The Oracle Problem: Unlocking the Potential of Blockchain

By Jennifer Yen

Advised by Brett Falk

School of Engineering and Applied Science

University of Pennsylvania

December 9th, 2020

Table of Contents

Abstract

A blockchain is a chain of blocks that contains information based on the premise that it should be impossible to change the information without it being apparent and validated. Haber and Stornetta introduced the first iteration of this concept in 1991 to timestamp digital documents, but it was not until 2008 that Bitcoin popularized this thinking. Subsequently, Bitcoin became a catalyst for several new cryptocurrencies and popularized blockchain as a distributed ledger. Blockchain decentralizes transactional systems and has the potential to revolutionize practically every conceivable domain. To transact on different problems, it is often necessary to pull in or push out data to and from external sources. This data exchange is done with secure middlewares named oracles but raises a new problem: How do we decentralize oracle networks? Even if the blockchain is decentralized, if the oracle is not decentralized, it exposes security flaws that blockchain set out to resolve in the first place. Chainlink, the leading decentralized oracle project, claims that it enables smart contracts in the blockchain to realize over 90% of their potential use cases [18]. This research thesis will explore the relationship between the blockchain and trustworthy oracle solutions and present a framework for further advancing oracle networks to unlock blockchain's potential.

Introduction

Since Satoshi Nakamoto's breakthrough Bitcoin whitepaper in 2008, there has been a massive surge in the popularity of cryptocurrencies and blockchains and the birth of an entire industry. Fast forward to a tumultuous 2020, and we are in a position poised for potentially explosive growth: Bitcoin surged on Monday, November 30 to hit an all-time high of $19,864, and Ethereum 2.0 is rolling out in phases beginning on December 1st [28]. At the time of writing, the global cryptocurrency market cap totals a stunning $577.6 billion (subject to fluctuation) according to CoinMarketCap [16]. Millions of dollars are spent daily to integrate crypto projects with businesses and organizations across domains like finance (DeFi), energy, voting, insurance, and more. At the crux of all of these feats are middlewares named oracles that push in external data to the blockchain, effectively expanding the blockchain's capabilities. However, there is still a lot of fogginess around how exactly oracle solutions resolve the Oracle Problem.

Specifically, there is a lack of clarity surrounding why oracles are necessary, how projects like Chainlink are shaping the crypto's advancement, and what alternatives to Chainlink are on the rise despite Chainlink monopolizing the oracle space. This paper attempts to elucidate these topics and is

divided into six parts. Section 1 will introduce the technicalities behind blockchain—the protocols and reasoning behind it from beginning to end. Section 2 will examine blockchain applications and the transactional capabilities unlocked by using digital agreements named smart contracts. Section 3 will elaborate on the Oracle Problem and how it undermines blockchain through a security lens. Section 4 will take a deep dive into trustworthy solutions such as Chainlink, Astraea, Town Crier, DECO, with a focus on Chainlink, the most prominent decentralized oracle network without significant competitors. A blockchain agnostic oracle, Chainlink could become an integral part of any smart contract platform. Section 5 will address the projected next steps for the advancement of oracle solutions and its involvement in prominent crypto projects. Finally, I will conclude the thesis with a summary of my research efforts.

1 An Overview of the Blockchain Protocol

1.1 Background Terminology

Before diving into complicated concepts, it may be helpful to define terminology that will be used repeatedly throughout the paper:

- **Decentralized network:** A network is decentralized if organized in a distributed manner to avoid a single point of failure. The nodes in such a network do not depend on a single central server and each have a copy of the network's configuration.
- **Ledger:** Transactions are grouped into blocks and recorded on digital ledgers. Instead of a centralized third party overseeing these ledgers, the blockchain enables distributed ownership of the ledger involving everyone in the network. In other words, every user in the network has a copy of the ledger and these copies "frequently update and sync data between [each other]" [38]. This distribution makes loss or destruction of the ledger very difficult because one would have to delete all the ledger copies across all users. Since the network is distributed among geographically diverse nodes worldwide, a digital ledger on the blockchain is not tied to any specific geographic location and is resilient to targeted network or service outages.
- **Smart contracts:** Smart contracts are programs that function as digital contracts to control the transfer of cryptocurrencies or data between parties. They execute automatically when certain conditions are met.

1.2 History of Blockchain

Blockchains are a type of tamper-resistant, distributed ledger that form the basis of many modern-day cryptocurrencies. Although the core ideas behind blockchain have been in motion since the late 1980s and early 1990s to timestamp digital documents, it wasn't until 2009 that it was popularized with Bitcoin's launch [24]. The technicalities behind the cryptographic protocols and individual components may be complicated, but the premise is relatively simple:

Blockchains are distributed ledgers of cryptographically signed transactions represented as blocks where a new block is linked to a previous one after cryptographic validation and undergoing a consensus decision [38]. When additional transactions are processed, and new blocks are added, older blocks become exponentially more challenging to modify. Every new block is replicated across all copies of the ledger in the blockchain, making it distributed.

The origins of blockchain concepts trace back as early as 1991 when research scientists Stuart Haber and W. Scott Stornetta introduced a cryptographic solution utilizing a secure chain of blocks

for timestamping digital documents [8]. Haber and Stornetta's timestamping method was based on two premises that have carried over to the present day blockchain protocol: 1) It should be impossible to change any bit of the document without it being apparent, and 2) It should be impossible to change the timestamp itself.

Haber and Stornetta proposed running digital documents through irreversible cryptographic hashing algorithms that produce unique hashes for each document. However, this still did not "[prevent] the timestamping service from colluding with a client," they wrote [22]. The documents and timestamps can be secure, but if the timestamping service is not trustworthy, it undermines the entire system. The question then became: Is there a way to devise a system that does not require a trusted central authority? Haber and Stornetta responded by printing customer hashes of notarized timestamps in the New York Times, essentially creating a distributed "immutable" ledger. The idea is that it should not be possible to change the hash in the New York Times, and users can trust the timestamps corresponding to the published hash.

Fast forward over a decade, and Haber and Stornetta's research went hardly noticed. That is until a pseudo-anonymous person named Satoshi Nakamoto released his Bitcoin whitepaper in 2008 and the subsequent software in 2009. He described a peer-to-peer electronic cash system that would allow users to transact with one another using a proof-of-work mechanism by individual minors instead of a centralized authority [9]. And so, the first blockchain was created.

It is important to note that blockchains have advanced to transact on all sorts of data, not only cryptocurrencies.

1.3 Types of Blockchain

Blockchain networks are categorized as either public or private. Sometimes texts may refer to them as permissionless or permissioned, respectively, but they refer to the same concepts.

Public blockchain networks are open to all who participate in the consensus model. In other words, anyone can join and leave the network, become a validator node, and read and write to the ledger. Because strict membership is not enforced on a public blockchain network, "permissionless blockchain platforms are often open source software, freely available to anyone who wishes to download them" [38]. Bitcoin and Ethereum are well-known examples of public blockchains.

Private blockchain networks require some authority to authorize users as validators capable of publishing blocks. In other words, only authorized users maintain the blockchain, and not everyone has read and write access to the ledger. Because of this, miners in a private network (users who verify and publish blocks—more on this later) usually have a level of trust with one another since they were all given authorization on the network that may be revoked if they misbehave. Hyperledger is one of the most popular private blockchain networks. [38]

Naturally, there are advantages and disadvantages to both types of blockchain, and usually, they are two sides of the same token [6]:

- **Anonymity:** In a public blockchain, anyone can trace down parties involved in a transaction using the public ledger. This is the opposite case in a private blockchain because the ledger is not publicized to everyone. The choice of anonymity or lack thereof depends on what the blockchain application is used for. Corporations may choose to use a private blockchain to secure information or, as we will see later on, use an oracle solution like DECO to handle the confidentiality overhead for them.
- **Security versus efficiency:** Private blockchains may offer practical benefits in efficiency because only authorized nodes are validating transactions. Meanwhile, public blockchains have to filter out redundant computations or submissions from unknown actors. This is ultimately a tradeoff between security and efficiency because the more calculations are made, the more secure the network is. It is much more feasible for actors in a private blockchain to collude and alter data points than on a public blockchain.
- **Regulation:** Because authorized nodes on a private blockchain are a much smaller subset of all the nodes in the network, it is much easier to implement updates and developments on a private blockchain. Only the authorized nodes have to agree on them. On the other hand, "public blockchains operate like public forums where any individual operating a full node would have a say in the governance," making updates and developments much slower [6]. Even so, public blockchains are less prone to censorship and regulation and more communal because there are usually healthy open source communities surrounding public projects.

1.4 Cryptographic Proof

Internet commerce today is typically regulated by third-party financial institutions that mediate transactions between consumers and sellers. They serve "to validate, safeguard, and preserve transactions" and effectively prevent some level of fraud [14]. Instead of a third party, the blockchain relies on public-key cryptography to execute transactions.

Public-key cryptography constitutes the backbone of virtually every cryptocurrency, and it would not exist without it. It is a form of asymmetric encryption where each user has a public-private key pair. In asymmetric encryption, the public and private keys are "functional inverses—the private key and vice versa can only decrypt something encrypted by the public key." Asymmetric encryption differs from symmetric encryption, where a single key is used to encrypt and decrypt the data. [33]

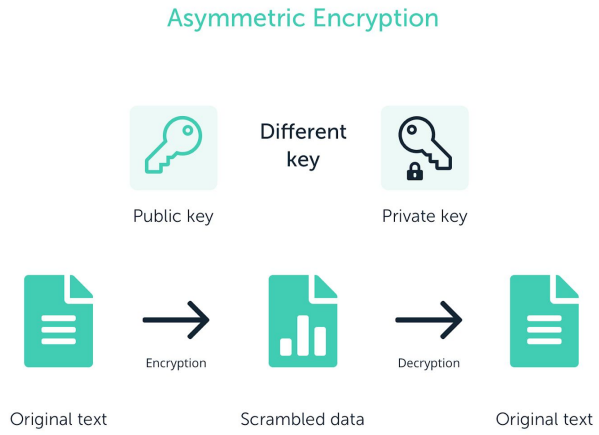


Figure 1: Asymmetric key encryption involves encryption with a public key and decryption with a private key. Source: [30]

Public keys are publicly known and used for identification, versus private keys are kept secret and used for authentication and decryption. When managing cryptocurrencies, your key pair is essentially your identity. By extension, generating an identity is as simple as generating a new key pair. You may wonder how the system prevents two individuals from generating the same key-pair. The answer is simple: the birthday attack problem models the probability of this happening. To have a 50% likelihood of two individuals generating the same key-pair, "you'd need to have about √N many key pairs registered, where N is the size of the total key space." Cryptocurrencies like Bitcoin and Ethereum use 256-bit keys, ensuring that there would need to be 2128 keys registered before a single collision. To put that into perspective, 2128 keys corresponds to "one key for every carbon atom in every human that has ever existed," effectively securing public-key cryptography as a valid method for the foreseeable future. [33]

The robustness of a public-key cryptographic system is only as good as its key generation. More specifically, keys must be reliably random and high-entropy. Although this paper will not touch on the details of generating random numbers, random number generation is critical because there have been instances where weak keys were exploited because they were not random enough.

Assuming we have a reliably random and high-entropy random number generator, a private key is generated by a client-side wallet. Crypto wallets store all kinds of cryptocurrencies and automatically generate and store private keys for users. Next, the public key is generated from the private key using the Elliptic Curve Digital Signature Algorithm (ECDSA) so that they are linked together. Although this paper will not delve into the specifics of ECDSA, it is such a computationally difficult problem to brute-force solve that "the energy required to crack a 228-bit elliptic curve cryptography (ECC) key would be enough to boil all of the water on Earth" [33].

