents consensus with partial synchrony. The consensus tolerates strictly less than a third of the participants exhibiting Byzantine behavior. The protocol utilizes a core of three-phases to come to agreement, not requiring any randomization or coin flipping and thus provides deterministic agreement and termination. HotStuff utilizes timeouts to progress, however, if the threshold of messages arrive earlier than the timeout, it will interrupt and progress without having to wait for the timeout to be fulfilled.

LibraBFT: LibraBFT [27] is a blockchain-based Byzantine Fault Tolerant consensus which builds upon HotStuff [219] and is at the heart of Facebook's Libra blockchain [179]. LibraBFT extends HotStuff by requiring a new leader for each block proposal. The leader proposes a block, which is then voted upon by the remainder of the committee. Once the leader has collected enough votes ($N-f$), they form a Quorum Certificate with the votes and append the block and certificate to the blockchain. A new leader is then chosen for the next round.

The LibraBFT also extends HotStuff through a reconfigurable consensus committee per epoch, where membership selection can be utilized, such as a VRF [140] as mentioned in the proposal.

VI. Structure

In this section we present overviews and graphical representations of proposed structural changes to the blockchain. The Bitcoin blockchain originally featured a single, canonical chain with homogeneous block types. New proposals, however, provide a variety of new block types and structures that provide improvements and alternative ways that the blockchain can operate.

A. New block types

Some proposals focus improvements primarily on block types. By moving away from homogeneous blocks and adding new block functionality, new opportunities arise for improvements to the blockchain.

1) Bitcoin-NG

Bitcoin-NG [84] aims at improving upon Bitcoin's throughput by utilizing heterogeneous block types. In particular, Bitcoin-NG introduces two types of blocks, namely *Keyblocks* and *Microblocks*. Keyblocks are produced through Bitcoin's Proof-of-Work, taking several minutes to produce a block. The keyblocks contain no transactions and each time they are created, the miner is chosen as the leader to produce microblocks, so in this sense the microblocks are linked directly to the keyblock. Microblocks contain transactions and are produced without Proof-of-Work, so transactions are batched and can be produced quicker. Figure 8 presents an overview of this structure.

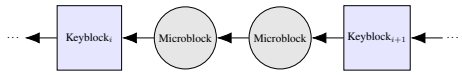

Figure 8. Bitcoin-NG Block Overview

2) ComChain

ComChain [209], the "Community Blockchain", proposes the *configuration block*, a new block type that defines a subset of nodes chosen to form the next consensus committee. When a new consensus committee is being proposed, a configuration block is sent to the current committee to be agreed upon. Once agreed, the new committee is formed and begins participating in consensus on the blocks containing transactions. The proposed configuration must be validated and all pending transactions and blocks should be transferred to the new committee to commit through the consensus. The structure is depicted in Figure 9.

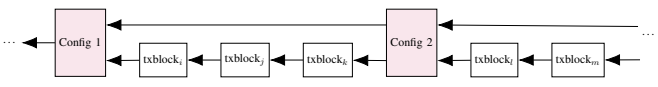

Figure 9. Community Blockchain (ComChain) structure.

B. New Structures

Rather than proposing changes to block types, some proposals focus primarily on the blockchain structure. The blockchain originated as a single, canonical chain appending blocks sequentially. However, new proposals provide a number of changes that allow for blocks, or transactions, to be committed

in a variety of ways. This presents opportunities for parallel block processing or new communication patterns between nodes.

1) Fruitchains

The Fruitchains [161] blockchain introduces an alternate structure to the blockchain and decouples transaction processing from block processing. Rather than transaction records being appended directly onto the chain inside a block, they are packed into a *fruit*, which is a batch of transactions. To append a fruit to a chain, it *hangs* from one block, referencing a recent block. The referenced block must not be too far from the block the fruit is attached to, as shown in Figure 10. By decoupling the transaction batching inside a block and having the fruits, transactions can be processed and committed to the chain with reference to the state they were processed in. This provides the ability to process more transactions while still ensuring the validity of the state.

In the Bitcoin blockchain, miners creating orphaned blocks will not get any reward, which is not fair to them as they also contributed their computing power. Fruitchains provides a fairer reward distribution, as the creator of fruits hanged from a recent block can also get its mining reward.

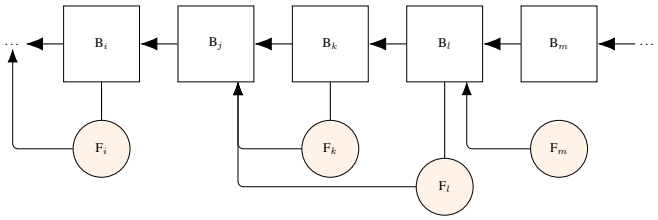

Figure 10. Fruitchains Structure

2) ByzCoin

ByzCoin [120], inspired by Bitcoin-NG [84], provides a new structure for the blockchain to improve transaction throughput by decoupling the transactions from the block creation. Unlike Bitcoin-NG, ByzCoin separated the microblocks and keyblocks into separate chains as shown in Figure 11, where two parallel chains store the information. The creation of Keyblocks is utilized to form a consensus committee. The consensus committee participates in BFT consensus to produce microblocks, which are appended to a second parallel chain every few seconds.

RepuCoin: Similar to ByzCoin, RepuCoin [221] presents a similar idea of decoupling transactions. RepuCoin utilizes the keyblocks for leader election and the consensus committee. When a keyblock is proposed, and the consensus committee agrees on this new block, it becomes *pinned*. From the moment the keyblock is pinned and committed, the microblocks are produced from the leader and consensus group. Like ByzCoin, RepuCoin follows a structure of parallel chains, where keyblocks are mined and committed, referencing each predecessor keyblock, whereas the microblocks run on a parallel chain referencing predecessor microblocks but also reference the most recent keyblock.

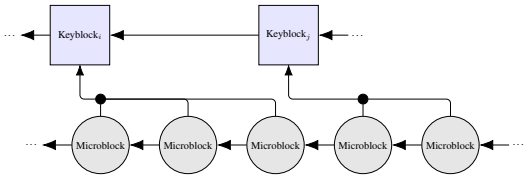

Figure 11. ByzCoin Microblock and Keyblock structure

3) HashGraph (Swirlds)

Swirlds [25] modified the traditional blockchain structure by proposing a graph structure where each "block" has multiple children or parents. Swirlds presents the *HashGraph*, a directed acyclic graph in which the "blockchain" utilizes all blocks that are mined. This removes the concept of stale blocks and encourages forked chains. As seen in Figure 12, the hashgraph contains vertices and edges, dubbed events and relationships respectively. An event is gossiped by a node can contain transaction information and must require hashes referencing two past events, which is then signed by the gossiping node. Through gossiping, the receiving node learns about the context of the event and all other information by previous nodes in the communication. The nodes can then agree on the ancestors of the event through offline virtual voting, as all needed votes from other nodes are contained in the received gossips. Although nodes may learn of events in differing places of their graph, an event is said to be consistent if it has equivalent ancestors in all node's graphs.

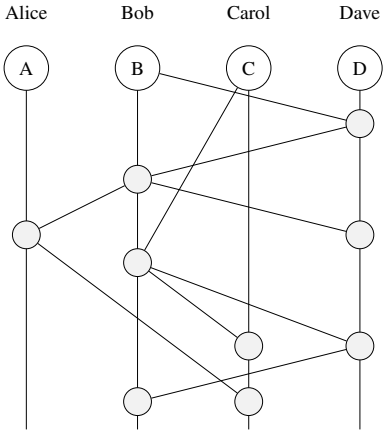

Figure 12. HashGraph Structure. An example round of communication and event gossiping.

4) Beacon Chain

Dfinity [100] envisioned a chain where a set of nodes selected through a *random beacon* would propose and notarize blocks. Ethereum Serenity [82, 208], proposed a similar concept but adopted a sharded blockchain model. The single, canonical beacon chain would serve as the backbone for the shards operating in parallel. The shard blockchains handle state transitions

and hold transaction information, whereas the beacon chain acts as a point of cross-shard communication and knowledge of shard states. Figure 13 illustrates the concept of the beacon chain with two shards.

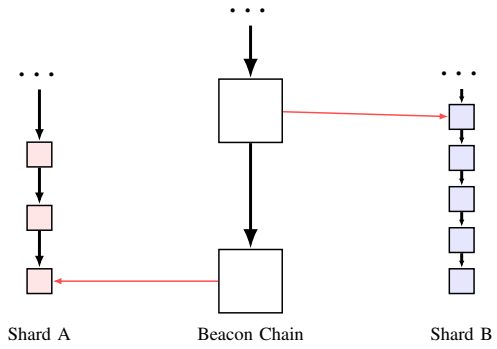

Figure 13. Beacon Chain as adopted by Ethereum Serenity

5) Tangle (IOTA)

IOTA [168] presents the *tangle*, a DAG which removes blocks from the storage structure. The tangle DAG, as illustrated in Figure 14, stores transactions as vertices and requires each vertex to have an edge between two (or more) previous transactions. This forms the directed structure and the edges constitute to the validity of the transactions. The transactions are gossiped through the network and are placed into the DAG of the node when received. Unconfirmed transactions, named *tips* are placed into the graph as leaves and are assigned a weight. The weight of the tip is proportional to the transactions it references, with respect to the recency and weight of referenced transactions.

For a tip to become validated, it must gain a high *confirmation confidence* by getting referenced by new transactions over time as the DAG continues to grow. Although the DAG may differ between nodes, the transaction ancestors will be common amongst nodes and will therefore be passed through the consensus to be agreed upon. Analysis [227] shows that such structure largely improves the scalability of blockchain.

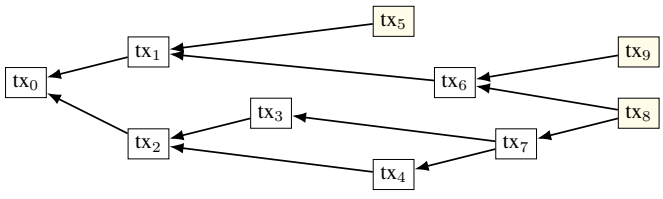

Figure 14. IOTA's tangle structure.

6) Block Lattice (RaiBlocks)

RaiBlocks [128] introduced the *block lattice*, depicted in Figure 15. The distinguishing feature of the block lattice is that each account maintains its own personal blockchain, the DAG ledger consists of the global set of accounts and their relevant blockchain. Each transaction sent from user to user requires two transactions, a "send" transaction signed by the transaction origin owner, and a "receive" transaction, signed by the party receiving the transaction. Similarly, each party must create a block on their respective chain, a "send block" and "receive block". Figure 15 depicts an example communication between parties. The pattern formed by the transactions forms a lattice of send and receive blocks, forming the DAG structure. Any conflicting transactions that are detected require a vote to be passed, in which the majority vote wins.

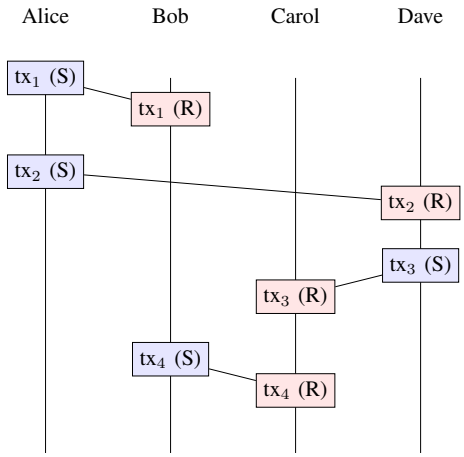

Figure 15. Block Lattice example execution and structure overview.

