UNIVERSITY OF THESSALY

BACHELOR THESIS

A Case Law Decentralized Public Ledger

Αποκεντρωμένο Δημόσιο Κατάστιχο Νομολογιών

Author:
Eleni PANAGOU

Supervisor:
Prof. Emmanouil VAVALIS

A thesis submitted in fulfillment of the requirements for the degree of Diploma in Engineering

in the

Department of Electrical and Computer Engineering

February 27, 2020

Declaration of Authorship

I, Eleni PANAGOU, declare that this thesis titled, "A Case Law Decentralized Public Ledger" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"I am not a visionary. I'm an engineer. I'm happy with the people who are wandering around looking at the stars but I am looking at the ground and I want to fix the pothole before I fall in."

Linus Torvalds

UNIVERSITY OF THESSALY

Abstract

Department of Electrical and Computer Engineering

Diploma in Engineering

A Case Law Decentralized Public Ledger

by Eleni PANAGOU

Case law is the term used to refer to reports of past court decisions and is considered an essential source of law. Current legal research services are centralized, have complete control over the data and often charge fees for its access. To address the aforementioned issues, this thesis attempts to leverage the potential of the Ethereum blockchain in order to form a public and decentralized platform which allows the submission of court decisions in a decentralized database, while achieving their validation, classification and evaluation via a network of contributors. The focus of this work is the establishment of a sybil-resistant voting protocol used for reaching agreement and the development of a tokenized economy used to incentivize participation.

Ο όρος νομολογία αναφέρεται σε δικαστικές αποφάσεις του παρελθόντος και αποτελεί σπουδαία πηγή νόμου. Οι τρέχουσες υπηρεσίες νομικής έρευνας είναι συγκεντροποιημένες, έχουν απόλυτο έλεγχο των δεδομένων και συχνά επιβάλλουν χρεώσεις για την πρόσβαση σε αυτά. Για να αντιμετωπίσει τα προαναφερόμενα ζητήματα, η παρούσα διατριβή επιχειρεί να εκμεταλλευτεί το Ethereum blockchain με σκοπό τη δημιουργία μίας δημόσιας και αποκεντρωμένης πλατφόρμας που επιτρέπει την εισαγωγή δικαστικών αποφάσεων σε μία αποκεντρωμένη βάση δεδομένων, επιτυγχάνοντας ταυτόχρονα την επικύρωση, κατηγοριοποίηση και αξιολόγησή τους μέσω ενός δικτύου συνεργατών. Έμφαση δίνεται στη σχεδίαση ενός πρωτοκόλλου ψηφοφορίας που αντιστέκεται σε επιθέσεις Sybil και χρησιμοποιείται για την επίτευξη συμφωνίας, καθώς και στην ανάπτυξη μίας tokenized οικονομίας που λειτουργεί ως κίνητρο για συμμετοχή.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor Prof. Manolis Vavalis who gave me great help and guidance during the development of this thesis. I would like to thank him for his support and faith in me and my ideas. Working with him gave me the opportunity to learn a lot about research, allowed me to gain confidence in my knowledge and capabilities as an engineer and helped me mature as a person.

I also wish to thank my father, Nikos, my mother, Kelly and my brother, Apostolos for their unceasing encouragement throughout the entirety of my studies. This accomplishment would not have been possible without their great love and support.

Finally, I would like to thank my partner David who was always supporting me and putting up with me during difficult times. I will be forever grateful for his unconditional love, for all the sacrifices he made to be near me and for never letting me quit, every time I was ready to.

Contents

List of Abbreviations

API	**A**pplication **P**rogramming **I**nterface
DApp	**D**ecentralized **App**lication
DAO	**D**ecentralized **A**utonomous **O**rganization
DID	**D**ecentralized **ID**entifier
ICO	**I**nitial **C**oin **O**ffering
IPFS	**I**nter**P**lanetary **F**ile **S**ystem
PoW	**P**roof **o**f **W**ork
PoS	**P**roof **o**f **S**take
UI	**U**ser **I**nterface
UX	**U**ser e**X**perience

Dedicated to my family and friends.