The security of ECDSA stems from the one-way nature of the public-private key pair. It is built from a trapdoor function, meaning it is much harder to compute than to verify, making it infeasible to brute force attack. These complex algorithms make it so the private key is virtually impossible to derive from the public key and vice versa. [32]

Another cryptographic primitive crucial to cryptocurrencies is a digital signature—a publicly verifiable, cryptographically unforgeable, and irrevocable designation that a particular individual "signed" some piece of data [32]. This is done by having an individual sign the hash of a transaction with their private key. A hash is an irreversible scrambling of a message into an obfuscated fixed-sized digest, and this is helpful because signing long messages can be very slow. Signing a hash often requires running it through a digital signature algorithm with a private key as a parameter.

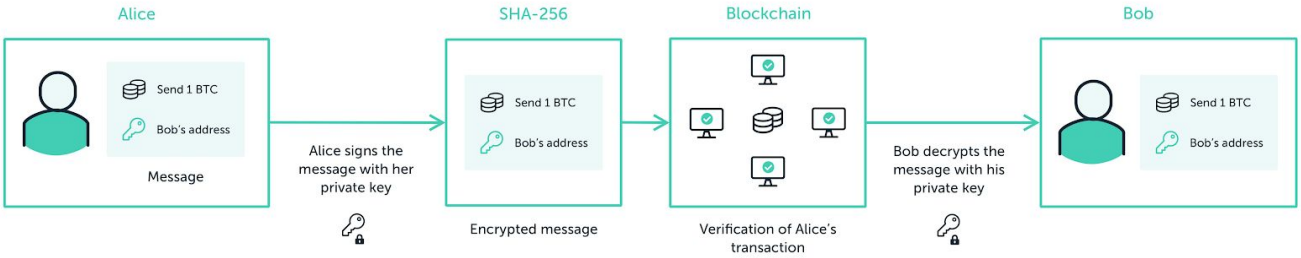


Figure 2: An example of a final transaction. Source: [30]

A transaction typically contains a list of the digital assets to be transferred and references the previous transaction where the sender received the money. The transaction also contains the recipient's public key and a digital signature of the sender's private key for authentication along with these assets.

Consider the transaction above where Alice sends Bob 1 BTC [30]:

1. Alice's transaction is hashed with SHA-256 and encrypted with Bob's public key.
2. Alice signs her transaction with her private key and sends it to the blockchain.
3. The blockchain verifies that the transaction came from Alice by reversing the digital signature algorithm with Alice's public key. Then, the blockchain forwards the message to Bob.
4. Bob decrypts the message with his private key.

Notice that an attacker cannot feasibly reverse-engineer the message or Alice's private key. This is because ECDSA is built using a trapdoor function and the attacker only knows their private key and Alice's public key. Ultimately, Alice and Bob's transaction is an example of the cryptographic proof that enables blockchain and cryptocurrencies to function smoothly.

1.5 Consensus Models

A block is a bundle of transactions verified by several nodes in the network nicknamed miners. Miners are incentivized to verify and publish blocks because they receive monetary rewards upon successful verifications of transactions.

The size of a block determines how many transactions may be verified per block. For example, as of October 2020, a Bitcoin block is limited to 4 million "weight units," corresponding to an average block size of 1.15 MB and an average of 2.2 K transactions per block [10]. Bitcoin's block size has increased significantly since its conception to scale with the number of transactions in the network. If the block size is not large enough, there may be increased transaction fees and delays in processing transactions [23]. The question of what to set the block size has long been debated in many cryptocurrency communities.

Whenever a block is added to the chain and the ledger is updated, changes are made to all copies of the ledger in the network. But how can nodes be sure that the block and all its transactions are trustworthy? This is where the consensus model comes into play.

Consensus models take advantage of the blockchain's distributed nature to maintain security and consistency across all transactions. They vary depending on the blockchain and cryptocurrency, but some of the most popular ones are proof-of-work (PoW) and proof-of-stake (PoS).

1.5.1 Proof of Work

We will be examining Bitcoin's model to understand PoW because the core ideas are similar across different cryptocurrencies. The core idea behind proof of work (PoW) is that a block is only accepted after an acceptable amount of computation has been done. Any individual who wishes to publish blocks in the network must make these computations.

For the sake of explanation, let T be the current block we are on and let T-1 be the chain's previous block. To add a block, miners must compute the cryptographic hash of T-1 with a random value called a nonce. Miners vary this nonce until they encounter a hash that is less than or equal to the hash of T, which can be formulated as:

SHA256("blockchain" + nonce) = hash digest starting with several leading 0s. [31]

It is essential to note although miners attempt to mine blocks containing the same transactions, they will never be mining identical blocks. Blocks of transactions may vary from individual to individual based on how they processed and sorted transactions. Each miner also includes their wallet address in the block header, which is unique by design. [15]

Cryptographic hash functions are designed so that it is virtually impossible to discern the input from the output, and even the smallest change to the input will result in a completely different output.

This is why it takes a large amount of computational resources for miners to solve these cryptographic hash problems.

The first miner to compute a correct hash publishes a new block and broadcasts that they have done so to all other miners in the network. Then, the recipient nodes verify that the new block solves the problem and adds the block to their copy of the ledger.

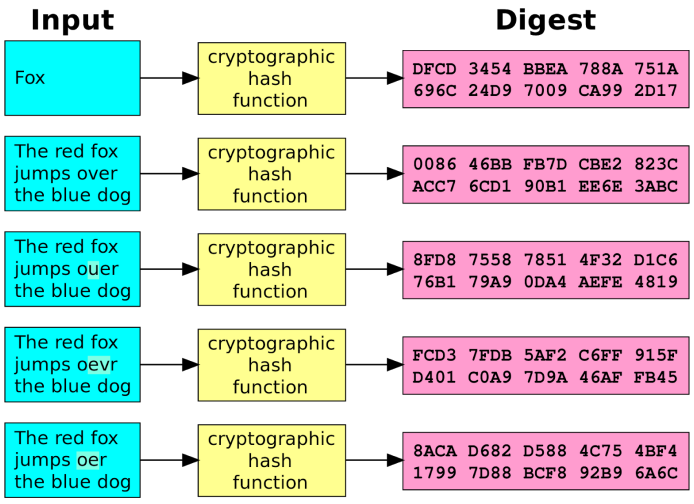


Figure 3: The one-way properties of a hash function. Source: [19]

To maintain consistency and security, the Bitcoin protocol maintains an expected time to mine a block, usually around ten minutes. The cryptographic problem's difficulty is adjusted by increasing or decreasing the number of leading zeros after every 2016 new blocks are added. Increasing the number of leading zeroes increases the problem's difficulty because it shrinks the solution space, whereas decreasing the number of leading zeroes decreases the difficulty because it expands the solution space. [38]

The change in difficulty after every 2016 blocks can be formulated as:

*new difficulty = previous difficulty * (time to mine last 2016 blocks / 20,160 minutes)* [34]

Each new block's problem is independent of one another, so solving one problem does not influence the likelihood of solving the next one. There is no shortcut to miners having to make brute force guesses to solve each block's problem, reaffirming the security strengths of Bitcoin's proof of work model. To ensure that all nodes have identical copies of the blockchain, all nodes follow the "longest chain rule." This rule says that if there are two distinct blockchains with conflicting transaction histories, default to the longer one because it has more work put into it. For this reason, Bitcoin's architecture makes it extraordinarily difficult for an attacker to falsify a transaction because they would have to maintain a false blockchain and out-mine all other miners in the network. [36]

1.5.2 Proof of Stake

Although reliable, the PoW model is computationally intensive. The central processing units (CPUs) used towards solving PoW's cryptographic problems could be diverted to other important jobs. Proof of stake (PoS) attempts to fix this concern and is based on the presumption that users with more currency staked in the network will not misbehave. Staking means to lock in a set amount of currency in a wallet as an investment into the system. Once staked, this currency is no longer able to be spent. PoS models use the amount of stake each user in the network has as the primary determining factor for who can publish a new block [38].

There are many variations for how to use stake [17]:

- **Random selection of staked users:** The ratio of a user's stake in the network correlates to how often they will be chosen to verify and publish a block. For example, if a node user has 25% stake in the network, they will be selected 25% of the time.
- **Multi-round voting:** The blockchain network selects several staked users at random. These users participate in several rounds of voting to decide on a new block.
- **Delegated PoS:** Users vote for a few nodes to become publishing nodes, where voting power is correlated with one's stake in the network. This system encourages nominated delegates to perform efficiently because otherwise, the delegate will be replaced with a new one in future rounds.

At the time of writing, Ethereum, the second-largest cryptocurrency by market cap, is transitioning from a PoW to a PoS consensus model (more on this in Section 2.2.3).

1.6 Resolving Ledger Conflicts

Sometimes it is necessary to update the blockchain network with protocol changes. Due to the distributed nature of the network and its composition of users from all around the world, this isn't easy to do. Changes to a network's protocol or data structures are called forks and are categorized as either soft or hard forks.

A soft fork is a change that is backward compatible with nodes that have not been updated. This means that updated nodes can continue to transact with non-updated nodes. In contrast, a hard fork is a change that is not backward compatible with nodes that have not been updated. This means that non-updated nodes will reject updated nodes. Networks may require that all nodes switch to using the updated protocol. If not all nodes update, this can lead to a split in the blockchain network (evidenced by Ethereum versus Ethereum classic), creating multiple versions of the same blockchain because nodes on different hard forks cannot transact with each other. [38]

2 Smart Contracts and Ethereum

2.1 Smart Contracts

Smart contracts are critical to this paper and explain why the Oracle Problem is a worthwhile one to solve. They are built-in programs running on the blockchain that execute automatically when certain conditions are met. Smart contracts enable the transaction of anything of value from currency to property to data and open a new world of possibilities for decentralized applications (DApps) on the blockchain.

For example, suppose you want to sell a house, and someone pays you the required amount in cryptocurrency towards your smart contract. Then, the house ownership will automatically be passed on to the buyer in whatever form that may be (updates to database(s), legal documentation, etc.). No lawyers, paper contracts, middleman fees, or long wait times are needed. Ultimately, all smart contracts can be boiled down to an "if X, then Y" statement.

Smart contracts are often mentioned in conjunction with Ethereum, the second-largest cryptocurrency by market cap because Ethereum was built for implementing them. This section will explore how smart contracts send and receive transactions and messages that provide the information required to carry out smart contract code.

2.2 Introducing Ethereum

Vitalik Buterin, the founder of Ethereum, recognized that Bitcoin's architecture was limited in functionality and not very flexible for applications beyond cryptocurrency transfer. So, he built the Ethereum blockchain, which includes several features that the Bitcoin blockchain does not have.

Although Bitcoin popularized blockchain technology, it has significant limitations. Some of these include:

- **Turing-incomplete scripting:** Bitcoin Script is Turing-incomplete, which means that there are many computations it cannot handle. Most notably, Bitcoin Script does not support for loops. This was intentionally designed to prevent infinite loops from ever occurring but leads to space-inefficient scripts (have to write out every iteration of a process instead of using a loop). [12]
- **Scalability and throughout:** Scalability is a natural concern for a PoW-based blockchain because it is computationally intensive, and "block size and frequency limitations constrain the network's throughput" [37].

Like Bitcoin, Ethereum is built on blockchain technology, has a distributed ledger and a decentralized peer-to-peer network, and shares many of the same protocols. Unlike Bitcoin, Ethereum is designed for building DApps and implementing smart contracts. Ethereum's design using a Turing-complete language allows anyone to write smart contracts and DApps "where they can create their own arbitrary rules for ownership, transaction formats, and state transition functions" [12].