VII. ATTACKS

Blockchains are susceptible to a number of attacks [58]. Each varying implementation requires assumptions which often leads to limitations and vulnerabilities. The known attacks utilize these assumptions to empower the adversary to double spend, or, deny the service of the blockchain to others. Table IV, Table V and Table VI present an overview of the attacks reviewed in this section.

A. Mining Power Attacks

The mining power attacks target systems that utilize Proof-of-Work membership selection combined with any consensus that suffers from forks. The mining power attacks are focused on an adversary gaining sufficient amount of mining power to double spend, or, take control of the chain. To allow for more profitable mining, nodes often collude together to form mining pools, allowing cost sharing of profits and higher probability of block rewards. The Bitcoin network consists of a number of pools [37]. In July 2014, the GHash.IO mining pool exceeded the 51% threshold. At the time of viewing, the largest pool is the BTC.com pool which controls 29.1% of the networks computational power. The second highest, AntPool, controls 13.3%, closely followed by ViaBTC (10.6%) and SlushPool (10.1%). It can be seen that if the top three pools collude, they control the majority of the network and can therefore control the chain. Similarly, in the Ethereum blockchain[14], mining pools, such as Ethermine (27.5%), f2pool_2 (16.6%),

[14]Data available online: https://etherscan.io/stat/miner?range=7&blocktype=blocks. Accessed 25 Apr. 2018

SparkPool (15.7%) and Nanopool (14.1%), control a large portion of the network mining power.

51%: The most noted attack is the majority mining attack, also referred to as the 51% Attack. The attack allows an adversary to control the blockchain, validating invalid transactions and censoring the entire network as they wish. If an adversary is able to gain the majority of the mining power, they have the ability to produce valid blocks faster than other nodes, or pools, with high probability, meaning they are able to create alternative chains with a false history, allowing arbitrary double-spending.

This attack targets the adversary assumption that the honest nodes hold the majority of the network's resources. Due to the distribution of computing power in current Proof-of-Work blockchains, gaining a majority of the network's resources can be seen as a practically infeasible task, as the cost would be too significant unless the attack was carried out in early stages.

However, proposed techniques [21, 38, 101, 152] make it plausible for an adversary to gain a 51% of the resources and exploit this assumption. It is also worth noting that large mining pools develop over time and collusion amongst the large pools may provide enough power to control the network.

1) Bribery Attack (Flash Attacks)

The Bribery Attack [38], also known as the flash attack, consists of an adversary temporarily gaining the majority of the mining power for malicious use. This attack allows for the adversary to gain temporary control of the chain, allowing a window where they are able to double spend or censor transactions. The adversary gains a majority of the mining power by renting computing power. This can be achieved through out-of-band bribes, in-band bribes or through renting computing power. The methodology follows an attacker issuing a transaction in the chain to a merchant. Once the merchant is satisfied with the block confirmations, the adversary utilizes the gained computing power to create a conflicting transaction in a forked chain. The adversary controls the majority of the network mining power, and is then able to extend the forked chain quicker than the network and win the longest chain in fork resolution, where the merchant will no longer gain the funds due to the conflicting transaction. The adversary is not concerned about the long-term health of the chain, so consequences of the bribery attack may be felt by the network.

This attack can be facilitated by the use of cloud services providing easily available computing power, or through collusion of mining pools. However, a large investment or bribe is needed to gain the majority of the network power, even for a short period of time.

B. Strategic Mining

Strategic mining attacks exploit the decentralized nature of the blockchain, as each node contributing to the system is required to uphold their own local copy of the chain. Strategic mining attacks consist of withholding information from other nodes for later use to cheat the system.

1) Selfish Mining attack

This selfish mining attack was proposed where pools mine blocks in secret [85]. The attack requires an adversary to control a significant amount (>25%) of mining power, such that it can keep up with the chain progress. The attack methodology requires the pool to mine blocks in secret, selfishly keeping the produced blocks from the other nodes in the network. Selectively, the pool can choose to release the newly produced blocks to waste the honest miners computational power on producing old blocks.

An extension to this work [189] highlighted optimal strategies for selfish mining and showed that the selfish mining had optimal and sub-optimal conditions for success. Such optimal suggestions include conditions where the adversary should adopt network blocks, override network blocks of their own, match the block or wait. These actions, if selected effectively, could lead to a more optimal selfish mining strategy.

This attack is highly dependent on the pool's mining power. The pool must be able to produce a blockchain branch that is longer than the public branch. When they choose to release their branch, they will win the longest chain rule, allowing them to falsely operate the chain.

2) Finney Attack

The Finney Attack [86] was proposed in the Bitcoin forums as a way to double-spend against a merchant. The prerequisite for this attack's success is that a merchant delivers an irrevocable product or service at the time of payment, meaning waiting for 0 block confirmations. The adversary engages with the merchant and sends coins for the service or goods that are irreversible. At this point, the merchant completes the trade. Simultaneously, the adversary then creates a transaction to themselves and mines a block, which includes this transaction. The adversary then quickly broadcasts the block with the transaction to themselves. If quick enough, the transaction to the merchant will conflict, and the transaction back to self will be confirmed in a block and accepted by the network.

This attack leverages the merchant's assumption that transaction confirmations do not require block confirmations. To remedy this attack, merchants are advised to wait the recommended block confirmation, implementation specific, which will give a higher probability of the transaction being confirmed into the chain.

3) Fast-Payment Double Spending

Similar to Finney's attack, the Double Spending attacks on Fast-Payments [117] leverages a merchant's acceptance of a transaction with 0 block confirmations. *Fast Payments* are payments where a transaction is considered accepted within a minute of it being proposed in the network. Conversely, *Slow Payments* require a threshold block confirmation.

The adversary begins the attack by connecting to the merchant as a peer. The adversary sends the transaction to the merchant for an irrevocable action. After some small delta time, the adversary sends a conflicting transaction to the other peers it is connected to. There is a high probability that the

Table IV
ATTACKS OVERVIEW: PART 1

Category	Attack	Overview	Attack Core	Cause
Mining	51%	An adversary that controls the strict majority of the mining power has the ability to double spend coins and control the chain.	• An adversary is able to obtain strictly greater than 50% of the total network mining power.	$f' > \frac{N'}{2}$
	Bribery/Flash	An adversary bribes others to gain a strict majority of the mining power to perform a 51% attack in a short time period.	• Mining power can be paid/bribed or rented. • The overall cost of the attack is (a) less than purchasing resources to perform a 51% attack, and (b) less than the overall reward gained from performing the attack.	$f' > \frac{N'}{2}$
Strategic Mining	Selfish Mining	A mining pool mines a fork in secret. They then release the fork to the network using a smart strategy to waste the work of others and ensure their private fork is considered the longest chain.	• Pool has significant amount of mining power, and, can mine quicker than network. • No transaction or blocks leaked to network.	Probabilistic Consensus
	Finney's	The adversary sends a transaction to a merchant for some irrevocable action, however, the adversary mines a separate conflicting transaction into a block. The merchant performs the action as soon as the transaction is sent into the network. At this point, the adversary releases the block containing the conflicting transaction and the original transaction to the merchant is invalidated.	• Merchant trades with 0 block confirmations. • Adversary's original transaction does not get mined into a block before releasing the conflicting block.	• Synchrony • Favor Availability • Probabilistic Consensus
	Double Spending Fast Payments	An adversary connects as a peer to a merchant and issues a transaction directly for an irrevocable action. The merchant accepts the transaction without any block confirmations and performs the action. Simultaneously, the adversary sends a conflicting transaction to the other peers. With non-zero probability, the conflicting transaction propagates faster and is mined into a block.	• A merchant accepts a transaction with 0 blocks. • The adversary's conflicting transaction propagates the network quicker. • The adversary's transaction is mined into a block before the merchants.	• Synchrony • Probabilistic Consensus
	Rosenfeld's	An adversary solo mines a private longest branch in secret, withholding all information from other nodes. They then interact with a merchant, who requires some number of block confirmations before executing an irreversible action. Once the merchant is satisfied and performs the action, the adversary releases the solo mined branch which wins the longest chain and is accepted as the main chain, invalidating the transaction to the merchant.	• An adversary can mine a branch in secret without any information leakage. • The adversary can mine the longest branch. • A merchant trades with a low block confirmation.	• Synchrony • Probabilistic Consensus
	Vector76	An adversary interacts with a merchant who performs an action as soon as the transaction is accepted into a block. The adversary quickly sends a self-mined block containing the transaction directly to the merchant and the action is performed, while simultaneously submitting a conflicting transaction to the network. Once the merchant learns of the network's chain, the adversary's block is invalidated and lost.	• A merchant trades with 1 block confirmation. • The adversary can mine a block quicker than the network. • The Merchant never learns of the network's block and the adversary can propagate the block directly to the merchant quicker than merchant learning of network block.	• Synchrony • Probabilistic Consensus
	Withholding Mining Pools	An adversary joins a mining pool, but hides all valid Proof-of-Work they mine. This lowers the potential revenue for the pool as they miss out on a number of blocks.	• Static pool membership; all members stay in the pool regardless of revenue profits. • Membership rules are not strict and do not require nodes to submit certain amounts of work.	This attack is not on membership selection nor on consensus. Rather, it is an attack on the mining pool software and infrastructure, which is independent to the blockchain system.
Strategic Mining + Communication	Stubborn Mining	An adversary mines blocks with the honest network, only selfishly mining when an optimal strategy proves successful. By releasing blocks periodically (rather than the entire chain at once), the adversary is able to mine a private chain and gain profit for longest chain.	• Adversary can mine blocks in a private chain and keep up with the network. • No information leakage of the adversary's chain unless specifically by the adversary.	• Synchrony • Probabilistic Consensus

conflicting transaction propagates through the network faster than the original transaction to the merchant and is mined into a block. Once the transaction has been mined into a block, the double spend has occurred.

This leverages the merchant accepting "fast payments" where a transaction is accepted with no block confirmations in a small period of time. Secondly, it requires the conflicting transaction to propagate the network and be mined into a block quicker than the original merchant transaction.

4) Rosenfeld's Attack

Rosenfeld's proposed attack [187] is a variant of the strategic mining attack, where nodes withhold block production from other nodes. The adversary solo mines a long branch in secret, keeping all information from other nodes. The adversary then interacts with a merchant, issuing a transaction for an irreversible action. Once the merchant has successfully completed the action, and is satisfied with block confirmations, the adversary releases the longer, solo-mined branch and conflicts with the main, public branch. Since the adversary holds the longest branch, the adversarial chain is accepted and the merchant's transaction is no longer in the chain.

This attack is highly dependent on the adversary controlling a major portion of the computational power. If the adversary is unable to solo-mine a longer branch, the attack will not be successful as the smaller branch will not be accepted by other nodes in the network. Furthermore, if the merchant waits for a large amount of confirmations, the adversary will be required to mine a longer branch for longer, making the cost of the attack rise.

5) Vector76 Attack

The Vector76 attack [206] was proposed on the BitcoinTalk forums by the user "vector76", where an adversary wishes to double-spend against a merchant. The requirement for this attack's success is the merchant's assumption that a transaction is confirmed within 1 block, meaning as soon as the transaction is accepted into a block it is confirmed. The adversary interacts with the merchant, and creates a transaction to pay for irreversible goods or services. The adversary mines a block in secret, containing the transaction to the merchant. When the adversary learns about a new block, they quickly send the block they mined to the merchant directly. At this point, the merchant can see the transaction in the block and performs the action, or trades the goods. When the merchant learns of the forked chain, they will select the other chain as correct, discarding the adversary's block, which includes the adversary's transaction. The adversary double spends the coins by including a conflicting transaction in the alternate chain.

This attack is dependent upon the merchant accepting the transaction as soon as it has been included in a block. Furthermore, the adversary must be able to provide a valid block faster than the network, and propagate it to the merchant quicker than the merchant learning of the alternate chain. If the merchant receives the network's block faster than the adversary's block, they may decide to reject the adversary.

