Deciphering Crypto Networks

An Investigation into the Potential of Decentralization, Real-World DApps, and Cryptoeconomics

By: Parth Chopra

Advisor: Brett Hemenway Falk

April 25, 2018

Introduction

Since Satoshi Nakamoto's momentus 2008 Bitcoin paper, there has been an explosion of interest into cryptography, blockchains, decentralized networks, and the cryptocurrencies built on top of these technologies. Currently, all cryptocurrencies have a combined market cap ~0.5 trillion dollars[1], and millions of dollars are being spent by existing organizations as they try to understand the underlying technology, and how it can help their individual businesses[2]. Developer activity working on new and existing protocols has increased tremendously, and we're seeing larger players (from businesses to institutional investors) begin to enter the space with both excitement and trepidation. Yet at the same time, the monetary attractions of crypto networks has also brought a lot of noise into coverage of the industry, and its become tough to delineate between reality, tangible and achievable goals, and fanciful dreams.

Specifically, there is still a lot of confusion on how exactly the underlying technology works, what the state of development is, the vision of the technology, and where it should actually be applied. This paper attempts to provide a more technical overview of the field and provide clearer frameworks on how to think through applications and limitations. Keeping this in mind, it may be helpful to further define the scope of this paper to the following three domains:

1) <u>Centralization/Decentralization (Section 3):</u> Understanding granularly what these terms mean, how to measure decentralization, and why it's beneficial and often seen as a goal for crypto networks.

2) <u>Making Decentralized Applications Useful (Section 4)</u>: Understanding how smart contracts and the Ethereum Virtual Machine (EVM) work, the current state in bringing external data onto blockchains, and attempts at interoperability between new protocols and legacy systems.

3) <u>Cryptoeconomics (Section 5):</u> Understanding the intersection between technical and incentive design to achieve particular goals. A particular focus will be given on consensus mechanisms, and this section will also look at Stablecoins as a case study.

Preceding the above topics will be a background in Section 2 where terminology, historical context, and an overview of the Bitcoin protocol will be provided. Each of the core topics will also include additional subsections to provide background, context, and real-world examples. We'll finally conclude with a holistic discussion that ties together core ideas and provides an examination on the future of crypto networks.

[1] ("Cryptocurrency Market Capitalizations | CoinMarketCap," n.d.)
[2] ("Blockchain Investment Trends In Review," n.d.)

2. Background

2.1 Defining Key Terminology

To begin the discussion, it may be helpful to define some of the key terms commonly used in the space. Given the interdisciplinary nature of the industry, we're hoping that a precise definition can provide contextual reference for the rest of the paper.

- Decentralized Networks: Formally in systems, a decentralized network is one where the allocation of processing and storage is done across a wide spectrum of devices rather than one single entity. When applied in an application context, it refers to a service with no significant central authority that dictates activities on a network.[3] This paper will dive more into the nuances of centralization/decentralization in Section 2.

- Blockchain: A blockchain is a sequence of records (called blocks) that are linked and secured together using cryptographic techniques. The popular instantiation involves using a decentralized network to agree on a sequence (temporal ordering) of the blocks. For public blockchains, each block is agreed upon and distributed across multiple clients on a public ledger. This makes data corruption very hard since every interlinked block in the network needs to be updated as well.[4]

- Cryptocurrencies: For most of human history, we've had some sort of central authority that everyone believes and trusts in and verifies that transactions take place. For example, if I pay a merchant with my credit card, we are trusting the banking entities that they will subtract X amount from my account and add it to the account of the merchant. The central authority is in charge of ensuring the transaction took place. Digital currencies aim to replace the single trusted entity with a decentralized network that stores and validates transactions. To make this work, we put into place a decentralized network of nodes that work to secure "blocks" of transactions onto the public ledger. Since everyone has access to the public ledger, we are able to for the first time guarantee trust without a central authority.[5]

- Smart Contracts: Software protocols that help agreements between two parties take place seamlessly, efficiently, and securely. Usually such contracts very specifically define what the conditions and corresponding actions should be, and are designed to lower transaction costs.[6]

- Crypto Networks: A generalized term referring to the entire ecosystem surrounding a cryptocurrency/project.[7] This includes the contextually relevant consensus mechanisms,

[3] (Buterin, 2017)
[4] (Economist, 2015)
[5] (Economist, 2015)
[6] (Economist, 2015)
[7] (Young, 2017)

blockchain governance, economic incentives, developer community, etc.[8] It will also be the de facto terminology used by this paper.

2.1 History of Crypto Networks

While Satoshi Nakamoto's Bitcoin paper ushered cryptocurrencies and cryptonetworks into the mainstream, it wasn't the first attempt to create a digital currency. We can trace that back to cryptography researcher David Chaum, who introduced the idea of "blind signatures" in 1983, and used that to build a digital currency called DigiCash in 1990. Rather than approach it from a completely decentralized approach, the currency instead was used more as what researcher Peng writes, "an anonymous pre-paid credit card". Ultimately because of failed partnerships and poor management, DigiCash went bankrupt.[9]

Inspired by Chaum's work was a cryptographic researcher named Nick Szabo, who in 1996 introduced and coined the term *smart contracts* (see formal definition in 2.1) in his publication *"Smart Contracts: Building Blocks for Digital Free Markets".* The idea of protocols that are programmatically built, secured, and maintained would go on to become the foundational layer for later projects.[10] Szabo continued his research and in 1998, published a proposal for a new digital currency called Bit Gold. Novel in this proposal was the idea of *unforgeable proof of work chains* that would become the architecture concept that Bitcoin would be built on top of.[11]

Cryptographic research and protocol design continued into the 2000s but applications began to focus particularly on gold-backed digitized currencies. These were essentially just digital currencies with gold collateral, and was a manifestation of the counterculture movement against fiat currency. The most popular of these projects was E-gold, which was started by oncologist Douglas Jackson, and at its peak in 2008 was processing over $2 billion in transactions with 5 million users scattered throughout the world. However, E-gold's success was short lived, and the US Department of Justice in 2008 filed the founders with "several counts of money laundering, conspiracy, and operating an unlicensed money transmitting business" and they were forced to shut down operations.[12]

Perhaps a little ironically, 2008 was also the year that Satoshi Nakamoto released the Bitcoin white paper, which described a "purely peer-to-peer version of electronic cash [that] would allow online payments to be sent directly from one party to another without going through a financial institution"[13]. The paper and protocol were released completely anonymously, and to this date it is still a mystery as to who Satoshi Nakamoto is. However, the success of his/her/their work is apparent, and one only needs to look at the market cap of Bitcoin over the last decade to see its massive growth and popularity:

[8] (Dixon, 2018)
[9] (Peng, 2013)
[10] ("Nick Szabo -- Smart Contracts: Building Blocks for Digital Markets," 1996)
[11] ("Bit Gold proposal - Bitcoin Wiki," n.d.)
[12] (Peng, 2013)
[13] (Nakamoto, 2008)

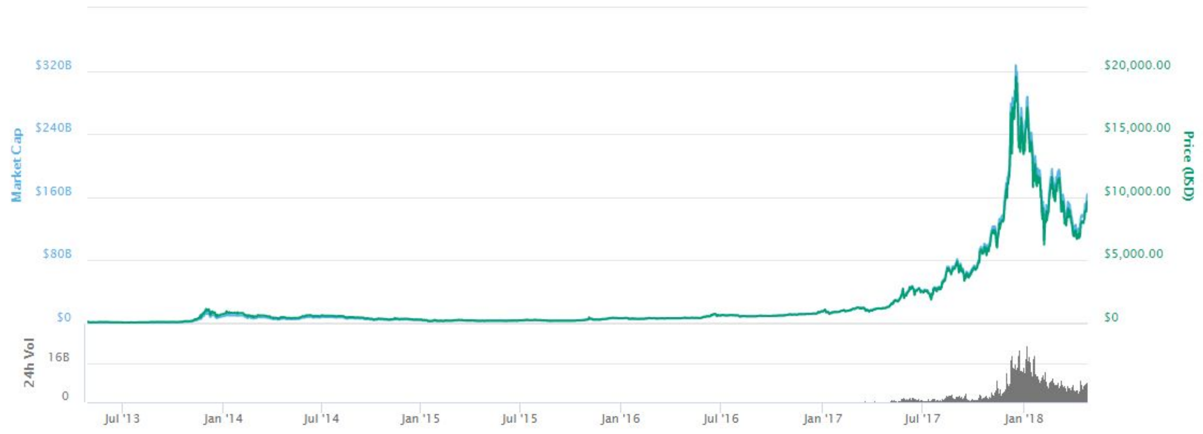

Image Credit: CoinMarketCap[14]

When we talk about crypto networks more broadly, however, there was another major turning point in the ecosystem in late 2013. This was when Vitalik Buterin, a researcher and contributing member in the Bitcoin community, described and published the Ethereum white paper to create a protocol to run smart contracts and decentralized applications. Alongside co-founder Dr. Gavin Wood, the Ethereum yellow paper was then released and was used to create the initial Ethereum clients. By mid-2014, Ethereum had conducted a pre-sale to raise 31,591 bitcoins ($18,439,086 at that time) that was used to finance development and operations of the Ethereum network.[15] Since then, the popularity of Ethereum and the value of its token, Ether, has also exploded:

