Amie Corso
Department of Computer and Information Science
University of Oregon
December, 2018

Literature Review
Blockchain Technology: Consensus and Performance

# I. Blockchain Basics

## A. Introduction

In January of 2009, the first version of Bitcoin [1] was deployed and the first coins created, marking the beginning of the first demonstrated success of a public, decentralized, anonymous, cryptographic currency implemented with blockchain technology. It took several more years for the success of Bitcoin to be proven and for the rich potential of other blockchain-based applications to spread in public awareness. Informed by Bitcoin's strengths and shortcomings, a major second-generation blockchain platform is Ethereum [2], which is capable of handling much more complex logic. While Ethereum supports a cryptocurrency ("Ether"), it has been put to many more diverse purposes given its additional capability. A good example of third-generation blockchain technology is the Hyperledger project [14, 16], an open-source blockchain framework managed by the Linux Foundation and completely dissociated from cryptocurrencies. While these are currently three giants in the industry, many smaller blockchain startups have created their own implementations, and others have created unique applications using existing frameworks. As of writing in 2018, blockchain technology is being heralded by many as comparable in revolutionary potential to the Internet. However, still in its infancy, the technology faces many barriers to adoption with regard to performance, security, ease-of-use, knowledgeable developers, and reputation. The following review introduces the key concepts and terminology in blockchain technology and covers the current state of research in blockchain consensus algorithms, which are the core of what makes blockchain technology uniquely useful.

## B. Key Properties

The term "blockchain" refers not to a single entity (such as Bitcoin), piece of software, data structure, or algorithm, but rather to a growing collection of various technologies that share certain unique properties. Blockchain technology is a subset of "distributed ledger" technology (DLT), which involves the synchronization of storage, access, and maintenance of digital data across a multi-party network of distinct participants. In other words, a blockchain is a form of distributed ledger -- a decentralized database capable of storing arbitrary information. A decentralized ledger is so-called because

data is stored and accessed not in a central repository, but is replicated across multiple “nodes” or “peers” in order to achieve properties such as robustness to error, efficient querying, or parallelized updates. The data stored in a blockchain can also be thought of as supporting an abstract finite state machine, a system that exists in exactly one of a finite number of possible states at any given time. The possible states and the valid transitions among them are customizable to the specific application of the blockchain. Thus, blockchains are not restricted to representing simple transactional information, such as sending and receiving of cryptocurrency, but can be capable of executing and storing arbitrarily complex logic and data.

Distributed ledger technology already faces a handful of difficult problems inherent to distributed systems, in which the network may experience latency or separation, and nodes may fail or commit errors. In particular, a distributed ledger must achieve consistency across replicas and support efficient updates despite the presence of concurrency challenges. The network must also be robust to failures of hardware or peers, and remain live despite a changing network topology. As a subset of DLT, blockchain technology faces the same problems, and tackles several more in addition. In particular, blockchain technology assumes a more hostile security model in which peers on the network may not trust each other and may exhibit intentionally malicious, selfish, or destructive behavior in addition to benign crashes or random error. To facilitate trustless transactions, the key properties of a blockchain data structure are *immutability*, *transparency*, and *tamper-evidence*. Immutability can also be understood as append-only updates; once data has been committed to the blockchain, it cannot be changed without destroying and re-creating the blockchain from that point forward (an extreme version of an append). Transparency means that all nodes on the network have access to the same data, which they use to check the validity of updates and collectively police the network. Tamper-evidence is a property that arises from the blockchain data structure itself which is a change-sensitive, hash-based structure that allows efficient detection of even the tiniest modification.

### C. Terms and Concepts

The following describes the important terminology and major components of a blockchain data structure and the network that maintains it. The data contained in a blockchain is a sequence of “transactions.” A transaction is the method of changing the state of the abstract state machine represented by the blockchain. As a concrete example, consider the cryptocurrency Bitcoin, in which users have accounts that can send and receive bitcoin (the currency of the same name). The state machine represented by the Bitcoin blockchain can be thought of as the balance of all user accounts. A transaction submitted to the Bitcoin network simply specifies a sender,

receiver, and amount of Bitcoin to be transferred, thus changing the state of the system. The blockchain, as an append-only, immutable structure, doesn't actually maintain a running balance for each account. Rather, the balance of a particular account is the result of every transaction ever applied to that account, and only the transactions are ever recorded in the blockchain. A "block" is the atomic unit of extending the chain, and is simply a bundle of transactions with some additional information needed to achieve the important properties of the blockchain (such as the hash of the previous block, timestamping information, other meta-data, and data that may be required for the particular form of consensus protocol being used on that blockchain).

**D. Data Structure**

A "blockchain" is so named due to the format of the underlying data structure, which largely differentiates a blockchain from other types of distributed database. A key cryptographic technique relied upon heavily in blockchain technology is "hashing." A "hash function" takes in any piece of data and produces a "hash value" for that data. Hash functions have several key properties that make them useful. They are one-way, meaning that knowing a hash value does not allow one to reconstruct the original data. They are unpredictable and change-sensitive, meaning that changing even a single bit of the input data will lead to an entirely different hash value compared to the original. Finally, they are collision-resistant, meaning that it is so unlikely as to be effectively impossible that two different inputs will yield the same hash value. Given these properties, hash values are an efficient way to "fingerprint" any arbitrarily sized piece of data.

A blockchain is a form of Merkle tree, a change-sensitive hash-based data structure in which hash values are successively aggregated (hashes are hashed) and can be used to quickly fingerprint large amounts of data. Transactions belonging to a block are stored in a Merkle tree, and the root hash of the Merkle tree is the only data that actually needs to be stored in the block. Sequential blocks are chained together using hash values as well. When a new block is added to the blockchain, part of the data inside the new block is the hash value of the previous block, which in turn contains the hash value of its predecessor, and so on until the "genesis" block, or first block in the chain. By chaining successive hash values, it becomes possible to efficiently verify that the data contained in one replica of the blockchain is perfectly identical to the data in another. [1, 27]

