Decentralized Consensus Systems Guide

for

Blockchains Demonstration

Project 9

Version 1.0

Prepared by Vanessa Soares, David Mistretta, Daniel Champagne, and Ian Eichorn

University of Massachusetts Dartmouth

Table of Contents

Chapter 1: What Are Distributed Consensus Systems?

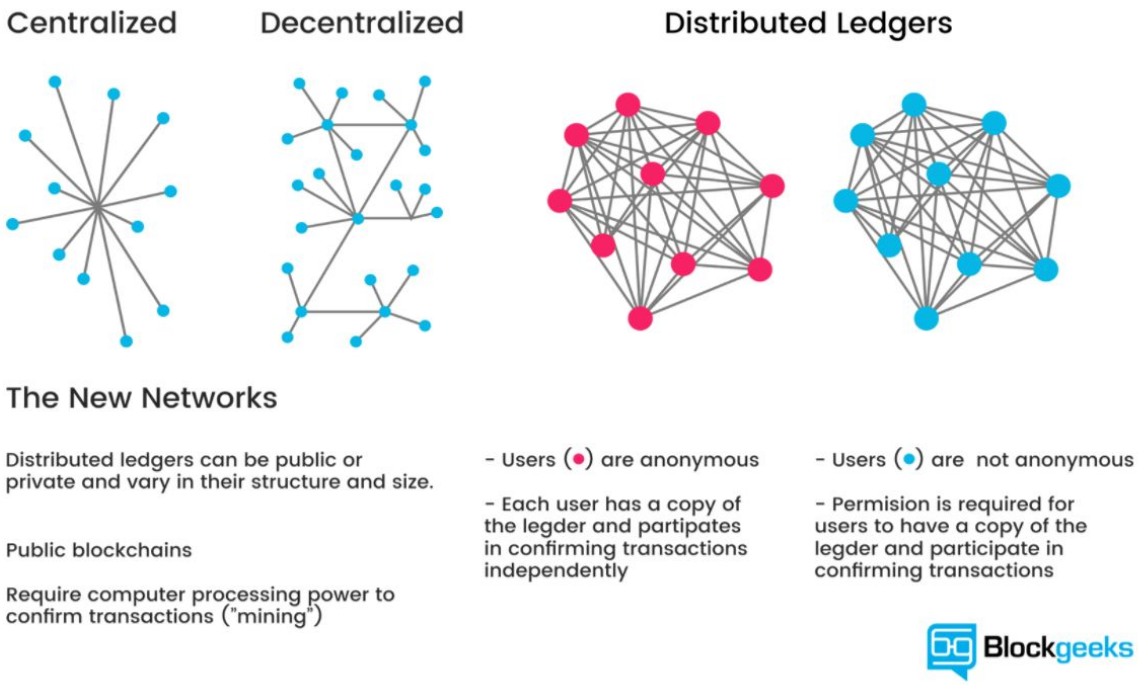
To understand what a distributed consensus system is, we must first understand the consensus problem. The consensus problem is an agreement among processes over a single data value. However, some agents may fail or be unreliable so there must be fail tolerance to accept the agreement. Generally a majority of the processes, being 51% must agree in order to accept the data value. A distributed consensus system follows the protocol of consensus systems, however distributed amongst many agents. Distributed consensus systems are often required to be replicated state machines in Byzantine fault tolerance [27].

Distributed consensus systems, like said before, is usually composed of replicated state machines. To create a distributed consensus system, you need to have a method do come to consensus, and a follow up action that occurs after consensus.

This guide provides in-depth research about various decentralized consensus systems like Blockchain, Hashgraph, Tangle, Hyperledger, and Corda. We proceed to provide a comparison of these technologies as well as tutorials on how to develop your own private blockchain.

Chapter 2: Core Concepts and Definitions

2.1: Centralization and Decentralization



Centralization implies that data is being stored in a single location, similar to data being stored on one database on one server. Decentralization implies that data is being stored in multiple locations across all nodes on the network.

2.2: Web 3.0

Web 3.0 is referred to as a third generation of Internet-based services that comprise what could be called the “Intelligent Web.” These services can include machine learning, semantic web, microformats, natural language search, recommendation agents, and artificial intelligence technologies.
In terms of the Blockchain and other decentralized technologies, distributed computing is a service that will be part of Web 3.0. Web 3.0 is considered to be the new web that has the potential to be the platform for decentralized apps.

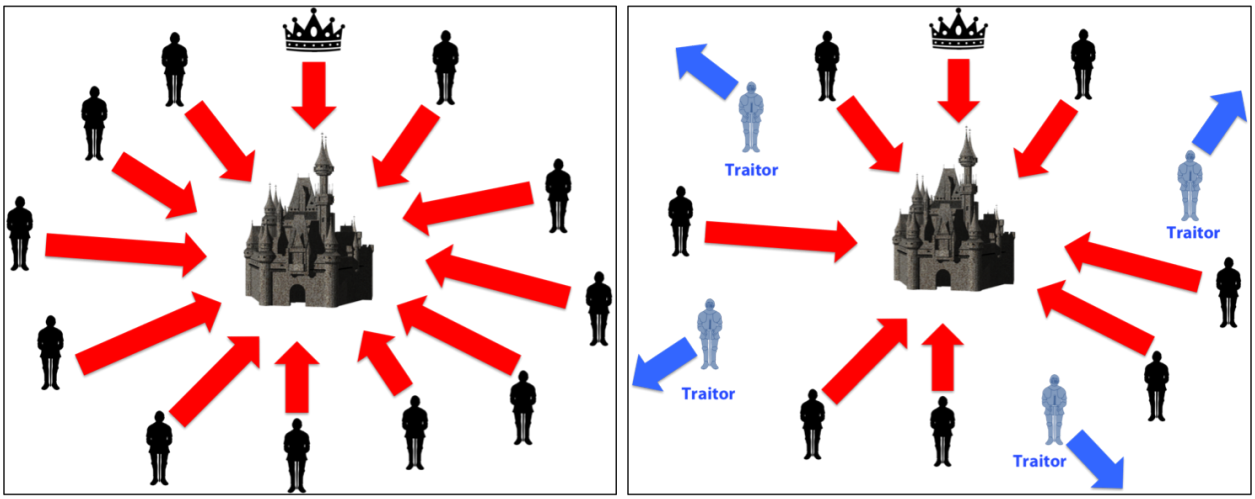
2.3: Byzantine Fault Tolerance

The original definition of Byzantine can be explained as the assumption that attackers will not collude or that communication is weakly asynchronous [27]. In its strong sense, Byzantine can be defined as up to just under ⅓ of the members can be attackers, they can collude, and they have the ability to delete/delay messages between honest members without bounds on the message delays [27]. Attackers are able to control the network to delete/delay messages [27]. If an honest member repeatedly sends messages to another member, then the attackers must eventually allow one through [27]. Attackers cannot undetectably modify messages due to the assumption that secure digital signatures exist [27]. It is also assumed that secure hash functions exist [27].

A distributed consensus system should guarantee a Byzantine fault tolerance in order to be determined secure, safe, and trustworthy. This distributed consensus system is often required to be composed of replicated state machines. The Byzantine fault tolerance aims to ensure that distributed consensus remains unaffected in the presence of Byzantine failures, that is, an incorrect consensus from a state machine within the distributed consensus system.

The Byzantine Generals problem

Distributed consensus problems can be explained using Byzantine Generals as an example.



Coordinated Attack Leading to Victory **Uncoordinated Attack Leading to Defeat**

Source: https://medium.com/@DebrajG/how-the-byzantine-general-sacked-the-castle-a-look-into-blockchain-370fe637502c

The Byzantine Generals problem

Byzantine Generals Problem. A commanding general must send an order to his $n - 1$ lieutenant generals such that

IC1. All loyal lieutenants obey the same order.
IC2. If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.

Source:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.9525&rep=rep1&type=pdf

In this problem let's consider a castle surrounded by several armies, each controlled by a general. There is one general who is the lead general (crown above). The only way for the attack on the castle is to be victorious, is if there is a simultaneous attack all at once. However, command is difficult due to there being multiple generals dispersed around the castle. The generals must agree upon a time to attack and they do so by sending messages. However, some generals may be traitors and relay incorrect attack times. In order to solve this problem we move from a general / messenger paradigm to a commander / lieutenant paradigm. In this paradigm, in order to achieve consensus, every general and lieutenant must agree on the same decision (either attack or retreat).

This problem is a metaphor to a problem that plagues many distributed consensus networks. When a group is trying to make a collective decision about how it will act, there is a vulnerability in aspect to certain members of the group may be traitors and send mixed messages about their preferences. This can cause problems for the group's ability to coordinate its actions efficiently.

Distributed Consensus Systems and Byzantine Fault tolerance

What does byzantine have to do with distributed consensus systems, such as blockchains? In order for a blockchain to be reliable, it must come up with a solution to solve the Byzantine Generals problem by providing a Byzantine Fault tolerance. Blockchains are decentralized; there is no central authority, yet it can verify legitimate transactions, which is what makes it a very powerful tool. If that verification was compromised, the blockchain would be less valuable. Because of this, byzantine faults and byzantine failures are a particular challenge that blockchains, and other decentralized consensus systems must overcome.

The blockchains solution to providing a reliable Byzantine Fault tolerance is their consensus protocols, namely Proof of work. Proof of work is a consensus protocol that requires nodes that give consensus to perform a hashing algorithm. This hashing algorithm can only be computed from true transactions, and 51% of the nodes must agree upon the next transaction. This works because it takes a fair amount of computing power to generate the correct hash, and in sufficiently large blockchains it would require an enormous amount of computer power to disrupt the truth of that blockchain. There are other consensus algorithms that attempt to tackle the challenge of providing a Byzantine Fault tolerance, namely Proof Of Stake, Virtual Voting(hashgraph), and tangle consensus.

Byzantine Fault tolerance is an important concept to understand when working with distributed consensus systems because all distributed consensus systems aim to provide a guaranteed Byzantine Fault tolerance, and if they can do so then the users of the DC systems can be comfortable in knowing their digital assets are secure.

2.4: Distributed Consensus Systems

I. Decentralization

Decentralized technology lets us store data on a network that can be accessed over the Internet. Through the decentralized technology, the owner of the data has direct control through their private key which is directly linked to the data [4].

Anything that happens on the blockchain network is a function of the network as a whole. Therefore, the blockchain is managed by all nodes on its network instead of a single entity. Decentralization is defined as the networking operating on a user-to-user/peer-to-peer basis [1].

Blockchains are defined as public ledgers of all transactions and communications that have ever been executed on the network [3]. They are comprised of a network of nodes that get a copy of the blockchain that is automatically downloaded once they join the blockchain network [1]. A block is the “current” part of the blockchain that records some or all of the recent communications. Once the communication or transaction is completed, the block goes into the blockchain as a permanent database. Once a block is completed, a new block is generated. These blocks are linked together in the form of a chain in linear chronological order with every block containing a hash of its previous block [3].

Since blockchains store blocks of data that are identical across the network, blockchains cannot be controlled by a single entity and they have no single point of failure. The blockchain network operates on a consensus mechanism that automatically checks in with itself every 10 minutes and reconciles every transaction that happens in these 10 minute intervals [1].

Reasons for Decentralization [4].

Empowered Users: allows users to keep control of their information and transactions.
Fault Tolerance: less likely to fail accidentally since they rely on many separate components that are not likely to fail.
Durability and Attack Resistance: Since the blockchain doesn’t have a central point of control and can better survive malicious attacks, decentralized systems are more “expensive” to attack and destroy or manipulate.
Free from Scams: it is much more difficult for users to cause harm to other users by scamming.
Removing Third-Party Risks: enables users to make an exchange without a third party as an intermediary which eliminates risks.
Higher Transaction Rate: transactions can reduce times to minutes and can be processed anytime compared to how transactions are done through banks now.
Lower Transaction Costs: done through eliminating 3rd party intermediaries and overhead costs for exchanging assets.
Transparency: changes to public blockchains are viewable by all parties and all transactions are immutable which means they cannot be altered or deleted.
Authenticity: the blockchain is complete, consistent, timely, accurate, and widely available.

II. Consensus

For a transaction to be valid, all participants must agree on the validity of the Transaction [5] In order to create a secure consensus protocol, the protocol must be fault tolerant.

III. Provenance

Participants know where the asset came from and how its ownership changed over time [5]

IV. Immutability

No participant can tamper with a transaction once it has been recorded to the blockchain. If a transaction was made by mistake and a change needs to be made, a new transaction must be used to reverse the error. Both transactions remain on the blockchain [5].

V. Finality

A single, shared ledger provides one place to go to determine who owns the asset or if a transaction has been completed [5]

VI. Transparency

The data recorded in a blockchain is transparent to each node as well as on updating its data. This allows for more trust in the data being stored [22].

VII. Open Source

The blockchain system is open to everyone. Records can be checked publicly and people can use the blockchain technology to create any application they want [22].

VIII. Autonomy

Due to the consensus mechanism, every node on the blockchain network can transfer and update data in a safe manner [22]. No one can intervene in the ability of adding data by bypassing the consensus [22].

IX. Anonymity

Blockchain technology solves the trust problem between nodes. Data transfers and transactions can be anonymous and only need to know the person’s blockchain address [22].

Chapter 3: Blockchain

3.1: What is Blockchain?

Definition of a Blockchain

A blockchain is a shared distributed ledger that allows for the recording of transactions and asset tracking on a network. An asset can be anything from a house, a car, land, or even patents, copyrights, or branding. Anything of value can be tracked and traded on a blockchain network. This reduces the risk and cuts costs for anyone involved in the tracking process. In terms of Bitcoin and the blockchain, think of the blockchain as the operating system and Bitcoin as the application that is running on the operating system. Bitcoin is only the first use case for the Blockchain [5].

Blockchains are a new and developing technology that may further protect against data breaches. They are defined as incorruptible digital ledgers of economic transactions that have the ability to be programmed to record everything of value. Blockchains have created the backbone of a new internet system by allowing the distribution of digital information but not allowing that digital information to be copied. [1].

They were originally devised for Bitcoin, the digital currency, but the technology community is finding other potential uses for them. Blockchains allow digital information to be distributed but not copied. Information shared on a blockchain exists as a shared and continually updated database. Information is public, easily verifiable, and is not stored in one single location. Therefore, it makes it extremely difficult for hackers to corrupt the data since no centralized version of this information exists. Since blockchains store blocks of information that are identical across the network, the blockchain can't be controlled by one single entity and it has no single point of failure. Therefore, data breaches may be prevented by potentially storing user information or any other confidential information on a blockchain. A blockchain can be viewed as a spreadsheet that is duplicated across a network of computers that's designed to regularly update this spreadsheet [1].