2.2.1 Ethereum Accounts

Every transaction on the Ethereum blockchain involves a state transition between accounts. An account is used for holding Ether, Ethereum's currency, and transacting with one another. An account contains four fields [37]:

- **Nonce:** a counter used to distinguish transactions
- **Ether balance:** the currency of Ethereum
- **Contract code (optional):** the hash code pointing to a specific smart contract
- **Storage (optional):** the hash code pointing to the account's storage, empty by default

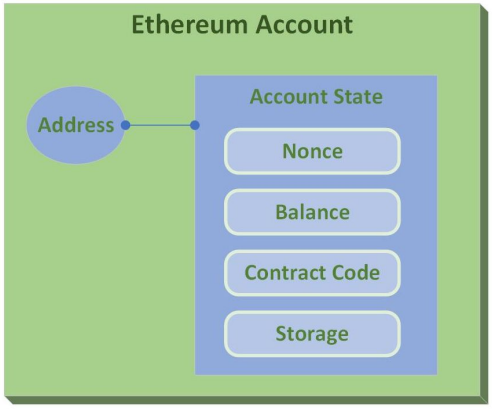


Figure 3: An Ethereum account. Source: [37]

There are two types of Ethereum accounts: externally owned accounts and contract accounts. Externally owned accounts are used for Ether transfer and are analogous to your checking account with a bank. Every externally owned account has a pair of cryptographic keys (public and private) where the account address is generated based on the owner's public key, and owners can manage their accounts with their private key. Owners digitally sign each transaction with their private key so that the blockchain can securely verify their identities. [37]

A contract account is used for executing smart contract code. The blockchain activates and executes the contract logic whenever the contract account receives a transaction from an externally owned account or another contract account message. The storage is where the smart contract code

reads and writes from memory. A state transition in a contract account may involve an update of the Ether balance, data in the storage, or both, depending on the contract. [37]

2.2.2 Ethereum Transactions and Messages

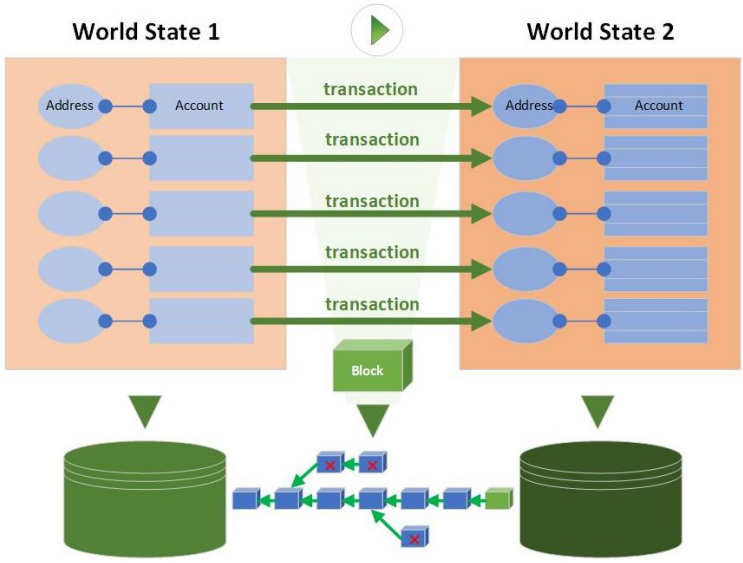


Figure 4: Transactions grouped into blocks and chained together. Source: [37]

Although they are often mentioned together, there is a distinct difference between Ethereum transactions and messages. Ethereum transactions follow the blockchain protocol, meaning they are grouped into blocks and chained together. Transactions are signed messages sent from an externally owned account to another account that contain: 1) recipient, 2) signature of the sender, 3) Ether amount to transfer from sender to recipient, 4) storage (optional) that may be used whenever smart contract code needs to transfer data, 5) a STARTGAS value representing the maximum number of computational steps the transaction execution is allowed to take, 6) a GASPRICE value, representing the fee the sender pays per computational step. [12]

Ethereum uses STARTGAS and GASPRICE to construct a robust denial of service model. "Gas" may be purchased with Ether and represents a unit of computation. The amount of gas required to execute computations depends on the code's complexity, so simpler steps require less gas, and complicated steps require more gas. By setting a limit on how many computational steps of code can be executed with gas, Ethereum prevents "accidental or hostile infinite loops or other computational wastage in code [and] requires attackers to pay proportionally for every resource they consume, including computation, bandwidth, and storage" [12].

In contrast to transactions, Ethereum messages are virtual objects sent by contract accounts to other contracts and are not recorded into the blockchain. Messages contain: 1) recipient, 2) sender, 3) Ether amount to transfer from sender to recipient, 4) storage (optional), and STARTGAS.

A message is very similar to a transaction. The critical difference is that it is only sent by contract accounts and not external actors. Whenever a contract runs its logic and fires a CALL opcode, it produces and executes a message. The message is sent to a recipient account that runs the smart contract code.

2.2.3 Ethereum 2.0

At the time of writing, Ethereum is in the midst of its long-planned upgrade to Ethereum 2.0 (Eth2) and is shipping out in phases. The Eth2 upgrade is only a change to Ethereum's infrastructure; users will still use ETH like they normally do. The planned launch date for the first phase is December 1st, 2020.

The main change is that Ethereum will transition from a proof-of-work to a proof-of-stake blockchain, using the PoS consensus model described in section 1.5.2. To recap, staking means to lock in ETH to become a validator for Ethereum. These validators are critical for verifying transactions and publishing new blocks. Currently, Ethereum can process about 15 transactions per second while Visa can process about 1,700—this disparity is a significant barrier to scalability. PoW blockchains are computationally heavy for security reasons; it should be difficult to attack or trick the network. Transitioning to Eth2 will reduce energy consumption and allow the network to process up to as many as 100,00 transactions per second. [35]

2.3 The World of DeFi

In recent years, the Ethereum blockchain has enabled the decentralized finance (DeFi) movement to rise in popularity. Coinbase employees described it best as a "global, open alternative to every financial service you use today—savings, loans, trading, insurance, and more—accessible to anyone in the world with a smartphone and internet connection" [13].

In a centralized financial system, financial institutions are the essential intermediaries between consumers and sellers. They help reduce transaction costs, protect against fraudulent activity, and ensure that transactions run smoothly. Despite their advantages, financial institutions also have clear limitations. For one, financial institutions usually grow to dominate economic activities and accumulate disproportionate market power and profits because of their critical role. Consider JPMorgan, Bank of America, Paypal, and Square—they all have significant power to shape the fintech space. This has many repercussions for what consumers are able and unable to do. In contrast, DeFi

utilizes decentralized peer-to-peer networks to circumvent the need for centralized financial mediators and prevent any one institution from monopolizing the system.

Freedom of Innovation

Free from governance control and unilateral changes, DeFi developers can freely innovate and open source their applications and platforms. For example, many cryptocurrencies such as Bitcoin and Ethereum share the same open source core technologies. While this collaboration promotes advancement, centralized financial institutions often safeguard their intellectual properties and compete with one another. [39]

Interoperability and Borderlessness

Each centralized financial institution maintains its own ledger for corporate reasons; for instance, JPMorgan would not record transactions from Bank of America users. Therefore, it is often cumbersome and time-consuming to move capital across platforms. On the other hand, DeFi is built upon public blockchains so that it is much easier to move capital across applications on the same blockchain. Moving capital across different blockchains is tricker because protocols vary depending on the network. Fortunately, there is ongoing development to increase interoperability between other blockchains, and projects like Cosmos and Polkadot are working to create an ecosystem of interconnected blockchains.

A corollary to DeFi's interoperability is that it is inherently borderless. Because cryptocurrency is not tied to any geographic location or fiat currency, it can be transferred between anyone worldwide without changing in value. The same cannot be said for centralized financial institutions because they are tied to specific geographic locations that have specific fiat currencies. [39]

3 The Oracle Problem

3.1 Introducing Oracles

The term "oracle" is derived from Greek mythology, which refers to a person who can communicate with the Gods and speak prophecies. Analogously, oracles in a blockchain network serve as bridges linking the on- and off-chain worlds. They push in external data, which can be anything from a basketball game's results to the value of a stock, and feed it into smart contracts. Introducing external data to a blockchain network dramatically broadens smart contracts' capabilities because they can form contractual relationships based on real-world events. [3]

Oracles are ideally trustless for smart contracts to work. Trustlessness means to minimize the amount of trust required from any single source; it does not eliminate trust. Since permissionless blockchains are trustless environments, "determinism is vital to guarantee that the nodes reach consensus." However, the Internet is non-deterministic—information frequently changes so that smart contracts cannot reliably fetch data as a server would. This is where oracles come in as the "layer responsible for querying, validating, and authenticating data before passing it on to the smart contract." [29]

3.2 Why Smart Contracts Need Oracles

The blockchain is a self-contained peer-to-peer network and does not have access to information outside of the network. Therefore, smart contracts are restricted to access information only available on the blockchain, but there is not much information on the blockchain itself.

Let’s imagine if the blockchain were to query information external to the blockchain. Recall that the blockchain is a consensus-based system, and each node must arrive at an identical state of information after each transaction. This is a strict requirement to validate transactions and uphold the distributed ledger (all nodes must have exact copies of the ledger). Therefore, the data shared to and from nodes must be completely deterministic, meaning there should be no way for distinct nodes to have differences. Dr. Gideon Greenspan, the CEO of Coin Sciences, goes as far as to say, “The moment that two honest nodes disagree about the chain’s state, the entire system becomes worthless.” [21]

By design, smart contracts are executed independently on individual nodes in the network. They query their information from external sources, and these actions are performed repeatedly and separately by each node. This introduces the vulnerabilities mentioned in the previous paragraph because there is no guarantee that every node will query data from the same source or that the data will

be identical [21]. Attackers can easily exploit this vulnerability to inject false data into the network and undermine the blockchain.

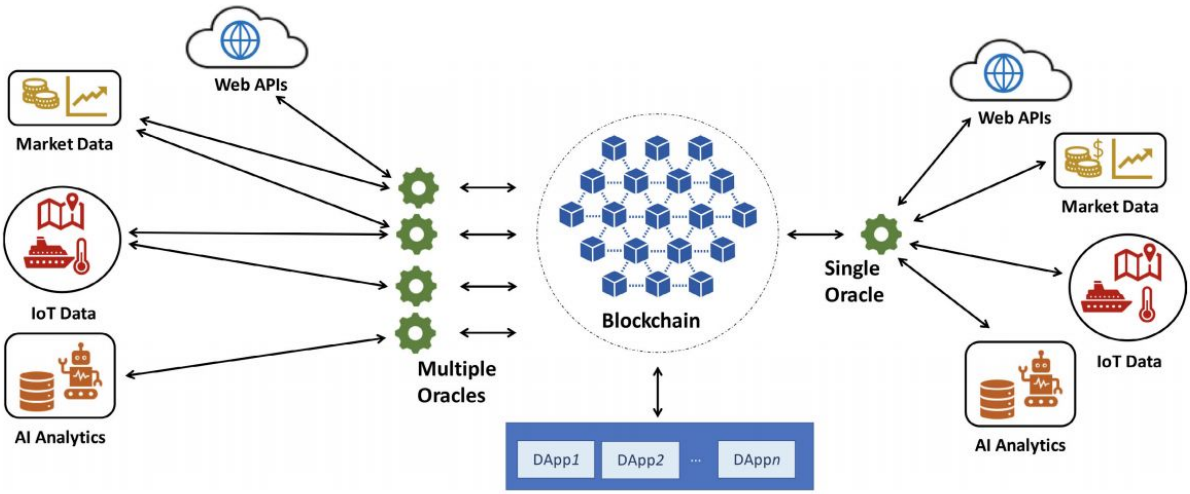


Figure 5: Depiction of an oracle network working with a DApp. Source: [1]

The resolution is relatively simple: instead of having a smart contract *pull* in data from external sources, have a trusted party (oracles) *push* in data into the blockchain. This is the very phenomenon we will be looking into.