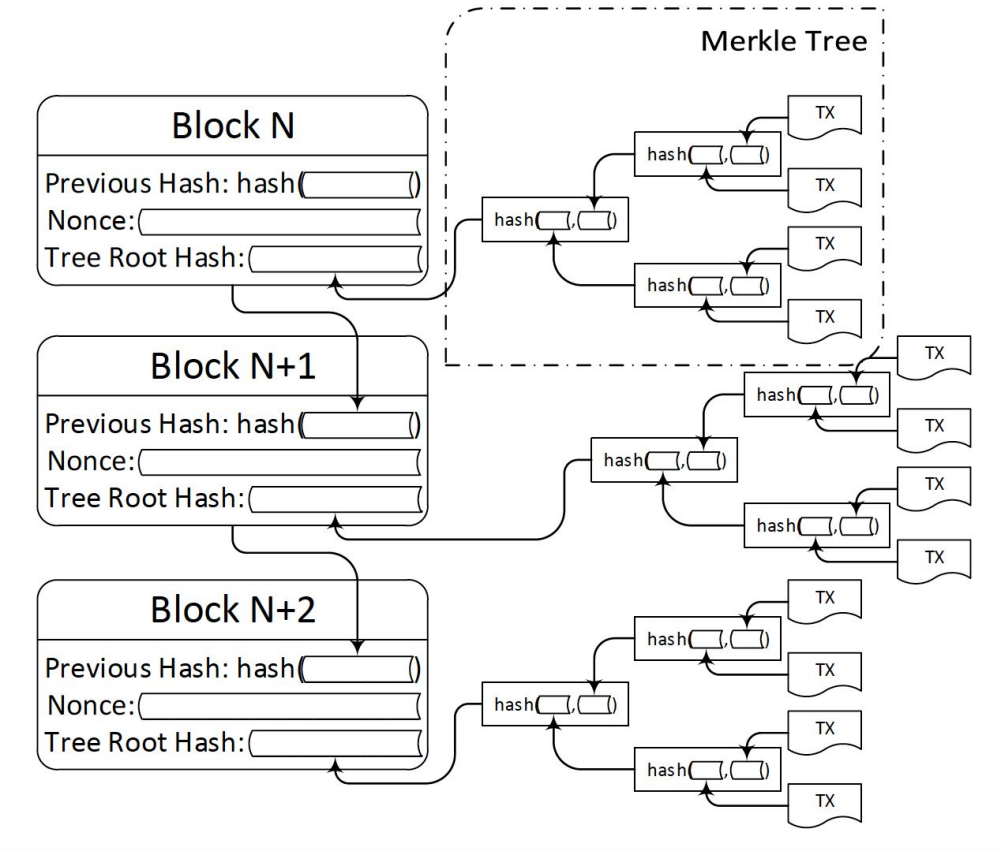


Figure 1. Merkle Trees and Block Structure [27]

## E. Smart Contracts

In the Bitcoin example above, a transaction is a simple piece of logic specifying a transfer of value between two parties. However, a blockchain state machine may be capable of representing more complex states than accounts and balances, and capable of moving between such states using complex logic. Complicated transaction logic can be implemented using "smart contracts" [2] or "chaincode" [14, 16] as termed by the Ethereum and Hyperledger frameworks, respectively. A smart contract is code that is stored within a transaction on the blockchain itself. Subsequent transactions can then reference and invoke this code. A simple example of what a smart contract might do is control the release of funds until multiple parties submit approval within some fixed window of time. A more complex usage might involve executing business logic and initiating transactions such as the exchange of physical goods based on the satisfaction of contract conditions.

### F. Use-cases

A natural use-case for a distributed ledger of trustless transactions is a currency. The cryptocurrency Bitcoin was the first successful implementation of a public blockchain, and as such blockchain technology has become largely associated with cryptocurrency in public awareness. However, the unique properties of a blockchain ledger may prove useful in a wide variety of other applications that currently rely on third parties and trusted middlemen to facilitate trust in exchanges. Some proposed use-cases include:

- creation of secure digital identities
- tokenization and control of physical assets or intellectual property, in which physical or digital assets are represented on the blockchain and ownership is managed and verified using blockchain technology
- improved voting and governance
- streamlining of clearing and settlement
- automation of business logic or contract fulfillment
- improvement of supply chain management and transparency

## II. Architectural Differences

### A. Public vs. Private

While any blockchain should provide immutability, consistency, transparency and tamper-evidence, there is great diversity of possible implementation and design choices that affect the properties and performance of the system. This means that different blockchain systems can be taxonomized along several axes and that the "best" architecture for a given use-case will depend on optimizing for that unique context.

A primary and important distinction is between "public" (or "unpermissioned") and "private" (or "permissioned") blockchains. Bitcoin is an example of a public blockchain. A public blockchain is so-called because it is truly open to the public--absolutely anyone can submit data to the chain, read data from the chain, and participate in its maintenance. A private blockchain, in comparison, is one that is not open to the public. There are various mechanisms by which access may be restricted, and various types and degrees of privacy, but any chain that isn't publicly accessible is considered private. In the case of private blockchains, the question may arise whether a blockchain is necessary at all. If every participant is already known and verified, isn't the system trusted, as opposed to trustless? In short, not necessarily. This is because multiple parties may be permissioned to contribute to the same blockchain, but may still be business rivals or have other conflicting interests. It is also true that

even parties believed to be trustworthy may still act maliciously, and using a blockchain data structure prevents them from wreaking havoc. Use of a private permissioning model may allow for the relaxation of certain security assumptions in the name of performance, while still maintaining security against rare but possible threats. Finally, a permissioned network has the ability to impose size constraints on the network, which may be prerequisite to other design considerations such as the choice of consensus algorithm. As research continues in blockchain network architecture and the technology adapts to a wider variety of use-cases, hybrid private-public architectures may also arise. See [18, 3, 7, 27] for further discussion of network permissioning.

## B. Consensus

The properties that make blockchain technology powerful, such as data immutability and the validation of transactions in a trustless environment, are often discussed without reference to what makes these properties possible. In essence, they arise from the inherent robustness of a large collection of united, honest actors against the goals of a smaller number of selfish or malicious actors who may seek to undermine the system. As an analogy, the more legs you add to a stool and the wider you make the base, the more difficult it becomes to topple. However, coordinating the behavior of a large number of dispersed nodes is no simple matter. In particular, the network must continually achieve consensus regarding the contents and order of the blockchain while also meeting security and performance requirements. The engine driving this coordination is known as the "consensus algorithm" (or "mechanism" or "protocol"), and has long been a focus of research in distributed computing, with the introduction of blockchain technology driving a resurgence of research. There are already many different consensus algorithms that provide different advantages, and their improvement is ongoing. While the permissioned vs. unpermissioned status of the network determines which consensus algorithms can be used, it is the choice of consensus algorithm that determines most other characteristics of the blockchain.