A verified piece of data forms a block that has to be added to the chain in order for a blockchain to be created. Blockchain users have to use their respective keys and computing systems to run algorithms that solve complex math problems. Once a problem is solved, the block is added to the chain and the data it contains exists on the network forever (can't be altered or removed). In order to update data, the owner has to add a new block on top of the previous block which creates a specific chain of code. The entire chain across the network changes accordingly. Every single alteration is tracked and no data is lost/deleted because users can always look at previous version of the block to identify what's different in the latest version [11].

A blockchain is a linked-list-like structure that acts as an immutable ledger. It is distributed among all members of a decentralized network. This ledger tracks and records any arbitrary piece of data in such a way that it cannot be changed. The distributed nature of the blockchain implies that multiple copies exist. The distributed nature is also what lends itself to being immutable. If there are multiple versions of the blockchain on the network, the official ledger is the version that is held by the majority of the peers on the network [18].

Components of a Blockchain

Hashing and Merkle Trees

A hash function is a tool that converts data of variable length to a unique fixed size digest. Hash functions exhibit pre-image resistance, second pre-image resistance, and collision resistance. Blockchain implementations typically use the SHA-256 hash [18].

A merkle tree is also known as a hash tree and it is a balanced binary search tree. Any parent node is the hash of the concatenation of its children. The tree root is the tallest node and is known as the Merkle Root. A condensed version of all the data stored in the block is the merkle tree. Hashing and merkle trees are the foundation of the immutable ledger [18].

5 Key Responsibilities of the Network Node

- Discover fellow peers and compute the genesis block
- Wait for data from a user node to signal the start of the race to compute/mine a block
- Repeatedly randomize or increment the nonce until the block header's hash meets the blockchain's criteria (e.g. a certain number of leading 0's)
- Once a solution is found, share it with the other peers on the network.
- Upon solving their own block, or receiving a block solved by a peer, each node must verify that the block fits onto the blockchain. This is important since nodes do not trust their peers on the network.

The above network structure gives Blockchain its distributed and decentralized properties [18].

Types of Blockchains [24]

Public

Public blockchains are based on the Proof of Work (PoW) consensus algorithm. They are open sourced and not permissioned. Anyone can participate on this blockchain without additional permission. Examples of public blockchains include Bitcoin and Ethereum . With public blockchains, anyone can download the code to the blockchain and start running a public node on their own device, validate network transactions, and participate in the consensus process. Anyone can send transactions through a public blockchain network and expect to see them added to the blockchain as long as the transactions are valid [24]. Public block explorers can be used by anyone to read transactions on the network [24].

Federated/Consortium

Federated blockchains operate under the leadership of a group and do not allow any person with access to the Internet to participate in the transaction verification process [24]. Federated blockchains are faster and provide more privacy in transactions [24]. Consortium blockchains are mainly used in the banking sector. The consensus mechanisms of these blockchains are controlled by a pre-selected set of nodes (e.g. a consortium of 15 financial institutions and each operates a node, 10 must sign every block for it to be valid) [24]. The right to read the blockchain can either be public or restricted to the participants on the network [24]. Examples of these blockchains include R3 and Corda.

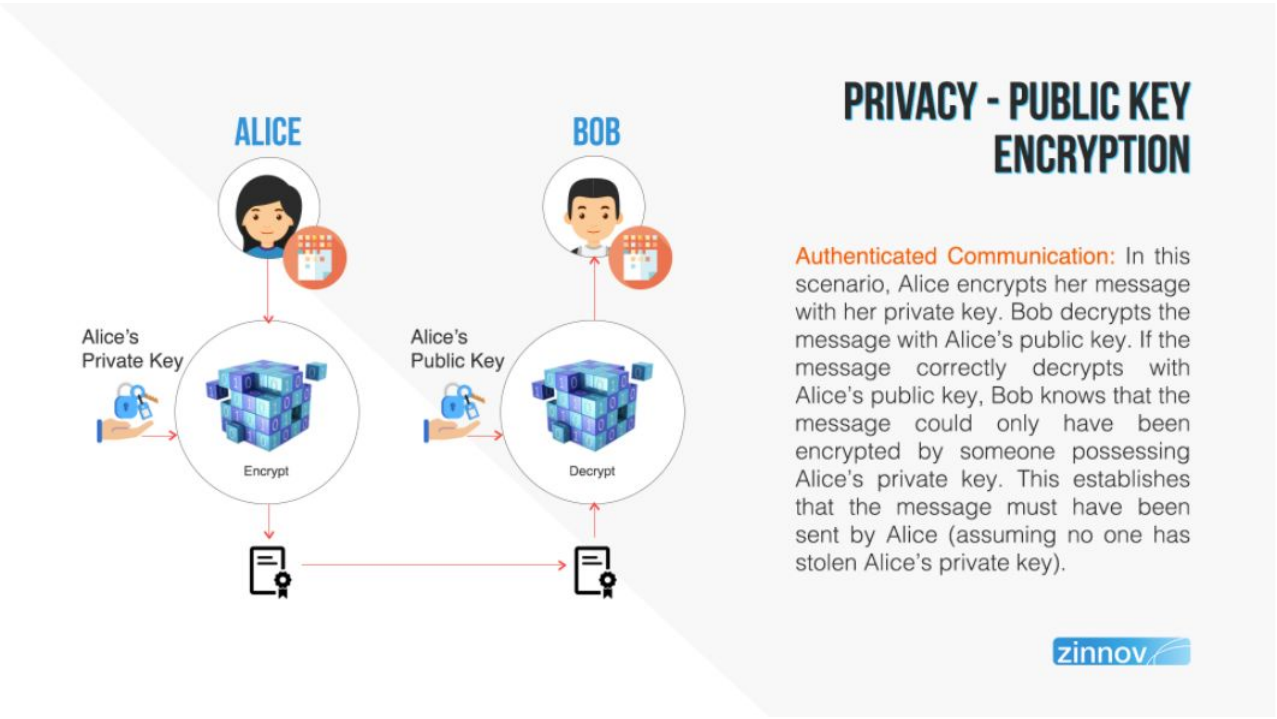
Private

With private blockchains, write permissions are centralized to one organization. Read permissions can either be public or restricted to a certain extent. Some applications include database management and auditing which are internal to a single company [24]. Private blockchains are a method of taking advantage of blockchain technology by setting up groups who can verify transactions internally. However, this puts you at risk of security breaches as with a centralized system, as opposed to public blockchains secured by incentive mechanisms [24]. However, private blockchains have their use case, especially with scalability and state compliance of data privacy rules [24]. Examples of private blockchains include MONAX and Multichain.

Byzantine Fault Tolerance

Systems such as Blockchain cannot be Byzantine since a member on the network never knows for sure when a consensus has been reached [27]. They only have a probability of confidence that rises over time.

Encryption Basics for Understanding the Blockchain [23]



3.2: Applications and Use Cases

The concepts of parallel blockchains and sidechains allow for tradeoffs and improved scalability using independent blockchains which allow for further innovation [3]. Various benefits of blockchains include decentralization, recording and validating every transaction which provides security and reliability, authorization of transactions by miners which make the transactions immutable and prevent hacking threats, and they discard the need for third-party or central authority for P2P transactions [3]. Various companies and institutions are studying blockchains to apply them to various areas such as money transfers, risk management, smart bonds, and cryptocurrencies.

Supply Chain Management

The immutability of the blockchain makes it suitable for tracking goods as they move and change locations in the supply chain [12]. Entries on a blockchain can be used to queue events with a supply chain [12]. Blockchain provides a new way to organize tracking data and putting it to use [12].

Healthcare

Age, gender, and basic medical history data would all be suitable information to be stored on a blockchain in the healthcare industry. None of this information should be able to identify a patient which allows it to be stored on a shared blockchain accessible by numerous individuals without any concerns for privacy [12]. Devices will be able to store data on a healthcare blockchain that can append data to a person's medical record [12].

Real Estate

The average person sells their house every 5-7 years, and the average person will move almost 12 times in their lifetime [12]. This information could be very useful to store on a blockchain for the real estate industry as it could expedite home sales by quickly verifying finances, reduce fraud as a result of blockchain's encryption, and offer transparency through the selling and purchasing process [12].

Media

Writers and content creators can spread their works on a blockchain and receive immediate payment [12]. Comcast's advanced advertising group developed a new technology to let companies may ad buys on broadcast and over-the-top TV through the blockchain [12].

Energy

Blockchains could be used to execute energy supply transactions and provide the basis for metering, billing, and clearing [12]. Other potential applications include ownership documentation, management of assets, guarantees of origin, emission allowances, and renewable energy certificates [12].

Record Management

National, state, and local governments must maintain records of all individuals such as birth and death dates, marital status, and property transfers [12]. Some of these records only exist in paper form. At times, citizens have to physically go to their local office to make changes which is time-consuming. Blockchain technology can simplify the record keeping process and make it far more secure [12].

Identity Management

Blockchains offer enhanced methods for managing identities as well as digitizing personal documents. Developing digital identity standards is a highly complex process. A universal online personal identity solution requires the cooperation of private entities and government.

Currently, Illinois is experimenting with blockchain technology to replace birth certificates as a form of identification [8]. This would give citizens more control over their data as well as reassurance that their information is more secure than before [8]. The Illinois Blockchain Initiative partnered with identity solutions firm Evernym to create an online ledger that is only accessible to the ID owner and additional granted individuals [8]. This is similar to how the technology could be used to share information between hospitals [8].

Voting

Blockchains are fault tolerant, do not allow someone to change the past records, hack the present records, or alter access to the system [12]. Every node on the blockchain with access

can see the same results and every vote on the blockchain can be irrefutably traced to its source without sacrificing voter anonymity [12]. End-to-end verifiable voting systems will give the voter the ability to verify that their vote was correctly recorded and counted [12]. Blockchains would especially be useful in the event that a ballot was missing, in transit, or modified and it can be detected by the voter and caught before the election is over [12].

Cybersecurity

Cisco believes that the blockchain can play a role in managing networks built on switches, firewalls, and other appliances [7]. Cisco has joined the Hyperledger Project, an open source initiative that is trying to develop cross-industry blockchain technologies. Blockchains can be used to manage switches, routers, firewalls, and internet of things gateways [7]. The blockchain technology is promising where network management is done through a centralized controller [7]. Blockchains could make management tasks possible across multiple vendors by keeping record of an appliance's current state and configuration history which ensures that changes made to a device don't cause a network outage [7].

Land Title Registration

Blockchains make record keeping far more efficient and can be applied to recording property titles as well as registered cars.

Connected Cars

Toyota is partnering with MIT to utilize blockchain technology for connected cars. This would allow an unspecified number of people to access data as well as manage payment between cars and charging spots [9]. Blockchains would also allow for accurately preserving vehicle mileage and maintenance records. It can link headquarters to vehicles to manage times and fees for car sharing [9]. This information is key to measuring the value of a vehicle and calculating insurance rates [9]. Blockchains would also allow car-sharing companies to locate any vehicle at any time and determine who to charge for any services [9].

Center for Disease Control

Blockchain technology could allow public health workers respond better to a crisis [10]. The CDC, state and local departments, and other organizations need to routinely share public health data so they can control the spread of various infectious diseases [10]. Blockchains can especially assist with public health surveillance by efficiently managing data during a crisis or allow for better tracking of opioid abuse [10]. Moving such important data between peers in a secure, compliant, and transparent manner as quickly as possible is key to the business model [10].

For example, let's evaluate the scenario of a pandemic. The CDC has an existing mobile app used by local health workers to log information about patients and determine medications that should be given to various patients. However, personally identifiable information cannot be

stored on the cloud. Storing it in an approved manner takes more time. By using the blockchain, storing and sharing data can be done much faster while complying with security and privacy laws [10].

Preventing DDoS Attacks

Hackers use several techniques to instigate an attack such as sending junk requests to a website, increasing traffic until the site can't keep up with requests [11]. The attack goes on until the site gets overwhelmed with requests and crashes [11]. The main difficulty in preventing DDoS is with the existing DNS (Domain Name System) [11]. DNS is a partially decentralized one-to-one mapping of IP addresses to domain names and works like an internet phone book [11].

Blockchains would allow for DNS to be fully decentralized and distribute the contents to a large number of nodes which makes it nearly impossible for hackers to attack a network [11]. Domain editing rights would only be granted to domain owners and no one else could make changes [11]. This reduces the risk of data being accessed/changed by unauthorized parties [11]. By using blockchains to protect data, a system can ensure that it is invulnerable to hackers unless every node on the network is wiped clean simultaneously [11]. If current DNS would operate on the blockchain, users would still be able to register domain names but only authorized users could make changes to their domains [11]. Data would be stored on various nodes and every user on the network would have a copy. It would be virtually impossible to hack/destroy it completely [11].

Safeguarding Data

Blockchains allow for the distribution of every piece of data to nodes throughout the system [11]. If someone tries to alter the data, the system analyzes the whole chain and compares them to the meta-packet and excludes any that don't match up [11]. The only way to wipe out the entire blockchain is to destroy every node on the network. Even if just one node remains on the system, the whole system can be restored despite all the other nodes being compromised [11].

Preventing Fraud and Data Theft

Blockchains prevent potential fraud and decrease the chance of data being stolen/compromised [11]. In order to destroy/corrupt a blockchain, a hacker would have to destroy the data stored on every user's computer in the blockchain network [11]. This can sometimes be millions of computers. It is nearly impossible for a hacker to simultaneously bring down an entire blockchain network [11]. Even if the Blockchain were impacted by an attack, nodes not affected by the attack would continue to run as normal on the network [11].
Bigger blockchain networks with more users have an infinitely lower risk of getting attacked by hackers since the complexity is higher to penetrate such a network [11].