Chapter 1

Introduction

1.1 Case Law

The essence of case law is captured by the principle of *stare decisis*, which is Latin for "to stand by decided matters". This principle compels courts to adhere to past rulings in order to avoid contradicting themselves when applying the law. In other words, cases that present similar facts or similar key points should be treated the same. Consistency is of utmost importance. Courts are not only discouraged from contradicting other courts' decisions, but are also outright forbidden from doing so when faced with the decision of a higher or equally ranked court in the hierarchy of the jurisdiction in question. The reason for adhering to this principle is the aspiration of maintaining a standard enforcement of the law.

It is common for the interpretation of the law to be unclear, as statutory language tends to be ambiguous [33] and at times the legality of a given act can be uncertain. In addition, when considering extraordinary cases, it is reasonable to assume that legislation could be insufficiently developed and thus incapable of offering a straightforward resolution to the matter. In these cases, case law, if available, becomes the catalyst for the case at bar as well as future cases of the same kind. Since courts are obligated to avoid contradictions with past rulings, case law constitutes an indispensable tool for legal professionals and is therefore both an extension and a source of law itself.

1.2 Motivation

The provision of reliable legal information for attorneys and law firms in a consistent, digitized format has been the focus of several companies in recent years. Services such as LexisNexis Case Law [27], WestLaw [50] and Bloomberg [9] provide comprehensive collections of case law. They employ attorney editors who review, analyze and summarize cases. Others, such as Luminance [29] and Eigen Technologies [16], a startup funded by Goldman Sachs, utilize AI in order to automate the extraction of useful data from legal documents. Finally, there are also services which strive for free and open access to case law for the public, such as Google Scholar [23], and the Caselaw Access Project [2], which exposes an API endpoint for this purpose.

Multiple issues arise with the approaches above, the most crucial of which is their high degree of centralization. A central authority has complete control over the data as well as the ability to restrict access to it. Restrictions may be applied by allowing minimal or no access to data without a subscription, while subscription fees may be freely raised and access to the data may be revoked at any time. Additionally, a centralized service presents a single point of failure that could fall victim to attacks

and censorship. Case in point, access to Wikipedia was blocked in Turkey in 2017 and continues to be blocked to this day [51].

Furthermore, [47] mentions that automatic categorization of case law documents can be challenging due to their complexity. As a result, the larger part of legal research services employ attorneys for the curation and quality control of the massive collection of documents they provide, a procedure which undoubtedly requires expert knowledge and immense effort. This leads to duplication of effort and a scatter of legal information across different sources and inconsistent formats, since no standard metadata structure has been agreed to represent a case law document. Lastly, as the volume of digitally available legal documents increases, the need for a more scalable and robust solution is apparent.

To this day, there is no decentralized, unified, comprehensive, aggregator of case law which is open access and offers opportunities for contribution by anyone. The open, trustless, immutable and publicly verifiable nature of a public blockchain could render it an essential component for the establishment of such a platform.

1.3 Purpose of this Thesis

This thesis aims to design and implement a public and decentralized case law ecosystem where the provision and curation of legal knowledge is a collaborative effort of an entire network of contributors. The ecosystem will attempt to enhance the transparency and ease of access to verified and categorized legal information, as well as eliminate single points of failure by relying on decentralized principles and components.

The regulations that define the ecosystem are permanently stored in a ledger as a set of smart contracts and enforced through the trustless execution offered by the Ethereum blockchain, which constitutes the cornerstone of the platform. The submission of court decisions is open to any user, while the validation and labeling of newly submitted decisions according to their legal subject and juristic value are performed through timed polls, which are open to any and all users of the platform. The influence of each user is measured by parameters such as their financial investment and positive contributions. Appropriate behavior is encouraged through financial incentives, which are formed through reward distribution using a digital token. Autonomy is maintained by employing the collective wisdom for reaching agreement, ensuring that no intervention by a central authority is required for the successful operation of the platform.

Consequently, the adoption of this platform on a national basis could potentially be of great value not only to legal professionals but also to any parties interested in case law research regardless of their legal expertise. Additionally, the establishment of a tokenized economy will directly benefit the platform's contributors. At the time of writing, a thorough search for decentralized applications of such type did not yield any results.

The rest of the thesis is organized as follows. Chapter 2 provides an overview of the background and fundamental technologies addressed throughout the paper. Chapter 3 presents the related work as well as a discussion of projects which could potentially cover the problem domain of this thesis. In chapter 4, a proposed design is presented, analyzed and evaluated. Chapter 5 presents the various components used for the implementation of the platform. Finally, chapter 6 concludes the thesis and mentions a number of recommendations for the potential enhancement of the platform's capabilities.