### 1. Consensus: Terms and Concepts

The nodes that are involved in the consensus process are known as "validators" or "validator nodes" (and in the case of proof-of-work consensus, often as "miners" or "mining nodes"). With regard to security and performance evaluation, the size of a network is only considered as large as the number of validators in the network, since these are the only nodes that participate in consensus and thereby secure the network. A "client" is any node capable of submitting transactions to the network, and may or may not also be a validator. The process of reaching consensus

involves broadcasting all data throughout the network and allowing all validators to scrutinize proposed transactions for conformity to network protocol (logic that may be arbitrarily complex, depending on the implementation of the blockchain). Such valid transactions are then prepared to be appended to the blockchain data structure by bundling multiple into a block. The way in which blocks are "published" (submitted for inclusion in the blockchain) depends on the specific consensus algorithm in use, but a published block is once again subject to the scrutiny of the network, and appended to the chain only if it is correct in terms of the relevant protocol. A block is considered "committed" to the blockchain when it has been circulated through the network and applied by all nodes (or by a sufficient threshold of nodes).

## 2. Consensus: Forks and finality

As blocks are published and committed to the chain, it is possible in some styles of consensus that more than one valid successor block is published on top of the current "head" (end) of the chain. If such multiple valid successors are released within a short enough time span, part of the network may first receive and start building upon one block, while the rest of the network may first receive another. This creates a "fork" in the chain, in which two or more equally valid versions of the blockchain exist concurrently in the network. Forks are resolved as one of the branches inevitably grows faster than the other, and the nodes of the network adopt this longest chain as the true blockchain, abandoning the shorter branches. The choice of "longest chain" is a criterion that would be defined by the specific consensus protocol that the network is using. In some protocols, "longest" may be determined simply by the number of blocks, whereas others might use a slightly different criterion such as "heaviest" with respect to some resource. A transaction is only considered "final" when it cannot be or is highly unlikely to be reversed. Of course, a blockchain is an immutable, append-only data structure, so "removal" of a transaction from the blockchain is equivalent to a longer fork of the chain being established in which the transaction is not included. Some consensus algorithms allow for "deterministic finality" in which forking does not occur and transactions are finalized as soon as they are committed. Other consensus algorithms allow only for "probabilistic finality," and transactions are only considered final when they are buried under enough subsequent commits that the probability of a different fork being accepted by the network is acceptably low.

## 3. Consensus: Security and performance properties

A viable consensus algorithm for a blockchain network has to meet many requirements, though the relative importance of these properties depend largely on

the use-case to which the blockchain is being applied and the environment in which it is deployed. First and most importantly, the algorithm must operate in a fully decentralized manner. There can be no centralized point of failure or corruption; otherwise, the other properties of a blockchain data structure become irrelevant. A single point of centralization renders the entire system centralized – or at least vulnerable in the same way as centralized systems. The algorithm must meet the security requirements of the network under relevant trust assumptions and environment, which in a trustless system means tolerating Byzantine faults, nodes that may exhibit not just crashes or errors, but malicious and strategic behaviors as well [7].

The security solution must not come at the expense of "safety" and "liveness," terms applicable to any system that deals with concurrency. "Safety" means, in a general sense, that "bad" outcomes won't happen in a system. In a blockchain network, this means that honest nodes sharing the same protocol and seeing the same data will agree on the same state. In other words, transactions and the state changes they cause are ordered and deterministic. "Liveness" means, generally, that *something* will eventually happen (deadlock or starvation will be avoided). In a blockchain network, this means that all nodes must eventually agree on state ("eventual consistency") and that progress will not stall indefinitely. An influential result in distributed systems theory known as the "FLP impossibility" states that it is impossible to theoretically guarantee both safety and liveness at the same time *in a fully asynchronous system* [11]. In practice, however, it is possible to develop performant and provable guarantees of both *given stronger assumptions*, such as time bounds on message delays and access by nodes to synchronized clocks. Different consensus algorithms place different emphasis on the relative importance of safety versus liveness, another example of how consensus properties should be customized to the particular needs and configurations of a given network.

Finally, in addition to security and concurrency concerns, a viable consensus algorithm must be adequately performant along several important dimensions. The "throughput" of a blockchain network is usually measured as transactions per second (tps). This metric is related to, but not derivable from "latency," which is a measure of the average time between submission of a transaction to the network and its inclusion on the chain. Latency may be driven up by high wait times for certain transactions, even if throughput remains constant. Again, the use-case of a particular blockchain will determine acceptable bounds on transaction throughput and latency. Bitcoin, for example, has a theoretical maximum throughput of ~7 tps, a number often compared to the ~10,000 tps processed by the Visa network and

cited as a fatal barrier to the replacement of fiat currencies (traditional, government-backed currencies) with Bitcoin. In other cases, time-sensitive applications may be more concerned with latency, and the guarantee that a transaction will be finalized and therefore usable within a bounded time window. "Scalability" is a related property that refers to the effect of network size on performance. A blockchain network is considered scalable if its required or potential growth preserves the adequacy of other properties. Consensus algorithms differ wildly in their ability to scale, but scalability represents trade-offs with other performance properties and is therefore also very case-dependent. See [1, 2, 3, 7, 25, 26] for further discussion of consensus in blockchain technology.

## III. Survey of Consensus Algorithms

### A. Proof of Work (PoW)

"Proof of Work" (PoW) is the name of the consensus algorithm that is used to extend and secure the Bitcoin blockchain. PoW is also used to refer to an entire family of consensus algorithms with similar properties, which will be discussed shortly. The specific PoW algorithm used on the Bitcoin blockchain is sometimes referred to more specifically as "Nakamoto PoW" or "Nakamoto consensus" after Satoshi Nakamoto, the pseudonym of Bitcoin's anonymous founder. See [1] for the most thorough, published treatment of Nakamoto PoW. The basics of Nakamoto PoW are as follows:

#### 1. PoW: The Algorithm

The Bitcoin network is consistently flooded with new incoming transactions that are submitted by clients and broadcast throughout the network, eventually to be received by all validator nodes. Validator nodes assemble these unpublished (or "uncommitted") transactions into blocks, verifying the validity of each transaction and candidate block based on the definition of valid in the Bitcoin protocol. The validator node then hopes to publish their newly assembled candidate block, because there is a financial reward for successfully publishing a block. The "winning" validator will receive the incentive fees submitted along with each included transaction, as well as a small amount of newly minted Bitcoin generated in the block itself. However, in order to publish a candidate block, a validator must solve a cryptographic challenge associated with that specific block. This cryptographic challenge is useless in nature – it adds no information to the blockchain and performs no useful work. The puzzle is simply a game of guess-and-check, in which miners systematically search a space of random values for one which, when hashed together with the block, produces a hash value that meets an established requirement. There is no way to gain an

advantage in this random puzzle apart from devoting more computational horsepower. If a miner happens upon this satisfactory value before other miners, she will get to extend the chain with her block, reap the rewards, and the race starts over with the next block.