Decentralized Storage, Recordkeeping, and Peer-to-Peer Sharing

Users are able to store all of the data in their network on their computer if they choose to do so. This results in earning money for renting “extra” storage space and they can ensure that the chain won’t collapse. If a hacker tries to tamper with a block, the whole system analyzes every block of data to find the one that differs from the rest. If the system finds this kind of block, it simply excludes it from the chain and identifies it as false. There is no central authority or storage location. Every user on the network stores some/all of the blockchain and everyone is responsible for verifying data to make sure false data can’t be changed and existing data can’t be removed [11].
A real life use case of using blockchain technology for peer to peer sharing is the blockchain TRON. TRON is a blockchain based decentralized protocol that aims to create a free content entertainment system using blockchain and distributed storage technology. The protocol allows each user to freely publish, store and own data, and in the decentralized autonomous form, decides the distribution, subscription, and push of contents and enables content creators by releasing, circulating and dealing with digital assets, thus forming a decentralized content entertainment ecosystem.
Another real life use case of peer to peer sharing on the blockchain is exemplified through Cindicator (CND for short). CND is a collective consciousness enhanced by artificial intelligence. It does this by polling all of its users and analyzing their pattern

3.3: Blockchain and Security

Blockchains eliminate the risk with data being held in a central location since they store data across its network [1]. Computer hackers will no longer be able to exploit central points of vulnerability when networks implement the blockchain for data [1]. Blockchain uses encryption for security with public and private keys as a basis. A public key is a long and randomly-generated string of numbers and is used as the user's’ address on the blockchain [1]. Any actions done by the user on that network get recorded as belonging to that specific public key [1]. The private key is used as a “password” that gives its owner access to assets on the network [1].

Why are blockchains considered secure? For a software to be secure, it must have Authentication, Authorization, and Accounting protocols. The information must also follow the CIA triad by having various degrees of Integrity, Confidentiality, and Availability.

Authentication
To be a useful node you must hold a true version of the blockchain on that node. In order to utilize most blockchains you must have an account authenticated with identification information.

Authorization
More than half of the nodes must authorize a change.

Accounting
All updates are stored on the blockchain.

CIA Triad

Confidentiality
Full encryption of blockchain data ensures that it won't be accessed by unauthorized parties while this data is in transit. End-to-End Encryption ensures that only those who have authorization to access the encrypted data i.e. through their private key can decrypt and see the data [2].

Integrity
The data is verified to be true because it was authorized by a majority of nodes.

Availability
When it comes to blockchain information availability differs based on the design of each one. But it has been shown to be possible to have no fee, instantaneous transfer of information.

Attacks on the Blockchain Network

Consensus Attack
A consensus attack occurs when a majority of the nodes on the network work together to commit a Shallow Block Attack to commit invalid data to the Blockchain. This attack takes advantage of the fact that the official ledger is the one present at a majority of the network's peers [18].

Consensus Attack Countermeasures
A high number of peers are the best method of countering a possible consensus attack since it is more costly to attack a network with a large number of peers. It would also be beneficial to keep track of who is allowed to become a miner on the network [18].

Shallow Block Attack
A shallow block attack occurs when a single malicious node computes blocks containing malicious data that fit into specific locations on the chain, and then swaps them with actual blocks. This attack is called "shallow" because it typically targets blocks at the top of the chain since too much hashing power would be needed to compute a block that is deep in the Blockchain [18].

Shallow Block Attack Counter

There isn't much you can do to counter shallow block attacks. However, in order for these to be effective, they must be coupled with a consensus attack and the counters for that attack are listed above [18].

3.4: Smart Contracts and Ethereum

Ethereum Virtual Machine

Smart Contracts are accounts holding objects on the Ethereum blockchain [6]. A smart contract is a piece of code that is executed on all nodes across the network. All nodes must execute the code and create the same result, verifying the code was executed as it should be. But what is the motivation for nodes to perform this code? WHat benefit do they get from following the rules of the smart contract. Besides the inherent functionality of the smart contract, what else is a reason a random node would want to execute a smart contract? That's when the concept of gas comes in. If you are a developer, and you create a smart contract, you have to declare the gas price you'll pay computers to execute the code. The gas price is determined by the size of the smart contract. This encourages smart and efficient coding and also encourages miners to continue running nodes.

ERC20 Tokens (Ethereum Request for Comments)

ERC20 Tokens, or Ethereum Standard tokens, are used within ethereum smart contracts. ERC20 defines a common list of rules that an ERC20 token must implement, which allows developers to implement coins in a general manner with the Ethereum ecosystem. ERC tokens are one of the most used resources in smart application development.

3.5: Performance and Maintenance

Computer Maintenance

To ensure blockchain performance, the blockchain must have multiple nodes constantly verifying the blockchain at all times. These nodes are called miners. Depending on which blockchain platform you use, the specs required to run a full node and the speed of consensus will differ. For the Ethereum blockchain. On go ethereum, you can sync to the blockchain at three different levels.

- "Full" Sync: Gets the block headers, the block bodies, and validates every element from genesis block.
- Fast Sync: The goal of fast sync is to utilize bandwidth rather than processing power. Instead of processing the entire blockchain like a full sync, it downloads the transaction receipts and pulls the entire recent state database. [36]
    - The fast sync can only be done on a nodes initial sync.

    - The fast sync is susceptible to a sybil attack; an attacker could isolate a single node and feed it a false truth making it believe it is in the correct state. Because fast sync skips the transaction processing, it is easier for an attacker to launch a sybil attack against a node in a fast sync. This is another reason fast sync is allowed only on a nodes initial sync.
- Light Sync: Gets only the current state. To verify elements, it needs to ask to full (archive) nodes for the corresponding tree leaves.

Comparison of Full vs Sync from [36]

“To benchmark the performance of the new algorithm, four separate tests were run: full syncing from scratch on Frontier and Olympic, using both the classical sync as well as the new sync mechanism. In all scenarios there were two nodes running on a single machine: a seed node featuring a fully synced database, and a leech node with only the genesis block pulling the data. In all test scenarios the seed node had a fast-synced database (smaller, less disk contention) and both nodes were given 1GB database cache (--cache=1024).

The machine running the tests was a Zenbook Pro, Core i7 4720HQ, 12GB RAM, 256GB m.2 SSD, Ubuntu 15.04.” [36]

Dataset (blocks, states)	Normal sync (time, db)	Fast sync (time, db)
Frontier, 357677 blocks, 42.4K states	12:21 mins, 1.6 GB	2:49 mins, 235.2 MB
Olympic, 837869 blocks, 10.2M states	4:07:55 hours, 21 GB	31:32 mins, 3.8 GB

Figure 3.5.1: Nodes run in Normal(full) sync compared to fast sync.

Blockchain Performance

If two blocks are mined at the same time, then the chain will fork until the community can decide on which branch to extend [27]. If the blocks are added slowly, then the community can add to the longer branch and stop growth on the other branch which can be pruned and discarded since it is stale [27]. This is inefficient since some blocks can be mined but discarded anyway. This also means that its necessary to slow down the speed of mining blocks so the community can jointly prune them faster than new branches sprout which is the purpose of proof-of-work: by requiring that miners solve difficult computation problems to mine a block, it can guarantee that the entire network will have long enough delays between mining events [27].

Proof-of-work blockchains require that electricity be wasted on extra computations and even expensive mining machines may need to be purchased [27].

3.6: Capabilities and Limitations

The nature of blockchains means that a large network of users is required to ensure that it is robust. [16] A blockchain with only a few users puts it at greater risk of failure and security risks such as a 51% attack. For example, a blockchain with 100 users versus one with 1000 is more likely to experience failures if users begin to disappear, or change the blockchains data . The large user base ensures the robustness of the blockchain, as authenticity is guaranteed if a very large amount of users can confirm it.

Capabilities

Transaction costs, network speed, transactions per second

Limitations

Another factor to think about when considering blockchain limitations is the size of the blockchain. As each new block is added to the blockchain, each node must reflect the addition. This becomes problematic when the blockchain itself reaches unmanageable sizes. Each block on the ethereum blockchain has an average size of 1.225 KB. Over the course of a month the ethereum blockchain grows approximately 187 MB. This makes it difficult for an average node run by an independent entity to continuously sync and maintain the blockchain.

Sharding

All blockchains face another scalability issue; they can not process more transactions than a single node can. Ethereum's blockchain can only process 7 - 15 transactions a second. As blockchains are adopted and become more mainstream, this number of verified transactions per second becomes cumbersome and the average time for a transaction to be verified increases. One way to combat this inadequacy is a method called sharding. A shard is a partition of an ethereum's state. An ethereum's state is a set of information that represents the current state of the blockchain. Currently, on most blockchains, each node must stores all states and processes all transactions of the blockchain In a sharded system, each node would only process a certain portion of the states and only a select amount of transactions. Granted there are enough nodes on the system, a blockchain could be verified by verifying a shard rather than a full block.This would decrease the amount of time needed to update the blockchain because it would cost less computing power, thus it can be done quicker. It also retains its innate security.

3.7: Alternatives to Ethereum

Eris

HydraChain
MultiChain
OpenChain
BigChainDB
Chain Core
Credits
Domus Tower Blockchain
Elements Blockchain Platform
Hyperledger Iroha
Hyperledger Sawtooth Lake
Quorum
Stellar
Symbiont Assembly

Chapter 4: Hashgraph

4.1: What is Hashgraph?

Hashgraph is a data structure and consensus algorithm that is fast, secure, and fair [25]. It grows upward over time and every participant has a copy of the hashgraph in its memory [26]. Each member (full node) on the Hashgraph starts by creating an event which is a small data structure that is stored in memory [26]. The goal of the Hashgraph consensus algorithm is for all members of the community to come to an agreement on the order of the events and the order of the transactions in the events and to agree on a timestamp for each event and transaction [26]. Using this system, it should be relatively difficult for attackers to prevent a consensus on the Hashgraph, to force different members to come to a different consensus, or to unfairly influence the order and timestamps that are agreed upon amongst the members of the Hashgraph [26].

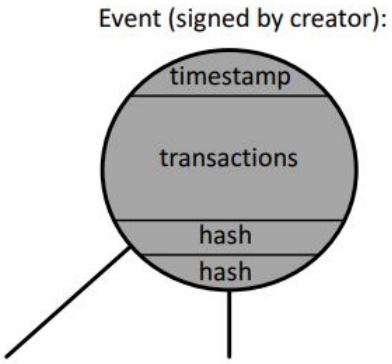
Core Concepts

Transactions - any member can create a signed transaction at any time. All members get a copy of it, and the community reaches Byzantine agreement on the order of those transactions.
• Fairness - it should be difficult for a small group of attackers to unfairly influence the order of transactions that is chosen as the consensus.
• Gossip - information spreads by each member repeatedly choosing another member at random, and telling them all they know
• Hashgraph - a data structure that records who gossiped to whom, and in what order.
• Gossip about gossip - the hashgraph is spread through the gossip protocol. The information being gossiped is the history of the gossip itself, so it is “gossip about gossip”. This uses very little bandwidth overhead beyond simply gossiping the transactions alone.
• Virtual voting - every member has a copy of the hashgraph, so Alice can calculate what vote Bob would have sent her, if they had been running a traditional Byzantine agreement protocol that involved sending votes. So Bob doesn’t need to actually her the vote. Every member can reach Byzantine agreement on any number of decisions, without a single vote ever being sent. The hashgraph alone is sufficient. So zero bandwidth is used, beyond simply gossiping the hashgraph.
• Famous witnesses - The community could put a list of n transactions into order by running separate Byzantine agreement protocols on O(n log n) different yes/no questions of the form “did event x come before event y?” A much faster approach is to pick just a few events (vertices in the hashgraph), to be called witnesses, and define a witness to be famous if the hashgraph shows that most members received it fairly soon after it was created. Then it’s sufficient to run the Byzantine agreement protocol only for witnesses, deciding for each witness the single

question “is this witness famous?” Once Byzantine agreement is reached on the exact set of famous witnesses, it is easy to derive from the hashgraph a fair total order for all events.
• Strongly seeing - given any two vertices x and y in the hashgraph, it can be immediately calculated whether x can strongly see y, which is defined to be true if they are connected by multiple directed paths passing through enough members. This concept allows the key lemma to be proved: that if Alice and Bob are both able to calculate Carol’s virtual vote on a given question, then Alice and Bob get the same answer. That lemma forms the foundation for the rest of the mathematical proof of Byzantine agreement with probability one.

Gossip about Gossip

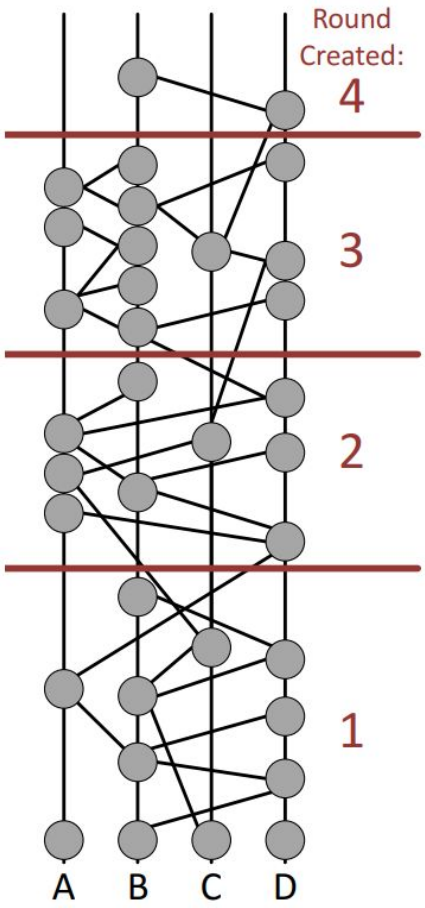
The members in the Hashgraph community run on a gossip protocol which basically means that members repeatedly call others at random to sync with them [26]. Members can avoid sending events to other members that already know about them by exchanging how many events they know about that were created by each member and exchanging events that are missing between them [26]. An event is a data structure that contains the two hashes of the two events below it (its self-parent and its other-parent) [26].



The graph formed by these gossips and transactions continues forever and therefore grows a directed acyclic graph upwards forever [26]. The graph is connected by cryptographic hashes thus leading to its name, Hashgraph. Every event contains the hashes of the events below it and is digitally signed by its creator [26]. The entire graph of hashes is cryptographically secure so it can always grow, but the older parts are immutable, as strong as the cryptographic hash, and a signature system is used [26].