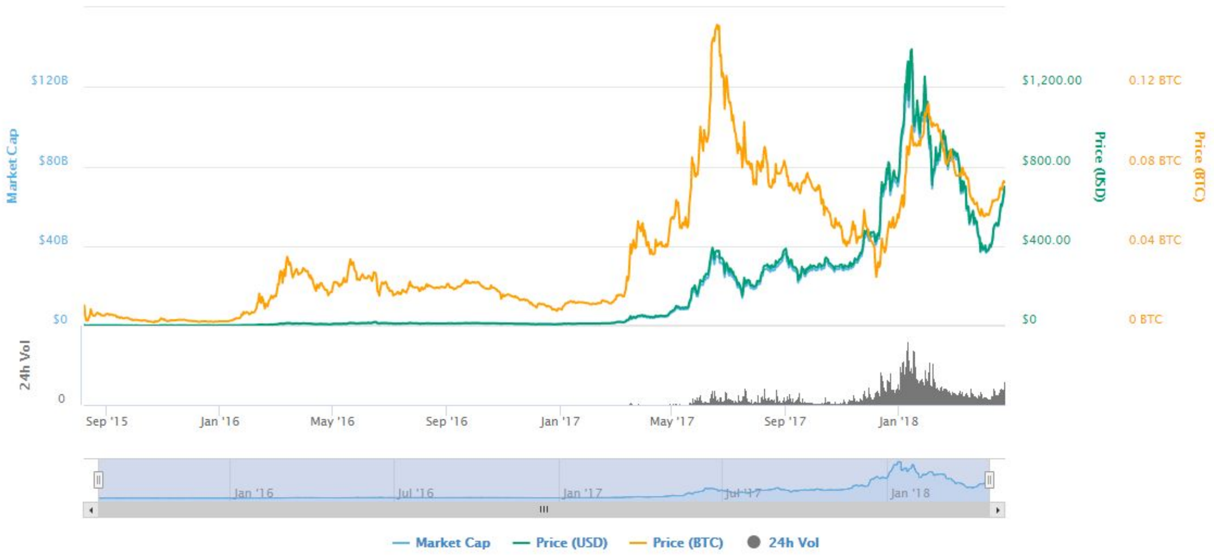

Image Credit: CoinMarketCap[16]

[14] ("Bitcoin (BTC) Price | CoinMarketCap," n.d.)
[15] ("History of Ethereum — Ethereum Homestead 0.1 documentation," n.d.)
[16] ("Ethereum (ETH) price," n.d., "History of Ethereum — Ethereum Homestead 0.1 documentation," n.d.)

This leads us to present day, where at the time of writing, there are 1399 decentralized applications (DApps) built on top of the Ethereum network, and the sum market cap of all cryptocurrencies is near half a trillion dollars.[17] [18]

2.2 An Overview of the Bitcoin Protocol

Given that the Bitcoin protocol and the concepts instilled in it are the inspiration for how the rest of crypto networks are built, we'll spend some time taking a look at the protocol from a broad viewpoint. The review below uses Satoshi's original white paper and a technical overview done by author Omar Fernandez.[19]

In the Bitcoin protocol, users transfer digital coins to another user in a process called a *transaction*. Each user has a public-private key pair, where the private key is a random number generated by a Bitcoin wallet software, and the public key is generated using the *Elliptic Curve Digital Signature Algorithm (ECDSA)*. The transaction is created by combining the recipient's public key, the previous transaction, and a digital signature of the sender's private key. Given the nature of the way digital signatures are chained, it is fairly easy to verify signatures, and thus verify the chain of ownership.

Because the entire system is decentralized and the goal is to remove a central authority keeping track of what happens, the protocol must find a way to maintain a *public ledger* and guarantee trust in the system. Bitcoin does this by having all transactions be known and have the ledger be present on millions of computers. More specifically, the protocol begins by combining a lot of transactions into something called a *block* and then cryptographically chains the blocks together to form a *blockchain*. In addition to all the transactions, each block also contains a timestamp, a variable called a *nonce*, and the hash of the previous block in the chain.

In order to "cryptographically chain" blocks together, and more importantly, to determine *consensus* that transactions actually occurred as stated, the protocol uses something called *proof-of-work* to ensure that enough computational resources have been expended to accept a block to an existing chain. The technical details for the Bitcoin implementation of proof-of-work are given in section 5.1.1. Essentially then, nodes in the network receive transactions, process them into blocks, and solve the blocks using proof-of-work. They then broadcast it to other nodes, and if nodes accept the block they start again on a new block, but this time using the hash of the accepted block. Nodes always consider the longest chain, and this becomes the ground truth and the "public ledger" that was referred to earlier.

Nodes whose primarily role is to solve the blocks using proof-of-work are known as miners, and whenever a block is solved and accepted, they receive newly minted Bitcoin (and transaction fees if applicable). This provides them with an incentive to continue mining and securing the system. Because of the decentralized nature of Bitcoin, the protocol is also susceptible to something called the *51%*

[17] ("State of the ĐApps — 1399 Projects Built on Ethereum," n.d.)
[18] (Fernandez, 2017)
[19] (Nakamoto, 2008)

attack which is when a group of nodes command greater than majority of the computation power in the network. If this occurs, the group can then compromise the system by undoing previous blocks and also choosing untruthful transactions/blocks to include in the main chain.[20] [21]

3. Understanding Decentralization

One of key reasons behind the rise of crypto networks is this notion of decentralization, and how a movement in that direction will lead to a substantially better state. In fact, we can turn to one of Satoshi Nakamoto's original blog posts launching Bitcoin, where in the second sentence he claims "...It's completely decentralized, with no central server or trusted parties, because everything is based on crypto proof instead of trust".[22] Yet the term decentralization is more nuanced than what is seen at first glance, and there are several issues one has to manage in measuring it. It is important to really understand what decentralization means because it serves as the underpinning for why crypto networks exist and are designed the way they are. In this part of the paper, we'll attempt to color the concept of decentralization, see how to measure it, and understand the benefits and tradeoffs.

3.1 Different Axes of Decentralization

When we talk about centralization/decentralization, it is important to contextualize it to a specific domain. When we look at software specifically, there are three major types of decentralization we can focus on: architectural decentralization, political decentralization, and logical decentralization.[23]

<u>Architectural decentralization</u> takes a look at the specific physical number of computers or other devices that makes up a system.[24] This concept can be applied to any software system - web services such as Netflix, Google, Facebook, etc. employ massive numbers and various types of devices, processors, and servers to maintain their systems. When we look at the internet more broadly, we can see that there were 30 million servers in 2008, with an estimated 75 million by 2013.[25] The analogy can be even extended to the undersea cables that control connections between different networks. In such a case, even though internet servers might be decentralized, trans-oceanic connections may actually be fewer (just given the complexity of deployment) and can be considered as architecturally centralized in that specific domain. Given this context and understanding, public blockchains can be considered as architecturally decentralized.

<u>Political decentralization</u> looks at the entities that actually *control* the system - that is the number and concentration of individuals and organizations with control. For example, corporations are often politically centralized with one central management team that has the authority and power to dictate the direction of the firm. In this context, public blockchains are politically decentralized as there is no

[20] (Nakamoto, 2008)
[21] (Fernandez, 2017)
[22] ("Satoshi Nakamoto Institute," 2009)
[23] (Buterin, 2017)
[24] (Buterin, 2017)
[25] (Richard, 2013)

one person or entity that controls them (Ex. There is no one individual that controls Bitcoin like there is a group that controls Google).[26]

Logical Decentralization looks at the degree of centralization of the interface, data structures, and logic systems that define a particular software application. One key heuristic recommended by Vitalik Buterin, is asking "if you cut the system in half, including both providers and users, will both halves continue to fully operate as independent units?".[27] We can actually take a look at nature for an example - when honey bee colonies grow too large, they engage in a process called swarming where half of the colony leaves with a new queen to form a new colony.[28] Bee colonies would thus be considered as logically decentralized because both colonies now operate independently of each other. When we look at blockchains under this definition of logical decentralization, we can see that they are logically *centralized* because there is one agreed upon state/ledger.[29] Ethereum, for example, transitions from one state to the next, and the entire system acts like a single entity. That's not to say splits can't happen - the evidence of hard forks in the Bitcoin (Bitcoin Cash[30]) and Ethereum (DAO fork[31]) networks suggest as much - but each split led to an entity that while independent, was quite different in functionality, characteristics, and goals.[32]

3.2 Measuring the Amount of Decentralization

So now that we've defined the different variations of decentralization, we can now look at different methods to actually measure the amount or degree of decentralization. Given the technical, economic, and political structure of crypto networks, there are a couple of different ways to do this.