Chapter 2

Background

2.1 Blockchain

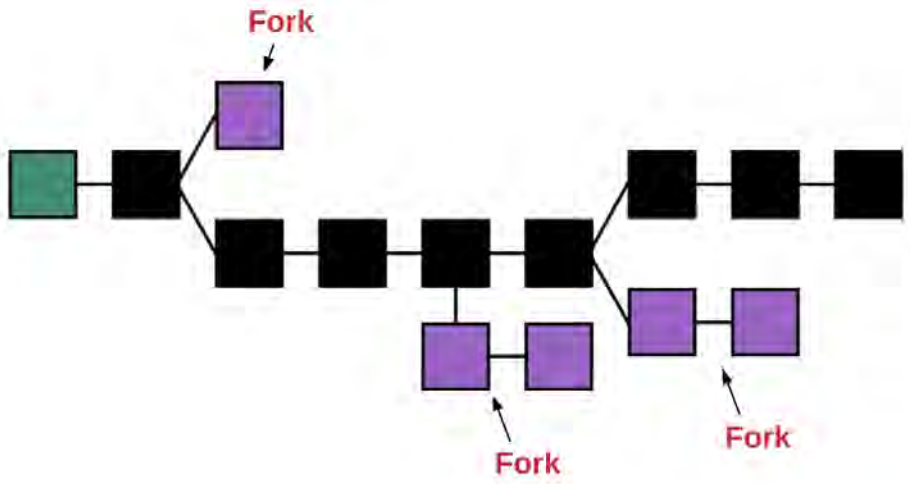
A blockchain is a peer-to-peer (P2P) network which maintains an immutable record of digital transactions. Transactions are signed messages that are propagated within the network and affect the global state of the chain. They are validated and grouped into cryptographically secured *blocks*. The *genesis* block is the first block of the chain and all subsequent blocks build on top of it by storing the hash of the previous block in their headers. This creates a permanent and tamper-resistant data structure.

The global state of the chain is updated with each newly generated block. In a public or *permissionless* blockchain, anyone may participate in the block creation process by connecting to the network with the chain's client software. This software implements the blockchain's specification and allows a peer to synchronize with the network by downloading a complete copy of the blockchain, from the genesis to the most recently generated block. The peer can then begin participating in the validation process.

There are various block validation schemes, the most widely used of which is *Proof-of-Work* (PoW), introduced by Nakamoto in [34]. In a Proof-of-Work blockchain, consensus is reached by competition. Peers participate in a process called *mining*, during which they use their computational power to solve a puzzle, which is usually a resource-intensive hashing process. The puzzle must be difficult to solve but easy to verify, in order to be able to quickly determine the miner's claim. The first miner to provide the proof of its solution is authorized to append the next block to the chain and gains an amount of *cryptocurrency*, the digital currency managed by the network, for doing so. Miners compete to solve the puzzle faster than other miners in order to reap the rewards and cover the financial costs associated with mining. Consequently, the blockchain is economically secured with cryptocurrency, since miners act beneficially to the network out of self-interest.

More than one different blocks may be generated and propagated to the network at the same time by different miners. A *fork* happens whenever more than one available path from the latest block to the genesis block is created. Nodes will attempt to keep building on the longest fork. As more blocks are appended to the existing branches, at some point in time, one of them will become longer than the rest and eventually, the fork will be resolved by nodes abandoning the shorter branches.

A *hard fork* is used deliberately to either perform radical protocol changes in the network, or to reverse the effects of a bug or malicious attack. If the protocol changes are backwards compatible with the old chain, the fork is termed a *soft fork*. Hard forks usually require nodes to update the software they use to connect to the network.



Source: https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

FIGURE 2.1: Example of multiple forks. The green block is the genesis and the black blocks represent the branch that prevails due to its length.

The hard fork between Ethereum and Ethereum Classic happened in order to amend an attack which, due to a contract vulnerability, successfully stole approximately $50 million worth of Ether in 2016 [48]. The network which did not accept the fork was named Ethereum Classic, while the network that accepted the fork by erasing the attacker's transactions from the chain retained the name Ethereum.