3.3 Types of Oracles

Blockchain oracles are classified depending on different qualities, such as where the data originates from, if it is inbound or outbound, and if the oracle is centralized or decentralized. There are many different flavors of oracles depending on the application, and here are some notable oracle categories [7]:

- **Software oracles:** Perhaps the most common type of oracle, software oracles are connected to the Internet and get data from online sources like APIs, databases, servers, and more. Some kinds of information transmitted by software oracles are exchange rates, digital asset prices, real-time flight information, etc.
- **Hardware oracles:** Hardware oracles are physical systems that allow smart contracts to interface with the real world. Some smart contracts may require information from electronic sensors, IoT devices, robotics, and other information reading devices. A hardware oracle could be a sensor that "translates" real-world occurrences into a numeric format readable by smart contracts, which can then execute decisions based on it. This type of oracle is most applicable in the supply chain and robotics space.

- **Human oracles:** Although uncommon, individuals with specialized knowledge may also serve as oracles. They synthesize their information from various sources, verify its authenticity, and translate it to smart contracts. Human oracles must verify and authenticate themselves with cryptographic proof (Section 1.4) to maintain the system's security.
- **Inbound/Outbound oracles:** Inbound oracles gather information from external sources, while outbound oracles send information off-chain to interact with the real-world. An example of inbound and outbound oracles working together is a smart lock. An inbound oracle could track funds deposited by an individual for a property. Once sufficient funds are allocated, an outbound oracle could trigger the mechanism to unlock the smart lock and allow the individual access to a property.
- **Centralized/Decentralized oracles:** Oracles may be centralized or decentralized, but recent oracle projects focus on ways to decentralize. A centralized oracle is a single source of data for smart contracts. By extension, it is also the single point of failure and is vulnerable to attacks from malicious actors attempting to interfere with its smart contracts. On the other hand, decentralized oracles rely on underlying aggregates from multiple nodes and evaluate input data with consensus models before feeding it to smart contracts. This way, there is no single point of failure if the oracle is attacked.

3.4 Oracle Design and Requirements

There are various ways to design an oracle depending on the application and the type of data queried [7]:

- **Immediate-Read:** Immediate-read oracles have the most straightforward design and provide data that is needed for an immediate decision by a smart contract, such as "Is this individual old enough to drink alcohol?" or "What is this student's ID number?" Immediate-read oracles typically communicate data that does not change often and stores it in its contract storage, where any other smart contract can read from.
- **Publish-Subscribe:** Publish-subscribe oracles add a layer of complexity from immediate-read and act as a broadcast service. It is continuously updated with new information, and parties (smart contracts, DApps, or the blockchain itself) can subscribe to these updates. Publish-subscribe oracles typically communicate data expected to change, such as price feeds, weather information, traffic conditions, etc.
- **Request-Response:** Request-response oracles are used when there is too much data to store in a smart contract, and only a small subset of the data is needed at any given time. In this case, an application would query the oracle, the oracle would retrieve relevant data from off-chain sources and sign it to verify its authenticity, and then the oracle would deliver the data to the

application, either by broadcasting it directly or via an oracle contract. This design differs from immediate-read because a smart contract does not automatically pick up the data; the decentralized application may do its own processing once it receives a response from the oracle.

To maintain trust on the blockchain network and in smart contract transactions, there are specific properties an oracle must fulfill [29]:

- **Correctness:** The data must be reliably untampered with, and this is ensured through cryptographic means. Clients should be able to trust that attackers are not able to falsify transactions on the blockchain.
- **Availability:** The oracle should be able to collect and communicate data from off-chain sources at any time. The infrastructure should be designed to minimize and be resilient to outages; this may be done by replicating distributed nodes, as in Ethereum.
- **Accountability:** The oracle system should be able to hold the data sources accountable for truthful data provisioning. Often, the reputation of these data sources are at stake and are attributed to their behavior on- and off-chain.

4 Trustworthy Oracle Solutions

Some centralized oracle solutions are exceptionally high-performing and use complex cryptographic methods to create an authenticated data feed between the data source and the blockchain. Although centralized oracles may be acceptable for some applications, they are inevitably single points of failure in the Ethereum network. This centralization problem exists even if the blockchain network is decentralized. To remedy this vulnerability, several decentralized oracle projects have been proposed as a means of securing the data-source-to-oracle pipeline.

In this section, we will be looking at decentralized oracle solutions such as Chainlink, Astraea, and DECO, as well as a centralized hardware solution named Town Crier. We will explore ways in which centralized solutions can improve decentralized solutions, rendering them worthwhile investments.

4.1 Chainlink

Chainlink is the leading decentralized oracle network by a far margin, holding a market cap of nearly $5 billion at the time of writing (Nov 2020). Chainlink's core functional objective is to bridge the on-chain and off-chain environments. It was initially developed for the Ethereum blockchain but is intended to "support all leading smart contract networks for both off-chain and cross-chain interactions" [7]. It was also developed with modularity in mind—that is, each component is distinct and functional so that they are individually upgradable. As better technologies and implementations arise, these components can be adjusted as needed.

Node operators use Chainlink's LINK token to stake deposits for participation in the network and are also paid in LINK for successfully operating a node/retrieving data. Staking LINK has many similarities to staking deposits in a PoS blockchain; that is, staking LINK tokens allows nodes to undertake jobs that require collateral, just as staking ETH will enable miners to publish blocks in the ETH2.0 blockchain. If a node is chosen to fulfill a contract, their LINK is tied to the contract until it is fulfilled to discourage nodes from misbehaving. Once the contract is fulfilled, the chosen nodes will get back their staked LINK and an agreed transaction fee. If a node is not selected for a contract, they get their staked LINK back, and if a node fails to deliver reliable data, they lose their staked LINK. These regulations set the framework for nodes to act honestly and reliably. [6]

4.1.1 Chainlink Architecture and Protocol

Chainlink nodes interface with both the blockchain and off-chain sources, so Chainlink's architecture must be examined on-chain versus off-chain. In the following diagram from Chainlink's

white paper, the purple block represents the on-chain components, and the green block represents the off-chain components:

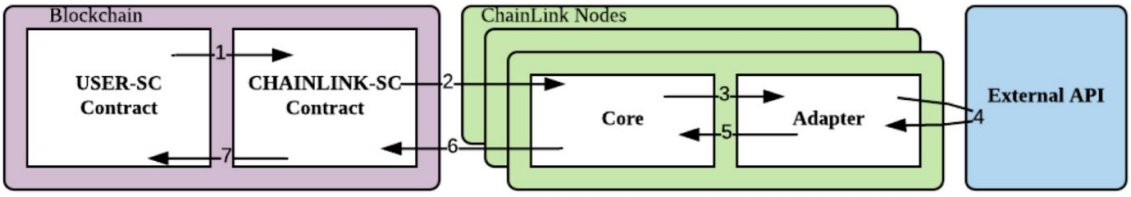


Figure 6: On-chain and off-chain components of Chainlink. Source: [18]

On-Chain

Chainlink supports two types of on-chain smart contracts: 1) user contracts (USER-SC) "execute the on-chain logic for decentralized applications and specify the data requirements from external data feeds" while 2) Chainlink contracts (CHAINLINK-SC) "ensure trust and security via aggregation operations and reputation calculation mechanisms" [1].

CHAINLINK-SC consists of three central contracts: a reputation contract, an order-matching contract, and an aggregating contract. The reputation contract tracks oracle-service-provider performance metrics. The order-matching contract oversees the oracle selection process detailed below. The aggregating contract collects the oracle providers' responses and outputs the Chainlink query's final collective result. These contracts are designed as modular parts that can be updated or replaced as advancements roll out. [18]

When a Chainlink node receives a data request or query, it goes through a series of steps to produce a final collective result: 1) oracle selection, 2) data reporting, and 3) result aggregation [18]:

- **Oracle selection:** An oracle services purchaser proposes a service level agreement (SLA) that includes details such as query parameters, number of oracles needed, and the reputation desired from oracles. With this SLA, qualified oracles undergo a bidding process managed by the order-matching contract. The order-matching contract communicates with the reputation contract and selects bids based on the oracle's performance metrics. It only accepts bids from those that meet the SLA's requirements. There is usually a bidding window in which the order-matching contract will accept bids. Once there are enough qualifying bids, or the bidding window has ended, the requested number of oracles is selected from the bidding pool, and the SLA is finalized.
- **Data reporting and collection:** Once an oracle is selected from the process detailed above, the off-chain oracles execute the SLA and report back on-chain.
- **Result aggregation:** Oracles reveal their results to an aggregating contract. The aggregating contract tallies the collective results from all oracles performing the task and computes a

weighted answer. Drawing from the consensus of oracles in the pool, the aggregating contract also detects outliers or possibly incorrect values and reports these validity scores to the reputation contract. Finally, the weighted answer is returned to the user in the smart contract.

Off-Chain

Chainlink's off-chain architecture consists of two main components: 1) core and 2) external adapters. Although Chainlink has its most extensive network of oracles on the Ethereum blockchain, it is blockchain-agnostic and can be integrated into any blockchain environment. Each node independently harvests responses to off-chain requests. These responses are then aggregated and analyzed through one of many consensus mechanisms before being returned to USER-SC. [18]

- **Core:** Living up to its name, the core node software is critical to the oracle system because it interfaces with the blockchain, partitioning work across its various services. Each necessary process is called an assignment, and each assignment is a set of subtasks, which are processed in a queue-like manner.
- **External adapters:** Adapters are software extensions that add flexibility to the oracle by allowing node operators to customize off-chain services for their application's needs. In other words, any individual can build their own external adapter. Adapters have a minimal REST API that enables nodes to perform requests to external APIs.

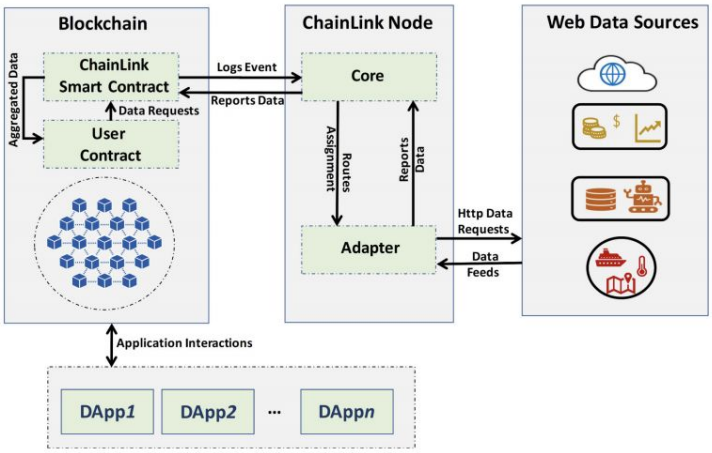


Figure 7: Chainlink architecture. Source: [1]

The steps for a simple Chainlink process can be summarized as follows:

1. USER-SC makes an on-chain request.
2. CHAINLINK-SC logs the request and passes it to the core.
3. Chainlink core routes the task to an appropriate adapter customized for the application.