3.2.1 Network Measurements

If we approach crypto networks from a pure graph theory perspectives, we can begin to apply traditional centrality analysis to understand the dynamics. Using a 2017 paper by Maesa, Marino, and Ricci as a guide, we can list key measurements used in analyzing a crypto network:[33]

Centrality Measurement	**Description**
Degree	The number of nodes that a node, *u*, has transacted with (both $u \rightarrow u_1$ and $u_1 \rightarrow u$)
In-Degree	The number of nodes that have transacted with *u* ($u_1 \rightarrow u$)

[26] (Buterin, 2017)
[27] (Buterin, 2017)
[28] (Morning Chores, 2017)
[29] (Buterin, 2017)
[30] ("A Short Guide to Bitcoin Forks - CoinDesk," 2017)
[31] ("A Short Guide to Bitcoin Forks - CoinDesk," 2017)
[32] (Buterin, 2017)
[33] (Di Francesco Maesa Andrea Marino Laura Ricci, 2017)

Out-Degree	The number of nodes that *u* has transacted with (*u* → u_1)
Harmonic	Defining centrality by the closeness in distance a node is from other nodes. More mathematically, it is defined as $\sum_{v \varepsilon V^{t}} \frac{1}{d(u,v)}$ where *d(u,v*) is the distance between nodes *u* and *v*
Page-Rank	Measures the "influence" of visiting a particular node. More formally, it can be defined by the equation $x_i = \alpha \sum_j a_{ji} \frac{x_j}{L(j)} + \frac{1-\alpha}{N},$ where $L(j) = \sum_i a_{ji}$ (*Image Credit: Wikipedia)*[34]
Eigenvector	Also measuring the influence of a particular node, with the assumption that connections to high-influence nodes are more valuable. Again, more formally, this can be defined by the equation $x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$ *(Image Credit: Wikipedia)*[35]

3.2.2 Inequality Measurements

Given the parallels of most crypto networks to economic systems, we can also bring in tools and methodologies from macroeconomic research. For our case, we can look more specifically at inequality research and different equations used to quantify that amount. Chief among them is the Gini Coefficient, which measures what concentration of wealth a segment of the population owns (ex. What % of wealth do the top x% own). More formally, the Gini coefficient G is defined by:

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2 \sum_{i=1}^{n} \sum_{j=1}^{n} x_j} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n \sum_{i=1}^{n} x_i}$$

Image Credit: Wikipedia[36]

Where x_i is the wealth of person *i* and the size of the population is *n*. The Gini coefficient is often visualized by the Lorenz curve which shows the amount of inequality. The ratio of the area of the curve from the 45-degree line to the Lorenz curve, divided by the total area under the 45-degree line is the Gini coefficient. A Gini coefficient of 1.0 is perfect inequality, and a coefficient of 0.0 is perfect equality.

[34] ("Centrality - Wikipedia," n.d.)
[35] ("Eigenvector centrality - Wikipedia," n.d.)
[36] ("Gini coefficient - Wikipedia," n.d.)

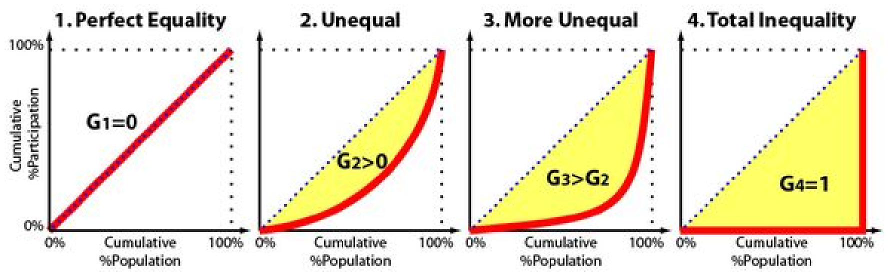

Image Credit: Matthew John [37]

We can extend the definition of the Gini coefficient to different protocol networks, and choose how we are defining "wealth" and the type of the population (ex. Owners of a coin, devices participating, etc.). Balaji Srinivasan and the team at Earn did something similar, where they defined Gini coefficients for various subsystems of Bitcoin and Ethereum. Specifically, they found the Gini coefficient for Mining (by reward), Client (by codebase), Developers (by commits), Exchanges (by volume), Nodes (by country), and Ownership (by addresses).[38]

From there, they developed a new coefficient called the *minimum Nakamoto coefficient* which is trying to measure the number of entities needed to compromise a subsystem (meaning get > 51% control). Under this methodology, one could find the Nakamoto coefficient for each subsystem, and then let the overall Nakamoto coefficient of the *entire system* be the minimum value from the Nakamoto values of the different subsystems. More generally, $N_{min} := min \{N_1, ..., N_S\}$ where N_s is the Nakamoto coefficient of subsystem s.[39] Such a precise definition does shed more light into the decentralization of a network, but as pointed out by the Nakamoto Institute, there are problems in 1) selection of subsystems, and 2) objectively and *accurately* measuring values in the chosen systems.[40]

Nevertheless, it is safe to say that with a new crypto network, multiple metrics need to be carefully and contextually chosen and measured to try to get a relative sense of the degree of decentralization.

3.2 Benefits of Decentralization

We can now attempt to address the question of if and why we should strive for decentralization. What are the benefits of decentralization? We'll begin by listing and addressing the major arguments.

<u>Fault Tolerance:</u> The classic statement, "Quis custodiet ipsos custodes?" (Who will watch the watchmen?) provides us with a good analogy for the single point of failure risk of a centralized entity.[41] When there is a concentration of power/resources in one single part of a system, removal of that part can lead to an entire system collapse. Decentralization aims to reduce this concentration by relying on

[37] (Srinivasan, 2017)
[38] (Srinivasan, 2017)
[39] (Srinivasan, 2017)
[40] ("Against the Minimum Majority Measure | Satoshi Nakamoto Institute," 2017)
[41] (Atzori, 2015)

many elements (some redundant, some not).[42] However, it's not enough just to create redundant systems, but instead it's necessary to make sure that the redundant components are statistically independent which can avoid *common mode failure.[43]* For example, a crypto network may have multiple nodes, but if they all run the same software, and that software has the same bug, then these nodes are *not* statistically independent and may all fail simultaneously.[44]

Attack Resistance: In cybersecurity, the surface of potential vulnerability is called the *attack surface* and generally the more complex the application, the larger attack surface it may have (for example, Ethereum has a larger attack surface than Bitcoin because the network is running actual scripts).[45] However, the economic nature of crypto networks and combined with decentralization changes the calculations of whether it makes sense to attack or not. For example, Vitalik Buterin points out an example where while it's easy for you to blackmail one person for $50M, it's much harder to do so if that money was spread out over ten people where all ten people would have to be blackmailed simultaneously (which of course raises the cost of the attack).[46] This of course has parallels to the commonly talked about 51% or majority attack first mentioned in Satoshi's white paper, which simply states that the security of a crypto network is compromised if the majority of CPU power coordinates to attack the network.[47] [48] However, new research has also shown that you might not necessarily need majority, and that at least in the Bitcoin network, selfish mining is possible by a pool of miners as small as ⅓.[49]

Collusion Resistance: Most crypto networks rely on the idea of *uncoordinated choice model* which is a game theory concept where all actors in a system have independent and separate incentives, and these incentives are all smaller than size X.[50] However, as mentioned briefly in the attack resistance section, there are incentives to collude and work together to achieve outsized profits. In parallels to something like the Tragedy of the Commons, the short-term benefits would only accrue to the colluders and the system would collapse when the vulnerability is exposed. Decentralization aims to prevent this collusion, and in crypto networks a combination of 1) technical protocol design, 2) economic incentives, and 3) social intervention, are employed to try to avoid collusion among participants. However, it is important to note that in crypto networks (especially premature networks) a certain balance is needed - these are evolving protocols, so you still want enough coordination for changes to be made to improve the protocol.[51]

Trust Ranking: As we start to talk more about bringing in real-world information on-chain, we need to start considering the trustworthiness of the data sources, and *the trustworthiness of the entity assigning*

[42] (Buterin, 2017)
[43] ("Common cause and special cause (statistics) - Wikipedia," n.d.)
[44] (Buterin, 2017)
[45] ("Attack Surface Analysis Cheat Sheet - OWASP," n.d.)
[46] (Buterin, 2017)
[47] (Nakamoto, 2008)
[48] ("51% Attack, Majority Hash Rate Attack - Bitcoin Glossary," n.d.)
[49] (Eyal & Sirer, 2014)
[50] ("Introduction to Cryptoeconomics," 2017)
[51] (Buterin, 2017)

trustworthiness. While this is a seemingly meta point, most crypto networks maintain a global state and verify every transaction through a consensus protocol. Decentralization of trust allows multiple entities to verify the trustworthiness and condition of a state rather than a central authority.[52]

[52] ("Decentralized Trust Management in Peer-to-Peer Systems - IEEE Conference Publication," n.d.)

4. Making Decentralized Applications Useful

The introduction of Ethereum by Vitalik in late 2013, and the subsequent development with Dr. Gavin Wood expanded and made mainstream the horizon of what is possible to do with blockchains.[53] Their work (and the work of countless other developers) in creating a "Next-Generation Smart Contract and Decentralized Application Platform" created the base foundation to allow a new generation of decentralized applications to be built on top.[54] Decentralized applications, or DApps, are software applications that accomplish a particular task by using decentralized protocols. DApps can have their own blockchain, be built on top of existing one (like Ethereum), or a combination of both. While DApps vary in the degree of usefulness, the value add for creating and using them involves a combination of decentralizing operations, capturing the value of a network in a tradeable economic asset, token and funding mechanism, etc.[55]

Since Ethereum's release, there have been countless applications built on top of the network, with varying degrees of usefulness. More so, some principles and technical concepts from the Ethereum project have been adapted by other protocols. While the state of the field is very exciting and a lot of progress has been made, it is still very young, and thus often times expectations of the technology don't match reality. This section will begin by providing a background into Ethereum smart contracts and its virtual machine, and the current attempts to make it more useful in the context of 1) Oracles (and bringing external, real-world data on-chain), and 2) Interoperability. Note that while decentralized applications are of course not restricted to just those built on top of Ethereum, a special focus is given to this protocol and its descendants because of the high amount of activity on or in relationship with Ethereum.

