Demystifying Blockchain by Teaching It in Computer Science*

Adventures in Essence, Accidents, and Data Structures

Alan G. Labouseur, Matthew Johnson, Thomas Magnusson
School of Computer Science and Mathematics
Marist College
Poughkeepsie, NY 12601 USA
{Alan.Labouseur,Matthew.Johnson,Thomas.Magnusson1}@Marist.edu

Abstract

This paper demystifies the advanced computer science topic of blockchain by placing it in the context of course and content development. In presenting suggestions for using blockchain as a tool to teach core computer science concepts, the authors reflect on student-centered, research-based projects spent understanding blockchain and developing an elementary implementation. Their experiences led to several teachable moments applicable to many topics across CS curricula including software design, algorithms and data structures, and distributed computing. The authors discuss many definitions of blockchain filtered through the philosophical lens of essence and accidents, give a precise definition of "essential" blockchain, and provide insight to understanding blockchain by presenting several of its structures and their implementation in the context of those curricular topics.

1 Introduction

As a buzzword, "blockchain" generates gratuitous hyperbole. Regardless of whether or not it really is "the next big thing", and in spite of the fact that

it is generally mystifying and resists explanation, blockchain presents a useful gateway into teaching core concepts in computer science. In this paper, we demystify blockchain by discussing its essence and accidents and making suggestions for using blockchain as a tool to teach core concepts in computer science.

1.1 Background

Having enjoyed (and endured) several student-centered, research-based experiences spent trying to understand blockchain and develop an elementary implementation of it, we encountered several teachable moments. We found that we could provide intuition to aid students' understanding by relating blockchain-specific teachable moments to various areas in traditional computer science curricula. These curricular areas include Software Design (in terms of defining the problem and discovering its essence and accidents), Algorithms and Data Structures (including hashing, linked lists, and trees), Distributed Computing (with issues like cooperating peers and peer discovery), and others (zero-knowledge protocols, graph theory, and more).

1.2 Contributions and Outline

Using the advanced topic of blockchain to teach core concepts in computer science is not well covered in existing literature. This paper – a step towards addressing that deficiency and demystifying blockchain in the process – makes the following contributions:

- In Section 2 we discuss some difficulties inherent in reasoning about blockchain and pose a critical question to motivate its exploration in academic computer science.
- In Section 3 we provide a precise definition of essential blockchain and suggest its use as a teaching tool.
- In Section 4 we dig into blockchain's details by exploring its fundamental nature through a discussion of the historic "essence and accidents" software design approach in this contemporary context. We also show examples of essential blockchain structures suitable for use in teaching several computer science topics.

Finally, Section 5 provides a brief conclusion, links to our blockchain source code, and comments on our future work.

2 Blockchain Difficulties

In reviewing the literature, we find blockchain difficult to define, difficult to characterize, and difficult to classify. It's frustrating. Yet it is within these difficulties that we also find teaching opportunities.

2.1 Difficult to Define

Block-chain has too many definitions. It has been defined as: "a public ledger" [17], "a distributed ledger" [16], "a peer-to-peer distributed ledger technology" [10], "a ledger replication system", an "incorruptible digital ledger of economic transactions" [15], "a linked list that is built with hash pointers" [14], a "terrible [...] database" [7], a "state transition system with a consensus system" [5], and a "distributed database of records or public ledger of all transactions or digital events" [6].

How is one to make sense of all these definitions? How can our students dig through all that noise and arrive at meaningful insight? These are motivating questions.

2.2 Difficult to Characterize

Blockchain's characterizations are as varied at its definitions. Blockchain is "an undeniably ingenious invention" unable to be controlled by a single organization or fail at a single point [2]. This "disruptive technology... opens the door for developing a democratic open and scalable digital economy" as well as presents "tremendous opportunities". The Bitcoin blockchain "solved fundamental problems in a highly sophisticated, original, and practically viable way" [16].

What does all this mean? Can our students dig through these implications and still arrive at meaningful insight? These are also motivating questions.

2.3 Difficult to Classify

Just as its definitions are varied and its characterizations wide-ranging, there are many classifications of many kinds of blockchain. These include "cryptocurrency, private blockchains, permissioned ledgers, distributed tech, or decentralized tech" [1]. The common term "Distributed Ledger Technology" (DLT) further stirs the cauldron of bubbling blockchain brands. Corda, a "distributed ledger" for financial transactions is a DLT [4], but its whitepaper explicitly states that, "there is no block chain" instead calling the structures "notaries" [8]. The Hyperledger Sawtooth documentation, however, claims that an "enterprise distributed ledger" is the same as a blockchain [9].