6) Withholding Attack against Mining Pools

The Block Withholding Attack [24, 186] is an attack targeted towards mining pools. An adversarial miner joins a mining pool and begins the mining protocol. Mining in a pool generates both partial Proof-of-Work and Full Proof-of-Work. The adversary can determine whether the proof-of-work they have mined is valid or invalid. When the adversary mines a valid hash, they withhold it from the pool, wasting the time of the pool and having the potential for the pool to miss out on a number of blocks. The motivation for this attack is to attack mining pools and lower their overall block reward.

This attack heavily relies on the share model of the mining pool, but can be mitigated by having strict conditions on the membership of the nodes in the pool as well as adjusting the pool based on the profits made.

7) Stubborn Mining

Selfish mining attacks alone may prove to be too impractical for real-world applications. The requirements for performing selfish mining lower the profit margin. However, through applying optimal mining strategies, and potentially combining with network-level attacks, the revenue may prove to be profitable.

The Stubborn Mining attack [153], proposed by Nayak et al., builds upon selfish mining by introducing mining strategies to increase success and revenue, rather than selfishly mining and releasing the private chain in one instance. The *Lead Mining* strategy is where the adversary leads by k blocks and releases blocks the same height as the network-mined blocks. Alternatively, the *Equal Fork* strategy means the adversary would conceal any new blocks even if they have no lead, and mine until they lead. The *Trail* strategy depicts a miner that continues to mine even if they fall behind, until a certain threshold j number of blocks behind. If the miner is j blocks behind, they accept the honest network chain and continue to mine, however, if they succeed and catch up to the network's chain they then introduce the fork resolution where they own the blocks that are in the chain and can profit. By applying these strategies, a stubborn miner is able to increase revenue and make their chain the chosen chain.

These strategies applied by a stubborn miner can be combined with network attacks to further improve the revenue potential and percentage of success of the attack. Examples such as the Eclipse Attack [101] can be used to isolate nodes and the adversary can then censor or collude with the eclipsed miner to gain profit.

This attack requires an adversary to mine blocks to try and keep up with the network. Furthermore, the strategies require no blocks to be leaked from the adversarial chain.

C. Communication Attacks

Alternatively to mining attacks, the adversaries can utilize the peer-to-peer network of the blockchain to attack the system. The segregation and isolation of nodes can often be leveraged to perform double-spending attacks.

1) BGP Routing Attack

BGP is the routing protocol responsible for connecting Autonomous Systems (AS) together and routing IP packets through the internet. Autonomous Systems are responsible for publishing IP prefixes, and Routers utilize this to route packets. Since the blockchain requires nodes to connect to peers over the internet, BGP is inherently used to route the packets to the other nodes. However, in BGP the validity of any route advertisements are unchecked, as any AS can advertise a given IP prefix and if they advertise a more specific prefix (e.g. \21 is more specific than \20) they will become the preferred route for the routing from other ASes. However, monitors [218] and methods exist to detect [191] and provide resilience [226] against BGP prefix hijacking.

The BGP Routing Attack [21] proposed against the Bitcoin blockchain exploits the lack of verification in BGP to isolate a group of nodes. Once the group of nodes has been successfully isolated, the adversary can then delay block propagation or leave the nodes uninformed, utilizing their mining power for their own benefit.

This attack relies on the ability of the adversary to isolate nodes in the network through BGP route hijacking, and upholding the advertised route for a long period of time.

2) Eclipse Attack

Hijacking BGP routes can be seen as a difficult task, as the adversary is required to advertise a specific prefix and maintain the preferred route for all isolated nodes. However, other methods exist to isolate specified node(s) on the network for an adversary to perform an attack.

The blockchain is a decentralized, peer-to-peer network where all nodes connect to others and form a gossip-based network. All nodes learn about information broadcast from other nodes. Different implementations allow nodes to connect to various numbers of other nodes, in Bitcoin it allows for 125 peers to be connected, 117 incoming and 8 outgoing connections, whereas Ethereum defaults to 25 peers.

The Eclipse attack [101] describes an attack in which a node can be eclipsed and isolated from the network by leveraging the peer-to-peer connections of the blockchain. The adversary populates the peer tables of the victim node, spoofing as many possible IPs into the victim node as possible. When the victim node restarts, they will try to reconnect to the nodes in the peer IP table. If the adversary has successfully populated a large percentage of the IPs in the peer table, the node will connect all peers to the adversary. The adversary can now control the information seen by the victim, and is now able to execute double-spending attacks or delay attacks.

This attack relies on the adversary being able to insert a large number of IPs into the victim's peer table. It also leveraged the eviction mechanism used by the node to replace old IPs. Countermeasures have been suggested, and implemented [101, 148], to become resilient to the attack.

3) The Balance Attack

Alternative to isolating a single node, the assumptions of the blockchain consensus can be used to assist an adversary to execute double-spending. Inspired by the Blockchain Anomaly [151], the Balance Attack [152] proposes an attack where the adversary partitions the network into groups of nodes with balanced computational power. Each subgraph of nodes will create a fork of the chain, which at some point, will require a resolution. The adversary issues a transaction into the fork containing the merchant. The merchant waits for the specified block confirmations, and performs the action providing irrevocable goods or services to the adversary.

At this point, the adversary then issues a conflicting transaction in another subgraph, and provides additional computational power, assisting to create a longer, or heavier, chain. The adversary slowly re-introduces the subgraphs together, growing the number of nodes that accept the adversary fork containing the conflicting transaction. When the merchant's subgraph learns of the fork, they will accept the adversary fork and the transaction to the merchant will be invalidated on the fork, executing the double-spending.

This attack heavily relies on the ability of the attacker to partition the network into balanced subgroups where their computational power will provide a significant benefit to the subgraph, and keep the network segregated until the fork can be resolved.

4) Man-in-the-Middle Attacks

The Man-in-the-Middle (MitM) attacks proposed against Ethereum [74] illustrates the applicability of BGP route hijacking to double spend on a public blockchain. By hijacking a BGP route, the adversary can manipulate the communication delay between nodes and subgroups of nodes. Once the adversary is in control of the communication delay, they can leverage their position to perform other attacks, illustrated with the Balance Attack [152] in the proposal.

Similarly, Ekparinya et al. also show the effectiveness of a MitM attack in a private blockchain context by using ARP spoofing [74] to double spend.

5) Attack of the Clones

The Attack of the Clones [75] presents an attack on the Proof-of-Authority blockchains in Ethereum [116, 156, 216]. This attack requires an adversarial authority, or a number of adversarial authorities, to duplicate their node instances and require them to have a exhibit the same public-private key pair. The adversary then partitions the network, with one cloned instance in each partition. The network continues to create blocks in separate partitions, and therefore double spending and conflicting transactions can be proposed and accepted into the partitions.

The proposed mitigations to the attack include the use of a partially synchronous consensus algorithm, such that message delays have lower impact on the operation of the system. Similarly, requiring a threshold of the sealers to sign the blocks helps to harden the protocol, as it requires the adversary to have a larger pool of sealers to allow the block to be produced in the forks.

Table V
ATTACKS OVERVIEW: PART 2

Category	Attack	Overview	Attack Core	Cause
Communication	BGP Routing	An adversary hijacks BGP routes and isolates groups of nodes. Once successful, and no information is being leaked, the adversary can control the information the nodes observe. This allows for potential double spending or utilizing the group's mining power for personal profit.	• Adversary can advertise (and hold) BGP routes without autonomous systems being suspicious. • No multi-homing for nodes. • No information leakage across the isolated group(s).	Synchrony
	Eclipse Attack	An adversary eclipses a node to isolate them from the network. The adversary achieves this by populating the victim's peer tables, due to eviction mechanisms the adversary can control enough peers to completely isolate the node. Once successful, the adversary now controls information observed by the victim and is able to double spend.	• Adversary can spoof a high number of peers, and keep the connections alive. • The peer eviction mechanism doesn't use "try before replace". • There is a low number of buckets in the peer table.	Synchrony
	Balance Attack	The adversary partitions the network into balanced powered subgraphs of nodes. Once the subgraphs have forked, the adversary transacts to a merchant in one subgraph, and joins another subgraph to increase it's block production. Once the adversary's subgraph has a longer fork, they begin to resolve the partition across subgraphs, resolving the fork and double spending the coins to the merchant.	• The adversary can partition and control the network. • The adversary can transact in two subgraphs without any information leakage or nodes knowing of the segregation. • Adversary can contribute enough block creation power that their chosen subgraph forms the longer chain.	• Synchrony • Probabilistic Consensus
	MitM Attacks	The adversary hijacks routes (BGP) or spoofs (ARP) to Man-in-the-Middle the communication against nodes. Once the adversary is positioned correctly, they are able to manipulate the communication delay between nodes and perform other communication attacks.	• Adversary can advertise (and hold) BGP routes without autonomous systems being suspicious. • No multi-homing for nodes/mining pools. • The adversary is the preferred route for communication and is situated in the middle. • The adversary is able to manipulate communication delay between nodes.	Synchrony
	Attack of the Clones	The adversary partitions the Proof-of-Authority network and duplicates their node to be present in each partition with duplicated public-private key pairs. Proof-of-Authority will then adjust to ensure progress, and the adversary will be able to produce blocks on each partition and therefore make conflicting transactions or double spend.	• Adversary can duplicate their instance without being detected as a duplicate. • Adversary is able to partition the authority groups and place one clone in each partition. • The block proposal/validation will still progress even with a significant number of authorities partitioned.	Synchrony

D. Stake Attacks

With the increasing popularity of Proof-of-Stake based approaches, identified attacks allow an adversary to gain power and profit in the chain by investigating minimal stake, or to make other selected nodes (a.k.a. validators or consensus group members) loss their stake. Proof-of-Stake, however, suffers from considerable validator collusion. If the majority of the validators collude to perform actions, then the Proof-of-Stake mechanism can fail in some cases.

1) Nothing at Stake

One of the first attacks on Proof-of-Stake is called "Nothing at Stake" [83], where rational validators aim at getting rewards by not following the protocol's specification. Proof-of-Stake based validation incentive means validators are encouraged to follow protocol to earn rewards. In the case of a fork, the most optimal strategy for validators is that they should validate on all possible chains for the chance of most reward.

Adversaries are able to leverage this behavior, and with non zero probability execute a double spending attack. The adversary creates a transaction to a merchant, and simultaneously creates a fork in the chain with a conflicting transaction. At this point, the validators will begin validating all chains due to the economic incentive. With non-zero probability, the adversary's forked chain may be selected as the correct chain to extend and the double spending has occurred with the attacker staking nothing during the process.

A solution to the Nothing at Stake problem is for the validators to commit value to sign and create valid blocks. This acts as a disincentive for the validators creating and signing blocks on all forks, or they will lose some money and be ultimately punished.

Alternatively, unforkable blockchains also present a solution to the nothing at stake problem as the adversary will be unable to create a fork with the conflicting transaction.

2) Discouragement

Alternatively, the adversary can discourage validator participation in the validation of blocks. In some proof-of-stake schemes, the validator group is punished if certain conditions are met and disagreement occurs. The Discouragement Attack [207] outlines an attack where the adversary can purposely act maliciously and broadcast their intention to other validators. Due to validators possibility to lose money, validators withdraw themselves from the committee. Eventually, the number of nodes in the validator set will diminish, due to discouragement, and the adversary can now perform a majority attack.

3) Censorship

Instead of validators working independently, adversarial collusion can lead to validators dictating the operation of the blockchain. The Censorship attack [44] can occur when validators collude to censor certain transactions from being placed into blocks. The attack ramifications lead to the destruction of the blockchain ecosystems and breaking the core values of the blockchain. This attack relies on the collusion of the validators, as the majority of the validators would have to agree on the censoring of specific blocks or transactions. However, adversaries may leverage other methods to lower the number of validators in the set until satisfied with the conditions.