4.1 Background on Smart Contracts and the Ethereum Virtual Machine (EVM)

Fundamentally, Ethereum is a transaction-based state machine that is quasi-Turing-complete (quasi coming from bounding the computation with a parameter called *gas*). There are two possible accounts that can be created in the network - external accounts and contract accounts with the difference being that contract accounts cannot initiate new transactions on their own (but can fire a transaction to activate another contract account). Transactions are what helps transition the machine from one state to another state, and these transactions are grouped together into immutable blocks that are verified through a consensus protocol. With this general overview, we can now dive deeper into the components of accounts, transactions, and the EVM and transaction execution. The following section about Ethereum is a summary of the Ethereum yellow paper and a helpful technical overview done by blockchain engineer Preethi Kasireddy.[56] [57]

[53] (Kariappa Bheemaiah, 2015)
[54] (ethereum, n.d.)
[55] (Brown, 2016)
[56] (Kasireddy, 2017a)

4.1.1 Ethereum Accounts

Again, there are two different types of accounts in the Ethereum network - external accounts (called "non-contract" accounts in the yellow paper) and contract accounts. External accounts are controlled by private keys and can send messages to other external accounts or directly to contract accounts. Contract accounts, on the other hand, are designed to hold scripts and code within them (hence we see the emergence of the term "smart contracts") and cannot initiate transactions on their own, but can fire a transaction to activate another contract account (called an "internal transaction"). The account state, formally defined as $\boldsymbol{\sigma}[a]$, consists of the following fields:

1) <u>Nonce:</u> Formally denoted as $\boldsymbol{\sigma}[a]_n$, this is a scalar value that for external accounts is the number of transactions sent *from* the address, and for a contract account, it's the number of contract-creations done
2) <u>Balance:</u> Value stored by the current account and formally denoted as $\boldsymbol{\sigma}[a]_b$
3) <u>StorageRoot:</u> A hash of the root node of a Merkle Patricia tree and formally denoted as $\boldsymbol{\sigma}[a]_s$
4) <u>CodeHash</u>: An immutable hash of the EVM code associated with the account and formally denoted as $\boldsymbol{\sigma}[a]_c$. For external accounts, this field is just the hash of an empty string (formally, $\boldsymbol{\sigma}[a]_n$ = KEC(()) where KEC is the Keccak-256 hash function

4.1.2 Transactions

Transactions are cryptographically-signed instructions that either result in a message call, or a contract creation. A message call contains a field called *data* which stores the byte array of a call. A contract creation, on the other hand, does exactly what it says - it calls an *init* function that is an EVM-code fragment that manifests a new account with the appropriate parameters (ex. Setting the right ether balance). Ethereum also supports internal transactions, where the code in contract accounts creates new contract accounts and ad infinitum until the gas runs out or execution sequence is finished. Combining Ethereum accounts with transactions gives us the following diagram to showcase how accounts and transactions interact.

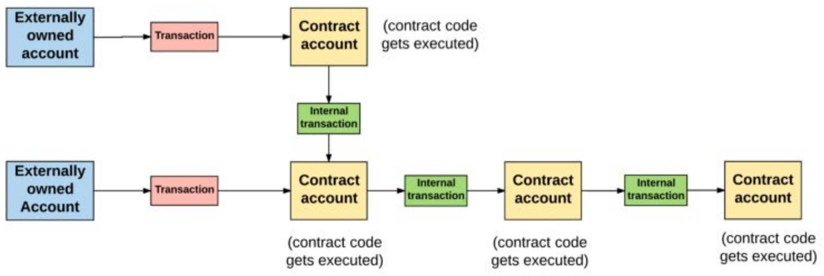

Image Credit: Preethi Kasireddy[58]

In addition to the respective *data* or *init* fields (again, depending on the transaction type), all

[57] (Wood, n.d.)
[58] (Kasireddy, 2017a)

transactions have the following fields:

1) Nonce: Number of transactions sent by the sender
2) Gas Price: Amount of ether the transaction is willing to spend on a unit of gas (gwei)
3) Gas Limit: Maximum amount of gas that can be allocated to the transaction
4) To: 160-bit address of the recipient
5) Value: Amount of ether to be set to the value of the recipient account
6) V, r, s: Signature of the transaction

4.1.3 Ethereum Virtual Machine (EVM) and Transaction Execution

The EVM is the heart of the Ethereum network and handles the actual running and processing of transactions. Unlike the typical von Neumann architecture, the EVM stores program code in a virtual ROM and access is only granted through specific and specialized instruction. The EVM uses a stack-based architecture (LI-FO), and each stack has a maximum size of 256-bits which was chosen to better facilitate the Keccak-256 hash scheme. Memory is volatile but storage is not and is maintained as part of the system state. A visual overview of the EVM is provided below:

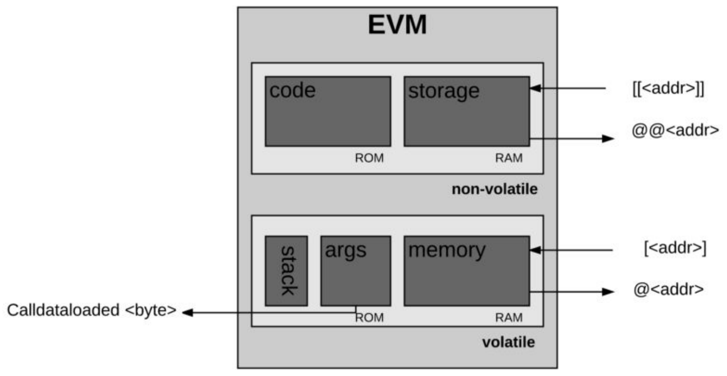

Image Credit: Preethi Kasireddy[59]

The EVM has its own language called *EVM bytecode* and the higher-level language typically used is called *Solidity*. Because each transaction can only occur if a certain amount of gas is used, each EVM instruction has a defined cost that is maintained by the Ethereum developer community. In addition, during the execution of each transaction, the EVM keeps track of the *substate* which holds any record information that may be needed.

We can now look at the execution of a transaction, and how we move from one state to the next. More formally, we can start with the system state σ, the gas amount *g*, the accrued substate *A*, and resultant output **o.** We then define a function Ξ such that:

$$(\sigma', g', A, o) \equiv \Xi(\sigma, g, I)$$

[59] (Kasireddy, 2017a)

Where σ', g' are the resultant state and remaining gas respectively, and I is a tuple that contains execution information. The transition function, Ξ, takes responsibility for ensuring valid transactions, and executing any code if necessary. It also ensures the gas payment procedure is properly executed. On the latter point, this procedure involves making sure that there is enough available gas, and if the gas *does* run out during execution, then the necessary self-destruct and state reversal methods occur.

4.2 Oracles: Bringing External Data On-Chain

Now that we have a general understanding of how Ethereum creates and processes smart contracts, we can dive more into the attempts to make the contracts more useful. Chief among this is bringing in data from external environments onto a blockchain (commonly referred to as "bringing data on-chain"). When we look at the problem, the key design of blockchains turns out to be a weakness - every single node has to process every transaction and reach consensus. Thus, because of the deterministic requirement, *you can't have smart contracts access external environments* like the Internet or another contract's storage. [60] [61]

The workaround to solve this technical problem is with something called *Oracles*. The key concept here is that rather than have a contract make an external call to seek data (which again isn't possible), a trusted authority *puts data onto the chain* which is then used. Every node gets an identical copy of the data and contracts can also use it to conduct whatever action they originally wanted.[62] The exact technical implementation on how to do this (and specifically, how much of the entire procedure is kept on-chain vs. off-chain) varies depending on what framework you're using. To illustrate this we've chosen the popular service called Oraclize as a case study.

4.2.1 Case Study: Oraclize

Oraclize is a European team that created an API service to check external data, and pass the data on-chain when needed. The procedure, generalized from looking at their various code examples, is as follows:

1) A smart contract is created that conducts at least the following two key operations:
    a) Sends a call to an Oraclize contract on-chain (specifically `oraclize_query`). This query can also be scheduled for the future.
    b) Defines a `__callback` function that takes as input, data from a future call
2) Oraclize has an external node that monitors the contract and sees the query
3) This external node executes the query off-chain and gathers the requested data
4) One of two things can now happen:
    a) The external entity directly calls your `__callback` function and sends it the data
    b) The external entity pushes the data to the Oracalize contract which then calls your `__callback` function

[60] ("Is there a way to give access of an external storage mechanism only to a smart contract?," n.d.)
[61] ("Why Many Smart Contract Use Cases Are Simply Impossible - CoinDesk," 2016)
[62] ("Why Many Smart Contract Use Cases Are Simply Impossible - CoinDesk," 2016)

5) The `__callback` function executes the actions with the now given data[63] [64] [65]

Oraclize can also provide authenticity proofs if necessary to verify the data. However, because the overall procedure requires multiple calls that have to go through the entire network, it can typically take 1-2 blocks for the data to make it back to the `__callback` function.[66] Lastly, Oraclize has payment mechanisms that help to pay both for the service and the gas costs to have the contracts make the query and callback transactions.[67]

The Oraclize case study provides us with an actual way that developers have put the Oracle philosophy in-production. However, even though this is state-of-the art, it's very clear that there are limitations to this strategy. First and foremost is privacy, especially if we're dealing with identifiable private information (ex. Banking information, health records, etc.) Since by nature of the way crypto networks work, every node in the system needs a copy of the data being used (thus increasing overall exposure) and because transactions need to be triggered, the data may be paired with public addresses leading to doxxing. While there are some possible solutions that cryptographically hash the necessary data in development, it does add an added layer of complexity and brings its own problems.[68]

Next is speed - with the Oraclize case study, there are at least two calls to get the data (and a third one if you want to do anything with that data). Scaling in Ethereum is a paper in itself, but trying to create a lasting application that requires fast actions is even more difficult if you need to make 2-3 calls per request. And lastly, going back to our earlier discussion about decentralization, there is inherent risk in ensuring the trust to a third-party Oracle. How can you trust and verify the authenticity of the Oracle? [69] [70]

Yet, despite all of this, there is also reason to be optimistic. The concept and execution of Oracles has at least given a good proof-of-concept for what a world where blockchain can access external data may look like, and a lot of development work is being done on the problems listed above.