Who are we to believe? Is there meaningful insight at all? These questions add still more metaphorical fuel to the equally metaphorical (and pedagogical) fire.

All of these difficulties led us to ask a motivating question: "What is blockchain?" But then we distilled it to a more important question:

What is *essential* blockchain?

3 Essential Blockchain

In the context of blockchain history from Bitcoin to Ethereum to Hyperledger, we build our definition of *essential blockchain* by first defining its structural parts.

Definition 1. *Transaction – a container for arbitrary data.*

Definition 2. *Block – a container for one or more transactions.*

Definition 3. *Blockchain – an ordered, append-only container for one or more blocks where the i^{th} block b_i depends on the prior block b_{i-1} to confirm b_i's permanent stasis where $i \geq 1$. (We will revisit this idea in Section 4.2.1.)*

Given the above definitions, and after examining many Bitcoin, Ethereum, and Hyperledger use cases, we find that blockchain is an overloaded term because it's more than a data structure, it's also a consensus network of peer instances of that data structure. Now we have its essence:

Definition 4. *Essential Blockchain – a peer-to-peer network of Blockchain instances cooperating for consensus.*

Let's have a look at some differences between blockchain as a data structure and blockchain as a consensus network.

3.1 Blockchain as a Data Structure

Blockchain is a data structure in that it is an *abstract structure* created to hold information. Satoshi Nakamoto [13], the anonymous inventor(s) of Bitcoin, the first instance of a blockchain, created blocks to contain monetary exchange transactions between entities. Blocks in a general blockchain are not limited to monetary exchanges. The Ethereum blockchain [5], in addition to holding payment transactions, also contains "smart contracts" that enforce various rules about various actions on the blockchain. The Hyperledger Fabric [10] blockchain stores many kinds of data. As one example, the Everledger project [11]

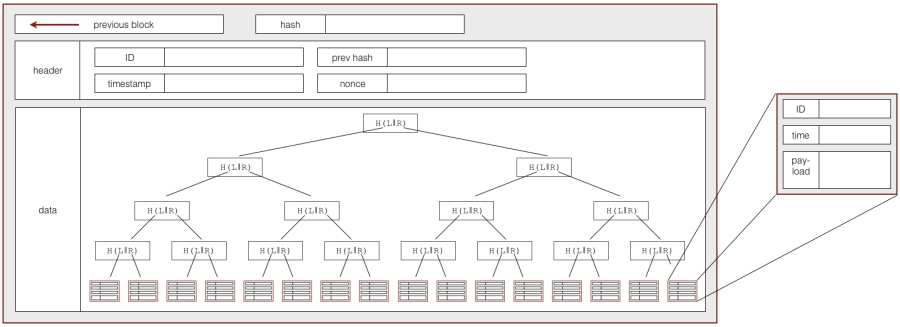

Figure 1: Block Data Structure with Transaction cutaway detail

maintains a Hyperledger blockchain for the Kimberly process of diamond mining, seeking to insure that none of those diamonds are mined unfairly.

The simplest blockchain data structure is the Transaction (Definition 1), an example of which is shown in the detail cutaway in Figure 1. A Block (Definition 2) is more complex, and is shown in the main body of Figure 1. Transactions and Blocks contain both essential and accidental attributes. (More on this in Section 4.1.2.)

3.2 Blockchain as a Consensus Network

Blockchain is a consensus network in that peers in a blockchain network cooperate to agree on which blocks of transactions are valid and which are not. To achieve this, Bitcoin uses a *proof of work*[1] mechanism [13] to ensure that Bitcoin peers who have performed more computational work have more influence on consensus decisions. Ethereum, likewise, adopted proof of work consensus, but aims to transition to *proof of stake*[2], in which those with more currency have more influence over consensus [5]. Hyperledger aims to have "modular, plug-and-play consensus" [10], meaning it is left to the developers of specific Hyperledger instances to choose the consensus method best for their needs.