Possible solutions to censorship arise with introduction of cost or mechanisms to the consensus. An example of changing the reward structure to be proportional to the number of validators that signed the block, for example if 98 of 100 validators sign the block, then the validators that signed would only be getting 98% of the reward. Unless full collusion amongst all validators occurs, then the validators censoring would not receive the full bonus and may not be economically incentivized to uphold the censoring. Rotation of validators for each block can also overcome censorship, as the validators performing the censoring would only be able to in the time window.

4) Grinding

Implementations of Proof-of-Stake have mechanisms to select a validator from the active set of validators to create and propose the next block. In some cases, the selection of a validator follows the rule that a node has a probability of being selected proportional to the amount of stake they have contributed as a percentage of the total. The Grinding [79] attacks are attacks where a validator takes steps to increase the bias of their selection as a validator, through some computation or other methods.

With PeerCoin, or PPCoin [119], the stake is based upon the coin-age, and the validators probability of success is determined based on the age they consume in the block creation process. The Grinding attack can be applied in this scenario where the adversary performs computation and grind through combinations of parameters in blocks to provide the best probability of their coins creating a valid block.

Similarly, NXT [155] implements Proof-of-Stake where the randomness for the next block is dependent on the signature of it's predecessor in the chain. Due to the nature of the signatures, the validator can execute a grinding attack and compute the randomness, selectively manipulating randomness [41] by skipping their turn to create a block. The validator can then calculate the optimal strategy such that they gain an above-average number of blocks to sign, leading to higher reward for the validator.

Known solutions [41, 79, 119] exist to the stake grinding attack. In PPCoin, validators can be asked to stake their coins well in advance of the block creation, such that the validator cannot grind the best probability for their coins to win. The randomness can also be improved through the use of secret sharing and threshold signatures, as well as validators all collaboratively generating the randomness.

5) Stake Bleeding

Proof-of-Stake provides a number of improvements to the current blockchain, however, it has been outlined that it suffers from a number of theoretical attacks and vulnerabilities. To oppose the vulnerabilities, proposed security measures have

Table VI
ATTACKS OVERVIEW: PART 3

Category	Attack	Overview	Attack Core	Cause
Stake	Nothing at Stake	Members have incentives to validate blocks on all possible chains to gain the most profit. This means an adversary can fork the chain and have a high probability of successfully getting their chain accepted.	• Members do not lose stake on validating an incorrect chain. • Adversary does not lose money by forking the chain. • The blockchain suffers from forks.	Incentive Punishment
	Discouragement	In Proof-of-Stake, members lose stake if agreement cannot be reached. An adversary threatens and discourages members from staying in the consensus committee by acting maliciously and purposely avoiding reaching agreement. As members leave the consensus committee, the adversary will gain a higher percentage of stake and can eventually perform a majority stake attack.	• An adversary can sacrifice enough stake for members to leave. • Honest validators do not join the validation and overthrow the adversary.	• Incentive Punishment • $f > \frac{N}{2}$
	Censorship	Adversarial collusion between members can lead to censorship of blocks and transactions from being accepted to the chain.	• Majority validator collusion. • No cost model that hinders rewards. • No adequate validator rotation.	$f > \frac{N}{2}$
	Grinding	An adversary grinds through parameters, or known computations, to find a strategy that will increase the number of blocks they validate, or, the amount of reward they get. The main aspect is to maximize the node's probability of selection by finding optimal parameters.	• Validators don't require to stake coins in advance. • Bad, or computable randomness.	Incentive Punishment
	Stake Bleeding	An adversary creates a hidden, forked chain. In the main chain, each time they are asked to validate they refuse and slow the block production. Eventually, the adversarial chain will grow and can win the longest chain if the main chain is delayed enough.	• No checkpoint mechanism implemented.	• Synchrony • Probabilistic Consensus
	Long Range	An adversary interacts with a merchant and sends a transaction for an irrevocable action. Once the action has been performed, the Adversary starts a fork from a significant distance in the past (e.g. directly after the genesis block) and eventually produces a longer chain. The adversary can then produce the longest chain and win the fork resolution.	• Timestamps are not used for authentication. • Context sensitive transactions are not used. • No checkpoint mechanism implemented.	• Synchrony • Probabilistic Consensus

been implemented which provide mechanisms countering the possible attacks. The introduction of checkpointing, a measure to solidify the blockchain state and eliminate all forks that are not complying with the latest checkpoint, allowed for a number of attacks to be mitigated.

For Proof-of-Stake blockchains that fail to implement checkpointing, as well as employ transaction-fee reward-based incentives, can be vulnerable to the Stake Bleeding attack [92], where an adversary can convince the network to adopt a secondary chain. The attack methodology requires an adversary that holds a moderate proportion of the stake in a blockchain that adheres to the longest chain protocol of Nakamoto's Consensus. The adversary creates a secondary chain, hidden from honest parties. Each time the adversary is chosen to extend the valid chain, they refuse and slow down the process, whereas they continue extending their private chain.

Over time, the adversary will lose stake in the honest chain, but will gain rewards in the fake chain and slowly with the inclusion of valid transactions will exceed the valid chain. At this point the adversary can release their private chain and all honest nodes will accept it as the valid chain due to it's superior length.

Proposed mitigations [30, 62, 92, 118, 127] to this attack consist of frequent checkpointing, chain and block density, and context-sensitive transactions which may include the hash of the most recent blocks in the transaction. Similarly, PPCoin's coin-age [119] also helps to prevent stake bleeding by requiring the investment of time.

E. Long Range

The Long Range Attack [42] was theorized during the early development of Ethereum by Vitalik Buterin. The attack details a sequence of events where an adversary is able to double spend. The adversary sends a transaction to the merchant, the merchant then accepts the transaction after a certain number of block confirmations. The adversary then creates a fork, but starts it from a much earlier block, for example just after the genesis block.

In Proof-of-Work, the adversary would be required to hold a majority of the network's computing power, constituting this attack as a majority mining attack. However, the methodology, where the attacker can utilize a smart contract to perform hashes that are very easy to mine, producing a known value for itself, but would be extremely costly to the other miners. For example, it can be a loop for a specific value that the adversary knows, but others will have to re-compute the value each time. In this way, the attacker is able to slow the progress of the honest chain. The adversary then continues to mine until they create the longest chain and can overtake the main chain. This attack requires a high amount of computing power, such that the adversary is able to create the longest chain. In Proof-of-Stake, however, the attack requirements are lower due to the validation process. An adversary with as low as 1% of all coins can perform the long range attack with more blocks trivially.

Proposed mitigations [42, 79, 119] exist for the long range attack. One example mitigation is to ensure that timestamps on the blocks are used as a rejection mechanism. Another alternative is to ensure that a large portion of the coins, for example 30%, are required to sign every N^{th} block.

VIII. Discussion

In this section, we discuss (a) how membership selection, consensus mechanism, and structure are combined in existing systems and their impact on each other; (b) other interesting areas of blockchains not explored in this survey; and (c) upcoming and future proposals integral to the improvement of the blockchain, but not mature enough for critical evaluations and inclusion in the survey.

A. Membership Selection and Consensus

This section provides an overview on how existing systems make use of different membership selection and consensus mechanisms. An overview is presented in Table VII.

Role of Membership in Consensus Mechanisms: Although consensus algorithms are powerful, there are some limitations that impede the security and scalability to high node numbers.

For security, consensus algorithms rely on the membership selection to provide a set of trustworthy nodes to run consensus, such that an attacker cannot become a consensus group member easily. For example, Bitcoin uses proof-of-work to select consensus group members, where an attacker with a temporally high mining power can become a consensus member easily. However, with proof-of-reputation in RepuCoin, it is much more difficult for such an attacker to become consensus member, thus it helps enforcing the security assumptions of consensus algorithms.

For scalability, running optimally, without saturating network bandwidth with messages, requires a smaller committee of members to run the consensus. Membership selection fills the gap by providing the consensus with a subset of nodes to form the small committee of consensus members. The consensus also provides the membership selection with the requirements and assumptions that it must fulfill.

Nakamoto's Consensus, for example, requires optimally one proposer of blocks but places an assumption that the blocks are propagated to the entirety of the network prior to any new proposals. This assumption is translated into the difficulty of the Proof-of-Work puzzle and limits the speed of block proposals and the number of nodes in the membership selection. This is also evident in the GHOST consensus mechanism, as it requires an amount of time between proposals so the heaviest sub-tree is observed by the nodes in the network.

Similarly, in AlgoRand as an example, the BA$\star$ consensus runs optimally with a smaller subset of nodes, rather than all nodes performing the consensus. The AlgoRand membership selection utilizes VRFs to provide a subset of nodes as the committee to run the BA$\star$ consensus.

Table VII
MEMBERSHIP SELECTION AND CONSENSUS

Chain	**Released**	**Membership**	*Category*	**Consensus Mechanism**	*Type*
Bitcoin	2008	Proof-of-Work	Work	Nakamoto's	Nakamoto
PeerCoin	2012	Proof-of-Coin-Age	Stake	Nakamoto's (Highest Age)	Nakamoto
Ethereum*	2014/20XX	Proof-of-Work/Proof-of-Deposit	Work/Stake	Nakamoto's /(Casper)	Nakamoto/BFT
Tendermint	2014	Proof-of-Lock	Stake	Tendermint BFT	BFT
PeerCensus	2014	Proof-of-Work	Work	PeerCensus (PBFT Variant)	BFT
Permacoin	2014	Proof-of-Retrievability	Capacity	Nakamoto's Consensus	Nakamoto
BurstCoin	2014	Proof-of-Space	Capacity	Nakamoto's	Nakamoto
SpaceMint	2015	Proof-of-Space	Capacity	Nakamoto's	Nakamoto
ByzCoin	2016	Proof-of-Work	Work	ByzCoin (PBFT Variant)	BFT
HoneyBadger BFT	2016	-	-	HoneyBadger BFT	BFT
Solida	2017	Proof-of-Work	Work	Solida (PBFT Variant)	BFT
Ouroboros	2017	Ouroboros PoS	Stake	Ouroboros' Consensus	BFT
AlgoRand	2017	AlgoRand	Stake	AlgoRand BA$\star$	BFT
HyperLedger Sawtooth	2017	PoET	TEE	Nakamoto's†	Nakamoto
RepuCoin	2018	Proof-of-Work	Work/Reputation	RepuCoin BFT	BFT
RedBelly Blockchain	2018	-	-	DBFT	BFT

* The Ethereum blockchain is moving to Proof-of-Stake, currently using Proof-of-Work with the longest chain, but moving to Casper Proof-of-Stake with a BFT variant (named Casper) for the consensus. This consensus has not been included in the survey as it is still being developed and has a lack of formal documentation at this time.

† The HyperLedger Sawtooth documentation [104] states that the PoET runs a measure of total time waiting. However, if a PBFT consensus is used, it will never fork, making the fork resolution and consensus in the system flexible.

Improvements on Bitcoin Membership Selection: To save the wasted energy in PoW, many membership selection mechanisms have been proposed. In particular, the proposed mechanisms either use a stake-based lottery protocol (e.g. PoCA, PoD) for membership selection, or repurpose the required work by replacing the to-be-wasted computational task with something useful (e.g. distributed archive in Proof-of-Space). Since they are only considering the issue of wasted energy, they keep using the Nakamoto's consensus mechanism.

With proposals of the former type, stake-based lottery, new members are selected at random having a probability proportional to the stake they have. The focus on these proposals is designing the "stake" they rely on. For example, PeerCoin makes use of coin age as the stake; and Proof-of-Activity uses the balance of a participant and its online activity as the stake.