4.3 Interoperability with Legacy Systems and Between Chains

Having addressed the challenge of bringing external data on-chain, we can now take a look at the challenges and solutions associated with interoperability with legacy systems. This is important because, as Thomas Hardjono at MIT Connection Science points out, *"...[a] company [that] is trying bring in new technology, this integration is a cost item. And people evaluate the ROI by also building in this cost of integration".*[71] In order for scalable real-world use of crypto networks, there has to be some level of

[63] ("How Oraclize and API call works," 2017)
[64] ("Oraclize Documentation," n.d.)
[65] ("How do oracle services work under the hood?," 2017)
[66] ("How to wait Oraclize result before running code further?," n.d.)
[67] ("Oraclize Documentation," n.d.)
[68] ("Why Many Smart Contract Use Cases Are Simply Impossible - CoinDesk," 2016)
[69] (Kasireddy, 2017b)
[70] ("Why Many Smart Contract Use Cases Are Simply Impossible - CoinDesk," 2016)
[71] (Capgemini Consulting, 2017)

interoperability with existing networks. In addition, with multiple protocol projects that are suited for different uses, there may also be interoperability requirements between different chains and protocols. As in the previous section, one of the best ways to highlight this is by looking at an actual case study. This is especially true when looking at legacy systems because the use case tends to be very industry specific.

4.3.1 Case Study: The Interledger Protocol (ILP)

An ambitious project started by engineers at the company Ripple, the Interledger Protocol is an attempt to create a protocol for payments across different payment systems. The project was invented in 2015 by Stefan Thomas and Evan Schwartz and is now run by the Interledger W3C Community Group. It is inspired by the Internet architecture first described in RFC 1122, FRC 1123, and RFC 1009 and hopes to create a system and standard where “sending value will be as easy as sending information is today”.[72] From a baseline technical perspective, the protocol uses connectors to move payments across different systems and secures them using something called *conditional transfers*. It also provides the packet and address format to assist the connectors in payment forwarding.[73] The key technical piece in the ILP is the security it provides with the conditional transfer, and more specifically its use and generalization of *Hash-Time-Locked Contracts (HTLCs)*. Given the importance of this mechanism, it’s worth taking a moment to really understand how this works.

Hash-Time-Locked Contracts (HTLC): A concept originally introduced in the Lightning network developer community, the idea behind this contract is that value is only transferred when a particular condition is met (with the “judge” being the ledger itself). Generally, the steps are as follows -

1) The sender sends a certain amount that is put on hold by the ledger
2) The receiver can only access the funds if they can produce a *preimage* (or the original data) of a *hashlock* (the resulting cryptographic hash function of the preimage). The preimage and hashlock are agreed upon by the sender and receiver before the transaction starts.
3) If they aren’t able to produce it in within a particular time period, then the funds are returned to the sender

This concept can be generalized with additional parties - for example, between three payment channels, the same hashlock and timelock requirements for A → B can be extended to B → C, making C the responsible party to produce the preimage to unlock the transaction.[74] [75]

ILP generalizes the idea of HTLCs with something called *Hashed-Timelock Agreements (HTLAs)*. Essentially, instead of a “ledger” (or contract if talking about pure blockchains), there is this entity called a connector which takes on the functionality of the hash and time locks. This way the independent ledgers don’t need to actually have implemented the hash and time lock functionality and can still participate in ILP. Furthermore this means that transactions can occur across different types of ledgers -

[72] (“About | Interledger,” n.d.)
[73] (“About | Interledger,” n.d.)
[74] (“Interledger Architecture,” n.d.)
[75] (“Hashed Timelock Contracts - Bitcoin Wiki,” n.d., “Interledger Architecture,” n.d.)

from new crypto networks to local and international banks.[76] The figure below shows the full architectural layout of the protocol suite including the interaction between senders, connectors, and receivers:

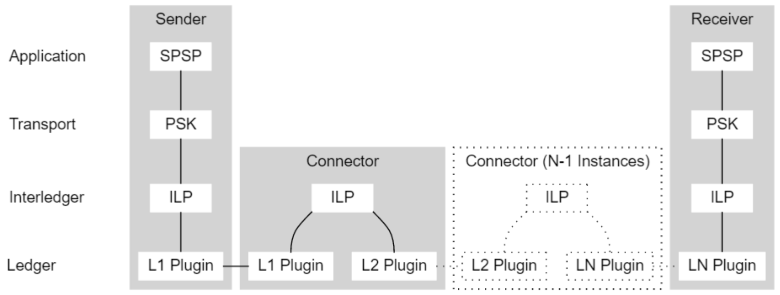

Image Credit: Interledger[77]

The ILP provides an example of a way of connecting with legacy systems using novel cryptographic techniques. The payment space is just one place this is occurring, and more interoperability measures may need to be added depending on the age of the system and industry. If blockchains are to be useful and interact with existing systems, interoperability has to be of utmost consideration.

[76] ("Interledger Architecture | Interledger," n.d.-a)
[77] ("Interledger Architecture | Interledger," n.d.-b)

5. Cryptoeconomics

Cryptoeconomics is the designing and building of systems using *cryptography* to guarantee certain properties, and *economics* to create the right incentives for the future.[78] It's very similar to the field of mechanism design, which is known as *reverse game theory* where a game is designed to achieve certain equilibrium outcomes. However, in this case, the economic incentives are built using cryptography and software, and operated by a system that is decentralized.[79]

Bitcoin is a perfect demonstration of cryptoeconomics in practice. The network uses cryptography to guarantee the past by first using public-private key cryptography to guarantee an entity has control over their coin. Transactions are all compiled into a block, and a SHA-256 hash function links blocks together. This guarantees *the past* as in order to reverse a transaction requires undoing all of the previous blocks from the time of the attack to the original transaction point. This is where incentive design and economic theory comes into play. The consensus mechanism of *proof-of-work* requires a certain amount of computation to chain blocks together, and miners are incentivized to do so because they themselves are rewarded with a tradeable and valuable coin (in addition to transaction fees). Another incentive design is the adjustment of the mining block difficulty to maintain a certain block time.[80]

As seen in the above example, cryptography helped guarantee *the past* and economic incentives ensure that those properties are *held in the future.*[81] While this field is nascent, it encompasses a large amount of subfields, and for sake of scope, this section will limit discussion to talk about consensus protocols, and a case study with the current work being done in a new category of tokens called stablecoins.

5.1 Consensus Mechanisms

Fundamentally, consensus mechanisms are a way to secure and update the state of a network. Given the decentralized nature of crypto networks, consensus mechanisms provide a structured way for all willing participants to give/prove a say on what the "correct" state is. Furthermore, these mechanisms are designed to be sophisticated enough both technically and economically to prevent cheating and collusion. There are several ways to do this with some mechanisms better suited for different protocol goals (ex. speed of transactions, size of transaction, certain

[78] ("Introduction to Cryptoeconomics," 2017)
[79] ("Introduction to Cryptoeconomics," 2017; Stark, 2017)
[80] (Nakamoto, 2008)
[81] ("Introduction to Cryptoeconomics," 2017)

payout schemes, etc.).[82] We'll begin by looking at the two most popular ones, Proof-Of-Work or PoW (focusing on the similarities and differences in implementation between Bitcoin and Ethereum), and Proof-of-Stake or PoS.