So how does this useless computational work keep the blockchain secure? It is first important to understand that the Bitcoin protocol, run by all nodes on the network, regards the *longest* valid chain to be the "truth," the universally accepted version of the blockchain. Therefore, malicious actors seeking to write a history that benefits themselves can only do so in one way: control the longest chain. Only by creating a branch of the chain that becomes and remains the longest can a version of reality be established that is *also* accepted by the other nodes of the network. The useless but computationally expensive puzzle associated with publishing a block makes extending the chain difficult and expensive. When an entire network of mining nodes around the world are racing to publish each additional block, it becomes unlikely that a single node or smaller group of nodes will be able to publish blocks faster than the rest of the network combined (thus writing the longest chain). In this way, the right to actually record data in the blockchain remains randomized and dispersed across many miners, and the ability of individuals or small subgroups to control the extension of the chain becomes limited by the expense of computational resources.

## 2. PoW: Properties

One of the strongest benefits of the PoW algorithm is that it provides a high degree of security, since write-access to the blockchain is limited by one's access to sheer computational resources and the proportionally increased probability of stumbling on a lucky solution. PoW algorithms can be susceptible to a "51% attack," in which any malicious individual or sub-group that controls over 50% of the mining power on the network can control what is and isn't written to the ledger. When a network is small, it is much more susceptible to a 51% attack because the absolute amount of computational power needed to control 51% the network is much smaller. However, once a network is formidable enough in size, a 51% attack becomes extremely expensive to carry out. Even in the case of a small network, there are economic incentive mechanisms in place that make a 51% attack financially fruitless.

It should be noted, however, that while the Bitcoin network has proven secure in practice, the validation network is not as decentralized, and therefore not as robust to aggregated attacks, as it was intended to be in theory. To gain a

computational advantage in the PoW algorithm, miners have created ASIC (application-specific integrated circuit) hardware designed specifically for the PoW computations. General-purpose CPUs and GPUs are no match for the speed of this specialized hardware, and as such, mining Bitcoin has become a capital-intensive activity that encourages the aggregation of mining behavior into "pools." Thus, the Bitcoin validation network is more of an oligarchy than the democracy it was intended to be. Other PoW-based networks such as Ethereum have created variations of PoW to render the algorithm ASIC-resistant.

As a result of this security, PoW has proven itself successful in completely unpermissioned, public networks such as Bitcoin and Ethereum, as well as many lesser-known cryptocurrencies. However, the required degree of security in a PoW network has meant compromising other desirable properties of the system. One such property is network throughput, measured in the number of transactions processed per second (tps) across the entire network. The reason for this has to do with the importance of maintaining a consistent and relatively long "block interval," the amount of time between publishing consecutive blocks. The length of the block interval has direct implications for the security of the consensus algorithm. In simplest terms, this is because the faster (easier) it is to publish blocks, the easier it would be for an adversary to create a longer version of the blockchain than the rest of the network [12]. Bitcoin's theoretical upper bound of 7 tps is due to the choice of a 10 minute block interval (and the maximum size of a block), and is a major barrier to its practical adoption as a replacement currency. Limited throughput may also pose a barrier to other high-volume, non-currency applications of blockchain technology.

Another major disadvantage of PoW consensus is the electricity spent performing the computational work. At the time of writing, the estimated annual energy consumption of the Bitcoin network is about 64 TWh or 0.29% of global energy consumption - roughly the same amount of energy used annually by the nation of Switzerland [4]. Even more alarming than this figure is the fact that energy consumption has increased exponentially since Bitcoin's creation and will continue to grow as the network grows. This has to do with the aforementioned need to maintain a rather long block interval. In Bitcoin this interval is approximately 10 minutes, meaning it should take an average of 10 minutes to find the solution to the computational puzzle. The speed with which a solution is found is related to both the difficulty of the puzzle and the computational resources available for solving it. As the network grows, the available computational resources grow, and therefore the difficulty of the puzzle is

increased in order to maintain the consistent 10-minute block interval. In short, the amount of electricity needed to secure the network is proportional to the size of the network, and that represents a major problem in sustainable scalability.

It should be noted here that PoW consensus is actually a scalable solution from a performance perspective, if not from an environmental one. To distribute information such as a published block, the network only requires a single round of multicast (each node forwards information to its peers who have not yet received it), which eventually allows the information to be distributed to every node in a fully connected network. Therefore the complexity of PoW consensus scales sub-linearly with the number of nodes in the network [9]. Of final note, finality (the point at which a transaction is considered immutable) in the PoW algorithm is probabilistic, not deterministic. Whether probabilistic finality represents a problem depends on the use-case of the blockchain. In the case of Bitcoin, probabilistic finality means that the most recent blocks added to the chain can't necessarily be trusted to remain as part of the longest chain, since forks at the tip of the chain are common. Once a block has been confirmed in the longest chain by the addition of several subsequent blocks (6 in Bitcoin), the probability of it being unconfirmed (usurped by a longer chain) becomes so infinitesimal that the blocks are regarded as immutable. The means that there is a delay between the submission of a transaction to the network, and the ability to safely use that transaction in subsequent transactions. This delay may be unacceptable in other cases.

## B. Proof of Elapsed Time (PoET)

Proof of Elapsed Time (PoET) is a promising consensus algorithm developed recently by Intel in conjunction with their Software Guard Extension (SGX) technology, and implemented in the open-source Hyperledger Sawtooth blockchain framework. PoET is a form of PoW consensus, but aims to eliminate the wasteful energy consumption associated with the original algorithm. See [21, 5, 9, 7] for further discussion of PoET.