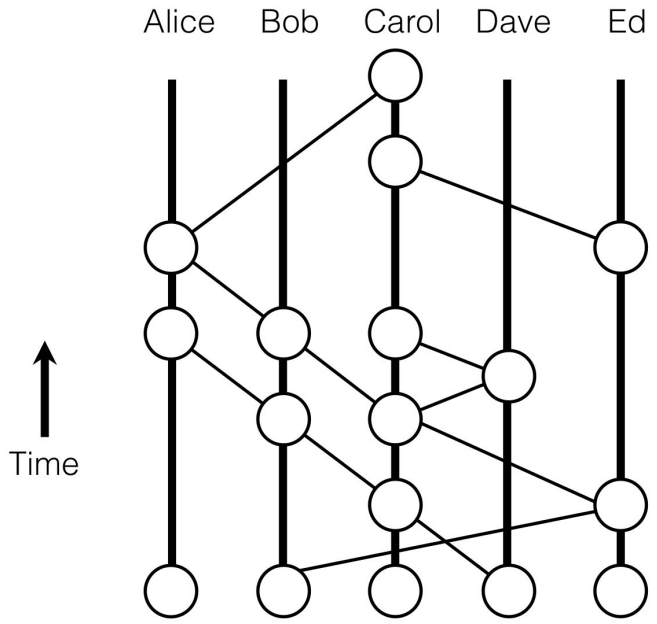
Round Created

A child never has a round created before one of its parents [26]. Therefore, a round can only stay the same or increase. The round created for an event is R or R+1 where R is the maximum of the round created of its parents. It is only R+1 if and only if it can strongly see a supermajority of round R witnesses [26]. A witness is the first event that is created in each round [26].



It is possible for a member to cheat by forking which consists of creating two events with the same self-parent [26]. As a result, there might be two witnesses in the same round created by the same member [26]. A famous witness is determined by considering the witnesses in the next round. If the witness is seen by many of the nodes in the next round then it is considered to be famous [26]. An election occurs when each of those witnesses vote to determine if the witness is famous [26]. There are separate elections for every witness. To strongly see a witness, there must be enough different paths to it so that the paths go through a supermajority of the population [26]. A supermajority is any number that is more than ⅔ of the population of members [26].

Swirlds Hashgraph is another consensus algorithm that is similar to the Blockchain and IOTA's Tangle. The Hashgraph is based on a gossip protocol, in which the participants don't *gossip* about transactions, they *gossip about gossip* about transactions. These participants conjointly build a *hashgraph* reflecting all of the gossip events. It is in this way of consensus that Swirlds achieves consensus that//to be continued



Distributed databases are often required to be replicated state machines with Byzantine fault tolerance. For swirlds hashgraph, Byzantine is used in a strong sense: up to 1/3rd of members can be an attacker without compromising the performance of the hashgraph.

Byzantine Fault Tolerance

The hashgraph consensus algorithm is entirely asynchronous, nondeterministic, and achieves Byzantine agreement with probability one [27].

Hedera Hashgraph

On March 13th, 2018, Hedera Hashgraph was announced to the public. It provides a new form of distributed consensus that allows people to securely collaborate and transact online without a trusted intermediary. Hedera is fast, secure, fair, and doesn't require proof-of-work like Blockchain technology. The Hedera Hashgraph Council will provide governance for a decentralized public ledger built on Hashgraph's consensus algorithm. Governance is provided by a council of up to 39 known and reputable global organizations.

Performance

The hashgraph distributed consensus algorithm provides near-perfect efficiency in bandwidth usage and can therefore process hundreds of thousands of transactions per second in a single shard. A single shard is defined as a fully-connected, P2P mesh of nodes in a network. Consensus latency is measured in seconds.

Security

Hashgraph achieves the "gold standard for security" by using asynchronous Byzantine Fault Tolerance (aBFT). Other platforms tend to be vulnerable to Distributed Denial of Service attacks

(DDoS). Hashgraph is resilient to DDoS and achieves the theoretical limits of security defined by aBFT. Hashgraph ensures both Fair Access and Fair Ordering.

Governance
Hedera Hashgraph is comprised of Council Governance and Consensus. Council Governance is used for the management of the business of the council. Consensus is used to determine the consensus order of transactions.

Council Governance Model
Hedera will be governed by up to 39 leading organizations in their respective fields. Membership criteria are designed to reflect a range of industries and to have highly respected brands and trusted marketing positions. The terms of the governance ensure that no single member will have control and no small group of members will have too much influence over the body as a whole.

Consensus Model
Each node casts one vote for each coin of the hashcap cryptocurrency that they own. New nodes join the network and will be compensated for their services in maintaining the hashgraph structure.

Stability
Regulatory Compliance

4.2: Performance and Maintenance

The Hashgraph consensus mechanism is similar to the blockchain in which the “chain” is always branching, but without pruning, no blocks are ever stale, and each miner can mine many blocks per second without proof-of-work and with 100% efficiency [27].

Hashgraph also has improved security since it is Asynchronous Byzantine Fault Tolerant. This means that no member can prevent the community from reaching an agreement and they also cannot change the consensus once it has been reached [28]. Also, no transactions ever become stale since every container is used and none are discarded [28]. Hashgraph is inexpensive since it avoids the Proof-of-Work mechanism. Every member can create transactions and containers whenever they want [28].

Fair [29]

- No individual can manipulate the order of transactions.
- No individual can stop or delay a transaction from entering the system.

Provable [29]

- Once an event occurs, within a couple of minutes every member in the community will know where it should be placed in history.
- Every member will know that every other member is aware of the event.
- Members don't need to remember old transactions or blocks which shrinks the amount of storage needed.

Byzantine [29]

- No single member or group of members can prevent the community from reaching a consensus.
- No member or group of members can change the consensus once it has been reached.
- Each member will eventually reach a point where they know with 100% certainty that a consensus has been reached.

ACID Compliant [29]

- Atomicity, Consistency, Isolation, Durability
- Applies to the hashgraph when it is used as a distributed database: a community of members uses it to reach a consensus. Once a consensus is reached, each member feeds the transactions to their local copy of the database.
- The community as a whole can be said to have a single, distributed database with the properties of ACID if the local database has all of those standard properties.

Inexpensive [29]

- Avoids proof-of-work mechanism which can impact performance
- Does not need to waste computations to slow itself down for blocks to be properly mined.

Timestamped [29]

- Every transaction is assigned a consensus time which is the median of the times that each member first received it.
- If a majority of the participating members are honest and have reliable computer clock times, then the timestamp itself will be honest and reliable.
- Consensus timestamping is useful for smart contracts since there will be a consensus on whether an event happened by a deadline.
- Timestamps are resistant to manipulation by an attacker.

Non-permissioned [29]

- A permissioned system is one where only trusted members can be part of the community and participate in transactions.
- An open system is not permissioned and allows anyone to participate.
- Hashgraph can be designed to work in various ways such as proof-of-stake.

4.3: Capabilities and Limitations

Hashgraph eliminates the need for massive computation and unsustainable energy consumption that is present in Blockchain systems [28]. Hashgraph is only limited by bandwidth at 250,000+ transactions per second pre-sharding while Bitcoin is limited to 7 transactions per second [28].

Hashgraph is more fair since no individual can manipulate the order of the transactions.

4.4: Hashgraph and Security

Hashgraph consensus does not use a leader and is therefore resilient to the denial of service (DoS) attacks on small subsets of members [27].

DoS Resistant [29]

- Distributed structure resists such attacks.
- An attacker might flood one member/miner with packets to temporarily disconnect them from the internet but the community as a whole will still operate as usual.
- An attack on the system as a whole would require flooding a large fraction of the population with packets which is much more difficult.
- Hashgraph avoids the problem of an attacker freezing the entire system by attacking one leader at a time.

Sybil Attacks

- Consensus algorithm provides a mathematical guarantee that these attacks cannot succeed as long as less than ⅓ of the population is dishonest.

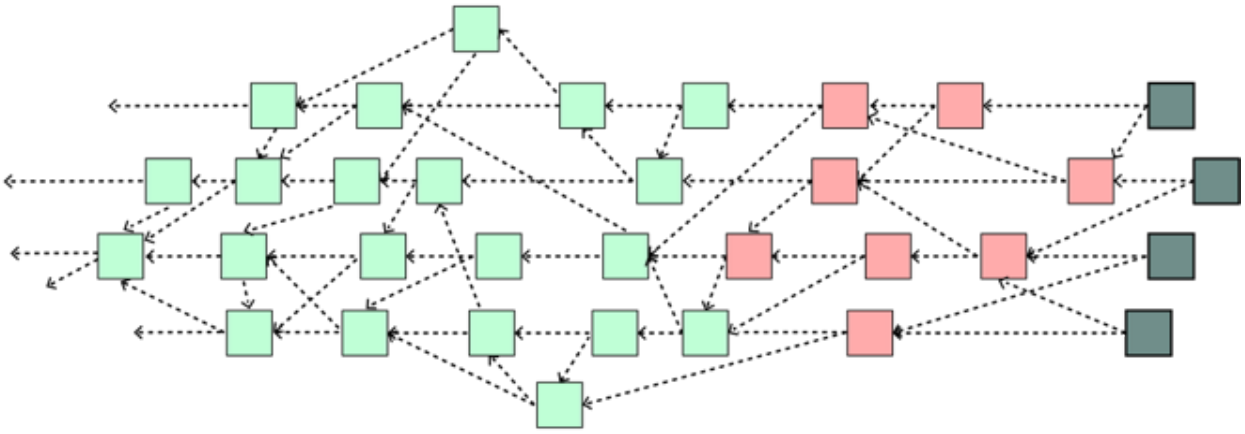
Chapter 5: Tangle

5.1: What is Tangle?

The Tangle is a Directed Acyclic graph which is used for storing transactions and distributed consensus[30]. The Tangle provides a solution similar to a blockchain, allowing for digital, decentralized, and immutable transactions of information. The Tangle was developed for IOTA, a cryptocurrency for the Internet-of-Things industry[30].

In general, tangle-based consensus transactions operate in the following manner. A tangle consists of *nodes*, which are entities that issue and validate transactions. The transactions issued by the nodes compose the site set of the tangle graph, which is the ledger for storing transactions. *Sites* are transactions represented on the tangle graph. Similar to the blockchains genesis block, the tangle has a *genesis* transaction. This genesis must be approved directly or indirectly by all other transactions[30]. The edge set of the tangle is produced in two following ways: when a transaction is put forth it must approve two other transactions. If transaction A *directly approves* transaction B, they are both connected to the same edge. If transaction A approves transaction B which has proved transaction C, we can say that A *indirectly approved* transaction C. As long as there is a directed path length of at least two between two transactions, then they are indirectly approving each other

The main idea behind the tangle is that in order to issue a transaction, you must approve multiple others. Thus for every transaction issued, the systems network security is enhanced. If a node finds a transaction that is in conflict with the tangles history, the node will not approve the transaction in an either indirect or direct manner.



IOTA Tangle: Each square box in this diagram represents a transaction being sent. For each new transaction, two random, unconfirmed transactions are validated in the tangle. Each validation (n) of a transaction increases the likelihood of a transaction being genuine, up to a threshold of *(c)*. In this figure, red boxes indicate transactions where $n > 0$, but below a certain confirmation threshold, $n < c$. The grey boxes represent transactions where $n = 0$. The green boxes represent transactions that have been validated a sufficient number of times, in order to be accepted as confirmed by the recipient address, $n \geq c$.

Core Concepts

Transactions

In order to issue a transaction, a node does the following:

I, The node chooses two other transactions to approve according to an algorithm. These transaction may or may not coincide.

II, The node determines if those transaction are conflicting or not, and does not approve the conflicting transaction.

III, The node must then solve a cryptographic puzzle, similar to blockchains proof of work. It does this by solving a nonce such that its hash is concatenated with the data of the approved transactions[30].

It is important to note the asynchronicity of IOTA's Tangle. It is also important to note that the tangle may in fact contain conflicting transactions, however these transactions are left on the tangles 'edge' and as more transactions are verified conflicting transactions become 'orphaned', and are prevented from being indirectly approved from incoming transactions. The tangle does this using its *tip selected algorithm*[30]. If a node doesn't approve enough transactions, it will be given status of 'lazy' and surrounding nodes will drop it. In this way, nodes are motivated to participate in approving transactions else they could never make a transaction themself.

Weights and Tips

All transactions have a *weight*. The weight of a transaction is proportional to the amount of work an issuing node has invested into it. The idea is that a transaction with more 'weight' is viewed as more important than a transaction with a similar weight.

The *cumulative weight* of a transaction is the sum of the transactions individual weight and all the transactions weight that *have approved* that transaction..

The *score* is the sum of the transactions individual weight and all the transactions weights that *have been approved by* that transaction.

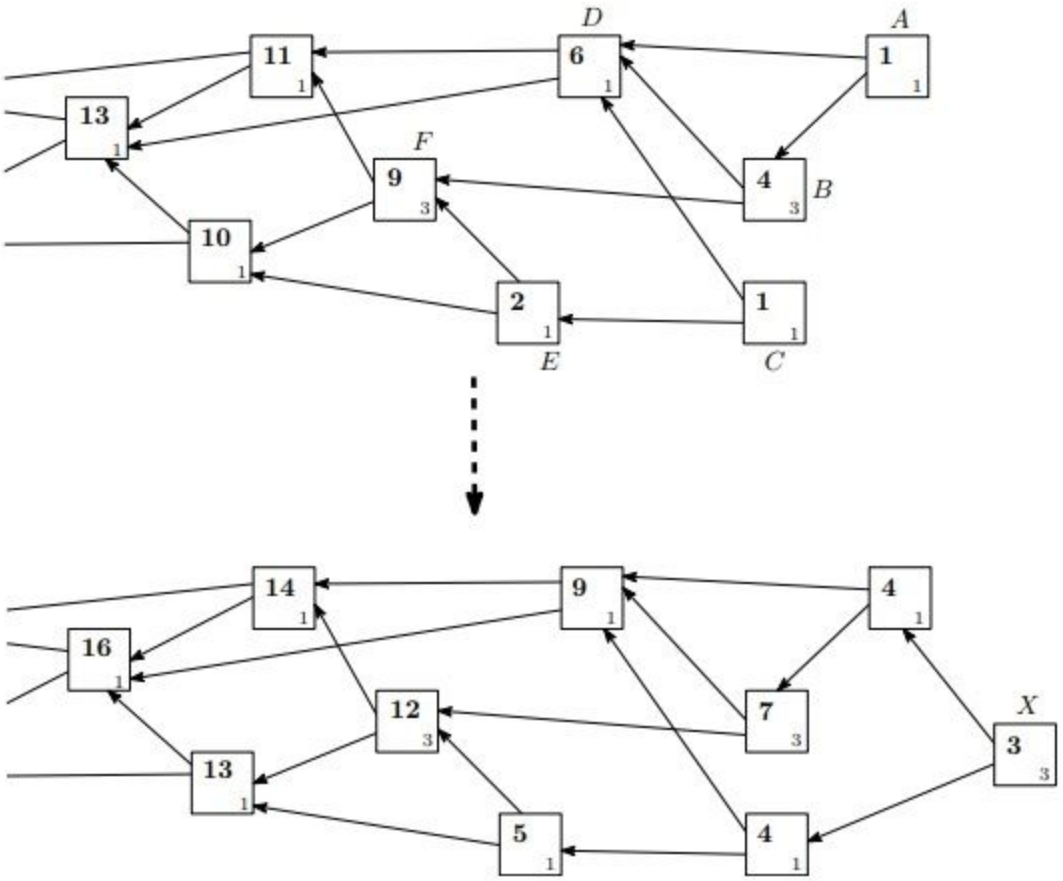


Figure 1: DAG with weight assignments before and after a newly issued transaction, X. The boxes represent transactions, the small number in the SE corner of each box denotes own weight, and the bold number denotes the cumulative weight.