4. The Chainlink adapter queries an external API for relevant information.
5. The adapter processes the response and passes it back to the core.
6. The core reports the data to CHAINLINK-SC.
7. Finally, CHAINLINK-SC aggregates and analyzes responses and returns a final answer to USER-SC.

4.1.2 Chainlink's Decentralization

To prevent a single point of failure, Chainlink utilizes three complementary approaches to mitigate the effects of faulty nodes: 1) distribution of data sources, 2) distribution of oracles, and 3) use of trusted hardware. [18]

- **Distribution of data sources:** Perhaps the most direct way to avoid faulty data sources is to gather data from multiple sources. A trustworthy oracle queries a collection of sources and aggregates them into a single answer. This can be done in various ways, from majority voting (if the majority of sources returns the same value) to eliminating outliers and outputting the mean. Sometimes, faulty data sources are linked. For example, site A may get its data from site B, which gets its data from site C. If Chainlink uses these three sites to collect sources, gathering from multiple data sources is not very helpful. Ongoing research is being done to provide better mappings of data sources to avoid these undesired correlations.
- **Distribution of oracles:** Similar to how data sources can be distributed, oracle nodes can be distributed as well. Chainlink uses a collection of oracle nodes where each node queries its own distinct set of data sources (overlap is okay), aggregates these responses, and produces a final answer.
- **Use of trusted hardware:** See Section 4.3.

4.2 Astraea

For breadth, let's now examine a completely different oracle approach. Astraea is a decentralized voting-based oracle that decides the truth value of Boolean propositions. Therefore, each proposition p has a truth value t that is either True (T) or False (F) but not both.

Participating entities may act as one of three roles: submitters, voters, and certifiers. [2]

- **Submitters:** These parties submit propositions to the oracle system by allocating money to fund their subsequent validation.
- **Voters:** Voters play a low-risk/low-reward role. A voter begins the voting process by staking a deposit into Astraea. The voter will then be given a proposition chosen uniformly at random and asked to vote on it (true or false). Finally, the voting process's outcome is aggregated as "a function of the sum of the votes weighted by the deposits" [1].

- **Certifiers:** Certifiers play a high-risk/high-reward role. A certifier chooses a proposition, stakes a large deposit on it, and certifies it as true or false. Whereas voters are given a random proposition, certifiers choose which proposition they wish to certify. Finally, the certification process outcome is aggregated as "a function of the sum of certifications weighted by the deposits" [1].

Figures 8 and 9: Player voting in Astraea (L) and player certifying in Astrada (R). Source: [2]

Let's define a player to be a voter or a certifier. The system functions based on the premise that a player's beliefs are independent of all other players' beliefs and that a player's beliefs in a particular proposition *p* are independent of their beliefs in a different proposition *p'*. An honest player always votes or certifies a proposition p to the best of their ability.

The influence of voters and certifiers is attributed to the system's parameters, such as deposit size and voting/certifying stake. Each voter must stake a deposit into Astraea to participate, and the maximum voting deposit "should be small relative to the total voting stake on each proposition". If this is not enforced and a single vote has a disproportionate amount staked into voting, an adversary can compromise the outcome of a randomly drawn proposition. However, if the maximum voting deposit is much smaller than the total voting stake on a proposition, then an adversary would be hard-pressed to repeatedly draw the same proposition to control its validity. On the other hand, the minimum certifying deposit "should be large enough that certifiers incur sufficient risk." This is the primary incentive for certifiers to behave honestly. [1]

Let's define some variables before elaborating upon the technicalities of the voting and certification process [2]:

- N players
- proposition list P of fixed size $|P|$
    - each $p_i \in P$ has a truth value t_i and bounty B_i
- two certifier reward pools R_T and R_F
- $s_{i,j,b}$ is the amount that player i has staked on voting a proposition p_j as b where $b \in \{T, F\}$
    - if player i has not voted on proposition p_j, then $s_{i,j,T}$ and $s_{i,j,F}$ are 0
    - otherwise, one or both (player i may vote more than once) are equal to the stake they put in
- $\sigma_{i,j,b}$ is the amount of stake that player i used to certify a proposition p_j as b where $b \in \{T, F\}$

- s_{max} and σ_{min} are the maximum voting deposit and minimum certifying deposit, respectively
- $S_{TOT,\,j,\,T}$, $S_{TOT,\,j,\,F}$, $\sigma_{TOT,\,j,\,T}$, $\sigma_{TOT,\,j,\,F}$ are the total voting stake for true, total voting stake for false, total certifying stake for true, and total certifying stake for false, respectively

The steps for an Astraea protocol can be summarized as follows [2]:

1. **Voting:** As described earlier, each voter stakes a deposit, receives a proposition uniformly at random, and submits a boolean vote. The voting process is played over all propositions P submitted by submitters simultaneously. Therefore, it is possible that a voter may vote on the same proposition more than once.
2. **Certifying:** As described earlier, each certifier chooses a proposition, stakes a deposit, and submits a boolean certification. Because certifiers can choose which proposition they wish to certify, it is possible that not all propositions will be certified.
3. **Termination and decision:** A proposition is decided when it accumulates a sufficient voting stake, and this is decided by a parameter of the system. Intuitively, voting outputs true if $S_{TOT,\,j,\,T} > S_{TOT,\,j,\,F}$ and outputs false if $S_{TOT,\,j,\,F} > S_{TOT,\,j,\,T}$. If $S_{TOT,\,j,\,T} = S_{TOT,\,j,\,F}$, the outcome is inconclusive. The same logic applies to certification. **Astraea outputs the voting outcome if it matches the certification outcome OR the certification outcome is inconclusive.**

Case I: outputting true or false

If the outcome is *T*, the voting reward is the share of the *T*- voting stake times the proposition's bounty amount minus the *F*- voting stake:

$$r_v = \left(\frac{s_{i,j,T}}{s_{TOT,j,T}}\right) \times B_j - s_{i,j,F}$$

The certifier's reward is the share of the *T*- certifying stake times the true certifier reward pool R_T divided by the certification target τ. The certification target is the number of certifiers the certifier reward pool can pay for for a certain proposition. For instance, if $R_T = 1000$ and $\tau = 10$, then the reward pool can distribute 100 monetary units to the certifiers and $R_T = 900$ for the next proposition.

$$r_c = \left(\frac{\sigma_{i,j,T}}{\sigma_{TOT,j,T}}\right) \times \left(R_T \times \frac{1}{\tau}\right) - \sigma_{i,j,F}$$

The calculations are similar in the case of an *F* outcome.

Case II: the outcome is unknown

If the outcome is unknown, voters are neither rewarded nor penalized while certifiers as penalized according to

$$r_c = -\left(\sigma_{i,j,F} + \sigma_{i,j,T}\right)$$

Voters are not penalized because they do not choose which propositions to vote on whereas certifiers choose which propositions to certify. It should be rare that certifiers choose to certify such a split proposition.

TABLE IV
SUMMARY OF REWARDS AND PENALTIES

	Reward		Penalty	
Outcome	**Voters**	**Certifiers**	**Voters**	**Certifiers**
T	$\left(\frac{s_{i,j,T}}{s_{TOT,j,T}}\right) \times B_j$	$\left(\frac{\sigma_{i,j,T}}{\sigma_{TOT,j,T}}\right) \times \left(R_T \times \frac{1}{\tau}\right)$	$s_{i,j,F}$	$\sigma_{i,j,F}$
F	$\left(\frac{s_{i,j,F}}{s_{TOT,j,F}}\right) \times B_j$	$\left(\frac{\sigma_{i,j,F}}{\sigma_{TOT,j,F}}\right) \times \left(R_F \times \frac{1}{\tau}\right)$	$s_{i,j,T}$	$\sigma_{i,j,T}$
Ø	0	0	0	$\sigma_{i,j,F} + \sigma_{i,j,T}$

Figure 10: Summary of rewards and penalties in the Astraea protocol. Source: [2].

The Astraea protocol encourages a Nash equilibrium in which all players are honest, assuming every voting outcome is correct and all feasible strategies are considered. This paper will not analyze the details of Nash equilibrium, but "even an adversary with perfect accuracy who controls 100% of the certifying stake is incentivized to play honestly or not at all" or risk losing all of their stake [1].

4.3 Town Crier

Before the groundbreaking launch of Chainlink, Town Crier (TC) was developed in 2016 by students and researchers at Cornell University. Unlike a decentralized oracle network, TC is a centralized oracle solution that ensures correct answers with a singular node and functions as an authenticated data feed (ADF). TC combines a "blockchain front end with a [trusted execution environment (TEE)] backend to scrape HTTPS-enabled websites and serve source-authenticated data to relying smart contracts" [42]. Nevertheless, we will examine how TC can be incorporated with Chainlink to provide even stronger security guarantees.

TC's TEE is implemented with Intel's Software Guard Extensions (SGX) technology, which will be discussed later. A TEE is a hardware-based capability that provides both integrity and confidentiality in an enclave. Integrity requires that code in an enclave be untamperable, and confidentiality means that it should be infeasible for an attacker to gain any information about the contents in an enclave. Under these assumptions, TC can handle sensitive data and run programs securely. [27]

Understandably, some individuals may be skeptical of TC's security because of the centralization problem (there is still technically a single point of failure). However, TC offers many security strengths and applications. In particular, it is possible to use both a decentralized structure, and TC's TEE guarantees to create a complementary and even more secure system. According to Ari Juels, a developer of Town Crier and member of Chainlink's research team, "a decentralized system that uses TEEs—even flawed ones—is strictly stronger than a system that doesn't" [27].

Intel's SGX is a set of instructions built into Intel's processors that offers hardware-based protection on user-level code. Integrity and confidentiality are maintained in a protected address space called an enclave. TC confirms that software is running in an SGX-protected enclave with an attestation, a digital signature derived from the program image, some user data, and a secret key known only to the hardware. The SGX enclaves "inherits the black box implementation properties of a TEE"—that is, neither the operating system nor other applications on the CPU can interfere with software running inside the enclave [1]. These capabilities enable data collection from multiple data sources and secure communication back to blockchain smart contracts.

4.3.1 Town Crier Architecture and Security

TC has three main components. The TC contract resides on the blockchain, while the enclave and relay reside on the TC server. [42]

- **TC smart contract:** The TC smart contract offers a simple API for the user contract to make requests to TC. There is no guarantee of confidentiality, and its execution may be expensive.
- **TC enclave:** As described earlier, the SGX-backed TC enclave allows the software to securely fulfill requests from the blockchain. Computation in the TC enclave is efficient because it runs on the CPU. However, there is no network stack, and it therefore lacks direct network access.
- **TC relay:** Because the enclave lacks direct network access, the relay "provides network connectivity from the enclave to three different types of entities": the blockchain, off-chain service requests from clients, and data sources [42].