With proposals of the latter, new members are selected also at random, but the probability is depending on the alternative useful work they perform. The focus of these proposals has been on repurposing the wasted energy. For example, Permacoin, proposes to use distributed archive of files as the work, and BurstCoin and SpaceMint propose to use distributed storage space as a means to selected members.

Improvements on Bitcoin Consensus: As mentioned previously, Nakamoto's consensus only provides probabilistic guarantees. This results in potential forks of the chain, and can be exploited by an attacker to launch double spending attacks. Prior works try to remedy this issue by replacing the Nakamoto's consensus with the classic consensus mechanisms, i.e., Byzantine fault tolerant protocols that provide deterministic guarantee.

To the best of our knowledge, PeerCensus [64] was the first blockchain to propose Proof-of-Work for membership selection, and a BFT-style protocol for reaching consensus. In particular, the miners who successfully solved the recent Proof-of-Work puzzles form the consensus committee to run the BFT consensus on the proposed blocks. Similar to PeerCensus, a number of other chains (e.g. ByzCoin, Solida, and RepuCoin) utilize Proof-of-Work to select consensus committee members, and BFT-style consensus for reaching consensus. The focus of ByzCoin is on scaling the PBFT consensus schemes. Solida is focusing on providing a solution for consensus committee reconfiguration. Hybrid consensus [162] aims at providing an efficient bootstrapping with fast response. RepuCoin uses reputation-based weighted BFT protocol for reaching consensus. It aims at exploring ways to prevent bribery attack and flash attacks, where an attacker bribes existing power miners or rents cloud mining power to obtain a majority of the mining power quickly for a short time period.

Hybrid systems: Other systems have been working on combining a number of different techniques. The Proof-of-Activity [28] combines both Proof-of-Work and Proof-of-Stake to perform the membership selection. The Proof-of-Work aspect produces the initial proposer of the empty block, then used to derive the validators that will sign off the block. This is a combination of two membership selection techniques to harmoniously utilize the guarantees of both.

Similarly, Thunderella [164] proposes the use of the BFT consensus during the optimal periods. In non-optimal conditions, or during the cooldown phases, they switch from using the BFT to Nakamoto consensus. This provides a hybrid nature between the BFT based model and the Proof-of-Work longest chain model.

B. Consensus Mechanism and Structure

This section discusses the impact of the structure on the consensus mechanisms of choice.

Role of Structure in Consensus: The structure defines the way transactions and events are recorded in blockchain systems. This definition places heavy requirements on the consensus algorithm and what blocks are proposed and decided upon. The traditional blockchain, as depicted in the original Bitcoin paper [150], describes one canonical chain with homogeneous block types. This gives the requirement to the consensus that a single block should be decided for each height.

Bitcoin-NG [84] introduces different block types, decoupling leader election (keyblocks) from transaction serialization (microblocks). In particular, miners create keyblocks, which do not contain any transaction, through PoW. The successful miner in PoW is selected as a leader for a period of time until a new keyblock is created. A leader includes transactions into microblocks, and is allowed to validate microblocks directly without requiring further proof-of-work. This greatly increases the scalability of the consensus algorithms, in terms of the system throughput.

The further evolution of Blockchains introduced the use of other structures. One such evolution is utilizing a Directed Acyclic Graph (DAG) as the structure of the blockchain, where multiple events are able to occur simultaneously but require new consensus approaches to provide finality on multiple events. DAG structures apply different techniques to achieve consensus on the events that have occurred in the graph.

Additionally, the integration of sharding techniques has exhibited similar aspects such as parallel transaction processing and the benefit of less storage space required. However, it also brings new challenges to achieving consensus, such as transaction and state finality requiring the context of the shards.

Improving Bitcoin Structure: The original bitcoin structure, the one canonical chain of homogeneous blocks, provides the requirement for a distributed ledger of transactions for a decentralized payment system. However, the evolution of blockchains involve a number of functionality updates and improvements, which propose new structure changes to provide the new functionality or improvements.

By decoupling the transactions from the block creation, this allows for a higher throughput of transactions to be processed and appended to the ledger, whilst still maintaining the guarantees of the original chain structure. DAG structures on the other hand require a significantly different consensus and membership selection model.

C. Membership Selection and Structure

The nature of the structure also imposes requirements on the membership selection. The structure denotes how blocks are to be accepted and added, imposing requirements on how the consensus is to be performed. This impacts the way members are selected and the type of members that are selected.

Role of Structure in Membership Selection: The structure may require different types of nodes to be selected in a variety of ways. The original Bitcoin structure requires one set of membership to be chosen for each block. The Microblock structure proposed by ByzCoin and RepuCoin require groups of nodes to be selected in intervals, differing from the original where optimally one node would propose and decide on the next block and be agreed upon by everyone. The introduction of DAG structures imposes a major change to the membership selection, where different nodes control different paths on the total graph, or, are in charge of managing their own ledger of information.

The introduction of sharding will also impose a number of changes to the way membership selection is performed. As each shard will operate in different conditions, different memberships can be employed for each shard and operate uniquely.

D. Scalability

Scalability of blockchains can be encompassed by three main aspects; the number of consensus nodes, the number of clients, and the number of transactions processed per second (TPS).

Typically, a client is a node that is not participating in the consensus and is not actively producing blocks, however is actively transacting and receiving and gossiping information around the network. The number of clients have minimal impact on the overall performance of the blockchain, however, may impact the overall time for message bandwidth and propagation in the network. All types of blockchain consensus scale well [212].

With respect to the number of consensus nodes, non-BFT blockchain consensus offers impeccable scalability due to the simplicity of its protocol. BFT protocols, however, have limited scalability due to message complexity and the types of communication patterns in the protocols. For example, Nakamoto's consensus only requires nodes to choose the longest chain locally to reach an agreement, so consensus is able to be reached even in the presence of thousands of nodes. PBFT, on the other hand, requires several rounds of communication with all-to-all message passing involving all consensus members. So in practice, PBFT can only scale to a few dozen nodes.

For the number of processed transactions per second, Nakamoto's consensus, often seen paired with Proof-of-Work, performs poorly due to the requirement of proposals needing to propagate through the network. In particular, with Bitcoin, each block is 1Mb in size, and the size of each transaction is on average approximately 256 bytes. This limits the block to contain only approximately $4,000$ transactions, which is proposed every 10 minutes on average leading to an average of roughly 7 TPS. Increasing the size of a block is not an effective solution to improve the performance. For example, Bitcoin Cash, a hard fork of Bitcoin, increases the maximum block size from 1 MB to 8MB. Later, in 2018, Bitcoin Cash[15] has been split into two — one is called Bitcoin Adjustable Blocksize Cap (Bitcoin ABC)[16] with maximum block size of 32 MB, and the other is called Bitcoin Satoshi's Vision (Bitcoin

[15]https://www.bitcoincash.org
[16]https://www.bitcoinabc.org

SV)[17] which supports block size up to 128 MB. However, while providing a better number of TPS, the performance is still limited. Additionally, since the PoW-based membership selection requires competition in physical computing power, this may put an extra constraint on the actual number of consensus nodes in practice. For example, currently (as of Dec 2018), 14 mining pools collectively have 80.5% mining power of the entire network, which shows the limit of scalability w.r.t. the number of consensus nodes in practical [18].

On the other hand, BFT-based blockchain consensus provides good throughput [212] leading to potentially tens of thousands of transactions per second. However, as BFT protocols normally require a pre-defined consensus group, they cannot be applied directly into blockchains. This promotes systems, such as ByzCoin and RepuCoin, to combine different membership selection algorithms with BFT consensus algorithms, to adapt BFT protocols in Blockchain. The new combinations also impose changes on the blockchain structure. For example, with the combination of PoW+BFT, systems (such as ByzCoin and RepuCoin) deployed paralleled chains.

Off-chain payments: Another proposal to improve blockchain scalability is the use of payment, or state, channels. These channels allow for transactions to be performed off-chain with the same security as on-chain payments, which results in overall higher throughput. Techniques such as the Lightning Network [167] and Raiden [16] utilize off-chain messages to perform transactions amongst parties. The messages and transactions are signed off-chain by both parties and the on-chain transactions are used for final settlement.

E. Ancillary Components

Although this paper provides insight into three blockchain components, namely the membership selection, consensus and structure, there are a number of other components that form the blockchain and heavily contribute and influence the design and operation.

1) Cryptography

The blockchain heavily relies on cryptography for many aspects of it's operation. The cryptographic properties integrate into the security of the blockchain, allowing it to provide critical guarantees. The use of cryptography falls into a number of aspects of the blockchain, each providing unique properties and security.

Hash Functions: Hash functions provide the basis for the mining performed on blockchains as well as integrating with the coins and transactions. A cryptographically secure hash function is a deterministic one-way function, that is pre-image resistance and collision resistant. The pre-image resistance property ensures that for any given hash value, it should be difficult to find the original input message. The collision resistance property guarantees that it is difficult to find two different inputs, such that they share the same hash value.

During the mining process, the pre-image resistance property of hash functions makes it difficult to find a number that fits the threshold provided by the difficulty. The collision resistance property makes it difficult[19] to find a collision where the hash of an entity leads to the same hash of another. This is critical during the mining process, but also in the way transactions and blocks are formed, as the hash should be unique to identify each transaction and block uniquely. Furthermore, the collision resistant deterministic one-way function also ties into the user accounts, as each user account is addressed by a hash, which if found to be non-unique may cause issues.

Authentication and Authorization: The use of public key infrastructure forms the basis of wallets and user accounts. Each account is given an associated private key and public key which an address is then derived from. This ties a user to valid signatures when they sign transactions, and is a large component of the blockchain. Each transaction must be signed with a valid signature from a user, and can be verified using the wallets associated public key. Recently, a number of hardware-based wallets have been released, such as the Trezor [12] and the Ledger [7], providing hardware private keys to increase the security for the users.

Transaction privacy: A major component of the blockchain is transaction privacy. Zcash [190] and Monero [147] are two blockchains that are heavily working on the privacy aspect of the blockchain.

Ring Signatures [185] allow a user in a group to create a signature, in the way that a verifier can verify that the signature is created by a member of the group, but does not know exactly which member has created this signature. This allows the user to stay anonymous when submitting transactions as part of the group. This provides privacy for groups of users, however, it can lead to double-spending of coins as a user can create two transactions spending the same coin without being detected. Linkable Ring Signature [131] addresses this problem by revealing the identity of the signer if the signing key is used more than once. A variant of Linkable Ring Signature [88] is adopted by CryptoNote [99, 204]. In CryptoNote, an input of a transaction is a Traceable Ring Signature, where the ring contains one coin to be spent, and several decoy inputs known as "mix-ins". An observer only knows that one of the coins in the ring is spent, but does not know which coin. Monero further improved the work on privacy with ring signatures, extending CryptoNote's implementation by designing "Ring Confidential Transactions" (RingCT) [154]. RingCT provides anonymity of the transaction protecting both the user and the transaction amount, allowing for a higher level of privacy amongst transactions.

However, attacks on the privacy of CryptoNote-style blockchains have been identified. Two concurrent papers [122, 149] found that there exists "zero-mix-in" transitions, where

[17]https://bitcoinsv.io

[18]https://www.blockchain.com/pools

[19]With the current computing technology and known mathematical principles.

a ring only contains the real coin to be spent. This not only provides no privacy guarantee on the transaction with zero-mix-in, but also reduces the privacy of other transactions that use the input of the zero-mix-in transitions. They also found that other attributes of a coin, such as its age, leak information on the likelihood of this coin being spend in a transaction. Other work [215, 220, 222] show that even assuming all transactions contain mix-ins, and all attributes do not leak any information, the different ways to choose mix-ins or having a hard fork would also leak information about the trace of transactions. There are also network level analysis [50] on the topology of such blockchains, as knowing the network topology would help compromising the security and privacy of a system.