[1] *Proof of work* systems are based on members of the consensus network performing computationally expensive calculations that are easy to verify to insure the validity of blocks. This is mining. From Nakamoto's Bitcoin paper[13]: "The proof-of-work involves scanning for a value that when hashed . . . begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash."

[2] *Proof of stake* systems are based on certain members of the consensus network (called *validators*) pledging a portion of their assets (their stake) to vouch for the integrity of new blocks. The more they stake, the more blocks they validate, the more assets they have to lose if their blocks are found to be invalid.

Regardless of the consensus method, a blockchain cannot be called a blockchain unless it can also be called a peer-to-peer consensus network.

The duality of blockchain as both a data structure and a consensus network invites its use in many areas across computer science curricula. The next section presents a detailed look at a few of those areas.

4 Blockchain in Computer Science Curricula

Our faculty and student researchers developed a rudimentary blockchain implementation in Java[3], initially as an in-memory data structure, adding file-based persistence later, and eventually building cooperating peers. In doing so, we discovered that the duality of blockchain as both a data structure and a consensus network makes it a useful teaching tool in many computer science areas, including the study of software design.

4.1 Software Design

In his famous essay, "No Silver Bullet" [3], Frederick P. Brooks Jr. addresses some of the difficulties inherent in software development. He points out that software development bridges the chaotic world of "arbitrary complexity, forced without rhyme or reason by many human institutions and systems" with the abstract, yet precise domain software affords. Thus, complexity is software development's proverbial "middle name", and managing that complexity is a primary software development challenge.

Motivated by the desire to manage complexity while developing a custom blockchain implementation, we presented our students a fundamental question: What problem we are trying to solve?

4.1.1 Defining the Problem

Analyzing the landscape of commonly used blockchain "solutions" is daunting and difficult. As noted in Section 2.1, there are many definitions of blockchain. This makes generalization a challenge. The difficulty in characterizing and classifying blockchain systems only makes matters worse. Having found that blockchain is a fundamentally distributed system (via Definition 4 and Section 3.2), we assert that applications of blockchain to use cases that are not fundamentally distributed are wrong. In other words, if we're using blockchain, we should be trying to solve problems where multiple decentralized parties need concurrency, collaboration, cooperation, and fault tolerance. If that is not the

[3]The source code is freely available and we invite you to use it. See Section 5 for details.

case then we should not use blockchain and would be better served by an existing, well-established approach like relational, graph, or even NoSQL databases. We reached and support this conclusion through an exploration of blockchain's essence and accidents.

4.1.2 Essence and Accidents

Brooks takes inspiration from the mother of all sciences, philosophy, to approach complexity. He defines two terms to discuss complexity:

essence Difficulties inherent in the nature of software.

accidents Difficulties that attend its production but are not inherent.

These ideas are not easy to self-appropriate. Instead of conjuring essence and accidents from the ground up, it is often easier to take something familiar and strip away accidents until, like a miner extracting precious metals from muddy rock, essence emerges.

By way of illustration, consider an instance of a coffee cup, in this case from Starbucks: it has a green logo showing the famous mermaid; it has a lid; it is made from a cardboard derivative; and it's filled (one hopes) with wonderfully-fragrant, rich coffee. But our Starbucks cup need not have its logo, nor a lid; those features are *accidents* of the cup. It need not be made from cardboard, nor does it need to be filled with coffee, as those features are also *accidents*. Removing those accidental features (or properties), we are left with a cylindrical object with the capacity to contain and ability to pour liquids. This is the *essence* of a cup... its "cupness".

We take the same approach with blockchain. By embracing the difficulties in defining, characterizing, and classifying blockchain over its short history from Bitcoin to Ethereum to Hyperledger, we can identify its properties and categorize each of them as *essential* or *accidental* in nature. Table 1 shows the results of this process when applied to Transaction and Block structures in the context of Definitions 1, 2, and 3.

We note an interesting meta-question about the Block structure: Is the use of a Merkle tree (discussed in the next section) an accident or is it essential to a block? It's certainly a solution to an essential problem of blockchain, but it is not the only solution. So is it essential or accidental? (That sounds like a good essay question for our students.)

4.2 Algorithms and Data Structures

Before we can implement blockchain, we must first become familiar with its structures and some algorithms that work with them. Understanding these

Structure	Property	Nature
transaction	id	accidental
	timestamp	essential-ish$^{\alpha}$
	payload	essential
block	id	accidental
	hash	essential
	timestamp	accidental-ish$^{\alpha}$
	previous block's hash	essential
	nonce$^{\beta}$	accidental
	Merkle tree	essential?