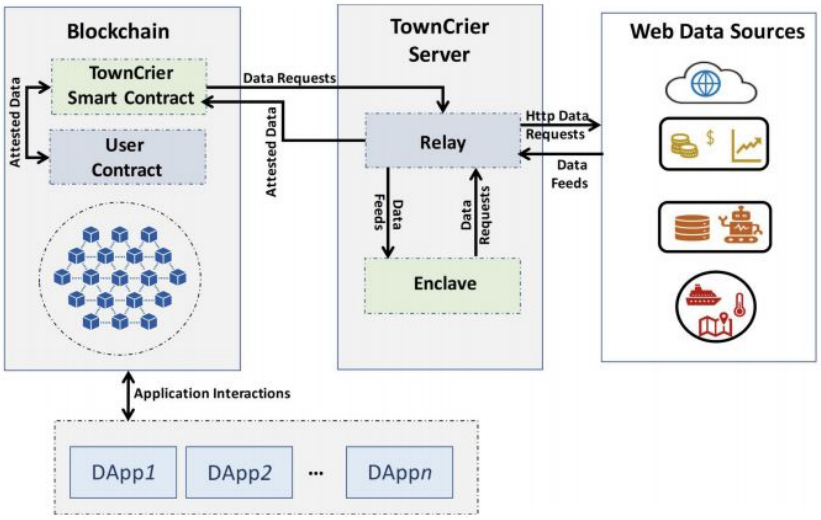


Figure 11: Town Crier architecture. Source: [1].

We assume the following for TC's security model [40]:

- The TC contract is secure because it is publicly visible on the blockchain, and its source code is also published. Therefore, we assume it behaves honestly.
- Data sources behave honestly and provide reliable information because they have a reputation to maintain.
- The SGX-backed enclave provides security and confidentiality guarantees.
- Communication on the blockchain is trustworthy because it is authenticable.
- Since the relay does not have SGX-backed integrity protection, it may be compromised by an adversarial operating system.

Despite the relay's vulnerabilities, TC is designed to guarantee authenticity at each point of communication between the components mentioned above. Consider the diagram below, which illustrates how an adversary controlling the network could corrupt query *Q* and convert it to *Q'*.

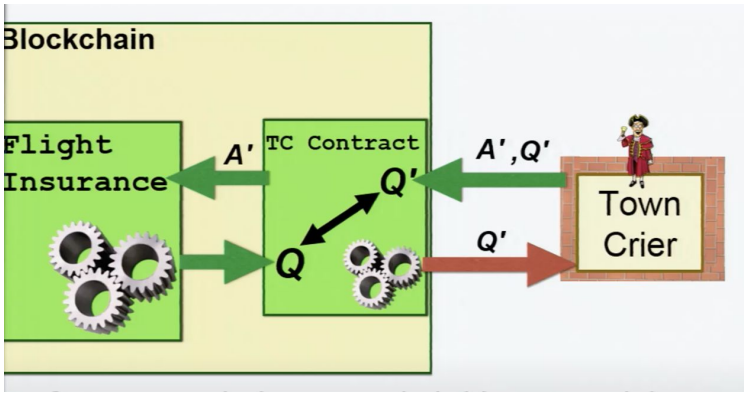


Figure 12: The blockchain interacting with TC. Source: [40]

The solution is rather simple. TC should go ahead with processing the potentially corrupted query *Q'*. Then, TC digitally signs and returns answer *A'* with the query *Q'* together as (*A', Q'*). Next, the TC contract checks if *Q* = *Q'*. If so, it returns *A,'* and otherwise, it throws out the response.

Now let's examine the relay's communication with web data sources. Consider the diagram below, which illustrates where a malicious OS may alter the data retrieved from a trusted website. Because the HTTPS protocol does not digitally sign data for out-of-band verification, TC is unable to verify data delivered by the OS.

To provide context, here is some important background terminology related to network security protocols [25]:

- **HyperText Transfer Protocol Secure (HTTPS):** HTTPS is a secured protocol used for communication over the Internet.
- **Transport Layer Security (TLS):** TLS is a widely adopted security protocol used to secure HTTPS by encrypting the communication between websites and servers.
- **Transmission Control Protocol (TCP):** TCP handles packet ordering and error checking to ensure that communication between parties in the network is successful.

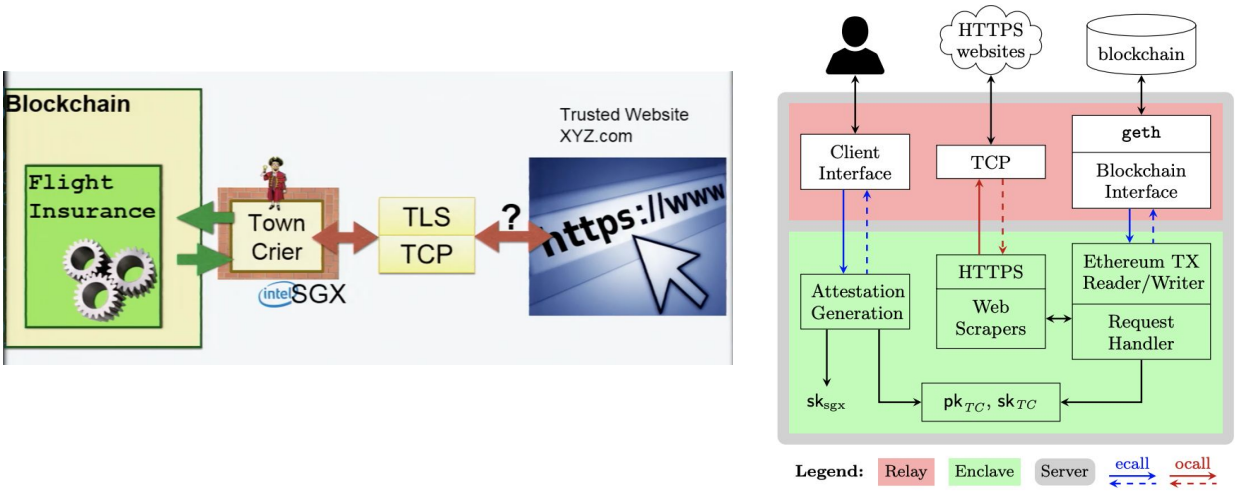


Figure 13: Network vulnerability (L) and TC's resolution (R). Source:[40]

TC remedies the problem of a potentially malicious OS by partitioning HTTPS into "a trusted layer consisting of HTTP and TLS code and an untrusted layer that provides low-layer network service, specifically TCP" [42]. This arrangement allows the TLS stack to execute inside the enclave. The enclave performs the TLS handshake with a target server and performs all cryptographic operations internally in the SGX environment. At the same time, the remaining untrusted process

TCP acts as a simple network interface. This new arrangement protects TC from a malicious OS and protects the channel between the enclave and the data source.

In addition to the advantages of an SGX-backed enclave, TC ensures gas sustainability, meaning a user never raises an out-of-gas error. As covered in Section 2.2.2, gas is the unit of computation that represents the cost to perform a transaction on the Ethereum blockchain. TC requires an up-front deposit of gas (Ether) from a relying contract to prevent users from running out of gas in the middle of executing a smart contract.

4.3.2 What can Town Crier offer Chainlink?

Town Crier and Chainlink have very different architectures and approaches to the Oracle Problem. Nevertheless, Chainlink acquired TC in 2018 to expand oracles' capabilities. Despite TC's offerings, Chainlink continues to monopolize the oracle space. I reached out to Fan Zhang, the lead developer of TC, and asked his thoughts on why. Fan echoed Ari Juels's sentiments that **when used together, TC can enhance Chainlink in many ways.**

The main advantage a decentralized oracle network like Chainlink has over TC is "an infrastructure for timely, robust, and efficient delivery of data to blockchains." Data sources could, in theory, sign their data on-chain (more on this in Section 5.1), but there are several engineering and operational challenges that make this difficult. It is conceivable that data sources may not wish to invest in such an infrastructure, in which case they can rely on Chainlink to provide trustworthy oracle services customizable for every conceivable application. Using Chainlink as a backbone, TC can offer various enhancements to Chainlink's decentralized architecture.

One enhancement TC provides is the ability "to generate publicly verifiable TLS proofs, which is hard to do otherwise." However, various attacks have recently emerged to attack TC and its trusted execution environment (TEE). In Section 4.4, we will explore a system called DECO and show how we can use a committee of trusted nodes to cryptographically generate TLS proofs. We will also reinforce the benefits of incorporating such a system to enhance Chainlink, similar to what TC strives to do.

Another enhancement that TC provides is performing privacy-preserving computation, one of its main advantages over other oracle solutions. Due to the blockchain's public nature, the smart contract has no confidential state to store private information. Fortunately, TC's trusted hardware can guarantee confidentiality regardless. Consider the following scenario: Alice has flight insurance and is using an oracle network to determine whether her flight is delayed or not. She does not want to send her flight information to the blockchain because it would be made public to the entire network. Instead, she sends an encrypted version of her flight information to TC. Then, TC uses its private key to decrypt Alice's information, extract the relevant information (flight delayed/not delayed), and send

it to the blockchain. Finally, the flight insurance smart contract is executed with the data, transferring funds to Alice if her flight is delayed.

Although it is unlikely that Chainlink will be superseded because of compelling infrastructural reasons, Chainlink is actively acquiring novel oracle technologies such as Town Crier and DECO (Section 4.4) to enhance its network and reinforce it as the primary oracle solution for the blockchain.

4.4 DECO

Due to the ubiquity of Transport Layer Security (TLS) on the Internet, users can securely communicate data over channels with end-to-end confidentiality and integrity. Despite this, they cannot "prove to third parties the provenance of such data, i.e., that it genuinely came from a particular website." As a result, there are lots of unusable private data locked up in web servers, such as bank account balances, identity information, and private enterprise data, that greatly limit smart contracts' capabilities. [41]

This is where DECO (short for Decentralized Oracle) comes in. DECO aims to liberate this data so it can be used in blockchain systems. According to the white paper, "DECO allows users to prove that a piece of data accessed via TLS came from a particular website and optionally prove statements about such data in zero-knowledge, keeping the data itself secret" [41]. Another critical advantage of DECO is that it is source-agnostic and can be used by any website running standard TLS. There is no need to customize per-website support, enabling anyone to become an oracle for any website.

To better understand the problem that DECO solves, let's consider the following scenario: Alice wants to prove to Bob that she has at least $10,000 in her bank account. Her first idea is to send a screenshot of her bank account balance to Bob, but screenshots can easily be forged with photo editing software. Her second idea is to send her login credentials to Bob, but this is a poor habit to get into and introduces undesirable trust assumptions (i.e., Bob is trustworthy).

The commonality in these two ideas is that Alice uses TLS to create a secure channel to connect with a server. It may seem like we can leverage TLS to authenticate Alice's data to a third party, but there are some limitations of TLS. Although TLS uses public-key cryptography and digital signatures to perform key exchange, TLS exchanges symmetric keys for the session. In other words, both Alice and the server share the same key called a message authentication code (MAC) when they exchange data. Therefore, Alice is just as capable of signing the data as the server is, and "forwarding TLS data to Bob is no better than forwarding a screenshot" [26]. There is still no provable way for Alice to show Bob that the data came from the server. There are proposals to change TLS, but this would require sizable infrastructural overhead with ramifications on the entire Internet, so instead, we

turn to DECO for the blockchain. With DECO, Alice can prove to Bob that she has at least $10,000 in her bank account without revealing her bank account balance or modifying the server.

4.4.1 DECO Architecture and Security

This paper will give an overview of the DECO architecture but will not dive into the complexities and technicalities of the full protocol. For notation purposes, let P represent the prover, V the verifier, and S the TLS server. The workflow of DECO is shown below.