ZCash, as well as a number of other blockchains [173], are working on applying *Zero Knowledge Proof of Knowledge* (ZKPK) to the blockchain. ZKPK allows parties to prove their knowledge without revealing any information about the knowledge. For example, it allows users to prove possession of information without revealing the actual information, retaining privacy. The use of Zero Knowledge Proof of Knowledge on the blockchain is becoming increasingly popular, allowing for the verification of transactions and the verifying of arbitrary computations [173]. The introduction of *zk-SNARKs* [224] in the Zcash [190] blockchain sparked widespread interest into the adoption of Zero Knowledge Proofs on the blockchain. This will lead to an advancement of the cryptographic techniques used as well as provide more insight into better privacy and efficiency. Similar to the effort on security analysis of CryptoNote-style blockchain, reducing the privacy guarantee of Zcash is shown to be possible [115].

2) Internal Structures and Storage

Another component critical to the operation of the blockchain is the use of internal structures. These structures are integral to the operation of the blockchain and can help define what functionality the blockchain can provide. The structures handle the storage, retrieval and handling of data that is saved to the chain, or, used as state.

In Bitcoin variants, the most common structure is the in-memory map of storing Unspent Transaction Outputs (UTXOs) [32], which tracks any unspent transaction outputs, representing coins.

Ethereum uses prefix trees, or "Tries" [76], to handle state, storage, transactions and receipts. Ethereum updates state through blocks, the use of the state trie helps to append new state to the chain, updating new balances, changing the state and storage of smart contracts.

Other blockchains use a number of internal structures to handle state, balances, account and transaction information, and more. The choice of structures can impact on the chain capabilities, differing with each implementation but still a core component to consider.

3) Virtual Machines and Platforms

The Ethereum blockchain utilizes the Ethereum Virtual Machine (EVM) [109, 216] to run smart contracts and handle state. Recently there have been efforts to move away from the EVM and move into eWASM [171], a WebAssembly [15] based virtual machine that will then handle the state. The shift from the specialized Ethereum virtual machine to eWASM will provide new support for different languages and provide a more optimal execution.

F. Whitepaper Information

The rising interest of blockchains and its applications promote an influx of proposals. These proposals encompass a number of components, including membership selection, consensus, structure, cryptography or others. Due to the nature of the surrounding community, many of the proposals are detailed informally by means of whitepapers [91, 124, 128, 150, 155, 168, 196, 201, 204, 205], which in some cases lack peer review and are often shallow in terms of technical details.

Many whitepapers are written as public-facing documents to support the future developments, or provide content for potential investors to understand. This often leads to the lack of formal models and technical details that form critical components of the new proposals. Other whitepapers are presented in a formal manner, clearly addressing the technical details and formalizing models consistent with the literature. The proposals conforming to this format are often able to be critically evaluated and discussed thoroughly, however they make up a minority of the proposed whitepapers. Although this survey aims to provide an aggregation of proposals, it is not exhaustive.

G. Deployment Issues

Currently, all mainstream blockchains are PoW-based. The transition from Proof-of-Work to Proof-of-Stake can be seen as a deployment challenge for employing the energy efficient solutions. In particular, to constitute fairness for existing miners, the transition period must reward the miners in such a way that they are incentivized to drop mining. Most Proof-of-Stake mechanisms planned today, for example Casper for Ethereum [79, 225], increase the block difficulty to slowly phase out mining by making it impractical. Other new systems, such as the Proof-of-Activity [28], integrated the Proof-of-Stake with the existing PoW-based system.

H. Future Outlook

One major upcoming release is the adoption of sharding of the blockchain. The issue of scalability has been a factor on improving the blockchain. Pruning of state [45], light clients [43, 45, 81, 157, 229], block sizes [51] and others. However, the issue of the size of the blockchain remains unanswered, as the blockchain is growing over time, issues of storage space and availability to smaller nodes arise. The proposal of having a sharded blockchain [80] allows for a reduction in the size of the chain by splitting it amongst participants. However, the challenge of applying this while maintaining the blockchain guarantees is currently in work.

Similar to payment channels, there have been proposals [23, 166] to have side-chains operating alongside the

main blockchain, enabling the use of "tokens" or other added functionality.

Recently, a number of cross-chain techniques have been devised [124, 158, 203] to address the issue of governance and allow for transactions across multiple blockchains, providing an opportunity to diversify the blockchain model and open possibilities to offer different guarantees for users.

As the blockchain matures, new mechanisms will be proposed to improve the blockchain. The future developments will impact performance, scalability or security of the blockchain in a variety of ways.

IX. Conclusion

Although in its infancy, the blockchain paradigm is generating widespread interest. As with all successful concepts, the current blockchain ecosystem quickly became quite complex and difficult to understand, from both an introductory view and an in-depth dive into the state of the art.

To address this problem, we deconstructed the blockchain into three main critical components: membership selection, consensus mechanism and structure. We introduced an evaluation framework to get insight into system models, desired properties and analysis criteria, using the decoupled components as parameters. We tested our approach by analyzing the relevant state of the art, and performing a categorization of how systems fit in the different criteria outlined.

The number of proposals for blockchains increases by the month. However, we trust our framework to be comprehensive enough that proposals can be added and classified in the future. Improvements to the framework may include the addition of quantitative examination of proposals.

This survey provided clear insight into the blockchain proposals landscape, through a novel perspective — deconstructing the blockchain complex into few simple but complete critical components. We expect this work to be useful to the community, providing direction and helping to simplify future designs, such as inspiring innovative coherent combinations revealed by our decomposition and categorization.

Acknowledgment

This research is in part supported under Australian Research Council Discovery Projects funding scheme (project number 180104030) entitled "Taipan: A Blockchain with Democratic Consensus and Validated Contracts" and Australian Research Council Future Fellowship funding scheme (project number 180100496) entitled "The Red Belly Blockchain: A Scalable Blockchain for Internet of Things".

This work is in part supported by the Fonds National de la Recherche Luxembourg through PEARL grant FNR/P14/8149128 and by Intel through the ICRI Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS).

References

[1] "AntMiner R4." [Online]. Available: https://asicminermarket.com/product/antminer-r4-mute-8ths/

[2] "Bitshares." [Online]. Available: https://bitshares.org

[3] "Cardano." [Online]. Available: https://www.cardano.org/en/home/

[4] "Dash." [Online]. Available: https://www.dash.org/

[5] "Dogecoin." [Online]. Available: https://dogecoin.com/

[6] "EOS." [Online]. Available: https://eos.io

[7] "Ledger Wallet." [Online]. Available: https://www.ledger.com

[8] "Lisk." [Online]. Available: https://lisk.io/

[9] "Neo." [Online]. Available: https://neo.org/

[10] "Ripple." [Online]. Available: https://ripple.com/

[11] "Steem." [Online]. Available: https://steem.io/

[12] "Trezor hardware wallet." [Online]. Available: https://trezor.io

[13] "Verge currency." [Online]. Available: https://vergecurrency.com/

[14] "Waves platform." [Online]. Available: https://wavesplatform.com/

[15] "Web Assembly." [Online]. Available: https://webassembly.org

[16] "Raiden Network Overview," 2015. [Online]. Available: https://raiden.network/faq.html

[17] M. Abadi, M. Burrows, and T. Wobber, "Moderately hard, memory-bound functions," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*, 2003.

[18] I. Abraham and D. Malkhi, "The blockchain consensus layer and BFT," *Bulletin of the EATCS*, vol. 123, 2017. [Online]. Available: http://eatcs.org/beatcs/index.php/beatcs/article/view/506

[19] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solida: A blockchain protocol based on reconfigurable byzantine consensus," in *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, 2017, pp. 25:1–25:19.

[20] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain," in *Proceedings of the Second Italian Conference on Cyber Security, Milan, Italy, February 6th - to - 9th, 2018.*, 2018. [Online]. Available: http://ceur-ws.org/Vol-2058/paper-06.pdf

[21] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking Bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 375–392.

[22] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," in *International Conference on Trust and Trustworthy Computing*. Springer, 2015, pp. 163–180.

[23] 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," *http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.

[24] 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, 2017.

[25] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance," 2016. [Online]. Available: https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[26] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," *CoRR*, vol. abs/1711.03936, 2017. [Online]. Available: http://arxiv.org/abs/1711.03936

[27] M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman, and A. Sonnino, "State machine replication in the libra blockchain," 2019. [Online]. Available: https://developers.libra.org/docs/state-machine-replication-paper

[28] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake [Extended Abstract]," *SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014. [Online]. Available: https://doi.org/10.1145/2695533.2695545

[29] I. Bentov, R. Pass, and E. Shi, "The sleepy model of consensus," *IACR Cryptology ePrint Archive*, vol. 2016, p. 918, 2016. [Online]. Available: http://eprint.iacr.org/2016/918

[30] ——, "Snow white: Provably secure proofs of stake," *IACR Cryptology ePrint Archive*, vol. 2016, p. 919, 2016. [Online]. Available: http://eprint.iacr.org/2016/919

[31] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem," in *23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[32] Bitcoin, "Bitcoin Core 0.11 Data Storage," 2016, updated January 2016. [Online]. Available: https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_2):_Data_Storage#The_UTXO_set_.28chainstate_leveldb.29

[33] "Bitcoin Wiki: Difficulty," 2018, https://en.bitcoin.it/wiki/Difficulty.

[34] Bitshares, 2013-2017. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/

[35] ——, "Delegated proof of stake," 2013-2017. [Online]. Available: http://docs.bitshares.org/bitshares/dpos.html

[36] A. Black, "Hashcash - A denial of service counter-measure," Cypherspace, Tech. Rep., 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf

[37] blockchain.com, "Hashrate distribution," 2019. [Online]. Available: https://www.blockchain.com/pools

[38] J. Bonneau, "Why Buy When You Can Rent? - Bribery Attacks on Bitcoin-Style Consensus," in *2016 International Workshops on Financial Cryptography and Data Security (FC), BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, 2016, pp. 19–26.

[39] G. Brambilla, M. Amoretti, and F. Zanichelli, "Using block chain for peer-to-peer proof-of-location," *CoRR*, vol. abs/1607.00174, 2016. [Online]. Available: http://arxiv.org/abs/1607.00174

[40] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," Master's thesis, The University of Guelph, 2016.

[41] V. Buterin, "Validator Ordering and Randomness in PoS." [Online]. Available: https://vitalik.ca/files/randomness.html

[42] ——, "Long Range Attacks: The Serious Problem with Adaptive Proof-of-Work," 2014. [Online]. Available: https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/

[43] ——, "Light Clients and Proof of Stake," 2015. [Online]. Available: https://blog.ethereum.org/2015/01/10/light-clients-proof-stake/

[44] ——, "The problem of censorship," 2015. [Online]. Available: https://blog.ethereum.org/2015/06/06/the-problem-of-censorship/

[45] ——, "State Tree Pruning," 2015. [Online]. Available: https://blog.ethereum.org/2015/06/26/state-tree-pruning/

[46] ——, "Casper FFG with one message type, and simpler fork choice rule," 2017. [Online]. Available: https://ethresear.ch/t/casper-ffg-with-one-message-type-and-simpler-fork-choice-rule/103

[47] ——, "On public and private blockchains," 2015. [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

[48] bytemaster and bitsharestalk.org, "Transactions as proof-of-stake & the end of mining," 2013. [Online]. Available: https://bitsharestalk.org/index.php?topic=1138.msg13602#msg13602A

[49] C. Cachin and M. Vukolić, "Blockchain Consensus Protocols in the Wild," *CoRR arXiv*, vol. abs/1707.01873, 2017. [Online]. Available: http://arxiv.org/abs/1707.01873

[50] T. Cao, J. Yu, J. Decouchant, X. Luo, and P. Veríssimo, "Exploring the Monero Peer-to-Peer Network," *IACR Cryptology ePrint Archive*, vol. 2019, p. 411, 2019. [Online]. Available: https://eprint.iacr.org/2019/411

[51] B. Cash, "Bitcoin Cash," 2017. [Online]. Available: https://www.bitcoincash.org/#faq-section

[52] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

[53] B. Chase and E. MacBrough, "Analysis of the XRP ledger consensus protocol," 2018. [Online]. Available: http://arxiv.org/abs/1802.07242

[54] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On Security Analysis of Proof-of-Elapsed-Time (PoET)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 282–297.