The aim of a Proof-of-Work validation scheme is to be resource-intensive in order to prevent an attacker from easily creating new blocks that modify the history of the chain or maintaining a malicious fork. However, if an attacker succeeds in acquiring more than half of the network's *mining power*, they will be able to mine faster than anyone else and create the longest and heaviest chain which will be then accepted by the network (*51% attack*). The more miners participate in the network, the harder it is to perform this attack.

Proof-of-Stake (PoS) is an alternative consensus algorithm which is based on peers staking amounts of cryptocurrency in order to become validators who have the ability to propose new blocks or vote on whether a proposed block is valid. Voting power is proportional to the staked amount, meaning the amount of cryptocurrency locked into a deposit by each validator. The validators are chosen with a round-robin protocol. They gain a reward, based on their stake, if their proposed block is accepted by the majority of validators, while they can lose the entirety of their stake if their proposed block is rejected.

2.2 Ethereum

Ethereum [11] is a Proof-of-Work blockchain which allows the execution of programs called *smart contracts* in an auditable, transparent and trustless environment. The blockchain is used as storage for the global state of the system, while a consensus algorithm allows state synchronization within the peers of the network. Smart contracts are written in high-level programming languages and compiled to bytecode. They can then be executed on the *Ethereum Virtual Machine* (EVM), a fully sandboxed

and isolated runtime environment. It has no access to other processes, the network stack or the file system as it is entirely virtual. This makes code execution completely deterministic.

Every Ethereum node has an instance of the EVM which is used to execute smart contract code triggered by transactions. The EVM has its own instruction set and is able to access block and account information such as addresses and their respective balances, number and timestamp of the latest block.

There are two account types available in the Ethereum blockchain. **Externally owned accounts**, which consist of a public and private key pair, are owned and controlled by individuals. They can use their private key to sign and send transactions to other externally owned accounts or contract accounts, and they can receive Ether through their public key or *address*. Generating a new identity happens locally on a user's machine and has virtually no cost. Thus, users may have one or more accounts at their disposal. **Contract accounts** represent the address through which a smart contract can be invoked. Contract accounts may not send transactions independently, rather, their execution can be triggered by a transaction originating from an externally owned account or another contract account. Both account types have an Ether balance.

Ethereum transactions are cryptographically signed messages from an externally owned account to another account on the chain. They can be used to transfer Ether to another account, invoke contract code or deploy a smart contract to the network.

A cryptocurrency named *ether* (Ξ), which is acquired as a reward for successfully mining a block, is used by the network as a means of paying for computation. Ethereum's smallest currency denomination is called *wei* and it is equal to 10^{-18} Ether.

The average Ethereum block creation rate or *block time* is maintained between the range of 10 and 20 seconds. It may be calibrated by various parameters, such as *block difficulty*, which adjusts how small the *block hash* has to be in order for a block to be considered valid and ultimately determines how much energy has to be expended in order to generate the Proof-of-Work. The smaller the valid hash threshold, the harder it is to find one that satisfies this condition, increasing the time required to complete the mining process. The block difficulty automatically adapts to the network state so that the average block time remains within the expected time. For instance, the block time is evaluated every few blocks, and if it exceeds the expected block time, the difficulty decreases in order to reduce it.

On the one hand, lower block times allow for higher transaction throughput. On the other hand, they lead to a higher probability of multiple miners solving the PoW puzzle at the same time, which causes the creation of orphaned blocks, meaning blocks that are valid but were not included in the main chain.

In Ethereum, these blocks may be appended to the chain at a later time and are called *uncle blocks*, although they do not grant the full block reward to their miner. The main causes of a high uncle rate are network latency due to a large block size and high block rate due to low block difficulty. The reason why they are not discarded is that they increase the security of the network by guaranteeing a reward for valid blocks regardless if they are orphaned or not. Otherwise, validating blocks would only be profitable for mining pools and not for independent miners, which would endanger the decentralization of the network.

2.2.1 Gas

Solidity, the most widely used contract programming language for Ethereum, is Turing-complete. This implies that contract code written in Solidity can be non-terminating, which, in the Ethereum world, can essentially perform a Denial of Service (DoS) attack on the network. This is because all peers that attempt to validate the program will inevitably be running it forever.