α The essential or accidental nature of timestamps is unclear. On the one hand, a transaction or a block clearly comes into existence at some particular point in time. On the other hand, it may not be essential to record that.
β A nonce is a value that, when combined with the previous block's hash, will result in a new hash that satisfies the proof of work.

Table 1: Blockchain Essence and Accidents

structures and algorithms, along with their common implementations, helps build fundamental computer science skills necessary to comprehend emerging technologies such as blockchain. We begin with hashing.

4.2.1 Hashing

Primarily a mathematical concept, hashing requires a hash function, H, that maps many possible inputs to a smaller number of outputs. There are many kinds of hash functions. For blockchain, we are particularly interested in *cryptographic* hash functions. A *cryptographic hash function* is one that fulfills these properties [14]:

Collision-resistant It is infeasible to find two values, x and y, such that $x \neq y$, yet $H(x) = H(y)$.

Hiding If a secret value r is randomly chosen from a set of values with equal probability, then given $H(r||x)$ it is infeasible to find x. ($||$ represents concatenation.)

The primary purpose of a cryptographic hash function in blockchain is to create a "fingerprint" of a given piece of data without revealing any information about it. This "fingerprint" output from a hash function is commonly called a *digest* or a *hash*.

Most (if not all) blockchains use hashing to achieve the "chain" part of their nature with *hash pointers*, which we describe in the next section. Hashing also provides an efficient method for "confirming permanent stasis" (as noted in Definition 3) because comparing hash values is a fast, easy, and anonymous[4] way to detect whether or not content has been altered since its initial hash.

4.2.2 Linked Lists

A useful and concrete definition of blockchain as a data structure states that, "A block chain is a linked list that is built with hash pointers." [14]. A *hash pointer* is "a pointer to where data is stored together with a cryptographic hash of the value of that data at some fixed point in time" [14].

Since linked lists are a fundamental topic in virtually all introductory or intermediate computer science courses, enhancing them with hash pointers to support blockchain applications makes for a nice modernization. Combining a cryptographic primitive such as a hash with a basic data structure like a linked list enables students to learn computer science fundamentals and emerging technologies simultaneously. Such teachable moments build understanding of and excitement for computer science. The excitement continues (one hopes) into discussions of $O(n)$ operations like list (or blockchain) traversals and linear searches. Once we've found a target block, we'll need to do a tree traversal to find transactions.

4.2.3 Trees

Instead of storing all transactions in a simple list within a block, Satoshi Nakamoto used a version of a balanced binary tree called a Merkle tree [12, 13]. A Merkle tree is:

binary Nodes have at most two children.

ordered Nodes maintain some ordering appropriate for the data they carry, e.g., lexicographic or numeric.

balanced For every node, the heights of its subtrees differ by at most 1.

leftmost All nodes are placed as far left as possible.

cryptographically hashed The root and all internal nodes contain the hash of the content of their children.

[4] Anonymous in the "zero knowledge" sense that we can confirm whether or not some value has changed without knowing the actual value itself.

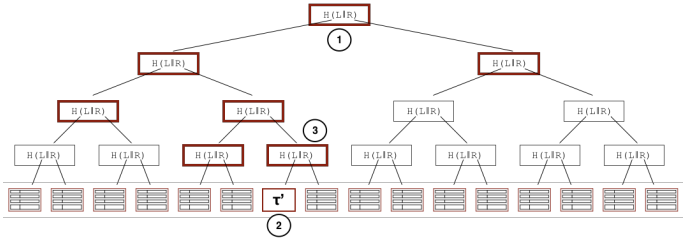

Figure 2: Hash Validation of Transactions in a Merkle Tree

This structure allows all transactions within a block to be validated by a single hash stored at the root of the Merkle tree and called, appropriately enough, the *Merkle root*. The node at #1 in Figure 2 is its Merkle root. Note that it contains a hash of the content of its (left and right) children. They in turn contain a hash of the content of their children, and so on all the way down to the penultimate nodes, which contain a hash of the content of (their children) the leaf nodes, which are the actual transactions. (See Figure 1 for a closer look at a Transaction.) In this manner, if any transaction τ in a block (i.e., any leaf node in a Merkle tree) is altered (e.g., #2 in Figure 2), that node's parent (e.g., #3 in Figure 2) will change as well. This change will propagate up the tree, level by level, until it reaches the root. The root therefore reflects all changes in the tree so it is the only datum a block needs to validate to confirm the permanent stasis of the transactions contained therein.