[55] N. Christin, "Traveling the silk road: A measurement

analysis of a large anonymous online marketplace," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13. New York, NY, USA: ACM, 2013, pp. 213–224. [Online]. Available: http://doi.acm.org/10.1145/2488388.2488408

[56] A. City, "Arcade city," 2018. [Online]. Available: https://arcade.city/

[57] CoinTelegraph, "Coinbase: Ethereum Classic Double Spending Involved More Than $1.1 Million in Crypto," 2019. [Online]. Available: https://cointelegraph.com/news/coinbase-ethereum-classic-double-spending-involved-more-than-11-million-in-crypto

[58] M. Conti, S. K. E, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018. [Online]. Available: https://doi.org/10.1109/COMST.2018.2842460

[59] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, "DBFT: efficient leaderless byzantine consensus and its application to blockchains," in *17th IEEE International Symposium on Network Computing and Applications, NCA 2018, Cambridge, MA, USA, November 1-3, 2018*, 2018, pp. 1–8.

[60] T. Crain, C. Natoli, and V. Gramoli, "Evaluating the red belly blockchain," *CoRR*, vol. abs/1812.11747, 2018. [Online]. Available: http://arxiv.org/abs/1812.11747

[61] W. Dai, "B-money," 1998. [Online]. Available: http://www.weidai.com/bmoney.txt

[62] P. Daian, R. Pass, and E. Shi, "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake," in *23rd International Conference on Financial Cryptography and Data Security FC'19 February 1822, 2019 St. Kitts Marriott Resort St. Kitts*, 2019.

[63] 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*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 66–98.

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

[65] C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," in *2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2013, pp. 1–10.

[66] N. Dhinsa and T. Marquez, "Chasyr," 2017. [Online]. Available: https://www.chasyr.com/

[67] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for iot," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*. ACM, 2017, pp. 173–178.

[68] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *2017 IEEE International Conference on Pervasive Computing and Communications workshops (PerCom workshops)*. IEEE, 2017, pp. 618–623.

[69] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[70] C. Dwork, A. V. Goldberg, and M. Naor, "On Memory-Bound Functions for Fighting Spam," in *23rd Annual International Cryptology Conference CRYPTO 2003, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, 2003, pp. 426–444.

[71] C. Dwork, N. A. Lynch, and L. J. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, vol. 35, no. 2, pp. 288–323, 1988. [Online]. Available: http://doi.acm.org/10.1145/42282.42283

[72] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*. Springer, 1992, pp. 139–147.

[73] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *35th Annual Cryptology Conference (CRYPTO) 2015, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, 2015, pp. 585–605.

[74] P. Ekparinya, V. Gramoli, and G. Jourjon, "Impact of man-in-the-middle attacks on ethereum," in *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2018, pp. 11–20.

[75] ——, "The attack of the clones against proof-of-authority," *CoRR*, vol. abs/1902.10244, 2019, Presented at EDCON Sydney 2019. [Online]. Available: http://arxiv.org/abs/1902.10244

[76] Ethereum, "Patricia Tree," 2014, updated June 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Patricia-Tree

[77] ——, "Solidity," 2014. [Online]. Available: https://solidity.readthedocs.io/en/latest/

[78] "Ethereum," https://www.ethereum.org/, 2017.

[79] Ethereum, "Proof-of-stake faq - ethereum," 2017. [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[80] Ethereum, V. Buterin, J. Ray, T. Ishizaki, and V. Griffith, "Sharding FAQs - On Sharding Blockchains," 2016-2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQs

[81] Ethereum, J. Ray, V. Buterin, J. McKinney, and Heiko, "Light Client Protocol," 2014-2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Light-client-protocol

[82] Ethereum Foundation. Ethereum, "Ethereum 2.0 specs," 2018. [Online]. Available: https://github.com/ethereum/eth2.0-specs

[83] Ethereum Wiki, "Problems," 2017. [Online]. Available: https://github.com/ethereum/wiki/wiki/Problems

[84] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: 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.

[85] I. Eyal and E. G. Sirer, "Majority Is Not Enough: Bitcoin Mining Is Vulnerable," in *18th International Conference on Financial Cryptography and Data Security, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, 2014, pp. 436–454.

[86] H. Finney, "Finney's attack," February 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384

[87] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[88] E. Fujisaki and K. Suzuki, "Traceable ring signature," in *Proceedings of the 10th International Conference on Practice and Theory in Public-key Cryptography*, ser. PKC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 181–200. [Online]. Available: http://dl.acm.org/citation.cfm?id=1760564.1760582

[89] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[90] J. A. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol with chains of variable difficulty," in *37th Annual International Cryptology Conference CRYPTO 2017, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, 2017, pp. 291–323. [Online]. Available: https://doi.org/10.1007/978-3-319-63688-7_10

[91] S. Gauld, F. von Acoina, and R. Stadler, "The burst dymaxion," 2017. [Online]. Available: https://www.burst-coin.org/wp-content/uploads/2017/07/The-Burst-Dymaxion-1.00.pdf

[92] P. Gaži, A. Kiayias, and A. Russell, "Stake-bleeding attacks on proof-of-stake blockchains," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 85–92.

[93] 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, Vienna, Austria, October 24-28, 2016*, 2016, pp. 3–16.

[94] 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*. ACM, 2017, pp. 51–68.

[95] J. Göbel, H. Keeler, A. Krzesinski, and P. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, July 2016.

[96] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache Attacks on Intel SGX," in *Proceedings of the 10th European Workshop on Systems Security*, ser. EuroSec'17. New York, NY, USA: ACM, 2017, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/3065913.3065915

[97] V. Gramoli, "From blockchain consensus back to Byzantine consensus," *Future Generation Computer Systems*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17320095

[98] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, no. 2, pp. 99–111, Jan 1991. [Online]. Available: https://doi.org/10.1007/BF00196791

[99] R. Han, J. Yu, J. K. Liu, and P. Zhang, "Evaluating cryptonote-style blockchains," in *14th International Conference on Information Security and Cryptology, Inscrypt 2018, Fuzhou, China, December 14-17, 2018, Revised Selected Papers*, 2018, pp. 29–48.

[100] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," *CoRR*, vol. abs/1805.04548, 2018. [Online]. Available: http://arxiv.org/abs/1805.04548

[101] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, 2015, pp. 129–144.

[102] E. Heilman, N. Narula, G. Tanzer, J. Lovejoy, M. Colavita, M. Virza, and T. Dryja, "Cryptanalysis of curl-p and other attacks on the IOTA cryptocurrency," *IACR Cryptology ePrint Archive*, vol. 2019, p. 344, 2019. [Online]. Available: https://eprint.iacr.org/2019/344

[103] S. Huh, S. Cho, and S. Kim, "Managing iot devices using blockchain platform," in *19th international conference on advanced communication technology (ICACT) 2017*. IEEE, 2017, pp. 464–467.

[104] Hyperledger, "Hyperledger sawtooth journal," 2015-2017. [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/journal.html

[105] "Hyperledger," https://www.hyperledger.org/, 2017.

[106] "IBM Blockchain: Hyperledger Fabric," https://www.ibm.com/blockchain/hyperledger.html, 2017.

[107] IBM, "Bank of Montreal, CaixaBank, Commerzbank, Erste Group, IBM and UBS Collaborate to Advance an Open, Blockchain-based Trade Finance Platform," 2017. [Online]. Available: https://www-03.ibm.com/press/us/en/pressrelease/53208.wss

[108] ——, "Abu Dhabi National Oil Company (ADNOC)," 2018. [Online]. Available: https://www.ibm.com/case-studies/abu-dhabi-national-oil-company-adnoc

[109] H. Ibrahim, J. Ray, liuxiopai, kakousis, A. Xiong, F. Vogelsteller, L. Zeug, K. Chia, H. C. Stockhausen, B. Joseff, jn pn, ghasshee, V. buterin, J. Kwon, and G. Wood, "Ethereum Development Tutorial," 2014-2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial

[110] Intel, "Intel software guard extensions," 2014. [Online]. Available: https://software.intel.com/en-us/sgx

[111] Intel and HyperLedger, "Hyperledger Sawtooth PoET Documentation," 2017. [Online].

Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html

[112] Intelledger, "Proof of elapsed time," 2016. [Online]. Available: https://intelledger.github.io/introduction.html

[113] Y. Jang, J. Lee, S. Lee, and T. Kim, "SGX-Bomb: Locking down the processor via Rowhammer attack," in *Proceedings of the 2nd Workshop on System Software for Trusted Execution*. ACM, 2017, p. 5.

[114] 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*. Acm, 2007, pp. 584–597.

[115] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in zcash," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, 2018, pp. 463–477.

[116] karalabe and Ethereum, "Clique poa protocol & rinkeby poa testnet," 2017. [Online]. Available: https://eips.ethereum.org/EIPS/eip-225

[117] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in Bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 906–917.

[118] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[119] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[120] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, 2016, pp. 279–296.

[121] N. Kshetri, "Can blockchain strengthen the internet of things?" *IT professional*, vol. 19, no. 4, pp. 68–72, 2017.

[122] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of monero's blockchain," in *22nd European Symposium on Research in Computer Security (ESORICS) 2017 , Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, 2017, pp. 153–173.

[123] J. Kwon, "Tendermint: Consensus without mining," http://tendermint.com/docs/tendermint_v04.pdf, 2014.

[124] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," 2017. [Online]. Available: https://cosmos.network/cosmos-whitepaper.pdf

[125] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[126] L. Lamport, R. E. Shostak, and M. C. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[127] D. Larimer, "Transactions as proof-of-stake," 2013. [Online]. Available: https://bravenewcoin.com/assets/Uploads/TransactionsAsProofOfStake10.pdf

[128] C. LeMahieu, "Raiblocks: A feeless distributed cryptcurrency network," 2017 (original 2014). [Online]. Available: https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf

[129] Lisk.io, "Delegated proof of stake," 2018. [Online]. Available: https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/delegated-proof-of-stake

[130] Litecoin, "Litecoin," 2011. [Online]. Available: https://litecoin.info/index.php/Litecoin

[131] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract)," in *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, 2004, pp. 325–335.

[132] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolić, "XFT: Practical Fault Tolerance beyond Crashes," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 485–500.

[133] A. Loibl, "Namecoin," *namecoin. info*, 2014.

[134] D. Malkhi and M. K. Reiter, "Byzantine Quorum Systems," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, 1997, pp. 569–578.

[135] B. Marr, "35 Amazing Real World Examples Of How Blockchain Is Changing Our World," 2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/01/22/35-amazing-real-world-examples-of-how-blockchain-is-changing-our-world/#351e61a543b5

[136] J.-P. Martin and L. Alvisi, "Fast Byzantine consensus," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 202–215, 2006.

[137] D. Mazieres, "Stellar." [Online]. Available: https://www.stellar.org/

[138] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1987, pp. 369–378.

[139] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on*. IEEE, 2016, pp. 1–3.

[140] S. Micali, "ALGORAND: the efficient and democratic ledger," *CoRR*, vol. abs/1607.01341, 2016. [Online]. Available: http://arxiv.org/abs/1607.01341

[141] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2013, pp. 397–411.