### 1. PoET: The Algorithm

Traditional PoW can be thought of as a lottery where the chance of winning is proportional to the amount of computational work expended. While the chance of winning this lottery is proportional to the amount of computational work expended, the winner is still random--whichever node happens on a solution first. This randomness helps prevent any one party from systematically controlling the writing of blocks. PoET creates a similar lottery-based system for block

publishing, but instead of spinning computational wheels, each node is assigned a random wait time from code running inside a “trusted execution environment” (TEE). The node with the shortest wait time will get to be the first to publish the next block. In the case that more than one block is published almost simultaneously, a measure of time-spent-waiting is used to resolve the potential fork, favoring the chain with lowest aggregate wait time.

The natural question is how can a malicious node be prevented from faking short wait times and thereby controlling the chain? Intel SGX technology makes possible hardware-assisted TEEs, secure areas of a main processor that protect application level code even from processes running at a higher privilege level. TEEs are execution environments that have an extremely small attack surface and are equipped with cryptographic verification procedures that can provide externally verifiable attestations and tamper evidence. Each node that participates in PoET must be equipped with this specialized hardware. The random wait time is generated from a sufficient source of entropy within the TEE, and cryptographically signed by the TEE such that the validity of the wait time is verifiable by other nodes running the PoET protocol. While the details of TEE security are outside the scope of this review, there are theoretical attacks that can be made on a TEE. In the case of compromised TEEs, PoET implements a second line of defense: a statistical z-test that examines whether any given node’s rate of publishing is anomalously fast in a statistically significant way. Therefore, even if a node manages to fool the network with regard to wait time attestation, they will be prevented from publishing too many blocks by statistics.

## 2. PoET: Properties

Despite the implementation differences between Nakamoto PoW and PoET, PoET is also a Proof-of-Work *style* algorithm, and therefore shares many properties with Nakamoto PoW in terms of security, throughput, latency, scalability, and probabilistic finality. A given instantiation of a PoET-based network is configurable (as is Bitcoin, if it were to be deployed anew), and different choices will lead to different metrics. For example, in the Bitcoin network, the block interval is set at 10 minutes and cannot be changed, which limits the maximum theoretical throughput. A PoET-based network may choose to use a shorter block interval, thus raising throughput at some expense to security. However, the theoretical properties of both systems and the way they respond to changes of configuration is very similar.

Like Nakamoto PoW, PoET can be used in public, unpermissioned networks, but the Hyperledger Sawtooth implementation is also designed to accommodate a private network environment. The vastly more robust on-chain logic (smart contract / chaincode) capabilities of the Sawtooth platform allow for node permissioning in a way that the Bitcoin protocol does not. The major advantage to PoET over Nakamoto PoW is the elimination of wasteful electricity consumption. It is also interesting to note that PoET can be used in a system that has no associated cryptocurrency. Since PoET doesn't require the contribution of expensive computational resources, validators don't need an equally strong incentive in the form of cryptocurrency block rewards or transaction fees before they are willing to participate in validation. The primary disadvantage to PoET is the reliance on specialized hardware, which represents a barrier to adoption and potentially centralizes too much power in the hands of hardware manufacturers such as Intel.

### C. Proof of Stake (PoS)

Another alternative to PoW which is also aimed at reducing electricity consumption is known as "Proof of Stake" (PoS). Put simply, PoS grants mining/validation power to a user in proportion to the amount of cryptocurrency that he or she is willing to stake for a chance at publishing the next block. For the incentive structure of PoS to function, it requires the existence of a valued cryptographic token, which makes it unsuited to enterprise frameworks such as Hyperledger that make no use of cryptocurrency. The idea is that a user stands to lose her staked tokens if she is selected as a validator but doesn't act in accordance with the consensus protocol (in other words, if she attempts to publish a block that is rejected by the rest of the network). Furthermore, for a user to perform a 51% attack in PoS consensus, they would have to own 51% of the currency. In theory, this would dissuade them from undermining the very network in which they are storing significant value. As of fall 2018, the Ethereum network is poised to transition to a PoS consensus algorithm named "Casper," the success of which is still unclear [2].

### D. Proof of Luck (PoL)

Proof of Luck functions almost identically to PoET and also takes advantage of TEEs. Unlike PoET, PoL is not used in any production-ready blockchain frameworks, but there is a proof-of-concept implementation called Luckychain. Whereas PoET uses a verifiable code base running in a TEE to assign a random wait time to the validator, PoL uses the TEE to assign a "luck" value which is a random number sampled from a uniform distribution on the interval (0, 1]. This luck value is then converted into a wait time, with higher luck values corresponding to

shorter wait times. In the case of a fork, the chain with highest aggregate luck value is selected. An interesting proposed extension to the PoL algorithm is "Luckiest *m*", in which the *m* blocks with the highest luck values for each round are all broadcast and deterministically aggregated into a super-block, which is then used to extend the chain. This is a defense mechanism against potentially compromised TEEs; even if up to *m* - 1 CPUs are compromised, the attacker still cannot fully control the superblock. This addition may limit scalability as it increases the number of messages that must be passed among nodes. For the full whitepaper, see [29].

**E. Proof of Space (PoSpace)**

Also known as "Proof of Storage" or "Proof of Capacity", PoSpace is another PoW-style approach to consensus that seeks to minimize the energy waste associated with traditional PoW. The algorithm is based on a 2015 paper [31] and there are currently two cryptocurrency implementations based on PoSpace consensus: SpaceMint [30] and Burstcoin. Instead of requiring computation to be performed in order to publish blocks, PoSpace requires the dedication of hard disk space. The chance of a node publishing the next block is proportional to the amount of dedicated disk space. The algorithm works by requiring an initial non-trivial amount of setup time in which the dedicated disk space is filled with a collection of randomly generated potential solutions to a "hard-to-pebble" graph problem. PoSpace is a relatively new consensus approach and isn't well understood or widely adopted yet. PoSpace faces the problem of "nothing-at-stake," the situation in which there are minimal disincentives to selfish or malicious mining strategies. Unlike traditional PoW which requires electricity expense in proportion to the likelihood of publishing, dedicating disk space is a one-time, low-cost expense. Therefore successful PoSpace algorithms must include additional complexities to prevent or sufficiently disincentivize selfish behavior. For example, SpaceMint implements additional logic that punishes miners for attempting to publish blocks on more than one fork. The more complex the underlying algorithm, the more difficult it becomes to prove its properties and the more points of potential attack are introduced.