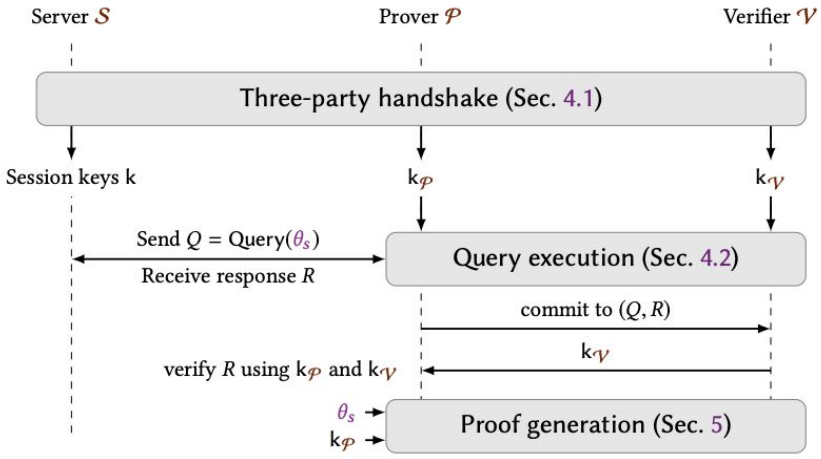


Figure 14: DECO architecture. Source: [41]

The main idea behind DECO is "a novel *three-party handshake* protocol between the prover, verifier, and web server that creates an *unforgeable commitment* by the prover to the verifier on a piece of TLS session data D" [41]. If the verifier is honest, it can check that D is authentic and from the server. Otherwise, it will not be able to.

When the prover establishes a session key K with the server, this key is split on the client side between the prover and the verifier, where K_P is the prover's and K_V is the verifier's. Therefore, $K = K_P + K_V$. Here is a simplified version of the DECO three-party handshake:

1. The verifier V generates a private key X_V at random and uses X_V to create a public key Y_V, which V sends to the prover.
2. The prover P also generates a private key X_P, a corresponding public key Y_P, and sends the server the product of P's public key and X's public key, denoted $Y_P Y_V$.
    a. **It is important to note that P commits Y_P to the session data *before* receiving Y_V, making the commitment unforgeable, while establishing communication with the server.**

3. By way of additive homomorphism, the private key X corresponding to the public key sent to the server is the sum of X_P and X_V.

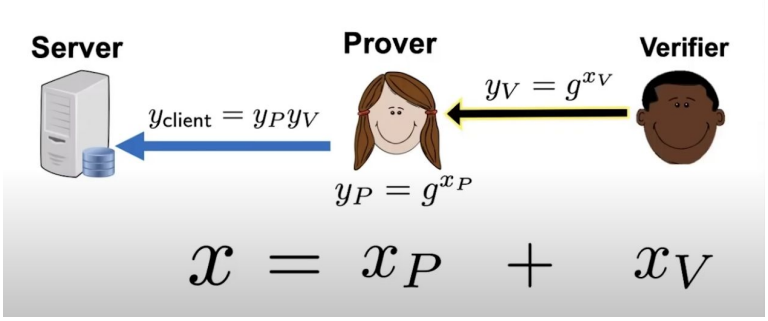


Figure 15: Simplified depiction of the DECO protocol. Source: [26]

The intuition is that the prover and verifier generate a shared key K for the TLS session, which they use jointly. The prover cannot forge data D from the server because it commits Y_P to the session data before receiving Y_V. However, the prover can prove things about D to the verifier in zero-knowledge. Although the session key is shared between P and V, the prover is strictly the only party that communicates with the server, and any queries it sends should not be leaked to the verifier.

DECO offers two techniques to support proof generation in zero-knowledge: 1) selective opening of a TLS session transcript to reveal specific lines of data and 2) redacting a substring containing confidential information to conceal it from V. A prover would not want to reveal the decryption key for its TLS session because it would reveal the entire session data.

However, the two methods mentioned above are not enough to provide context integrity protection because "the context of a substring [could] affect its meaning" [41]. To remedy this, DECO utilizes a technique named zero-knowledge two-stage parsing. The idea is relatively simple: The prover parses its session data locally to remove context integrity vulnerabilities. Only then does it send any data to the verifier.

Finally, the verifier can easily verify the authenticity of the data with simple decryption.

4.4.2 What can DECO offer Chainlink?

A user cannot unilaterally generate a DECO proof to place onto the blockchain. **DECO must be integrated with an oracle network like Chainlink that provides the infrastructure for data delivery to the blockchain.** Just as it is advantageous to integrate Town Crier with Chainlink, it is advantageous and perhaps *necessary* to integrate DECO with Chainlink. DECO alone does not have built-in support for aggregating multiple DECO proofs and arriving at a final answer; this is the task of a layer of Chainlink oracle nodes.

The following are a few of many applications of DECO for Chainlink [26]:

- **Private enterprise data:** Private enterprises would like to take advantage of smart contracts to execute an automatic transaction when goods are shipped to their destination. However, they may not want to disclose what goods they are shipping (maybe highly valuable) on the blockchain. The shipment details may be represented in a cryptographically concealed form C in the smart contract. The client can query the shipment service for delivery status information, generate a DECO proof for the oracle nodes, and the oracle can confirm that C has arrived without revealing the contents of C.
- **Decentralized identity:** Suppose that Alice wants to take out a loan from a bank and needs credentials to prove that she is over 18. She can access a government website with an authoritative record of her birth date, use it to generate a DECO proof for the oracle network, and have the oracle network report to the bank's smart contract that she is indeed over 18. Using this method, Alice can take out her loan without revealing her exact birth date.
- **Confidential DeFi:** Suppose Alice and Bob are betting on the value of some asset X, and both stake 100 ETH into the smart contract. However, Alice and Bob do not want the oracle network to learn what X or their contract terms are. So, they store a cryptographically concealed representation of X and its terms into the smart contract. Alice and Bob access a trustworthy site and use the DECO protocol to prove to the oracle network, which of them won the bet. Using this method, the oracle enables the smart contract's successful execution without learning X or its terms.

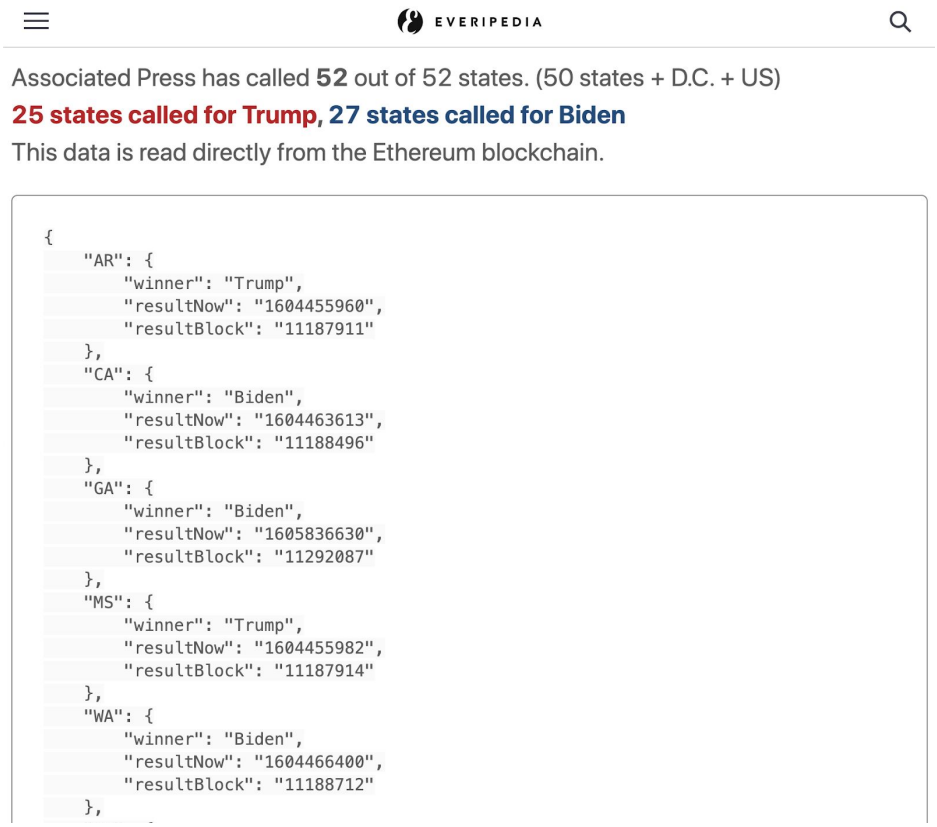
5 Future Prospects

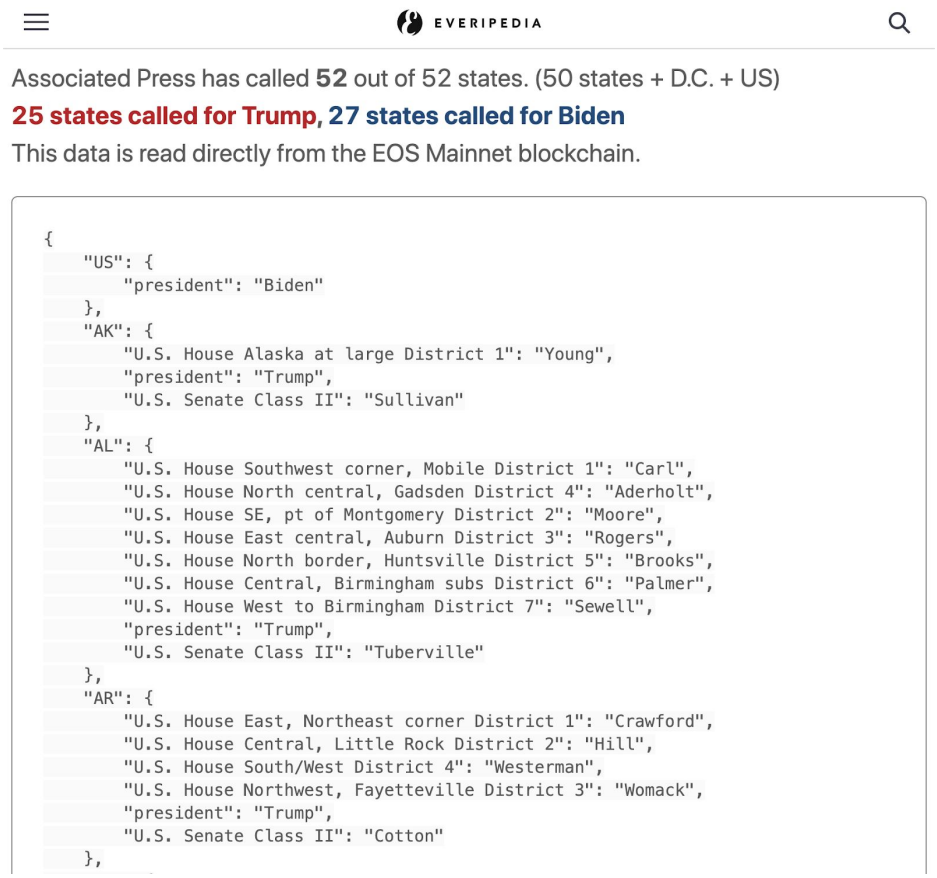
5.1 Case Study: Everipedia's Collaboration with the Associated Press

I recently took advantage of my Kleiner Perkins Fellow alumni network and reached out to Dawson Botsford, a software engineer at Everipedia. Everipedia is a decentralized knowledge platform coined as "the world's first encyclopedia to use blockchain technology."

Everipedia collaborated with the Associated Press for the 2020 US Presidential Election to fetch election results and publish them to the EOS and Ethereum blockchains. In other words, the Associated Press directly transferred its data to the blockchain.

In general, gathering data traditionally requires a third party to hit some HTTP API endpoint from a data source and communicate it to its platform. Consider large news networks like BBC, CNBC, and Fox that broadcast election results. They probably hit these endpoints and get their data from a primary source like the Associated Press. However, placing a news network as a middle point between the primary source and the audience creates a medium with which the news network can alter data to fit their narrative. A conservative news network may highlight more Republican results, and a Democratic new network may highlight more Democratic results, fogging up data transparency. This is an inherent risk with centralized systems.