We constructed a Java class called `MerkleTree` that satisfies the above properties and accepts any data and any hashing algorithm. This exercise might be appropriate in a classroom setting as a bridge from simple binary trees to more complex data structures. It may also be a good way to teach the importance of choosing an appropriate abstract structure for the task at hand and determining which properties it must satisfy **before** implementing it in code, as bugs are easier to fix while planning than while programming.

Constructing a Merkle tree from an arbitrarily-sized collection of transactions is fairly interesting. The details of such an algorithm are out of the scope of this paper, but would be very appropriate in an upper-level algorithms course. As Niklaus Wirth said, algorithms plus data structures equals programs. Tying algorithms to data structures in this manner reinforces the link between the two and their importance in programming fundamentals.

4.3 Distributed Computing

The essentially distributed nature of blockchain (Definition 4) serves as a natural example by which to teach topics of networking, routing, and consensus

algorithms. Our preliminary work used multiple blockchains on one computer running many JVM instances. This makes for a nice simulation of a distributed system, as each socket requires an IP address and a port. It's easy to see how, for example, 127.0.0.1:2007 exchanging messages with 127.0.0.1:2112 is similar enough to 172.217.10.78:42 exchanging messages with 137.254.120.50:71 as not to matter. Once we had that working, we moved with ease to multiple blockchains on multiple cooperating peers (both physical and virtual).

4.3.1 Cooperating Peers

We devised a rudimentary REpresentational State Transfer (REST) Application Programming Interface (API) over Hypertext Transfer Protocol (HTTP) with a handful of endpoints that cooperating peers use to communicate. These REST endpoints allow peers to request (1) the entire blockchain, (2) a specific block in the blockchain, or (3) a specific transaction in the blockchain by using HTTP verbs with the following paths:

1. `GET /blockchain`
2. `GET /blockchain/{block id}`
3. `GET /blockchain/transaction/{transaction id}`

Peers can POST new transactions with the intention that they be accepted as soon as possible. But POSTed transactions do not immediately become part of the blockchain. The peer looking to accept transactions into the blockchain puts them into its *transaction buffer*, a container for pending transactions. A peer mines its buffer when another requests it via the `POST /mine` endpoint. This triggers the *proof of work* mining process. When complete, the previously buffered transactions officially constitute the next block of the blockchain.

4.3.2 Peer Discovery

Peer discovery is accessed through the REST API endpoint `GET /friends`. It provides the IP addresses of known peers on the blockchain network. (A friend is a peer with whom another peer has communication over the network through the REST API.) When starting a peer for the first time, the user must list the IP addresses participating in the blockchain network. This constitutes the initial friend list for the newly-started peer. The network effect of requesting the friends list of its friends allows peers to discover each other.

While discovering its peers, each node builds a graph of the blockchain network by denoting itself as a *from* vertex and its friends as one-hop *to* vertices. This process is repeated for all of the *to* vertices, but this time with them as the *from* vertex. The result of the whole process is a snapshot of the entire

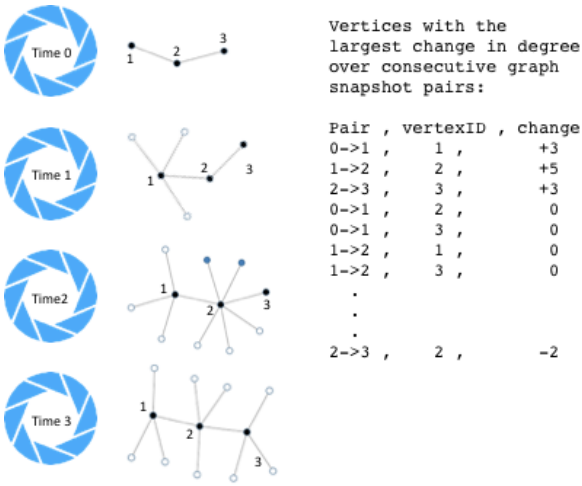

Figure 3: Blockchain network evolution captured through graph snapshots with output from a query about changing influence

network as a graph. The network will change over time (they all do), providing a opportunity to visualize and analyze blockchain evolution with a graph analytics, an example of which is shown in Figure 3.