Figure 1. [30]

In the first DAG of figure 1, the *tips* of the graph are transaction A and C. Each individual weight is represented in the bottom right corner of the square representing it. In the second DAG, we have a new transaction X, which becomes the only *tip*. (Note: we are going to use the word tip to describe algorithms in future sections. A tip is a transaction that has yet to be approved by other transactions). It directly approved two transactions and indirectly approves seven other transactions. This loop continues to go on as new transactions are added to the tangle. The more times a transaction is approved or indirectly approved, the deeper into the tangle it get and the more value it has for the system. These transactions have a *high confidence level*.

Quantum Computing Resistance

Resistance to quantum computation

It is well known that an attacker with quantum computing can bypass security measures that rely on trial and error, such as generating the correct hash(proof of work). A ‘large weight’ attack

could also be much more effective on a quantum computing attack against the tangle. This is prevented by IOTA being able to set an upper limit on computing power. This prevents nodes from verifying transactions too quickly and setting an imbalance in the system.

5.2: Performance and Maintenance

Regimes

There are two ruling regimes within tangles architecture; low load and high load. We must understand what these are in order to talk about performance; mainly expected time it takes for a transaction to receive its first approval

The two types are *low load regime* and *high load regime*.
In a *low load regime* the number of tips present are typically few, and often 1. This usually occurs when there is a slow flow of transaction such that the probability that several different transactions would approve that tip is sufficiently small[30].
In the *high load regime* the number of tips present are typically large. This happens when there is heavy transaction traffic flow and is usually aided by lag making it likely that several different transactions approve the same tip. [30]

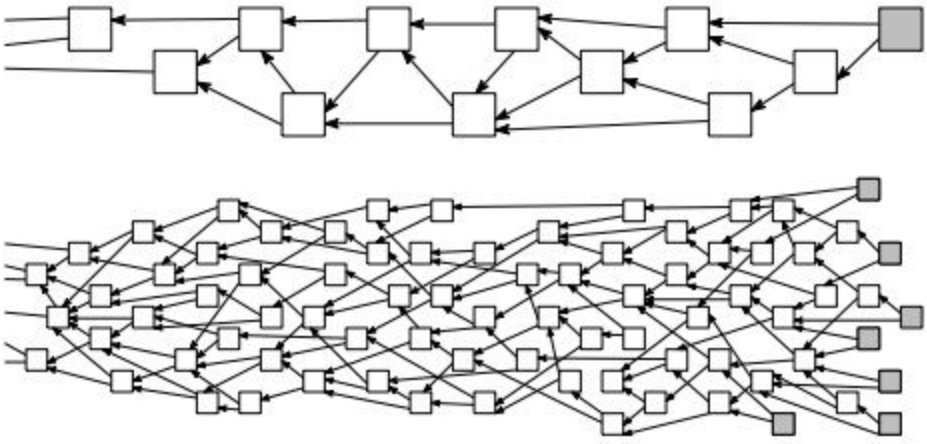


Figure 3: Low load (top) and high load (bottom) regimes of incoming transaction flow. White squares represent verified sites, while gray squares represent tips.

Approval Rate

Performance is measured by the speed and accuracy of the approval rate of a transaction. The approval rate differs between each regime.
Low load regime transaction first time approval happens at a rate of $\Theta(\lambda^{-1})$, with λ being the rate of the rate of the incoming flow of transactions (the Poisson Process represents this rate).

The typical time for a tip to be approved is Θ(h) in the high load regime, where h is the average computation/propagation time for a node. However, if the first approval does not occur in the above time interval, it is a good idea for the issuer and/or receiver to promote that transaction with an additional empty transaction[30]

So the performance of the system depends on the regime that system has at that time. For a more in depth explanation of how the approval rate was calculated, check out Section 3 in IOTA's white paper, source 30.

It's important to note that the system has more complex tip selection algorithms that optimize the tangles security but also changes its performance to a degree. These algorithms are entwined with security and maintenance aspects as well.

5.3: Capabilities and Limitations

The Coordinator

Currently the tangle is supplemented with an entity called the Coordinator because it is not a sufficiently large network yet. This makes the Tangle not yet decentralized. The coordinators job is to checkpoint valid transactions, which are then validated by the entire network. The coordinator is a centralized agent being run by the entire IOTA foundation, The coordinator issues a new milestone every minute, which references valid transactions. The coordinator will be made optional by summer. This is one of the current limitations of the tangle. [31]

5.4: Tangle and Security

IOTA's Tangle has implemented many attack prevention algorithms to defend from a variety of threats. The Tangle is not Byzantine Fault Tolerant.

Potential Attacks

Giant Weight Attack

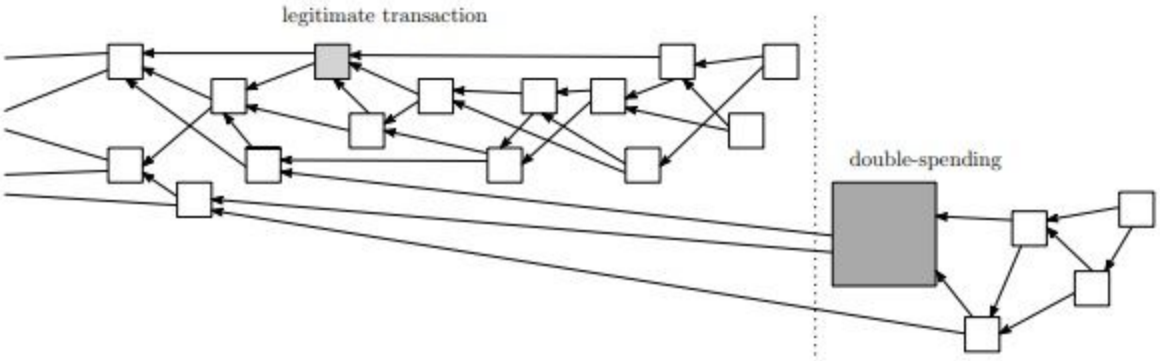


Figure 5: The "large weight" attack

In a large weight attack, a malicious entity will issue a false transaction and use a massive amount of computing power to make its weight sufficiently large. If a host can issue a sufficiently large weight on a transaction by building a secret subtangle, it can overtake the main tangle, orphaning the main tangle, and taking control. In an ideal situation, this attack will always work with success. In order to mitigate this risk we must have a better way to decide how to select which tips to directly or indirectly prove. We can also cap the weight of each transaction to 1 which will reduce the chance of a secret subtangle and a double spending transaction to overtake the main tangle with a large weight.

Parasite Chain Attack

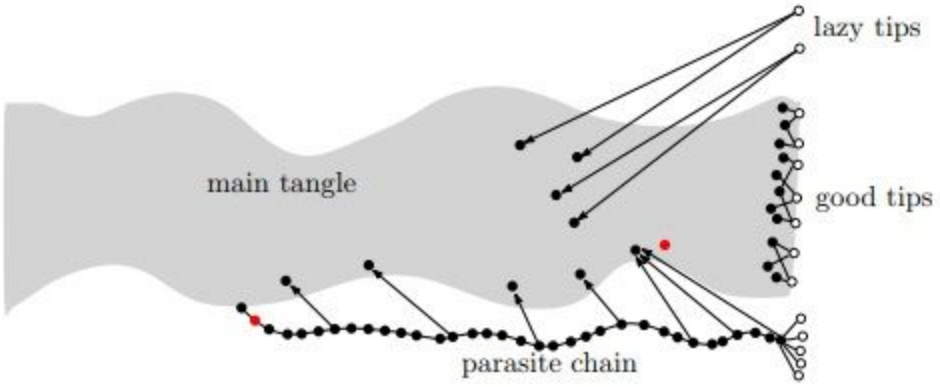


Figure 6: Visual representation of the tip selection algorithm for honest tips, as well as the parasite chain. The two red circles indicate an attempted double-spend by an attacker.

A parasite chain attack is similar to a large weight attack. A malicious entity builds a parasite chain, or a subtangle, that occasionally references the main tangle in order to increase its weight and thus increase its confidence interval in the system. Given sufficiently large computation power, that malicious entity can release a whole load of artificial transactions that are approved by its own subtangle. To defend against this attack the tangle relies on the assumption that the main tangle consists of more hashing power than a single entity can produce, and thus retain a larger cumulative weight than the parasite chain. The tangle proposes a Markhov Chain Monte Carlo (MCMC) algorithm to help combat this. The idea is to place random walkers on sites in the tangle, and let them 'walk' towards tips in a random way. The tips selected by the walkers are then candidates for approval. In this way, there is a very low chance that a random walker will select a node on the parasite chain to approve and also a low chance that it will select a lazy or a selfish node.

Splitting Attack

The splitting attack was proposed as a potential attack against the tangle that is using the MCMC algorithm. It suggest that in a high load regime, an attack could potentially split the main tangle while retaining a balance between the two halves. The attacker does this by placing two conflicting transactions at the point of the split to prevent honest nodes from joining the network. Then, the attacker hopes that the branch will be roughly split in half so half of the network contributes to each branch similarly. If the attacker succeeds, the attacker could issue the same spending transactions on both chains.

To defend against this, the tangle employs a sharp threshold rule that makes it really difficult for a split tangle to retain balance between both of them.

Quantum Computing Resistance

As quantum computing becomes more prevalent, they become a bigger risk within distributed consensus systems because they are billions of times more efficient at solving hashing algorithms, making them an ideal host to exploit vulnerabilities in systems that rely on those algorithms for security. With Tangle, by limiting the weight of each transaction to 1, you can effectively prevent a quantum computer from overloading the network with a large weight attack. While a quantum computer may hit the ideal rate of giving and seeking approval, it will always remain in an acceptable parameter of increased rate.

Chapter 6: Hyperledger

6.1: What is Hyperledger?

Hyperledger Fabric

Fabric's understanding of a consensus mechanism is broad and encompasses the entire transaction process from start to finish [20]. Nodes take on different roles and tasks during this consensus process. This is different to Ethereum's mechanism where roles and tasks of nodes participating in the consensus are identical [20].

Nodes are differentiated based on their role: client, peer, or orderer [20]. A client acts on behalf of the end-user by creating and invoking transactions and also communicate with peers and orderers [20]. Peers maintain the ledger and receive ordered update messages from orderers that wish to commit new transactions to the ledger [20]. Endorsers are a special kind of peer that is responsible for endorsing a transaction by checking if they fulfill the necessary and sufficient conditions [20]. Orderers provide a communication channel to clients and peers where messages containing transactions can be broadcasted [20]. The channels ensure that all connected peers are delivered the same messages in the same logical order [20].

There are problems that arise with Fabric's mechanism of consensus as well. For instance, faults in the delivery of messages may occur when many mutually untrusting orderers are put in place [20]. A consensus algorithm has to be used in order to reach a consensus despite faults [20]. The algorithm put in place by Fabric is "pluggable", meaning that various algorithms can be used depending on specific application requirements [20]. Fabric's channels partition the flow of messages so clients only see the messages and transactions of the channels they are connected to while remaining unaware of other channels [20]. This allows for restricting access to transactions to involved parties only with the consequence that consensus only has to be reached at the transaction level and not at the ledger level as with Ethereum [20].

Role of Nodes in Transaction Flow [20]

- Client sends transaction to connected endorsers to initiate ledger update.
- Endorsers must agree on the proposed transaction. A consensus has to be reached on the ledger update proposal.
- Client collects approval of all endorsers.
- Approved transaction is sent to connected orders to reach consensus.
- Transaction is forwarded to peers holding the ledger for committing the transaction.

Hence, Fabric provides fine-grained control over consensus mechanisms and restricts access to transactions which provides improved performance scalability and privacy.

Hyperledger is an organization focused on the development and research of blockchains while providing a platform for projects to collaborate. Hyperledger plans on proving a set of guidelines that architecture of blockchains can follow. Hyperledger consists of multiple projects who all follow a common design philosophy. The architectural design that projects follow is based off of the following components.

- Consensus Layer
- Smart Contract Layer
- Communication Layer
- Data Store Abstraction
- Crypto Abstraction
- Identity Services
- Policy Services
- APIs
- Interoperation

Their main focuses are on a modular architectures with high security that is natively not based on cryptocurrency.

Hyperledger provides some core functionality that consensus must follow. It must confirm the correctness of the transactions proposed by on endorsement and polies. All nodes must agree on order and correctness to execute the transaction. Last, it must interface with the smart-contract layer to verify the correctness and order. Hyperledger states that consensus processing is based on an environment of partial trust, much different than bitcoin or ether who rely on anonymous nodes. This is because hyperledger is based on a business sense where nodes will most likely be trusted as they will be within a closed secure system. Consensus can be approached in multiple ways, but hyperledger focuses on three, permissioned lottery-based, permissioned voting-based, and stand proof of work. However, based on the modularity of hyperledger framework aims to provide any other methods of consensus can possibly be implemented. Hyperledger essentiality gives a platform that provides interface between modular aspects so they may be modified without affecting how other pieces work. Hyperledger fabric is just one architectures that follows these guidelines, there is also Hyperledger Sawtooth, Burrow, Iroha and Indy. All of which have unique features that allow them to function different from one another but all follow Hyperledgers standards.

Hyperledger Fabric:

Hyperledger Fabric uses a consensus algorithm known as Kafka and approaches consensus as permissioned-voting based. The consensus is broken into 3 phases, Endorsement, ordering and a validation phase. The endorsement phase is controlled by a policy such as having m out of n signatures participate in the transaction. Once the transaction is endorsed, the ordering phase then accepts it and commits it to the ledger. The validation phase then verifies the correctness of the transactions. Based on hyperledgers modularity, these three phases are pluggable, meaning that different ways of endorsement, ordering, and validation can

be implemented based on the users needs. For example, the ordering service offers many API plugins that create a variety of ways the services works.