Like PoW, PoSpace is designed to operate in a public network environment. Assuming PoSpace proves sufficiently secure, its primary benefit over traditional PoW is the massive reduction in energy consumption. It is also resistant to specialized hardware advantages since general-purpose memory is already optimal, which favors decentralization and allows nodes of all resource levels to participate in consensus.

## F. Practical Byzantine Fault Tolerance (PBFT)

PoW consensus offers strong security as is required in large, public networks, and also scales well with large numbers of nodes. However, this comes with a performance trade-off in other areas such as transaction throughput, which is quite low in PoW. In the case of a private (permissioned) network, the security assumptions may be less threatening. Additionally, the size of the network may be fixed in advance or bounded. In this case, a different consensus algorithm may provide a suitable solution. One such algorithm that is used in the Hyperledger Fabric platform (and modifications of which are used in the Ripple and Stellar protocols) is Practical Byzantine Fault Tolerance (PBFT). For PBFT to function, it is assumed that the number of malicious or faulty nodes doesn't equal or exceed 33% of the total number of nodes in the system (a weaker security goal than the 50% malicious threshold in PoW).

### 1. PBFT: The Algorithm

Most generally, PBFT can be described as a voting algorithm, in which a quorum of votes is collected by each node and used to determine the validity of a given transaction or group of transactions. Nodes on the network are given an ordering, and during any given "round" of PBFT (an update to the blockchain) one of the nodes acts as a leader. The leader collects transactions and bundles them into a block, and then multicasts the proposed block to the rest of the nodes. The rest of the nodes run and validate all contained transactions, and then broadcast their results to all nodes, either approving or rejecting the proposed block, based on the relevant validation logic. When a node receives a quorum of at least f + 1 identical responses (where f represents the maximum number of malicious nodes) it accepts that response as the result of the proposal. If the network has failed to make progress after a time interval, the leader is automatically replaced. The leader for each successive round is automatically chosen in a round-robin fashion [10, 19].

### 2. PBFT: Properties

PBFT has several important advantages. Unlike PoW consensus, block commits are immediately final, which is potentially important for applications that may need to respond to committed transactions in a time-sensitive manner. There is no need to wait for a predetermined number of additional commits before accepting the finality of the transaction. PBFT also uses such a negligible amount of electricity as compared to PoW that it is of no concern. The throughput of PBFT is orders of magnitude higher than in PoW systems. This is

because security in PoW consensus depends on making it difficult for an adversary to write a longer chain faster than the remainder of the network and therefore depends on block commits being slow and painstaking, which limits transaction throughput. The majority-vote security model of PBFT can be run as quickly as nodes can validate transactions and communicate with each other, with no impact on security. Hyperledger Fabric has been shown to handle up to 10,000 tps in a network size of up to 16 nodes [10, 20].

The major drawback to PBFT is that the complexity of the message passing scheme is quadratic in the number of nodes. Therefore PBFT can only handle very small consensus group sizes before the number of messages being exchanged becomes prohibitively large. While the Bitcoin and Ethereum networks have already scaled to thousands of nodes using PoW consensus, Hyperledger Fabric using PBFT has only been shown to scale well up to 16 nodes, after which the network failed entirely due to congestion in message channels [10]. PBFT is also susceptible to "sybil attacks," in which a single adversary assumes multiple identities on a network. Since voting power isn't tied to a physical resource, an adversary need only pose as multiple independent identities to gain voting power and potentially control 33% of the network. For this reason, PBFT is only secure in a permissioned network setting, preventing the inclusion of unknown identities in the consensus group.

### G. PBFT Variants: Ripple RPCA

Ripple, the real-time gross settlement system, uses a version of the PBFT algorithm that allows for network scalability despite the complexity-restricted consensus group size. Ripple has its own cryptocurrency, XRP, which is used as an intermediary currency to facilitate cross-border exchange of fiat currencies. The "Ripple Protocol Consensus Algorithm" (RPCA) works by establishing many smaller but overlapping groups of nodes that concern themselves only with establishing consensus within the group. Each node maintains a "Unique Node List" (UNL) which are the network peers with whom they perform consensus. During each round of consensus, a node collects all transactions that it has seen since the last round and have not yet been committed, amalgamating them into a candidate list and finally sharing them with the other nodes on the UNL. Each node then runs and verifies the validity of every transaction, submitting votes to its UNL. A transaction requires a minimum threshold of 80% yes votes to be committed, meaning the Ripple network can only tolerate at most 20% Byzantine failures (as opposed to the 33% handled by PBFT). RPCA includes several additional heuristics and procedures that help police the network and eliminate faulty behavior. For example, nodes that vote no on every transaction,

consistently propose invalid transactions to the network, or take too long to respond are automatically removed. While users are free to choose their own UNLs, Ripple provides curated UNLs to maximize trustworthiness for naive users. The protocol also requires that the UNLs of any pair of validating nodes overlap by at least ⅕ the size of the larger list in order to prevent a split network. The RPCA is a promising advancement in enterprise-grade, scalable, performant consensus that achieves secure, deterministic finality. The primary criticism of Ripple and RPCA is the relative degree of centralization currently present in the system. RPCA is not a centralized algorithm, but Ripple itself owns more than 60% of XRP, which gives them great influence over the price of XRP. Ripple also controls five trusted validator nodes that trust each other and no other node. As of 2017 Ripple was unable to recommend any other validators in the default and recommended list of validators, thus many validator nodes depend specifically on Ripple validators to comprise their UNLS, making the consensus algorithm far more centralized than it could be [7, 22, 23].

### H. PBFT Variants: Stellar SCP

Stellar is also an exchange system that facilitates currency exchanges and fast cross-border transacting. It was founded by one of the co-founders of Ripple, and is somewhat more focused on providing services to individual people, especially those in the developing world. Stellar uses a similar protocol called “Stellar Consensus Protocol” (SCP). SCP uses a protocol called “federated Byzantine agreement” in which each validator declares its own “quorum slice” (very similar to the Unique Node List in RPCA), or set of multiple quorum slices. A quorum slice for a node is simply a subset of the network that also contains the node itself, and in SCP, as in RPCA, each node decides which other nodes it trusts based on the nodes’ reputation both on and off the network. Quorum slices are subject to the same overlapping requirements as the UNL to prevent network forks. [7, 24, 28]

## IV. Performance analysis

Given the many design choices and environmental factors that comprise a specific blockchain instance, it is important to develop meaningful performance evaluation techniques to assess the fitness of a system for its intended purpose. More specifically, standard benchmarking techniques need to be established to allow meaningful comparisons among different platforms and configurations. The complexity of blockchain systems makes even instance-specific performance evaluation difficult to design, implement, and understand. This, combined with the nascent status of the industry and the incredible diversity of designs, makes establishing meaningful comparative benchmarks even more challenging. A paper from Hyperledger in October

2018 outlines many of these challenges and states that performance evaluation will be a necessary first-step toward formal benchmarks [15]. The ability to benchmark performance will be necessary to facilitate adoption by industry and allow businesses to choose optimal solutions for their needs [10, 12, 15].