5.1.1 Proof Of Work (PoW)

The core idea behind PoW is that a block should be secured only after a certain amount of computation has been done, and it can be proven to have been done. This computation is essentially solving a particular cryptography problem. To maintain the security of the network, protocols that implement PoW have a particular block time (the average expected time to mine a block) they are trying to maintain, and adjust the difficulty of the "problem" accordingly. Miners then expend computational resources to solve the problem.[83]

Bitcoin Implementation of PoW

In the Bitcoin protocol, Satoshi expected it to take on average 10 minutes to mine a block, and currently the difficulty is reevaluated every 2016 blocks. The equation for the new difficulty is as follows:[84]

$$newDifficulty = \frac{oldDifficulty \times (2016\ blocks \times 10\ minutes)}{Time\ in\ minutes\ to\ mine\ the\ last\ 2016\ blocks}$$

The difficulty variable is unitless, but we can compare it to the difficulty of other blocks to contextualize it - for example, if the current block has a difficulty of 10, then it is 10x more difficult to mine the current block then the genesis block (the very first block).[85] At the time of the writing of this paper, the most current Bitcoin block #519600 and had a difficulty of 3,839,316,899,029.67.[86]

The next step is to create the target hash value for the current block, which will be the "computational problem" that miners will attempt to solve. We begin by taking the hexadecimal representation of the variable called *bits* that was in the previous block's header. The first two digits are called the *exponent* and the last six are called the *coefficient*. We can then use these variables in the equation below to calculate the target hash value:[87]

$$target = coefficient \times 2^{(8 \times (exponent - 3))}$$

[82] (Baliga, 2017)
[83] (Baliga, 2017)
[84] (Siriwardena, 2017a)
[85] (Siriwardena, 2017a)
[86] ("Bitcoin Block #519600," n.d.)
[87] (Siriwardena, 2017b)

Leading zeros are added to the target to represent it in 256 bits. Once the target for the genesis block is calculated, all future blocks are calculated using the equation:[88]

$$newTarget = \frac{oldTarget}{newDifficulty}$$

An example of what the target hash may look like is below (for *bits*=1D00FFF and *difficulty*=1):

```
0000000000000000000000000000000011111111011011100100011100011000
1111001001000000011111101110000110010101000010100110110011100000
0011001000101110011111001010001101011001000110011100100010000010
0000000000000000000000000000000000000000000000000000000000000000
```

Now that a 256-bit target hash has been set, miners now attempt to “solve” this problem. What this means is that the miner takes a hash of the previous block and a variable called **nonce**. The miner varies the nonce until they find a hash that is less than or equal to the target hash. A simpler way to think about this is that the miner is trying to find a nonce such that the combined hash leads to the right number of leading zeros.[89]

Ethereum PoW (Homestead)

Very similarly to the Bitcoin implementation, the homestead release of Ethereum’s PoW also 1) aims for a particular block time that it then uses to find a difficulty, 2) uses that difficulty to create a target hash, and 3) has miners vary the nonce until the resulting hash is less than the target hash. However, there are a couple of differences in design principle.[90]

First, ethereum is designed to have an average block time of 10 to 19 seconds. The main reason is that this time was deemed to be the shortest time that the network could process blocks while also limiting the security consequences of network latency. This is also aligned with current studies. In a 2013 study done by Decker and Wattenhofer, they found that while there was a massive tailend in block propagation to all nodes (see image below), the mean time for 95% of nodes to receive a new block was 12.6 seconds.[91]

[88] (Siriwardena, 2017a)
[89] (Siriwardena, 2017a)
[90] (Siriwardena, 2017a)
[91] (Decker & Wattenhofer, 2013)

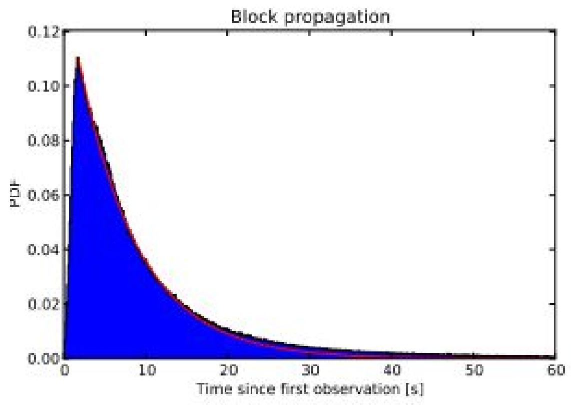

Image Credit: Decker and Wattenhofer[92]

Ethereum thus has a different way of calculating the *newDifficulty* variable in the Bitcoin equation. The code snippet from the Ethereum codebase used to calculate this value is below (with the *newDifficulty* variable referred to as *block_diff*):[93]

```
block_diff = parent_diff + parent_diff // 2048 * max(1 - (block_timestamp -
    parent_timestamp) // 10, -99) + int(2**((block.number // 100000) - 2))
```

Ethereum also varies in how miners go about implementing PoW. Called **Ethhash**, the algorithm has miners generate a *mixHash* consisting of slices of a cached dataset and a nonce. The mixHash is kept on being generated until the target nonce is created. Tangentially, the use of a cached dataset (which is created using a calculated seed that is the same for every 30,000 blocks) enables *light nodes* in Ethereum. These light nodes can be used to verify blocks with fairly low bandwidth and storage requirements. Lastly, Ethhash is memory-hard which means that not only are there computation requirements but certain storage requirements. Combined with its use of Keccak-256 hash scheme rather than SHA-256, this is used to prevent excessive decentralization among miners by democratizing hardware requirements. More specifically, the design ensures there is no advantage in using specialized chips like ASICs, which creates a higher barrier to entry and thus more centralization (something that has happened in Bitcoin). General purpose hardware should in theory allow for a larger base of eligible miners and validators, which leads to more decentralization.[94]

[92] (Decker & Wattenhofer, 2013)
[93] ("How is the Mining Difficulty calculated on Ethereum?," n.d.)
[94] (Kasireddy, 2017a)

5.1.2 Proof of Stake

PoS is a consensus mechanism where *validators* propose and vote on a current block/state, and the weight of their vote corresponds to the amount of coin they "stake". While there are different variations of the PoS mechanism, the general steps of a PoS implementation are:

1) Users tie up a certain amount of currency into a deposit (a smart contract) and become validators
2) These validators are then pseudo-randomly selected to validate a particular block, with the probability of selection linear to the amount of currency they have staked
3) If the produced block is successfully included in the chain, then the validator is rewarded a block reward. If the produced block is not included in the chain then the validator loses part of the deposit[95]

The Ethereum network has publicly stated their intention to move to a PoS mechanism with the release of Casper, their personal version of PoS designed for Ethereum. There's a couple of key advantages over PoW they list. Chief is the prevention of needing to use massive amounts of electricity to perform computations. Not only is this sustainable from an environmental standpoint, but as mentioned before, there are no unfair gains from having different types of equipment. Thus, that can help prevent the current situation of *centralized mining cartels* where a few major entities control majority of the hash power in the network.[96]

From a cryptoeconomics perspective, the current PoS designs are also very interesting in the way they think about incentives. For example, when compared to PoW, PoS relies on penalties and not just rewards. This can be powerful because it 1) doesn't require the creation or destruction of new tokens (and thus avoiding inflationary/deflationary pressure on the token), and 2) it subtly discourages abnormal behavior. The latter point is particularly important when we consider the *nothing at stake* problem, where validators just make blocks on every chain because they are rewarded for producing block. Take the following diagram for example: [97]

[95] ("Proof of Stake FAQ," n.d.)
[96] ("Proof of Stake FAQ," n.d.)
[97] ("Proof of Stake FAQ," n.d.)

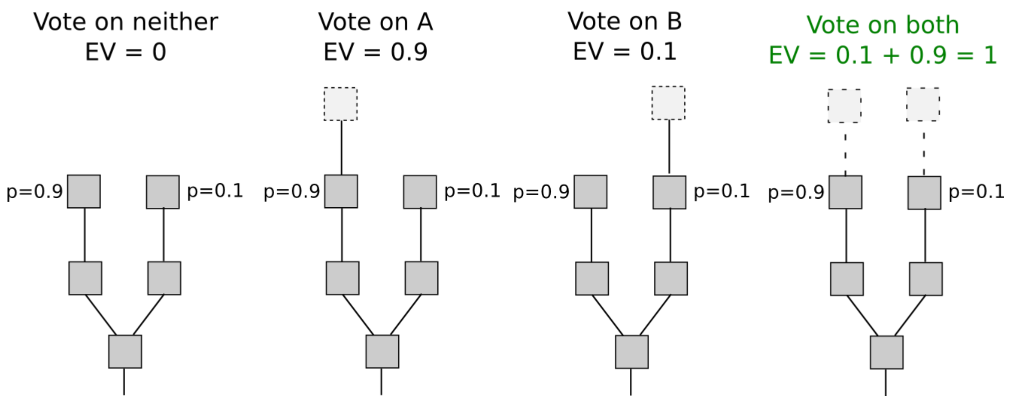

Image Credit: Ethereum Github

Here the validator is incentivized to place or vote blocks on every chain because it receive some time of award. While not fully finalized, the Casper implementation of PoS attempts to solve this by *penalizing* when a validator adds to the wrong chain. Using the same diagram, we see that:

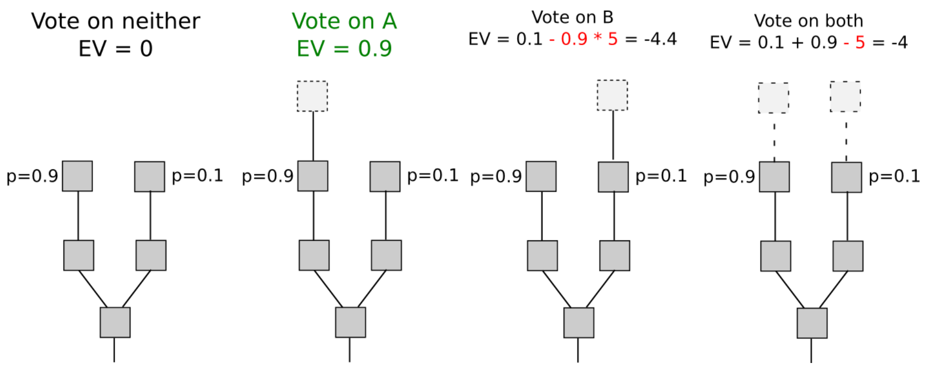

Image Credit: Ethereum Github

While the above is a specific example, more generally the loss of a validators deposit occurs under *slashing conditions* that are programmatically built into the network. This essentially serves as a *cryptoeconomic proof* where an actor is asserting a state is true and willing to suffer a massive economic loss if it is not.[98] [99]

Another interesting concept that has required cryptoeconomic thinking is when to finalize a block, also called *economic finality.* Essentially it's a "guarantee" that a block has been finalized because if it is not included in future chains, then all the validators lose an extremely large amount of money. This is also known as a *cryptoeconomic security margin* and essentially is

[98] ("Proof of Stake FAQ," n.d.)
[99] ("Introduction to Cryptoeconomics," 2017)

guaranteeing a particular state with the collateral being funds high enough that it wouldn't be worth it to conduct that action.[100]

5.2 Case Study: Stablecoins

We'll end the section on cryptoeconomics by looking at the case study of stablecoins, which is often considered as the "holy grail of the cryptocurrency ecosystem".[101] Fundamentally a stablecoin is just a cryptocurrency that is able to maintain a consistent value over time. Such a coin would allow for easy liquidity, medium of exchange, and most importantly, protect against the fluctuations and volatility of most current-day crypto networks. Lastly, the ideal stablecoin design concept would also be decentralized and allow self-governance through the design of the protocol.[102] There are three major designs of stablecoins that we will take a look at - fiat collateralized, crypto-collateralized, and Seigniorage Shares (non-collateralized stablecoins).