The hyperledger fabric model consisted of some key features. They are assets, chaincode, ledger features, privacy, security, and consensus. Assets are essentially what is part of the transaction, from a tangible item being sold or a contract. These assets are modified by the next feature, being the chaincode. The chaincode is the rules enforced for modifying assets. This is essentially what needs to be done in order for a transaction to take place as well as the execution and writing to all peers. The ledger features are a main part of what sets Hyperledger Fabric apart from others. The blockchain consist of many channels, each of these channels has its own ledger as well as special rules in members. For example, a peer can have access to a channel involving every other peer and a channel involved only 2 peers. This brings up the privacy feature within Hyperledger fabric. Based on the channel system a channel can be made private between chosen peers, or a channel can be open to everyone. This can further allow obscuring data within a transaction to only the peers allowed, but also displaying to all that a transaction took place. Hyperledger fabrics security is also a promptly based on a trusted level with business use in mind. Each member has a known identity which essential for conducting business transactions where the user would need to know who they having a transaction with.

Consensus in hyperledger fabric emcompasses the lifecyle of a transaction. Through the process of a transaction, there are multiple checks that take place to ensure that the transaction is correct. The extends between endorsements and that the correct chaincode policies were met. There final check involves a versioning check to make sure the ledger is correct before appending the new transaction. Consensus within fabric is based around the whole transaction from start to finish.

Hyperledger Sawtooth:

Hyperledger Sawtooth once again aims to create a platform where customized applications are built off of a system core. This separation of application from core allows developers to create customized applications, whether they be smart contract VMs much like ethereum or business logic like Hyperledger fabric, to run on this core. Sawtooth even allows for multiple types of applications to exist in the same blockchain network. Sawtooth aims to be used in private networks, and the blockchain as a whole stores who are participants, their roles, and identify which is available to other members within the network to view.

One of the distinct features of sawtooth is the parallel scheduler which can split transactions into parallel channels to be executed while maintaining the changes to the blockchain. This gives sawtooth a potential to have greater performance compared to traditional transaction processing.

Consensus in sawtooth allows the probability of multiple types as well as allowing different types on the same blockchain. Sawtooth currently supports a few consensus types, Proof of Elapsed Time(PoET), PoET simulator, and Dev mode. Proof of Elapsed Time is designed to support large networks, while PoET simulator is based on hardware. Dev mode is primarily used for testing and is a simple random-leader algorithm. The default consensus is the lottery

based PoET, which is lottery leader is selected based on the guaranteed wait time. This way of consensus is based on fairness, investment and verification. Fairness should distribute a leader across all participants, investment should correlate the cost controlling the leaders election to the value of it, and verification that all participants verify the leader is legitimate. Sawtooth builds off of Intel's Software Guard Extensions to ensure that the leader is selected safely, randomly, and is cost efficient. This in return mainly means the necessary computing power required in a proof of work blockchain is not needed. Leaders are chosen based on a request for a random time to wait, whichever has the shortest time to wait is elected leader and so on.

Sawtooth also provides transactions families, there are data models that reflect the requirements of the ledger. These families are highly customizable and can be made to a users needs but Sawtooth provides some example families. These families declare what is essentially stored within the state. This correlates to consensus, for example PoET requires a target wait time setting which can be implemented into the family setting and adjusted to the users desire.

Hyperledger Burrow:

Hyperledger burrow is designed to execute permissioned Ethereum smart-contracts. Burrow consists of a consensus engine where transactions are executed in order based on a byzantine fault-tolerant Tendermint protocol. Transactions are validated and applied based on the order the consensus engine finalized them.

Hyperledger Iroha:

Hyperledger Iroha is aimed to provide reusable components to compliment other projects such as Hyperledger fabric and Hyperledger Sawtooth. The project plans to implement these components in C++ and provide a long term and robust library of features developers can take advantage of. Irohas main goals are to provide a C++ environment for hyperledger, provide mobile application support, and provide an environment for new components (Such as other consensus algorithms) to potentially be incorporated into hyperledger. Iroha has a major focus on C++ design and mobile applications, two environments lacking in other hyper ledger projects.

Irohas architecture provides a number of useful features. The system will work on a peer to peer basis, currently all peers are validating peers but there are plans to implement client peers, validating peers, and normal peers together. Iroha will feature a membership service where $2f+1$ signatures are needed to add or remove a node. There are future plans to implement membership groups and more features to how membership permissions are given. Iroha provides a default cytophrahy through SHA-3. Iroha also supports chaincode, implementing domains and assets, and running transactions with these elements. Iroha stores transactions in the form of a merkle tree where each transactions represents a leaf.

Consensus through Iroha is done through a Byzantine fault tolerant consensus algorithm called Sumeragi. Consensus through this method consist of validate peers of at least $2f+1$ peers to sign the transaction. First the client submits the transaction to a validating peer, who then signs it and broadcasts it to the $2f+1$ peers. The servers then verify the transaction from the

broadcast then commit once the 2*f*+1 signatures are met. Iroha also features a peer reputation system as a way of establishing trust. This is called the hijiri reputation system. The system revolves around a set of rounds that validating peers run that test data throughout, version, computation, and consistency.

Hyperledger Indy:

Hyperledger Indy is focused around a decentralized identity system. Indy features a system of trust where users can exchange verifiable claims with a strong focus on privacy. A major part of this privacy is Indys use of decentralized identifiers. These are the primary keys to the ledger which do not require a centralized registry service. These keys are verified through cryptography which creates a "web of trust". Indy also does not write any personal data to the ledger, instead it is all exchanged through peer-to-peer encrypted connections. Indy also has support for zero-knowledge proofs to avoid any unnecessary disclosure of identities.

6.2: Performance and Maintenance

Hyperledgers projects can offer a solution to one of blockchains biggest issues, scalability.

With traditional blockchains every node has a copy of the entire chain. Because of this, when the chain becomes very large and there are many nodes, the size of data storage needed drastically becomes bigger. Each hyperledger architecture processes consensus and data storage differently, however mainly provide a way of storing a ledger that is much more efficient to many blockchains. As mentioned in Hyperledger fabric, the architecture revolves around channels. Essentially when a transaction is done, everyone in the system can see that a transaction was made between 2 entities while only disclosing some of the information about this transaction. (Hyperledger allows the developers to decide what information is shared to everyone). The 2 parties involved however are the ones who store their copy of the transaction and all of its parts. This major change in data storage means that what would be considered wasted data is separated from what everyone needs access to, in turn improving scalability.

Hyperledger architecture often include a membership service for maintaining nodes and usually a fail safe in case of an issue with a transaction. Hyperledgers membership services can range from single entities who oversee and validate new nodes, to consensus validation through all the nodes to add or remove a node. Because of hyper ledgers projects modularity, whatever the needs of the system can easily be met to maintaining the nodes.

6.3: Capabilities and Limitations

Hyperledger projects are capable of a wide range of functions that separate it from other blockchains. Their main feature being its design around modularity. Hyperledgers goal is to create a platforms that can be manipulated to the users needs. This extents from functionalities

such as consensus, and validation to how entities are allowed onto the blockchain. Hyperledgers performance capabilities are highly based on how it is implemented and what it is planned to be used for. They are capable of giving even more options into the architecture by offering systems like Iroha that provides C++ and mobile applications, or like barrow which implements an ethereum VM. There is a wide range of capabilities and limitations to hyperledger projects, because of its modularity many of the limits will be controlled by how they are implemented.

6.4: Hyperledger and Security

Security in hyperledger is largely designed around permisioned based systems. Because of the large focus around business needs, hyperledger plans to implement systems that require the entities to be permissioned identities. This is often done by some membership system that approves who is allowed on the blockchain. The idea is that the members using will the system will be conducting business together, often needing to know who they are doing business with and validating the authenticity that it is them. These membership services create a private closed blockchain where trust is inherit because each member was approved to be a member. Although membership is implemented, once again because of hyperledgers modularity, this can change to the users needs. We see in hyperledger Iroha that the default system of allowing membership is based on a number of nodes approving a member be added. Users can decide how private they wish their system to be, and what method they wish to be conducted when giving membership.

Hyperledger also offers many methods of consensus security. Hyperledger fabric offers consensus that can be based on only a few nodes. For example user A can trade user B and may only need user C, who is a bank, to approve that the transaction is legitimate. This way of consensus can also be changed depending on the need. We see in hyperledger sawtooth consensus is done through a lottery based system.

Hyperledger continuously demonstrates its modularity and choice when it comes to security. A developer of a system can not only choose their method of consensus to validation and membership, but also fine tune the methods chosen. We see this throughout all of hyperledgers projects, as they each offer different security methods, but the main focus on security is through a private closed blockchain.

Chapter 7: R3 Corda

7.1: What is Corda?

R3 Corda

Corda is similar to Fabric in the sense that consensus is also reached at the transaction level only by the parties involved [20]. Uniqueness and validity of the transaction are subject to consensus. Validity of a transaction is ensured by running the smart contract code that is associated with the transaction, checking for required signatures, and assuring that any transactions that are referred to are also valid. Uniqueness is concerned with the input states of a transaction and it has to be ensured that the transaction is the unique consumer of all of its input states [20]. This is done to avoid double-spends. Consensus through uniqueness is achieved by notary nodes whereas the algorithm is “pluggable” with Fabric.

Corda is a technology that uses distributed ledgers as a form of financial services in order to create transactions can be processed seamlessly and kept secured. The vision of corda is a “global logical ledger”, which will allow all agents to interact in their own agreements while being secure, reliable, private, and authoritative. Essentially any agent within the network will be able to create secure transactions while deciding who is involved with this transaction. This essentially means that unlike a blockchain, every node will not need to store every transaction on the network. Cordas implementation is based off of many principles including:
-Records are accepted as evidence and legally bound to the parties involved.
-Records finalized on the ledger cannot be changed, another transaction is needed to fix the error.
-The only parties who have access to transactions are those who need to know. Example: Two parties of a transaction and their banks.

Corda also has a large vision for the future use of this technology. Its goals include reduction of cost, risk, and maintenance as well as establishing new services. They also wish to gain adoption through many financial communities.
Corda is designed for regulated financial institutions, and has many key features that make it useful for this environment. It is designed for the recording of agreements between parties while adhering to all legalities and existing regulations. The network allows seamless workflow between groups as well as supporting consensus between individuals instead of a whole as well as a variety of consensus methods. This allows for transactions to be validated between only those who are involved and restricts access of the agreement to those verified to see it.

7.2: Performance and Maintenance

Cordas design separates it from the limitations that many blockchains have, specifically with scaling and TPS(Transactions per second). Because corda is designed in a way that only the members part of a transaction are required to have the data involving said transaction, a massive amount of wasted space that traditional blockchains use is eliminated. This is because instead of having every transaction saved to every member only the members involved in the transaction save the information. This will in turn save space but also increase scalability with a large number of transactions taking place. Cordas second consensus service of uniqueness however can cause bottlenecking. All transactions are determined uniqueness by a notary system, essentially specific systems that determine the uniqueness of all transactions. However if all transactions are accommodated by a single system, a bottleneck may occur. By having multiple systems spread out to deal with transactions, the risk of a bottleneck can be lessened.

7.3: Capabilities and Limitations

Cordas main source of limitation is its notary system of uniqueness. As mentioned above, this uniqueness system is prone to bottlenecking. Essentially cordas speed is limited to how many transactions are handled by a single notary system. This bottleneck can be accommodated for by a having many systems handle the input of transactions. The more systems, the less likely of a bottle neck.

7.4: Corda and Security

Consensus in corda is dealt in two parts, transaction validity and transaction uniqueness. Transaction validity is when the parties involved agree to consensus, this in turn means only these parties can see the transaction. At least two parties are needed in order to record a transaction to the ledger. Transaction uniqueness can be implemented in a few ways. Corda allows for this service to be “pluggable” meaning that different concesnus methods may be used, such as a byzantine fault tolerance algorithm or a simple single machine system. The uniqueness systems are not required to see the information within the transaction as well, they are only there to confirm that the state has not been consumed before.

Chapter 8: Comparison of Blockchain, Hyperledger Fabric, Corda, Tangle, and Hashgraph

All 3 of these frameworks have very different visions in mind with respect to fields of application [20]. Hyperledger Fabric and Corda are driven by concrete use cases. However, Corda's use cases are drawn from financial applications [20]. Hyperledger Fabric provides a modular and expandable architecture that can be applied to various industries [20]. Ethereum also intends to be independent of any specific application field [20]. The table below provides a summary of all 3 frameworks.

Comparison of Ethereum, Hyperledger Fabric, and Corda

Characteristic	Ethereum	Hyperledger Fabric	R3 Corda
Description of Platform	Generic Blockchain Platform	Modular Blockchain Platform	Specialized distributed ledger platform for financial industry
Governance	Ethereum Developers	Linux Foundation	R3
Mode of Operation	Permissionless Public or Private	Permissioned Private	Permissioned Private
Consensus	Mining - Proof of Work Ledger Level	Broad understanding of consensus that allows multiple approaches. Transaction Level	Specific understanding of consensus (i.e. notary nodes) Transaction Level
Smart Contracts	Smart Contract Code (e.g. Solidity)	Smart Contract Code (e.g. Go, Java)	Smart Contract Code (e.g. Kotlin, Java) Smart Legal Contract (legal prose)
Currency	Ether Tokens via Smart Contracts	None Currency and tokens via chaincode	None

Table 1.1: Comparison of Ethereum, Hyperledger Fabric, and Corda [20]

Participation of Peers

With centralized and typical data storage mechanisms, a single entity keeps a copy of the database. This entity controls what data is distributed and what other entities can contribute to the database [20]. With the advancement of decentralization, multiple entities now hold a copy

of the database and are naturally permitted to make changes [20]. All entities that participate in this process of distributed storage form a network of nodes or peers [20]. However, with distributed data storage, the difficulty arises to make sure that all nodes agree on a common block to be added to the blockchain since a change made by one node has to be propagated to all other nodes on the network [20]. The result of arriving at an agreement is called a consensus.

There are two methods of participating to consensus: permissioned and permissionless. If participation is permissionless, then anyone can participate as a node on the network [20]. This is true for the Ethereum public blockchain. However, if participation is permission, then participants are chosen in advance and access is restricted to these selected participants [20]. This is true for Hyperledger Fabric and Corda. The mode of participation, whether it be permissioned or permissionless, has a significant impact on how a consensus on the network is reached.