Standardization of terminology is prerequisite to intelligent discussion of performance, and the aforementioned paper by Hyperledger is currently the most complete published attempt. For example, the term “transaction” becomes complicated in a performance context. In some networks (e.g., Bitcoin), a transaction is the atomic unit of state change, but in other networks (e.g., Hyperledger) transactions are bundled into “batches” in which either all batched transactions are committed, or none are. To further complicate matters, not all transactions are created equal: some may involve complicated smart contract logic that itself becomes a bottleneck to throughput. Therefore, it doesn’t suffice to consider only the number of transactions, but potentially also the type when considering a throughput metric such as tps. The term “node” is another example of complicated terminology. A node is an abstract concept: it is a single locus of computation, but doesn’t necessarily represent an independent piece of hardware or independent user, and nodes may vary drastically in the computing resources available to them. Many platforms differentiate nodes into distinct roles, such as load-generating clients, validators, or “ordering service” nodes as in Hyperledger Fabric.

These examples serve to point out that an evaluation metric is meaningless without a complete description of the system being tested. Such a complete description must include the consensus protocol being used, the geographic distribution of the network (and any other factors relevant to network latency or bottlenecks), the hardware and software environments of all peers, the number and types of nodes involved in various roles, tools used for testing and observation, the type of data store used to represent the blockchain, and the nature of delivered workloads [15].

A 2017 paper, “BLOCKBENCH: A Framework for Analyzing Private Blockchains” [10] provides an excellent discussion of the complications that arise while trying to understand component functionality in a complex system. They identify four primary layers in a blockchain system, each of which has the potential to create performance bottlenecks.

1. The first layer is the consensus protocol, which is responsible for achieving consistency among all nodes. Performance constraints in the consensus layer may be intentional in the algorithm itself (PoW is designed to be slow and computation-heavy for security purposes), or an unfortunate side-effect of

expensive, high-complexity protocols such as PBFT with its need for high-volume message passing.

2. The second layer is the data model, or the actual data structure used to store the blockchain. Performance may be hindered by the data model layer if transactions are IO-intensive, requiring many queries of the underlying data structure.
3. The third layer is the execution environment, the environment that executes the logic of smart contracts or chaincode. This execution must be fast, since all validator nodes must execute the logic of every transaction. Ethereum executes smart contract logic using its own blockchain-specific virtual machine (the Ethereum Virtual Machine or EVM), whereas Hyperledger executes chaincode written in one of several common high-level languages inside a Docker image.
4. The fourth and final layer is the application layer, which represents any application that would use a blockchain as its underlying data structure. In the application layer, it's the workflow of the application itself that may be a bottleneck, which is of lesser concern for system benchmarking. Without targeting the performance of a specific layer, it may be unclear where performance bottlenecks are occurring. The BLOCKBENCH framework delivers layer-targeted workloads in order to apply isolated stress to potential bottlenecks.

Performance analysis and benchmarking for blockchain technology is still in its infancy. This is especially true for the many emerging approaches to consensus that differ from traditional PoW, which has thus far received the most attention from researchers. A deep understanding of these systems will come from a combination of empirical evidence, simulation, and theoretical analyses. Expert Christian Cachin of IBM points out that "Simulation alone isn't enough to prove the security of a model since truly exhaustive tests are usually impossible in a given scenario space, so we resort to mathematical tools and analysis" [7]. At the same time, the extreme number of variables introduced in real-world instantiations may render mathematical analyses insufficient for predicting the actual behavior of a system. It will be important to identify which design factors dominate the performance of the system, in what way, and in which combinations.

## V. Considerations for Supply Chain Management

A commonly proposed and widely encompassing use for blockchain technology is in supply chain management. A "supply chain" is a system of organizations and processes involved in taking material from a raw resource, through all transfers and transformations, and finally delivering it to an end user. The global supply chain is a wildly complex, interconnected network of supply chains with no central authority.

“Supply-chain management” is the oversight and coordination of supply chain activity, and its objectives include reducing costs, maintaining product quality, moving goods quickly, efficiently, and dependably, minimizing risks, promoting sustainability, and maintaining flexibility [17]. Consumers and governments have additional desires, including transparency of product origin and traceability of goods, which aids in preventing food and pharmaceutical safety problems, promoting sustainable decision-making, and reducing illegal or fraudulent activity. Modern supply-chain management faces many problems in meeting these objectives that may be eased or eliminated with the help of blockchain technology.

Several aforementioned problems are the result of the disconnected state of information in a supply chain. Individual organizations maintain their own ledgers and business data, but there are no structures in place for creating inter-organization record keeping such that data from multiple organizations can be integrated without painstaking, manual assembly. Such integrated records could make it possible to trace a product all the way from storefront to its sources of raw materials, thus helping consumers make conscious, competitive choices about which industries to support. Better integration would also help pinpoint the source of foodborne illnesses or other safety hazards, and could aid in optimizing the supply chain itself by identifying and removing inefficiencies.

Blockchain technology may also prove prerequisite to widespread incorporation of Internet of Things (IoT) technology in supply chain management. IoT technology consists of internet-connected sensors, identifiers or other “smart” devices that are being increasingly used to monitor shipping conditions, delivery times, and product identities. For example, IoT sensors can be used to monitor the environmental temperature of heat-sensitive food or drugs while in transit. The use of blockchain technology in conjunction with IoT would serve to prevent fraudulent sensor data and to create an automated, trustworthy way of collecting and using such data. The use of smart contracts (embedded logic) within a supply-chain blockchain could allow for trustworthy automation of business logic that has previously been manual. For example, payments could be released automatically upon the receipt of a particular shipment based on logic existing on the blockchain. The establishment of such a contract can create a guarantee that the sender will automatically be paid if they fulfill their end of the contract, and speeds the process of releasing funds. More complex logic could be established to optimize supply and demand, enforce quality control, or reduce the number of required middlemen, to name a few examples. Blockchain technology allows for the establishment of such logic in a trustworthy and agreed-upon way, after which all parties know the logic will be executed exactly as written.