Fiat-Collateralized Stablecoins

This type of stablecoin is the simplest to understand and implement - a centralized company sells a coin for fiat and holds the fiat in a bank account. An example implementation of this is with the project called Tether, which at its beginning sold 1 tether for $1, and held the amount in a banking system in the "real world". If you wanted to exchange your tether for fiat, you could easily exchange it through normal and legacy systems. However, while simple in theory and design, fiat-collateralized stablecoins go against many of the goals of the crypto network that we've highlighted before. First and foremost is the centralization of the custodian, or in other words, one needs to trust a central authority to store the fiat. This is in addition to the trust in the fiat currency itself, which takes us back to Bitcoin's original motivation to remove the centralization that fiat currencies have in the first place. Lastly, as we talked about interoperability in earlier sections, there are a lot of technical challenges and vulnerability that come into play in creating the rails between crypto networks and traditional legacy banking/regulatory systems. [103] [104]

Crypto-Collateralized Stablecoins

This type of stablecoin is similar to the fiat-collateralized idea, but instead of using fiat as the collateral, these stablecoins just use other cryptocurrencies. However, because of the fluctuation in the market as a whole, these coins require an *overcollateralization (OC)* to

[100] ("Introduction to Cryptoeconomics," 2017)
[101] (Qureshi, 2018)
[102] ("An Overview Of Stablecoins - Multicoin Capital," 2018)
[103] (Qureshi, 2018)
[104] ("An Overview Of Stablecoins - Multicoin Capital," 2018)

maintain the stability. Essentially, the protocols for these coins offers the exchange of a stablecoin for 1-2+ times the underlying crypto collateral (so $1 of stablecoin for $2 ether) and then locks up the collateral in a smart contract. Programatically, the contract sells the collateral if the price of the collateral falls below a certain threshold. Some designs also have the coin pay an interest on any gains made by the collateral. While crypto-collateralized stablecoins are more decentralized and transparent, the inherent risk of crypto makes it less stable than the fiat-collateralized solution.[105] [106]

Lastly, these coins also have to find a decentralized way to determine the baseline price of a coin. One interesting way to do this is by using a schelling point scheme. The goal here is to determine the "true answer" to a given question (in the stablecoin scenario, it's what is the true price of the stablecoin at a particular moment) using independent actors who vote. The simple algorithm for the schelling point scheme is as follows:

1) Users stake token and provide price inputs with votes weighted by the amount of token staked
2) Software uses the weighted voting to determine the range of answer
3) Users who submitted answers below the 25th percentile or above the 75th percentile, lose their entire stake. The slashed tokens are then redistributed to the users who submitted correct answers[107] [108]

The schelling point scheme has its own disadvantages and vulnerabilities, like the *P + epsilon attack*, but is an attempt to reach consensus on what a ground truth should be.[109]

Seigniorage Shares (Non-Collateralized Stablecoins)

The last design schema we'll cover is the seigniorage shares approach, which essentially algorithmically expands/contracts the supply to maintain price-stability. It is the most "crypto-native" to all the approaches. A peg is maintained in a fairly decentralized manner with consensus being determined similar to how it is in crypto-collateralized stablecoins using mechanisms like the schelling point scheme.[110] Depending on whether the price is above/below the peg, different mechanisms are used to bring it back to a stable level:

[105] (Qureshi, 2018)
[106] ("An Overview Of Stablecoins - Multicoin Capital," 2018)
[107] ("SchellingCoin: A Minimal-Trust Universal Data Feed - Ethereum Blog," 2014)
[108] ("An Overview Of Stablecoins - Multicoin Capital," 2018)
[109] ("The P + epsilon Attack - Ethereum Blog," 2015)
[110] ("An Overview Of Stablecoins - Multicoin Capital," 2018)

- <u>Price below the peg:</u> Here the stablecoin will *decrease/contract* the money supply, and most of these schemas do so by issuing bonds that users can purchase (thus removing the stablecoin from the network). The bonds pay out over time and can expire.
- <u>Price above the peg:</u> Here the stablecoin will *increase/expand* the supply and does so by issuing more stablecoin. These newly minted coins go first to bondholders, and then to shareholders who exist to maintain some price stability and earn returns.

From a cryptoeconomic standpoint, seigniorage shares involve a certain level of risk because it requires continual growth - that is there needs to be expanded demand to create new stablecoins that can pay off existing bond holders. At the same time, there is a risk of a *death spiral* on the contraction side, where bonds are issued at lower prices, which in turn requires more bonds to be printed to remove the right amount of stablecoins.[111]

In conclusion of this case study, we see that stablecoins are a perfect manifestation of cryptoeconomic concepts, as incentive and economic mechanisms need to be combined with technical prowess to make it fit with current crypto networks. Here tradeoffs need to be made between decentralization, real-world utility (using Oracles), and also incentives to ensure that the goals of a particular protocol are actually met.

[111] ("An Overview Of Stablecoins - Multicoin Capital," 2018)

6. Conclusion

In a few months from the date of publication of this paper, Satoshi Nakamoto's original Bitcoin paper and the cryptocurrency movement will officially turn a decade old. And in that time, we've seen the advent of countless crypto networks and a sequential rethinking of what the world should look like, the likes of which we haven't seen since the Internet era. While there is much to look and be appreciative about, it's also important to be realistic and understand the limitations of the technology. Particularly, we need to be intellectually honest about what *is* possible today and what *can* be possible in the future.

From decentralization, making DApps useful, and instilling cryptoeconomics, this paper has weaved together a survey on why crypto networks are important and how they are going about achieving their potential. It's important to note that these topics are not distinct but rather build on one another. Understanding decentralization gives us the motivation to build DApps that are useful enough that people will want to use them over centralized alternatives. And building useful DApps requires some interaction with the external world and integrating with existing legacy systems when necessary. And finally, because of this motivation of decentralization, we have to be smarter in how we design protocols and need to tap into human psychology, game theory, and economics to ensure their consistency and stability. Thus, cryptoeconomics is emerging as as its own study dedicated to exploring these concepts.

Lastly, from an academia standpoint, what's particularly exciting about crypto networks is just the amalgamation of so many distinct fields and concepts into a particular ecosystem. From cryptography, distributed systems, game theory, philosophy, and economics (among many other fields), the interest in crypto networks has brought together experts who may have never had a chance to interact before. It's the interdisciplinary nature that brings complexity and dynamism to these networks, and solutions to problems highlighted in this paper will require cross-team thinking.

It may be cliched and hyperbolic to think about crypto networks as a living, growing organism, but the metaphor does hold up more than a cynic may give it credit for. There is a lot to be cautiously optimistic about crypto networks, and time will tell if this technology becomes one of humanity's crowning achievements, or just a footnote in history.