There is more work to be done, including an implementation of consensus algorithms as required by Definition 4 and supporting smart contracts. These are first up in our future work.

5 Conclusions and Future Work

In studying new technologies like blockchain and creating their own implementation of it, students engage in a holistic learning process bridging theory and practice that promises to be both challenging and rewarding, not to mention demystifying in terms of the new technology itself. We encourage institutions to promote this kind of learning and to teach core concepts through emerging technologies like blockchain. To that end, the source code for our blockchain implementation is available on GitHub at `https://github.com/Marist-Innovation-Lab/blockchain`. All are welcome to take it, use it, and build on it.

We are pursuing two paths in our future work: implementation and research. On the implementation path, we will be programming consensus algo-

rithms and adding support for smart contracts. On the research path, having achieved a network of cooperating blockchain peers and the ability to capture snapshots of their evolution over time, we will be simulating common network attacks (man in the middle, denial of service) and uncommon attacks (Sybil subversion in peer-to-peer networks) so we can test theories of blockchain's security characteristics. Both promise to be fun.

Acknowledgements

The authors wish to thank Daniel Gisolfi and the other students (and faculty and staff) of the Marist/IBM Joint Study and the NSF-funded *SecureCloud* research grant (#1541384). We appreciate their technical and emotional support. Thanks go out as well to the reviewers for their thoughtful comments. Writing papers like this, just as with writing software, relies on critical feedback from external stakeholders.

References

[1] Berkeley Blockchain. What is blockchain?, 10 2016. `https://drive.google.com/file/d/0ByBe1QJVC-EJRUJqVWcyY2VNdlU/view`. Accessed on 2018-11-18.

[2] BlockGeeks. What is blockchain technology? a step-by-step guide for beginners, 2017. `https://blockgeeks.com/guides/what-is-blockchain-technology/`. Accessed on 2018-11-18.

[3] Frederick P. Brooks, Jr. No silver bullet essence and accidents of software engineering. *Computer*, 20(4):10–19, April 1987.

[4] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: An introduction, 2017. `https://docs.corda.net/_static/corda-introductory-whitepaper.pdf`. Accessed on 2018-11-18.

[5] Vitalik Buterin et al. Ethereum white paper: A next-generation smart contract and decentralized application platform, 2013. `https://github.com/ethereum/wiki/wiki/White-Paper`. Accessed on 2018-11-18.

[6] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology: Beyond bitcoin. *Sutardja Center for Entrepreneurship & Technology*, 2:6–10, 2016. `http://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf`. Accessed on 2018-11-18.

[7] Trent McConaghy et al. Bigchaindb 2.0: The blockchain database, 2016. `https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf`. Accessed 2017-8-21.

[8] Mike Hearn. Corda: A distributed ledger, 2016. `https://docs.corda.net/_static/corda-technical-whitepaper.pdf`. Accessed on 2018-11-18.

[9] Hyperledger. Sawtooth introduction, 2016. `https://intelledger.github.io/introduction.html`. Accessed 2018-11-18.

[10] Hyperledger. Hyperledger whitepaper. Google Drive, 2017. No specific authors or publish date were given. `https://docs.google.com/document/d/1Z4M_qwILLRehPbVRUsJ3OF8Iir-gqS-ZYe7W-LE9gnE/`. Accessed on 2018-11-18.

[11] Everledger Ltd. Everledger industry applications, 2016. https://www.everledger.io/industry-applications. Accessed on 2018-11-18.

[12] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1979. AAI8001972. http://www.merkle.com/papers/Thesis1979.pdf. Accessed on 2018-11-18.

[13] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. bitcoin.org/bitcoin.pdf. Accessed on 2018-11-18.

[14] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, Princeton, NJ, USA, 2016.

[15] Don Tapscott and Alex Tapscott. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World.* Brilliance Audio, Brilliance Audio, 2016.

[16] F Tschorsch and B Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016. https://eprint.iacr.org/2015/464.pdf. Accessed on 2018-11-18.

[17] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *Big Data (BigData Congress), 2017 IEEE International Congress on*, pages 557–564, ., 2017. IEEE.