Despite advanced database technology, there are many clear reasons why inter-organizational record keeping hasn't already been established. Without the properties of blockchain technology such as data immutability and cryptographic identity verification, there would be little in place to stop malicious actors from tampering with records for their own competitive, financial or legal gains. Another problem that remains to be solved and is a current research focus in the blockchain community is that of data privacy. Businesses have an obvious interest in keeping the details of transactions private to all but the directly involved parties. Research is ongoing in cryptographic data access control and "zero-knowledge proofs," with the goal being that data can be written to the blockchain and used to check the validity of transactions or queried for other reasons without revealing that data publically.

Privacy concerns are just one factor that will shape the architecture of blockchain solutions for supply chain. For integrated supply chain data to be useful, many or even most players involved in a particular supply chain need to be participating in the blockchain network. Not all organizations will adopt such technology at the same time, or buy into the same platforms. There is no central global authority in place to establish a single blockchain infrastructure or to govern any required network permissioning. Therefore, blockchain solutions for supply-chain management will likely be diverse, fragmented, and industry-specific. There is a need for interoperability among different chains, various types of privacy guarantees, and continued research in all forms of consensus algorithms. Permissioned consensus algorithms (such as PBFT and its variants) may be most beneficial in smaller or intra-organizational networks, or in any situation requiring high transaction throughput (such as IoT networks). Public consensus algorithms (such as PoW and PoET) will be important for securing networks in which permissioning is infeasible due to lack of central governance. Continued work improving and understanding the performance and security properties of blockchain consensus mechanisms will be important for advancing real-world solutions to current challenges in supply chain management.

## VI. References


[1] A. Antonopoulos, *Mastering Bitcoin: programming the open blockchain*. Beijing: OReilly, 2017.

[2] A. Antonopoulos, G. Wood, *Mastering Ethereum building smart contracts and dapps*, DRAFT. O'Reilly Media, 2018. (https://github.com/ethereumbook/ethereumbook)

[3] A. Baliga, "Understanding Blockchain Consensus Models," Persistent Systems Ltd. Corporate Report. 2017.

[4] Bitcoin energy consumption. https://digiconomist.net/bitcoin-energy-consumption

[5] L. C. B, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On Security Analysis of Proof-of-Elapsed-Time ( PoET )," *Springer Int. Publ.*, pp. 282–297, 2017.

[6] BlockBench: private blockchains benchmarking. https://github.com/ooibc88/blockbench.

[7] C. Cachin and M. Vukolić, "Blockchain Consensus Protocols in the Wild," *arXiv*, no. IBM Research-Zurich, 2017.

[8] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based Traceability in Agri-Food Supply Chain Management : A Practical Implementation," *IEEE*, pp. 3–6, 2018.

[9] H. Dang, A. Dinh, E.-C. Chang, and B. C. Ooi, "Chain of Trust: Can Trusted Hardware Help Scaling Blockchains?," *CoRR*, 2018.

[10] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A Framework for Analyzing Private Blockchains," *ACM*, pp. 1085–1100, 2017.

[11] J. Fischer, A. Lynch, and S. Paterson, "Impossibility of Distributed Consensus," *ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[12] A. Gervais, K. Wüst, and H. Ritzdorf, "On the Security and Performance of Proof of Work Blockchains," *IACR Cryptol. ePrint Arch.*, 2016.

[13] M. Hulea, O. Rosu, R. Miron, and A. Astilean, "Pharmaceutical cold chain management: Platform based on a distributed ledger," *2018 IEEE Int. Conf. Autom. Qual. Testing, Robot.*, vol. 6, pp. 1–6, 2018.

[14] Hyperledger. Blockchain technologies for business. https://www.hyperledger.org

[15] Hyperledger, "Hyperledger Blockchain Performance Metrics," *Hyperledger.org*, pp. 1–17, 2018.

[16] Hyperledger Sawtooth. https://sawtooth.hyperledger.org/docs/core/releases/1.0/contents.html

[17] N. Kshetri, “Blockchain’s roles in meeting key supply chain management objectives,” *Int. J. Inf. Manage.*, vol. 39, no. December 2017, pp. 80–89, 2018.
[18] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame, “Towards Scalable and Private Industrial Blockchains,” in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts - BCC ’17*, 2017.
[19] B. Liskov and M. Castro, “Practical Byzantine Fault Tolerance,” Proceedings of the Third Symposium on Operating Systems Design and Implementation. New Orleans, USA. February, 1999.
[20] Q. Nasir, I. A. Qasse, M. A. Talib, and A. B. Nassif, “Performance Analysis of Hyperledger Fabric Platforms,” vol. 2018, 2017.
[21] PoET Specification. https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html
[22] Ripple. https://ripple.com
[23] D. Schwartz, N. Youngs, and A. Britto, “The Ripple protocol consensus algorithm,” Ripple Labs Inc. White Paper, 2014.
[24] Stellar. https://stellar.org
[25] D. K. Tosh, S. Shetty, X. Liang, C. Kamhoua, and L. Njilla, “Consensus Protocols for Blockchain-based Data Provenance : Challenges and Opportunities,” *IEEE*, pp. 469–474, 2017.
[26] A. Wahab and W. Memood, “Survey of Consensus Protocols,” pp. 1–12, 2018.
[27] K. Zhang and H. A. Jacobsen, “Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains,” Middleware Systems Research Group, Univ. Toronto, Canada, 2018.
[28] D. Mazieres. The Stellar consensus protocol: A federated model for Internet-level consensus. Stellar, available online, https://www.stellar.org/papers/stellar-consensus-protocol.pdf, 2016.
[29] M. Milutinovic, W. He, H. Wu, and M. Kanwal, “Proof of Luck: an Efficient Blockchain Consensus Protocol,” *Conf. Proc. SysTEX*, 2016.
[30] J. Alwen, G. Fuchsbauer, P. Gazi, S. Park, and K. Pietrzak, “Spacecoin : A Cryptocurrency Based on Proofs of Space,” *IACR Cryptol. ePrint Arch.*, pp. 1–26, 2015.
[31] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, “Proofs of Space,” 2015.