Consensus

Ethereum

All participants on Ethereum's network must reach a consensus on the transaction independent of whether or not that participant took part on the transaction [20]. The order of transactions is crucial for the consistent state of the blockchain ledger [20]. If an order of transactions cannot be established, then there's a chance that double-spends may have occurred, meaning that two parallel transactions transfer the same coin to different recipients and therefore making money out of thin air [20]. The network may involve mutually distrusting and anonymous parties [20]. Therefore, a consensus mechanism is put in place to protect the ledger against fraudulent or adverse participants that try to double-spend [20].

In Ethereum's current implementation, a proof-of-work (PoW) mechanism is used to mine and agree on a common ledger [20]. All participants have access to all entries that have ever been recorded on the blockchain [20]. However, there are consequences to using this proof-of-work system such as affecting the performance of transactions processing. Even though the records on the ledger are anonymized, they are accessible to all participants which may be a problem for applications that need more privacy [20].

Fabric's and Corda's interpretation of consensus mechanisms is more refined and isn't simplay mining based on PoW or another system [20]. Since Fabric and Corda operate in a permissioned setting, they provide a more fine-grained access control to records and enhances privacy as a result [20]. Performance is also enhanced using Fabric or Corda because only parties involved in the transaction have to reach the consensus as opposed to Ethereum's consensus mechanism [20].

Hyperledger Fabric

Fabric's understanding of a consensus mechanism is broad and encompasses the entire transaction process from start to finish [20]. Nodes take on different roles and tasks during this consensus process. This is different to Ethereum's mechanism where roles and tasks of nodes participating in the consensus are identical [20].

Nodes are differentiated based on their role: client, peer, or orderer [20]. A client acts on behalf of the end-user by creating and invoking transactions and also communicate with peers and orderers [20]. Peers maintain the ledger and receive ordered update messages from orderers that wish to commit new transactions to the ledger [20]. Endorsers are a special kind of peer that is responsible for endorsing a transaction by checking if they fulfill the necessary and sufficient conditions [20]. Orderers provide a communication channel to clients and peers where messages containing transactions can be broadcasted [20]. The channels ensure that all connected peers are delivered the same messages in the same logical order [20].

There are problems that arise with Fabric's mechanism of consensus as well. For instance, faults in the delivery of messages may occur when many mutually untrusting orderers are put in place [20]. A consensus algorithm has to be used in order to reach a consensus despite faults [20]. The algorithm put in place by Fabric is "pluggable", meaning that various algorithms can be used depending on specific application requirements [20]. Fabric's channels partition the flow of messages so clients only see the messages and transactions of the channels they are connected to while remaining unaware of other channels [20]. This allows for restricting access to transactions to involved parties only with the consequence that consensus only has to be reached at the transaction level and not at the ledger level as with Ethereum [20].

Role of Nodes in Transaction Flow [20]

- Client sends transaction to connected endorsers to initiate ledger update.
- Endorsers must agree on the proposed transaction. A consensus has to be reached on the ledger update proposal.
- Client collects approval of all endorsers.
- Approved transaction is sent to connected orders to reach consensus.
- Transaction is forwarded to peers holding the ledger for committing the transaction.

Hence, Fabric provides fine-grained control over consensus mechanisms and restricts access to transactions which provides improved performance scalability and privacy.

R3 Corda

Corda is similar to Fabric in the sense that consensus is also reached at the transaction level only by the parties involved [20]. Uniqueness and validity of the transaction are subject to consensus. Validity of a transaction is ensured by running the smart contract code that is associated with the transaction, checking for required signatures, and assuring that any

transactions that are referred to are also valid. Uniqueness is concerned with the input states of a transaction and it has to be ensured that the transaction is the unique consumer of all of its input states [20]. This is done to avoid double-spends. Consensus through uniqueness is achieved by notary nodes whereas the algorithm is “pluggable” with Fabric.

Smart Contracts

Smart contract code denotes software written in a programming language and acts as an agent or delegate of the party that employed the contract with the intention that it fulfills certain obligations, exercises rights, or can take control of assets within a distributed ledger in an automated way [20]. Smart contract code takes on tasks that emulate contract logic in the real world [20].

Built-in Currency

Ethereum has its own built-in cryptocurrency known as Ether. Ether is used to pay rewards to nodes that contribute to the consensus mechanism by mining blocks and paying transaction fees [20]. Decentralized apps (DApps) can be built for Ethereum that allow for money transactions [20]. A digital token for custom use cases can be created by writing a smart contract that adheres to a predefined standard [20].

Fabric and Corda don’t need a built-in cryptocurrency as consensus since it isn’t reached through mining. Fabric does allow the development of a native currency or digital token through its chaincode while Corda does not since it is not intended for that purpose [20].

Chapter 9: Blockchain Tutorials

Creating a Private Ethereum Blockchain in Windows
https://medium.com/mercuryprotocol/how-to-create-your-own-private-ethereum-blockchain-dad6af82fc9f
Installing Bash for Windows 10
https://www.windowscentral.com/how-install-bash-shell-command-line-windows-10
Development using Windows Terminal with Cygwin

Cygwin is a terminal for windows 7 used to recreate a linux OS terminal. If you use windows 7, or windows 8.1, Cygwin is the terminal that you want. If you have windows 10, you can use Bash for Windows. If you have an UNIX OS, you can use the default terminal.
Download Cygwin: https://www.cygwin.com/

Cygwin Instructions:

Follow the download prompts for Cygwin. It will have you choose a mirror to download from. Don't be afraid to choose whichever one and then follow through keeping all the defaults. After you've downloaded, you can create a shortcut on the desktop.
Download Bash for Windows:
https://blogs.msdn.microsoft.com/commandline/2016/04/06/bash-on-ubuntu-on-windows-download-now-3/

In this tutorial I will walk you through the solidity development process. This tutorial is directed towards those who are new to new to blockchain development. I will show you how to get all the applications you need to begin work as a blockchain developer and also walk you through creating your first solidity contract and the mechanics behind it.
Installing Node.js, npm, git, web3.js, truffle, and connecting to the ethereum testnet.

Node.js, npm, git, web3.js, and truffle are essential ingredients for Blockchain development. //add short explanation of why

1. Install nodejs : https://nodejs.org/en/ 8.9.0
    a. Install the appropriate node.js version, then confirm it's installed by typing $node -v in your terminal.
    b. Note: Npm should be installed with nodejs. Check with npm -v in the console. Skip step 2 if version was installed.
2. Install npm via terminal if needed.
    a. $npm install --global. https://docs.npmjs.com/getting-started/what-is-npm
    b. You can check your version of npm by typing
        $npm -v
3. Install git - https://git-for-windows.github.io/

    a. Follow install instructions and keep default selections. Once you finish, restart your terminal. Type
        $git version
4. Install web3.js via terminal
    a. Web3.js is the main ethereum javascript library that comes packed with useful API. Web3.js can be very tricky to install. The first command to try is
        $npm install web3
        A lot of times you will run into errors, especially if you are using a Windows OS (although sometimes you won't). Try
        $npm install web3@^0.20.0
5. Install truffle via terminal
    a. $npm install -g truffle
    b. To check if you have truffle installed, type
        $truffle
        This will give you a list of the commands you can use in truffle
6. Install ethereum testnet via terminal
    a. $npm install -g ethereumjs-testnet

These are the main tools you need to begin life as a blockchain developer. For editing solidity and javascript files, we recommend Visual Studio Code; but you can use whichever IDE you prefer.

Using Truffle to create smart contracts

Once you have everything installed and downloaded (node.js, npm, git, web3.js, truffle, and ethereum testnet), you are now ready to create your first blockchain project! The first thing you want to do is open up your terminal and navigate to a workspace you enjoy. If you use Cygwin, the terminal will open up in C:\cygwin64\home\UserName . This is the workspace I use. To create a folder within that workspace, type

$mkdir SD_Blockchain

The name of my project in this instance is SD_Blockchain.To navigate into this workspace type

$cd SD_Blockchain

If you are familiar with Linux you will recognize a lot of these commands. That's because Cygwin is a terminal meant to emulate linux. Once you are in your workspace, you can check which files are in your terminal with the command

$ls

Once you are in your workspace, for the first time, you should have no files in it. To begin your first contract you will use a truffle command

$truffle init

This will initialize truffle within that workspace and create three folders and two Javascript files. The folders are [contracts] [migrations] [test] and the files are truffle.js and truffle_test.js.

[contracts] will be where you make all of your contracts. To create a .sol file within the contracts folder first navigate into contracts

$cd contracts

You will notice there is already a contract in there, called migrations. Keep that there as it's used when you migrate any new contracts after compilation. Now you want to create a new .sol file. To do so,

$touch FileName.sol

This will create a new file with filetype .sol. You can edit this file with Visual Studio Code or whichever IDE you prefer.

Next you will have to add to the [migrations] folder. You will see their is already a javascript file in it called 1_initial_migration. You will want to create a new .js file

$touch 2_filename_migration.js

It doesn't really matter what you call it as long as it indicates which .sol file it's for.

Development using Ubuntu 16.04 (Bash/Cygwin for Windows)

Installing Ethereum

Use the following commands:
sudo apt-get install -y software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install -y ethereum

Creating the Genesis Block

Use the following commands:
mkdir genesisblock - make the directory
cd genesisblock - get into the directory
nano genesis.json - make the file

Copy and paste the following genesis block into your file:
[insert our genesis block here]
Save this new file.

Command List for Libraries
List of linux commands - https://ss64.com/bash/

Coding with Solidity and Javascript

Truffle Development
https://medium.com/@gus_tavo_guim/using-truffle-to-create-and-deploy-smart-contracts-95d65df626a2

Commands:

Command	Purpose
compile	initializes the Truffle console that lets you enter commands.
migrate	compiles your solidity code.
deploy	migrates your contract to a file that can be deployed.

Getting Your Application to Communicate with a Solidity Contract
First, you'll need to declare your web3 provider. Use the following line below to declare your provider as your local host.
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

You'll need to set your default account. For the purpose of development and testing, we'll set the default account to the first account in the testRPC.

web3.eth.defaultAccount = web3.eth.accounts[0];

This is the tricky part. You will need to pass the JSON representation of the contract ABI into the contract instance. Go to remix.ethereum.org and paste your solidity contract into a new text file. In the compile window, click the "Details" button and a popup window will appear. Click the clipboard icon for the Interface:ABI section. That is the JSON representation of your contract. Structure your statement like ours:

var addBlockContract =
web3.eth.contract([{"constant":true,"inputs":[{"name":"key","type":"uint256"}],"name":"getdata","outputs":[{"name":"","type":"bytes32"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"key","type":"uint256"}],"name":"getID","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"data","type":"bytes32"},{"name":"ublock_no","type":"uint8"},{"name":"uid","type":"uint256"}],"name":"addData","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[{"name":"key","type":"uint256"}],"name":"getblock_no","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"blockchain","outputs":[{"name":"block_no","type":"uint8"},{"name":"id","type":"uint256"},{"name":"data","type":"bytes32"}],"payable":false,"stateMutability":"view","type":"function"}]);

Now, we have a working contract. But, we can't facilitate communication yet. To do that, we'll need an instance of the contract to work with. Declare the following statement and replace the address with your current contract address. You'll find it as the second to last parameter after running addBlock.new('Constructor Initialization'):

var addingBlock = addBlockContract.at('0xcd09404926caf1ae13dbae49f4d853930b95a192');

Guide/Tutorial List:

- How to Configure Truffle
- How to set up an Ethereum Node
- How to write your first Solidity Contract in ethereum
- How to configure a lighttpd server
- How to configure visual studio code for github
- Basic Github tutorials
- How to configure metamask with your web application and blockchain
- Web3 and nodeJS tutorial basics

Chapter 10: Troubleshooting Issues with Development

Metamask

Set up using customRPC and http://134.88.13.89:8545 as RPC
Create account using address (private key) and password
If off campus, use checkpoint endpoint security VPN with umassd email and password
App address: http://134.88.13.89
Costs 12 cents to use our smart contract

Appendices

Resources for API/Language Documentation

Solidity Documentation
https://solidity.readthedocs.io/en/develop/

Web3js API Documentation
https://github.com/ethereum/wiki/wiki/JavaScript-API

Nodejs API Documentation
https://nodejs.org/api/index.html

JQuery Documentation
https://api.jquery.com

GETH Documentation
https://github.com/ethereum/go-ethereum/wiki/geth

Javascript Documentation
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

HTML Documentation
https://developer.mozilla.org/en-US/docs/Web/HTML

CSS Documentation
https://developer.mozilla.org/en-US/docs/Web/CSS

Bootstrap Documentation
http://bootstrapdocs.com/v3.0.3/docs/css/

Glossary

51% Attack
When more than half of the computing power of a network is controlled by a single entity/group. This entity or group may issue conflicting requests to harm the network.

Accounts
An account consists of a user ID and a password. In terms of the blockchain, our user ID is our public address and our password is our private key. Every transaction is validated with your private key.

Blockchains
Blockchains are shared, trusted, and public ledgers of transactions. Everyone is able to see the transactions but no single user controls them. Blockchains are cryptographed, secure, and tamper resistant. They are distributed databases that are perfect for storing information such as values, identities, agreements, property rights, and credentials. Any information placed into a blockchain will remain there forever. Blockchains are decentralized, disintermediated, affordable, and censorship resistant. Various applications include Bitcoin, Namecoin, Sia, and Ethereum.

Block
A block is a file that is used to permanently record data. It is a record of some or all of the most recent transactions that haven't been recorded in any previous blocks. New blocks are added at the end of the blockchain and can't be changed or removed once they are written. Each block memorializes what happened in the minutes before it was created and keeps a record of some or all recent transactions as well as a reference to its previous block.

Block Explorer
A block explorer is an online tool used for exploring the blockchain. It allows for watching and following all transactions in real time on the blockchain. They can serve as blockchain analysis and provide information such as total network hashrate and transaction growth.

Block Height
The number of blocks connected in the blockchain.

Chain Linking
Chain linking is the process of connecting two blockchains. This allows for transactions between the two chains and also allows blockchains to communicate with other sidechains.

Client

A software program executed by a user on a desktop, laptop, or mobile device to launch an application.

Consensus
A consensus requires that an agreement is made among a number of processes for a single data value.