[142] 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, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 2014, pp. 475–490.

[143] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The Honey Badger of BFT Protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-*

28, 2016, 2016, pp. 31–42.

[144] R. Miller, "Walmart is betting on the blockchain to improve food safety," 2018. [Online]. Available: https://techcrunch.com/2018/09/24/walmart-is-betting-on-the-blockchain-to-improve-food-safety/

[145] 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, SysTEX@Middleware 2016, Trento, Italy, December 12, 2016*, 2016, pp. 21–26.

[146] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "CacheZoom: How SGX amplifies the power of cache attacks," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 69–90.

[147] Monero, "Monero," 2014. [Online]. Available: https://getmonero.org/resources/research-lab/

[148] A. Morcos, C. Fields, dexX7, fsb4000, G. Andresen, G. Maxwell, I. Pustogarov, J. Nick, J. Schnelli, M. Corallo, mrbandrews, P. Wuille, R. de Vries, S. Daftuar, and W. J. van der Laan, "Bitcoin Core version 0.10.1 released," 2015. [Online]. Available: https://bitcoin.org/en/release/v0.10.1

[149] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, and N. Christin, "An Empirical Analysis of Traceability in the Monero Blockchain," *Proceedings of Privacy Enhancing Technologies Symposium (PETs)*, vol. 2018, no. 3, pp. 143–163, 2018.

[150] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[151] C. Natoli and V. Gramoli, "The Blockchain Anomaly," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2016, pp. 310–317.

[152] ——, "The Balance Attack or Why Forkable Blockchains Are Ill-Suited for Consortium," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2017, pp. 579–590.

[153] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 305–320.

[154] S. Noether, "Ring signature confidential transactions for monero," Cryptology ePrint Archive, Report 2015/1098, 2015, https://eprint.iacr.org/2015/1098.

[155] NXT, "NXT Whitepaper," 2016. [Online]. Available: http://nxtwiki.org/wiki/Whitepaper:Nxt

[156] Parity, "Proof of authority chains," 2017. [Online]. Available: https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains

[157] Parity Technologies, "Light Ethereum SubProtocol (LES) - Wiki," 2015. [Online]. Available: https://wiki.parity.io/Light-Ethereum-Subprotocol-(LES)

[158] Parity Technologies and Web3 Foundation, "Polkadot lightpaper," 2017. [Online]. Available: https://polkadot.network/Polkadot-lightpaper.pdf

[159] S. Park, K. Pietrzak, A. Kwon, J. Alwen, G. Fuchsbauer, and P. Gazi, "Spacemint: A cryptocurrency based on proofs of space," *IACR Cryptology ePrint Archive*, vol. 2015, p. 528, 2015.

[160] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *36th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) 2017, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, 2017, pp. 643–673.

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

[162] ——, "Hybrid Consensus: Efficient Consensus in the Permissionless Model," in *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, 2017, pp. 39:1–39:16.

[163] ——, "The sleepy model of consensus," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 380–409.

[164] ——, "Thunderella: Blockchains with optimistic instant confirmation," in *37th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) 2018, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, 2018, pp. 3–33.

[165] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A blockchain-based approach to health information exchange networks," in *Proc. NIST Workshop Blockchain Healthcare*, vol. 1, 2016, pp. 1–10.

[166] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," 2017. [Online]. Available: https://plasma.io/plasma.pdf

[167] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2015. [Online]. Available: https://lightning.network

[168] S. Popov, "The Tangle," 2017. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf

[169] QuantumMechanic, "Bitcoin Forum - Proof of Stake instead of Proof of Work," 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=27787.0

[170] R3, "Corda," Available from https://www.corda.net/, 2016.

[171] J. Ray, "eWASM Compendium," 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/EWasm-compendium

[172] M. K. Reiter, "A Secure Group Membership Protocol," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 31–42, 1996.

[173] C. Reitwiessner, "zkSNARKs in a Nutshell," 2016. [Online]. Available: https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/

[174] AMD, "Amd secure technology," 2013. [Online]. Available: https://www.amd.com/en/technologies/security

[175] ARM, "Building a secure system using trustzone technology," 2009. [Online].

Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[176] dacoinminster, "It's here: The second bitcoin whitepaper," 2012. [Online]. Available: https://bitcointalk.org/index.php?topic=56901.0

[177] ——, "The Second Bitcoin Whitepaper," 2012. [Online]. Available: https://sites.google.com/site/2ndbtcwpaper/2ndBitcoinWhitepaper.pdf

[178] Ethereum, 2013. [Online]. Available: https://github.com/ethereum/go-ethereum

[179] Libra, "Libra white paper," 2019. [Online]. Available: https://libra.org/en-US/white-paper/

[180] Parity, "Aura," 2018. [Online]. Available: https://wiki.parity.io/Aura

[181] Parity Technologies, 2016. [Online]. Available: https://github.com/paritytech/parity-ethereum

[182] Platin, "Platin Whitepaper," 2019. [Online]. Available: https://platin.io/assets/whitepaper/Platin_Whitepaper_v3.03.pdf

[183] POA Network, "Poa-network-whitepaper," 2018. [Online]. Available: https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper

[184] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[185] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 552–565.

[186] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *CoRR*, vol. abs/1112.4980, 2011. [Online]. Available: http://arxiv.org/abs/1112.4980

[187] ——, "Analysis of hashrate-based double spending," *CoRR*, vol. abs/1402.2009, 2014. [Online]. Available: http://arxiv.org/abs/1402.2009

[188] N. Rush, D. Ryan, V. Zamfir, and Ethereum, "CBC Casper Resource List," 2018. [Online]. Available: https://github.com/ethereum/cbc-casper/wiki/Resource-List

[189] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *Financial Cryptography and Data Security - FC 2016*, 2016.

[190] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized Anonymous Payments from Bitcoin," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2014, pp. 459–474.

[191] J. Schlamp, R. Holz, Q. Jacquemart, G. Carle, and E. W. Biersack, "HEAP: Reliable assessment of BGP hijacking attacks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1849–1861, 2016.

[192] F. B. Schneider, "Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial," *ACM Comput. Surv.*, vol. 22, no. 4, pp. 299–319, Dec. 1990. [Online]. Available: http://doi.acm.org/10.1145/98163.98167

[193] D. Schwartz, N. Youngs, A. Britto *et al.*, "The ripple protocol consensus algorithm," 2014. [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf

[194] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware guard extension: Using SGX to conceal cache attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2017, pp. 3–24.

[195] sifmelcara, "A Livelock Bug in the Presence of Byzantine Validator," 2018. [Online]. Available: https://github.com/tendermint/tendermint/issues/1047

[196] Slimcoin and P4Titan, "Slimcoin whitepaper," 2014. [Online]. Available: www.slimcoin.club/whitepaper.pdf

[197] Y. Sompolinsky and A. Zohar, "Secure High-rate Transaction Processing in Bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[198] I. Stewart, "Proof of burn. Bitcoin. it," 2012. [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_burn

[199] ——, "Proof of Burn - a potential alternative to proof of work and proof of stake," 2012. [Online]. Available: https://bitcointalk.org/index.php?topic=131139.msg1404195

[200] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities "honest or bust" with decentralized witness cosigning," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 526–545.

[201] Team Rocket, "Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies," 2018. [Online]. Available: https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV

[202] TheHackerNews, "Ethereum Classic (ETC) Hit by Double-Spend Attack Worth $1.1 Million," 2019. [Online]. Available: https://thehackernews.com/2019/01/ethereum-double-spend-attack.html

[203] S. Thomas and E. Schwartz, "A Protocol for Interledger Payments," 2017. [Online]. Available: https://interledger.org/interledger.pdf

[204] N. van Saberhagen, "CryptoNote v1.0," 2012. [Online]. Available: https://cryptonote.org/whitepaper_v1.pdf

[205] P. Vasin, "BlackCoin's Proof-of-Stake Protocol v2," 2014. [Online]. Available: http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf

[206] vector76, "The vector76 attack," August 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=36788.msg463391#msg463391

[207] E. Vitalik Buterin, "Discouragement attacks," 2017. [Online]. Available: https://github.com/ethereum/research/blob/master/papers/other_casper/discouragement.tex

[208] Ethereum Foundation. Vitalik Buterin, "Convenience link to Casper+Sharding chain v2.1 spec," 2018. [Online]. Available: https://ethresear.ch/t/convenience-link-to-casper-sharding-chain-v2-1-spec/2332

[209] G. Vizier and V. Gramoli, "Comchain: Bridging the

gap between public and consortium blockchains," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018*, 2018, pp. 1469–1474.

[210] W. Vogels, "Eventually consistent," *Communications of the ACM*, vol. 52, no. 1, pp. 40–44, 2009.

[211] M. Vukolić, *Quorum systems with applications to storage and consensus*. San Rafael, Calif: Morgan & Claypool, 2012.

[212] ——, "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," in *Open Problems in Network Security - IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29, 2015, Revised Selected Papers*, 2015, pp. 112–125.

[213] W. Wang, D. T. Hoang, Z. Xiong, D. Niyato, P. Wang, P. Hu, and Y. Wen, "A survey on consensus mechanisms and mining management in blockchain networks," *CoRR*, vol. abs/1805.02707, 2018. [Online]. Available: http://arxiv.org/abs/1805.02707

[214] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, and C. A. Gunter, "Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2421–2434.

[215] D. A. Wijaya, J. K. Liu, R. Steinfeld, D. Liu, and J. Yu, "On The Unforkability of Monero," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, 2019, pp. 621–632.

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

[217] N. Woolf, "Silk road sentencing: Why governments can't win war on darknet drugs," 2015. [Online]. Available: https://www.theguardian.com/technology/2015/may/31/silk-road-sentencing-darknet-drugs

[218] H. Yan, R. Oliveira, K. Burnett, D. Matthews, L. Zhang, and D. Massey, "BGPmon: A real-time, scalable, extensible monitoring system," in *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*. IEEE, 2009, pp. 212–223.

[219] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus in the lens of blockchain," 2018. [Online]. Available: http://arxiv.org/abs/1803.05069

[220] J. Yu, M. H. A. Au, and P. J. E. Veríssimo, "Re-thinking untraceability in the CryptoNote-style blockchain," *IACR Cryptology ePrint Archive*, vol. 2019, p. 186, 2019. [Online]. Available: https://eprint.iacr.org/2019/186

[221] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power," *IEEE Transactions on Computers (ToC)*, 2019.

[222] Z. Yu, M. H. Au, J. Yu, R. Yang, Q. Xu, and W. F. Lau, "New empirical traceability analysis of CryptoNote-style blockchains," in *23rd International Conference on Financial Cryptography and Data Security (FC)*, 2019.

[223] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, p. 218, 2016.

[224] Z-Cash, "What are zk-SNARKs?" 2016. [Online]. Available: https://z.cash/technology/zksnarks.html

[225] V. Zamfir, "Introducing Casper "the friendly GHOST"," 2015. [Online]. Available: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/

[226] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao, "Practical defenses against BGP prefix hijacking," in *Proceedings of the 2007 ACM CoNEXT conference*. ACM, 2007, p. 3.

[227] L. Zhao and J. Yu, "Evaluation DAG-based Blockchains for IoT," in *Proceedings of the 18th IEEE International Conference on Trust, Security and Privacy in Computing Communications (TrustCom)*, 2019.

[228] Z. Zhu and G. Cao, "Toward privacy preserving and collusion resistance in a location proof updating system," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 51–64, 2011.

[229] F. Zsolt, "Light Ethereum SubProtocol (LES)," 2016-2017. [Online]. Available: https://github.com/zsfelfoldi/go-ethereum/wiki/Light-Ethereum-Subprotocol-(LES)