7. Bibliography

1. 51% Attack, Majority Hash Rate Attack - Bitcoin Glossary. (n.d.). Retrieved April 22, 2018, from https://bitcoin.org/en/glossary/51-percent-attack

2. About | Interledger. (n.d.). Retrieved April 23, 2018, from https://interledger.org/about.html

3. Against the Minimum Majority Measure | Satoshi Nakamoto Institute. (2017, July 28). Retrieved April 22, 2018, from http://nakamotoinstitute.org/mempool/against-the-minimum-majority-measure/

4. An Overview Of Stablecoins - Multicoin Capital. (2018, January 17). Retrieved April 24, 2018, from https://multicoin.capital/2018/01/17/an-overview-of-stablecoins/

5. A Short Guide to Bitcoin Forks - CoinDesk. (2017, March 27). Retrieved April 25, 2018, from https://www.coindesk.com/short-guide-bitcoin-forks-explained/

6. Attack Surface Analysis Cheat Sheet - OWASP. (n.d.). Retrieved April 22, 2018, from https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet#Defining_the_Attack_Surface_of_an_Application

7. Atzori, M. (2015). Blockchain Technology and Decentralized Governance: Is the State Still Necessary? https://doi.org/10.2139/ssrn.2709713

8. Baliga, A. (2017). *Understanding Blockchain Consensus Models*. Retrieved from https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf

9. Bitcoin Block #519600. (n.d.). Retrieved April 25, 2018, from https://blockchain.info/block/0000000000000000002459ebb5bfb018003ee9fce583f9a0a191af634a851aea

10. Bitcoin (BTC) Price | CoinMarketCap. (n.d.). Retrieved April 24, 2018, from https://coinmarketcap.com/currencies/bitcoin/#charts

11. Bit Gold proposal - Bitcoin Wiki. (n.d.). Retrieved April 24, 2018, from https://en.bitcoin.it/wiki/Bit_Gold_proposal

12. Blockchain Investment Trends In Review. (n.d.). Retrieved April 21, 2018, from https://www.cbinsights.com/research/report/blockchain-trends-opportunities/

13. Brown, C. (2016, June 18). Why Build Decentralized Applications: Understanding Dapps - Due. Retrieved April 25, 2018, from https://due.com/blog/why-build-decentralized-applications-understanding-dapps/

14. Buterin, V. (2017, February 6). The Meaning of Decentralization – Vitalik Buterin –

Medium. Retrieved April 21, 2018, from https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274

15. Capgemini Consulting. (2017). *Smart Contracts in Financial Services: Getting from Hype to Reality*. Retrieved from https://www.capgemini.com/consulting-de/wp-content/uploads/sites/32/2017/08/smart_contracts_paper_long_0.pdf

16. Centrality - Wikipedia. (n.d.). Retrieved April 22, 2018, from https://en.wikipedia.org/wiki/Centrality#PageRank_centrality

17. Common cause and special cause (statistics) - Wikipedia. (n.d.). Retrieved April 22, 2018, from https://en.wikipedia.org/wiki/Common_cause_and_special_cause_%28statistics%29#Common_mode_failure_in_engineering

18. Cryptocurrency Market Capitalizations | CoinMarketCap. (n.d.). Retrieved April 21, 2018, from https://coinmarketcap.com/

19. Decentralized Trust Management in Peer-to-Peer Systems - IEEE Conference Publication. (n.d.). Retrieved April 22, 2018, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6108278

20. Decker, C., & Wattenhofer, R. (2013). Information propagation in the Bitcoin network. In *IEEE P2P 2013 Proceedings*. https://doi.org/10.1109/p2p.2013.6688704

21. Di Francesco Maesa Andrea Marino Laura Ricci, D. (2017, September 12). Data-driven analysis of Bitcoin properties: exploiting the users graph. Retrieved April 22, 2018, from https://link.springer.com/content/pdf/10.1007%2Fs41060-017-0074-x.pdf

22. Dixon, C. (2018, February 18). Why Decentralization Matters – Chris Dixon – Medium. Retrieved April 23, 2018, from https://medium.com/@cdixon/why-decentralization-matters-5e3f79f7638e

23. Economist, T. (2015). The great chain of being sure about things. Retrieved April 21, 2018, from https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable

24. Eigenvector centrality - Wikipedia. (n.d.). Retrieved April 22, 2018, from https://en.wikipedia.org/wiki/Eigenvector_centrality

25. ethereum. (n.d.). ethereum/wiki. Retrieved April 22, 2018, from https://github.com/ethereum/wiki

26. Ethereum (ETH) price. (n.d.). Retrieved April 24, 2018, from https://coinmarketcap.com/currencies/ethereum/

27. Eyal, I., & Sirer, E. G. (2014). Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *Lecture Notes in Computer Science* (pp. 436–454).

28. Fernandez, O. (2017, November 29). How Bitcoin Works Under The Hood (Technical) - Blog - Techlaunch.io. Retrieved April 24, 2018, from https://techlaunch.io/blog/how-bitcoin-works-under-hood-technical/

29. Gini coefficient - Wikipedia. (n.d.). Retrieved April 22, 2018, from https://en.wikipedia.org/wiki/Gini_coefficient

30. Hashed Timelock Contracts - Bitcoin Wiki. (n.d.). Retrieved April 23, 2018, from https://en.bitcoin.it/wiki/Hashed_Timelock_Contracts

31. History of Ethereum — Ethereum Homestead 0.1 documentation. (n.d.). Retrieved April 24, 2018, from http://ethdocs.org/en/latest/introduction/history-of-ethereum.html

32. How do oracle services work under the hood? (2017, January 23). Retrieved April 23, 2018, from https://ethereum.stackexchange.com/questions/11589/how-do-oracle-services-work-under-the-hood?noredirect=1&lq=1

33. How is the Mining Difficulty calculated on Ethereum? (n.d.). Retrieved April 23, 2018, from https://ethereum.stackexchange.com/questions/1880/how-is-the-mining-difficulty-calculated-on-ethereum

34. How Oraclize and API call works. (2017, November 23). Retrieved April 23, 2018, from https://ethereum.stackexchange.com/questions/31352/how-oraclize-and-api-call-works

35. How to wait Oraclize result before running code further? (n.d.). Retrieved April 23, 2018, from https://ethereum.stackexchange.com/questions/4423/how-to-wait-oraclize-result-before-running-code-further

36. Interledger Architecture. (n.d.). Retrieved April 23, 2018, from https://github.com/interledger/rfcs/blob/master/0001-interledger-architecture/0001-interledger-architecture.md

37. Interledger Architecture | Interledger. (n.d.-a). Retrieved April 23, 2018, from https://interledger.org/rfcs/0001-interledger-architecture/#connectors

38. Introduction to Cryptoeconomics. (2017, February 23). Retrieved April 23, 2018, from https://vitalik.ca/files/intro_cryptoeconomics.pdf

39. Is there a way to give access of an external storage mechanism only to a smart contract? (n.d.). Retrieved April 23, 2018, from https://ethereum.stackexchange.com/questions/9823/is-there-a-way-to-give-access-of-

an-external-storage-mechanism-only-to-a-smart-c/9825

40. Kariappa Bheemaiah, G. E. de M. (2015, January 6). Block Chain 2.0: The Renaissance of Money. Retrieved April 22, 2018, from https://www.wired.com/insights/2015/01/block-chain-2-0/

41. Kasireddy, P. (2017a, September 27). How does Ethereum work, anyway? – Preethi Kasireddy – Medium. Retrieved April 22, 2018, from https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

42. Kasireddy, P. (2017b, December 10). Fundamental challenges with public blockchains – Preethi Kasireddy – Medium. Retrieved April 23, 2018, from https://medium.com/@preethikasireddy/fundamental-challenges-with-public-blockchains-253c800e9428

43. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

44. Nick Szabo -- Smart Contracts: Building Blocks for Digital Markets. (1996). Retrieved April 24, 2018, from http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

45. Oraclize Documentation. (n.d.). Retrieved April 23, 2018, from https://docs.oraclize.it/#ethereum-quick-start

46. Peng, S. (2013, December 10). BITCOIN: Cryptography, Economics, and the Future. Retrieved April 20, 2018, from http://www.cis.upenn.edu/current-students/undergraduate/courses/documents/EAS499BitcoinThesis-StarryPeng.pdf

47. Proof of Stake FAQ. (n.d.). Retrieved April 23, 2018, from https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

48. Qureshi, H. (2018, February 19). Stablecoins: designing a price-stable cryptocurrency. Retrieved April 24, 2018, from https://hackernoon.com/stablecoins-designing-a-price-stable-cryptocurrency-6bf24e2689e5

49. Richard. (2013, December 6). Where in the World Does the Internet Live? at WhoIsHostingThis.com. Retrieved April 21, 2018, from https://www.whoishostingthis.com/blog/2013/12/06/internet-infographic/

50. Satoshi Nakamoto Institute. (2009, February 11). Retrieved April 21, 2018, from http://satoshi.nakamotoinstitute.org/posts/p2pfoundation/1/

51. SchellingCoin: A Minimal-Trust Universal Data Feed - Ethereum Blog. (2014, March 28).

Retrieved April 24, 2018, from https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/

52. Siriwardena, P. (2017a, October 15). The Mystery Behind Block Time – FACILELOGIN. Retrieved April 23, 2018, from https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a

53. Srinivasan, B. S. (2017, July 28). Quantifying Decentralization – news.earn.com. Retrieved April 22, 2018, from https://news.earn.com/quantifying-decentralization-e39db233c28e

54. Stark, J. (2017, November 16). Making Sense of "Cryptoeconomics" – Hacker Noon. Retrieved April 23, 2018, from https://hackernoon.com/making-sense-of-cryptoeconomics-5edea77e4e8d

55. State of the ÐApps — 1399 Projects Built on Ethereum. (n.d.). Retrieved April 24, 2018, from https://www.stateofthedapps.com/

56. The P + epsilon Attack - Ethereum Blog. (2015, January 28). Retrieved April 24, 2018, from https://blog.ethereum.org/2015/01/28/p-epsilon-attack/

57. Why Many Smart Contract Use Cases Are Simply Impossible - CoinDesk. (2016, April 17). Retrieved April 23, 2018, from https://www.coindesk.com/three-smart-contract-misconceptions/

58. Wood, G. (n.d.). Ethereum: A Secure Decentralised Generalised Transaction Ledger Byzantium version. Retrieved April 22, 2018, from https://ethereum.github.io/yellowpaper/paper.pdf

59. Young, A. (2017, December 20). Crypto Network Fundamentals – Andrew Young – Medium. Retrieved April 23, 2018, from https://medium.com/@andrew_young/crypto-network-fundamentals-dfa11f15d026