Figures 16 and 17: Everipedia dashboard view of data published from the Associated Press to the EOS and Ethereum blockchains. Source: [4]

Everipedia's election oracle is in its own category. Dawson provisionally calls it a "first-party oracle" because Everipedia's oracle bypasses a middle point altogether. The idea is that most news networks get their data from a single source: the Associated Press. If this is true and the Associated Press signs their data on-chain, there is no need for third parties—not even decentralized oracles—to hit any API endpoints. According to Botsford, "the implications of this election project was that we allowed the Associated Press to publish their data in a way that is more trustless than anything ever done before": in other words, directly to their audience. People are responding affirmatively as well. In fact, "there was over $250,000 locked into the smart contract to resolve the winner of the election," Botsford said. Moving forward, Everipedia wants to explore this "first-party oracle" idea with even more clients—weather, sports, international elections, and more.

The integrity of the first-party oracle system is based on the premise that data providers have a reputation to maintain both on- and off-chain. Therefore, they will not intentionally publish false data. Otherwise, users will lose trust in them.

Let's consider weather data and its effects on farming insurance. Farming insurance is a form of coverage designed to protect people who operate and live on farms. If a particular year experiences bad weather and affects agricultural production, these farm owners will still be financially supported. Conceptually, this idea is nothing new, but its methods of implementation are. Chainlink only recently started providing decentralized weather data to farming insurance startup Arbol to accomplish this in Aug 2020 [20]. But there is a catch: there are not many sources of weather data, so why not trust one primary weather network? This is where a first-party oracle may substitute a decentralized oracle network, potentially cutting down costs and resources.

The oracle methodology that Dawson is working on is quite different from Chainlink. As described in Section 4.1, Chainlink is a decentralized, consensus-based oracle that aggregates data from multiple sources, processes it, and outputs a trustworthy answer. However, it is possible to entirely remove Chainlink from the workflow if data providers sign the data on-chain to validate its integrity. Everipedia is working on pitching this workflow to data providers so that first-party oracles can become "the software solution for companies to publish their data in a trustworthy, verifiable way," says Botford. No consensus or decentralization is needed.

In Section 4.3.2, Fan Zhang cites engineering and operational challenges as one reason why data sources signing their data on-chain is not very prevalent. Often, data providers also profit off of selling multiple copies of their data. Dawson's work challenges these constructs, and it will be interesting to follow Everipedia's work moving forward.

5.2 Working Together

Cryptography is a rapidly developing field where attacks are often discovered for previously thought secure protocols. For instance, Foreshadow is "a software-only microarchitectural attack that decisively dismantles the security objectives of current SGX implementations," rendering Town Crier vulnerable [11]. This led to the subsequent development of DECO, which offers improved end-to-end performance without added trust assumptions.

Chainlink's long list of acquisitions suggests that they are aware of this and hope to integrate different technologies' strengths with one another. Chainlink's aggressive acquisition and partnership strategy may well be the reason for the explosive growth since its conception in 2017. As cryptocurrency continues to grow, it is wise to follow how oracle projects build off of and cooperate with each other, as evidenced by Chainlink, Town Crier, and DECO.

6 Conclusion

Blockchain networks are clearly capable of much more than the sole transfer of currency. With smart contracts, the possibilities become limitless, extending into every conceivable domain. Therefore, it is vital to invest in an infrastructure of trustworthy oracle solutions for blockchain networks to interface with the external world. Spearheaded by Chainlink, several oracle solutions have been developed in recent years to parallel the explosive growth of the blockchain. The major insight from this paper is the value of integrating novel solutions with each other, such as Chainlink and Town Crier or Chainlink and DECO. These oracle projects are not isolated competitors but rather complementary to each other. In the former, Town Crier's TEE provides greater security for Chainlink and in the latter, DECO provides confidentiality guarantees for Chainlink. In both, Chainlink serves as the infrastructural backbone for timely and efficient delivery of data to the blockchain. As cryptocurrencies continue to advance and especially in light of Ethereum 2.0, it will be interesting to follow developments in the blockchain/oracle space to watch which applications arise and thrive.

Works Cited

[1] Al-Breiki, Hambda, et al. “Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges.” Shibboleth Authentication Request, 2020, ieeexplore-ieee-org.proxy.library.upenn.edu/stamp/stamp.jsp?tp=.

[2] Adler, John, et al. “Astraea: A Decentralized Blockchain Oracle.” Arxiv.org, arxiv.org/pdf/1808.00528.pdf.

[3] Antonopoulos, Andreas M., author. “Mastering Ethereum : Building Smart Contracts and DApps /.” Mastering Ethereum : 2018. [4]

[4] “AP Election Mission Control.” Everipedia.org, everipedia.org/oraqle/ap.

[5] Beale, Mat. “Staking with Chainlink.” Medium, LinkPool, 10 June 2019, medium.com/linkpool/staking-with-chainlink-b58eb3de6f1b.

[6] “Benefits and Drawbacks of Permissioned Blockchains.” Coinstelegram, 15 Dec. 2018, coinstelegram.com/2018/12/15/benefits-and-drawbacks-of-permissioned-blockchains/.

[7] Beniiche, Abdeljalil. A Study of Blockchain Oracles, 14 July 2020, arxiv.org/pdf/2004.07140.pdf.

[8] Beyer, Dr. Stefan. “Blockchain Before Bitcoin: A History.” Block Telegraph, 23 Aug. 2018, blocktelegraph.io/blockchain-before-bitcoin-history/.

[9] Binance Academy. “History of Blockchain.” Binance Academy, Binance Academy, 19 Jan. 2020, academy.binance.com/blockchain/history-of-blockchain.

[10] “Blockchain.com Charts Summary.” Blockchain.com, www.blockchain.com/charts.

[11] Bulck, Jo Van, et al. “Foreshadow: Extracting the Keys to the Intel {SGX} Kingdom with Transient Out-of-Order Execution.” USENIX, 1 Jan. 1970, www.usenix.org/conference/usenixsecurity18/presentation/bulck.

[12] Buterin, Vitalik. “Ethereum Whitepaper.” Ethereum.org, 2013, ethereum.org/en/whitepaper/.

[13] Coinbase. “A Beginner's Guide to Decentralized Finance (DeFi).” Medium, The Coinbase Blog, 28 Aug. 2020, blog.coinbase.com/a-beginners-guide-to-decentralized-finance-defi-574c68ff43c4.

[14] Crosby, Michael. “Blockchain Technology Beyond Bitcoin.” Scet.berkeley.edu, scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf.

[15] CryptoChamp. “How Is the Bitcoin Mining Difficulty Calculated?” cryptochamp.com/how-is-the-bitcoin-mining-difficulty-calculated/.

[16] “Cryptocurrency Prices, Charts And Market Capitalizations.” CoinMarketCap, coinmarketcap.com/.

[17] D. Khan, L. T. Jung, M. Ahmed Hashmani and A. Waqas, "A Critical Review of Blockchain Consensus Model," 2020 3rd International Conference on Computing, Mathematics and

Engineering Technologies (iCoMET), Sukkur, Pakistan, 2020, pp. 1-6, doi: 10.1109/iCoMET48670.2020.9074107.

[18] Ellis, Steve, et al. Chainlink A Decentralized Oracle Network. link.smartcontract.com/whitepaper.

[19] "File:Cryptographic Hash Function.svg." File:Cryptographic Hash Function.svg - Wikimedia Commons, commons.wikimedia.org/w/index.php?curid=5290240.

[20] Foxley, William. "Chainlink to Provide Data for Farming Insurance Startup Arbol." CoinDesk, CoinDesk, 19 Aug. 2020, www.coindesk.com/chainlink-to-provide-data-for-farming-insurance-startup-arbol.

[21] Greenspan, D. G. (2016, April 17). Why Many Smart Contract Use Cases Are Simply Impossible. Retrieved from coindesk.com: https://www.coindesk.com/three-smart-contractmisconceptions/

[22] Haber, Stuart, and W. Scott Stornetta. "How to Time-Stamp a Digital Document." Journal of Cryptology, Springer-Verlag, 1 Nov. 1984, link.springer.com/article/10.1007/BF00196791.

[23] Haig, Samuel. "Bitcoin Block Size, Explained." Cointelegraph, Cointelegraph, 2 Dec. 2019, cointelegraph.com/explained/bitcoin-block-size-explained.

[24] "History of Blockchain." ICAEW, www.icaew.com/technical/technology/blockchain/blockchain-articles/what-is-blockchain/history.

[25] Hiwarale, Uday. "A Brief Overview of the TCP/IP Model, SSL/TLS/HTTPS Protocols and SSL Certificates." Medium, JsPoint, 1 Sept. 2020, medium.com/jspoint/a-brief-overview-of-the-tcp-ip-model-ssl-tls-https-protocols-and-ssl-certificates-d5a6269fe29e.

[26] Juels, Ari. "Chainlink Smartcon 2020."

[27] Juels, Ari. "Town Crier and Chainlink: Enriching the Function of Blockchain Oracles." Chainlink, Chainlink, 13 May 2020, blog.chain.link/town-crier-and-chainlink/.

[28] Keoun, Bradley. "5 Reasons Why Bitcoin Just Hit an All-Time High Price." CoinDesk, CoinDesk, 30 Nov. 2020, www.coindesk.com/five-reasons-why-bitcoin-just-hit-all-time-high-price.

[29] Kumar, Manoj, et al. Decentralising Finance Using Decentralised Blockchain Oracles.

[30] "What Are Public Keys and Private Keys?" Ledger, www.ledger.com/academy/blockchain/what-are-public-keys-and-private-keys.

[31] "PoW - Proof of Work." Horizen Academy, academy.horizen.io/technology/expert/proof-of-work/.

[32] "Public and Private Keys." Blockchain Support Center, 27 Sept. 2020, support.blockchain.com/hc/en-us/articles/360000951966-Public-and-private-keys.

[33] Qureshi, Haseeb. “Public-Key Cryptography.” NAKAMOTO, NAKAMOTO, 2 Nov. 2020, nakamoto.com/public-key-cryptography/.
[34] Siriwardena, Prabath. “The Mystery Behind Block Time.” Medium, FACILELOGIN, 8 July 2018, medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a.
[35] “The Eth2 Upgrades.” Ethereum.org, ethereum.org/en/eth2/.
[36] Walker, Greg. “Longest Chain.” The Longest Chain - Blockchain Guide, learnmeabitcoin.com/technical/longest-chain.
[37] Wu, Xun, author. “Learn Ethereum : Build Your Own Decentralized Applications with Ethereum and Smart Contracts /.” Learn Ethereum : 2019.
[38] Yaga, Dylan, et al. Blockchain Technology Overview. www.bkd.com/sites/default/files/2019-12/NIST.IR_.8202.pdf.
[39] Yan Chen, Cristiano Bellavitis, Blockchain disruption and decentralized finance: The rise of decentralized business models, Journal of Business Venturing Insights, Volume 13, 2020, e00151, ISSN 2352-6734, https://doi.org/10.1016/j.jbvi.2019.e00151.
[40] Zhang, Fan. “CCS 2016 - Town Crier: An Authenticated Data Feed for Smart Contracts.”
[41] Zhang, Fan, et al. DECO: Liberating Web Data Using Decentralized Oracles for TLS. Aug. 2020, arxiv.org/pdf/1909.00938.pdf.
[42] Zhang, Fan, et al. Town Crier: An Authenticated Data Feed for Smart Contracts. eprint.iacr.org/2016/168.pdf.