Consortium Blockchains
A consortium blockchain is a blockchain where the consensus process is controlled by a preselected set of nodes. Each user operates a node and they must sign every block for the block to be valid. The right to read the blockchain may be public or restricted to its participants. Consortium blockchains are considered to be “partially decentralized.”

Cryptographic Hash Function
A cryptographic hash function is a math algorithm that takes an input that can be any kind of digital data (ex: password file) and produces a single fixed length output. The main properties of cryptographic hash functions are as follows: easy to compute a hash value for any message; infeasible to generate a message from its hash function except through brute force; infeasible to change a message’s contents without changing the hash; infeasible to find two messages with the same hash; deterministic so that the same message always has the same hash output. Cryptographic hash functions may have security applications, indexing data in hash tables, fingerprinting, detecting duplicate data or uniquely identify files, and as checksums to detect accidental data corruption.

dApp (Decentralized Application)
An application is decentralized if it meets the following criteria: completely open-source, operate autonomously, and with no entity controlling the majority of its tokens and all changes being decided on by a consensus of its users; data and operation records must be cryptographically stored in a public and decentralized blockchain to avoid central points of failure; must use a cryptographic token necessary for access to the application; must generate tokens according to a standard cryptographic algorithm acting as a proof of valid contributing nodes.

Distributed Consensus System
A Distributed Consensus System(DCS) is usually composed of replicated state machines that utilize a consensus protocol (Proof of work, gossip about gossip, proof of stake, etc). This consensus protocol is used to guarantee the validity of your digital data and also make it very difficult for threat agents to corrupt, steal, or change any of that data. A DCS can utilize a number of mechanisms to implement these consensus protocols, such as a blockchain, DAG, or hashgraph.

Distributed Network
A network where processing power and data are spread over the network nodes instead of having a centralized data center.

Digital Signature
A digital code generated by a public key encryption that is attached to an electronically transmitted document to verify its contents and the sender's identity.

Ethereum
Ethereum is an open software platform that uses blockchain technology. It enables developers to write smart contracts and build and deploy decentralized applications.

Ethereum Wallet
An Ethereum Wallet is a piece of software used to connect to an Ethereum blockchain.

Fork
Forks are created when two blocks are created at the same time. This essentially creates two parallel blockchains with one of the two being the winning blockchain. The winning blockchain gets determined by its users by the majority choosing which blockchain the clients should listen to.

Genesis Block
The very first block in a blockchain.

Hash
Performing a hash function on the output data. This is used for confirming transactions.

Hardfork
A hardfork is a change to the protocol of the blockchain that makes previously invalid blocks or transactions valid. It requires all users to upgrade their clients.

Hashcash
Hashcash is a proof-of-work system that is used to limit email spam and DoS attacks.

Hashgraph

Light Node
A light node is a computer on the blockchain network that only verifies a limited number of transactions.

Lightning Network
A lightning network is a decentralized network that uses smart contract functionality on the blockchain to allow for instant payments across a network of participants. It allows transactions to happen instantly without worrying about block confirmation times. Lightning networks also allow two participants on the network to create an entry, conduct transactions between each other, and record the state of the transactions on the blockchain.

Merkle Tree
A Merkle tree's main concept is to have some piece of data linking to another. This can be accomplished by linking things together via a cryptographic hash. The content itself can be used to determine the hash and this allows us to address the content. The content becomes immutable because if you change anything in the data, the hash changes and the link is different. Every block points to the previous block and a block becomes invalid if it is modified.

Miner
A miner is an Ethereum node that is used to validate a transaction.

Mist Wallet
Mist Wallet is the software used to connected to the Blockchain network using its UI page. This software can also be used to connect with multiple blockchain networks. Mist Wallet lets you create accounts and transactions. It runs indirectly on top of the Ethereum Client software.
Node
A node is any computer that is connected to the blockchain network. Nodes are considered to be full nodes if they fully enforce all rules of the blockchain. Most nodes are lightweight nodes but full nodes form the network backbone.

Oracles
Smart contracts on the blockchain that cannot access the outside network on their own. Oracles sit between a smart contract and the external world. They provide data needed by the smart contract and send its commands to external systems.

Peer-to-Peer Network
The decentralized interactions between two parties or more in a highly-interconnected network. Participants of a P2P network deal directly with each other through a single mediation point.

Public Address
The cryptographic hash of a public key. These can act as email addresses that can be published anywhere unlike private keys.

Private Keys
String of data that allows you to access the tokens. These act as passwords that are kept

hidden from anyone but the owner of the address.

Private Blockchains
Private blockchains are blockchains where write permissions are centralized to a single organization. Read permissions can be public or restricted.

Proof of Authority (PoA)
A proof of authority is a method of consensus in a private blockchain. It gives one or more clients with one particular private key the right to make all blocks on a blockchain.

Proof of Work (PoW)
A proof of work system is a measure used to deter denial of service attacks and other service abuses such as spam. PoW systems require some work from the service requester. Typical work may include processing time by a computer.

Public Blockchains
A public blockchain is a blockchain that allows for anyone to read it, send transactions, and participate in the consensus process. Public blockchains are secured by crypto economics which is the combination of economic incentives and cryptographic verification by using proof of work or proof of stake. Public blockchains are considered to be fully decentralized.

Ring Signature
Ring signatures are cryptographic and provide a decent level of anonymisation on a blockchain. They ensure that individual transaction outputs cannot be traced. A message signed with a ring signature is endorsed by someone. It should be computationally infeasible to determine which group member key was used to produce the ring signature.

SHA (Secure Hash Algorithm)
A SHA is a group of cryptographic hash functions published by the National Institute of Standards and Technology. It takes an input of any size and form, mixes it up, and creates a fixed size output known as a hash. A hash can be viewed as a fingerprint of the data. They are one-way functions and cannot be decrypted back to their original form unless brute force is used.

Smart Contracts
Smart Contracts are computer protocols that facilitate, verify, or enforce the negotiation or performance of a contract. They usually have a user interface and emulate the logic of contractual clauses. They aim to provide security superior to traditional contract law.

Softfork
A soft-fork is a change to the protocol where only previously valid blocks are made invalid. It is

backward-compatible since old nodes will recognize new blocks as valid. This type of fork requires only a majority of miners to upgrade and enforce the new rules.

Solidity
A programming language used for developing Smart Contracts.

SVP (Simplified Payment Verification) Client
SVP Clients are lightweight clients that don't download and locally store the entire blockchain. They provide a way to verify transactions without having to download the whole blockchain. They only download the block headers by connecting to a full node.

State Channel
A state channel is an interaction that is made off of the blockchain without significantly increasing participant risk. This is done by moving interactions off of the chain which can lead to improvements in cost and speed. They function by locking part of the blockchain state so a set of participants can completely agree with each other to update it.

Swarm
A Swarm is a distributed storage platform and content distribution service. Its primary objective is to provide a decentralized and redundant store of Ethereum's public record. Essentially, it stores and distributes dApp code and data as well as blockchain data.

Token
A token is a digital identity for something that can be owned and is created as a sophisticated smart contract system with permission systems and interaction paths attached to them.

Testnet
A testnet is a second blockchain used by developers to test new versions of client software. This is done to prevent putting data at risk.

Transaction Block
Collection of transactions gathered into a block that can then be hashed and added into the blockchain.

User Keys (Public and Private)
Public Key: A User's address on the blockchain. (A long string of randomly generated numbers)
Private Key: Gives access to a User's digital assets.
Having a public and private key is what allows users to share digital assets through the blockchain safe and securely. You must safeguard and protect your private key however; and if you lose your private key you will be unable to retrieve your assets.

Wallet
A file that houses private keys and usually contains a software client that allows access to view and create transactions on a specific blockchain.

Web3.js
Web3 is a channel that is used to run ethereum commands from the front end or back end to interact with the blockchain.

Whisper
A whisper is a part of the Ethereum protocol that allows for messaging between users on the same network that runs the blockchain. Its main task is the provision of a communication protocol between decentralized applications.

References

[1]"What is Blockchain Technology? A Step-by-Step Guide For Beginners", Blockgeeks, 2017.

[2] E. Piscini, D. Dalton and L. Kehoe, "Blockchains and Cyber Security", Deloitte, 2017.

[3] "Know more about Blockchain: Overview, Technology, Application Areas and Use Cases - Lets Talk Payments", Lets Talk Payments, 2017.

[4]"4 Key Features of Blockchain", Techracers, 2017.

[5]M. Gupta, Blockchain for Dummies. for Dummies: A Wiley Brand, 2017.

[6] "Ethereum Project", Ethereum.org, 2017

[7] A. Gonsalves, "Cisco says blockchain ledger technology has networking role", SearchSDN, 2017.

[8] K. Leary, "Illinois is experimenting with blockchains to replace physical birth certificates", Futurism, 2017.

[9] N. Shimizu, "Blockchain's new client: Connected cars- Nikkei Asian Review", Nikkei Asian Review, 2017.

[10]M. Orcutt, "Why the CDC thinks blockchain can save lives", MIT Technology Review, 2017.

[11]"Using Blockchain Technology to Boost Cyber Security", Hacker Noon, 2017.

[12]A. Meola, "The growing list of applications and use cases of blockchain technology in business & life", Business Insider, 2017.

[13]"Blockchain Glossary." Blockchainhub, 16 October 2017.

[14]“Comprehensive Blockchain Glossary: From A-Z - Blockgeeks." Blockgeeks. 16 October 2017.

[15]"What Is Blockchain Technology? A Step-By-Step Guide For Beginners." Blockgeeks. 16 October 2017.

[16]"What are Blockchain's Issues and Limitations? - CoinDesk", CoinDesk, 2017.

[17] Antonopoulos, Andreas M. (2017-06-12). Mastering Bitcoin: Programming the Open Blockchain. O'Reilly Media. Kindle Edition.

[18] Church, Zach. “Newsroom.” Blockchain, Explained, 25 May 2017.

[19] Laurence, Paul. Blockchain: Step-By- Step Guide to Understanding and Implementing Blockchain Technology. Kindle Edition.

[20] P. Sandner, "Comparison of Ethereum, Hyperledger Fabric and Corda", Medium, 2017.

[21] S. Gr, "Ethereum Blockchain — concepts and terminologies – Seetharaman Gr – Medium", Medium, 2017.

[22]I. Lin and T. Liao, "International Journal of Modern Trends in Engineering & Research", 2017.

[23] H. Kancharana, "Blockchain 101 - A Beginner's Guide", Zinnov, 2017.

[24] "Blockchains & Distributed Ledger Technologies", BlockchainHub, 2017.

[25] **https://hashgraph.com**

[26] http://www.swirlds.com/downloads/SWIRLDS-TR-2016-02.pdf

[27] http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[28] https://squawker.org/technology/blockchain-just-became-obsolete-the-future-is-hashgraph/

[29]http://www.swirlds.com/downloads/Overview-of-Swirlds-Hashgraph.pdf

[30]https://iota.org/IOTA_Whitepaper.pdf

[31]https://domschiener.gitbooks.io/iota-guide/content/chapter1/current-role-of-the-coordinator.html

[32] https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419

[33] https://en.wikipedia.org/wiki/Quantum_Byzantine_agreement#Introduction

[34] http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[35] https://www.nasdaq.com/article/byzantine-fault-tolerance-the-key-for-blockchains-cm810058
[36] https://github.com/ethereum/go-ethereum/pull/1889

[37] **https://docs.corda.net/key-concepts-ecosystem.html**

[38] **https://docs.corda.net/_static/corda-introductory-whitepaper.pdf**

[39] https://lifeboat.com/ex/web.3.0

[40] http://ethdocs.org/en/latest/introduction/web3.html

[41] http://www.techradar.com/news/here-are-the-10-sectors-that-blockchain-will-disrupt-forever

[42] https://www.coindesk.com/5-blockchain-developments-coming-2018/

[43]
https://www.forbes.com/sites/bernardmarr/2018/01/22/35-amazing-real-world-examples-of-how-blockchain-is-changing-our-world/2/#3ce5ce071232

[44] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8038517

[45] https://medium.com/blockchain-blog/17-blockchain-platforms-a-brief-introduction-e07273185a0b

[46]
https://www.researchgate.net/publication/313249614_The_Blockchain_A_Comparison_of_Platforms_and_Their_Uses_Beyond_Bitcoin

[47] https://github.com/imbaniac/awesome-blockchain

[48] https://richtopia.com/emerging-technologies/review-6-major-blockchain-protocols

[49] https://www.b2bnn.com/2017/09/top-8-blockchain-platforms-check-now/

[50]
https://ethereum.stackexchange.com/questions/143/what-are-the-ethereum-disk-space-needs

[51] https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[52] https://www.hederahashgraph.com

[53] https://iota.org/IOTA_Whitepaper.pdf

[54]https://iota.org/IOTA_Whitepaper.pdf

[55]https://domschiener.gitbooks.io/iota-guide/content/chapter1/current-role-of-the-coordinator.html

[56]https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

[57] https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html#private-networks-with-the-sawtooth-permissioning-features

[58]https://sawtooth.hyperledger.org/docs/core/releases/latest/transaction_family_specifications/settings_transaction_family.html

[59]https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

[60] https://docs.corda.net/

[61] https://www.corda.net/

[62] https://www.r3.com/

[63] https://medium.com/think-consortium-on-blockchain/r3-corda-isnt-a-blockchain-what-does-this-mean-for-you-74cce6a09601

[64] **https://www.hyperledger.org/blog/2017/05/02/hyperledger-welcomes-project-indy**

[65] https://www.hyperledger.org/blog/2016/11/01/hyperledger-welcomes-iroha

[66] https://github.com/hyperledger/iroha/blob/master/docs/iroha_whitepaper.md

[67] https://github.com/hyperledger/burrow/blob/master/README.md

[68]https://hyperledger-fabric.readthedocs.io/en/release/fabric_model.html

[69] https://www.hyperledger.org/projects

[70] https://www.frankfurt-school.de/en/home/research/centres/blockchain.html

[71] https://iota.readme.io/docs/glossary

[72] https://hyperledger-fabric.readthedocs.io/en/latest/glossary.html

[73] http://fabrictestdocs.readthedocs.io/en/latest/glossary.html

[74] https://github.com/hyperledger/iroha/wiki/Glossary

[75] https://sawtooth.hyperledger.org/docs/core/nightly/master/glossary.html

[76] https://wiki.hyperledger.org/projects/indy/glossary

[77] https://docs.corda.net/glossary.html

[78] http://monsuite.readthedocs.io/en/latest/docs/glossary.html