For this reason, Ethereum employs *gas*, a mechanism for controlling the computational resources a transaction attempts to consume. Every EVM instruction has a predetermined cost measured in gas units. Thus, gas represents the amount of work a miner needs to do in order to validate a transaction and include it in a block. Gas is purchased with Ether for every transaction, and it is consumed as code execution progresses. Any unused gas is returned to the transaction sender. The sum of all transaction gas fees is sent to the miner of the block as a reward. If the gas sent with the transaction does not cover the cost of the operations needed to complete its execution, the transaction fails with an "Out of Gas" exception, any changes to the state are reverted and the price paid for the gas is lost, since the network already used computational power for the execution. Most wallets, which are responsible for constructing the transaction object, calculate the amount of gas needed to complete the transaction and set the correct amount for the user in the transaction headers.

The cost of gas is not fixed. The transaction sender is able to set the *gas price* of the transaction, which represents the conversion rate of ether to gas. It is thus used to calculate the amount of Ether to be paid for the gas spent during transaction execution. Miners naturally prefer to include transactions with higher gas price in their blocks due to gaining higher rewards for doing so. Thus, setting a higher gas price will result in a faster transaction confirmation time.

Currently, each block has a gas limit of about 10,000,000 gas units. Consequently, the number of transactions that fit in a block depends only on how much gas each transaction requires. Nevertheless, the block gas limit is adjustable. Miners are able to vote to increase or decrease it by a maximum of $\frac{1}{1024}$ of the previous block's gas limit [52].

Increasing the block gas limit leads to increased storage requirements for nodes due to the larger block size. A larger block size also implies that the time required to propagate blocks is increased, which leads to a higher uncle rate. A higher uncle rate will lead to a loss of revenue for miners. To counteract this, miners will avoid creating full blocks in order to ensure that their block gets validated and appended to the chain in time.

2.2.2 Smart Contracts

Unlike Bitcoin and most blockchains which only allow the management of cryptocurrency within the network, Ethereum provides the ability to instantiate and execute programs in a trustless manner. Smart contracts are code that has been stored on the blockchain and can be invoked by anyone who has the ability to send transactions to the network. When a transaction is sent, it is stored in a transaction pool and all peers in the network individually execute the contract code in order to agree on its output.

Smart contracts are written in high-level languages such as Solidity and Vyper, compiled to EVM bytecode and deployed to the network. Once deployed, they acquire their own address through which their functions and storage are accessible. They are permanent and immutable: the only way to update a smart contract is to

redeploy it. During the execution of a transaction, in case of interruption, for instance due to an EVM exception, all the state changes that have been caused by it are reverted. Furthermore, contract code cannot run autonomously. An account has to sign and send a transaction which includes all the input needed to invoke contract code in order to notify the nodes of the Ethereum network to execute it. As mentioned in 2.2.1, this does not come without cost.

All input and output of a contract is public knowledge and can be accessed by the whole network. Any sensitive input will be revealed as soon as the transaction is sent. A *commitment scheme* allows one to temporarily hide a chosen value stored in a public environment and having the ability to reveal it at some point in the future. Commitment schemes exploit the fact that hashing is a one-way operation and are performed in two phases. During the *commit* phase, a user publishes the hash of the desired value, while during the *reveal* phase, they publish the value itself. It is then possible for anyone to verify if the hash of the revealed value is equal to the committed hash. In that case, the user has successfully hidden their value during the commit phase and has proven that they have revealed the correct value. Applications of commitment schemes include secure coin flipping, zero-knowledge proofs and secure computation.

2.2.3 Tokens

The permanent, immutable and trustless nature of the blockchain makes it a suitable option for asset management. Programmable assets can be implemented on top of Ethereum through smart contracts for cases when Ethereum's native cryptocurrency is not sufficient. An asset may be abstracted and implemented on the blockchain as a *token*. Tokens may represent currency, physical or digital resources, voting power, or access rights. There is absolute freedom to establish custom rules of ownership as well as asset management such as transfers, purchases and sales, while maintaining trustlessness and transparency.

Various token standards have been introduced that cover basic token concepts. They define an interface, meaning a set of methods which have to be implemented to cover the specification of the token. ERC20 [17] is the most widely used token standard. It represents ownership of value which can be divisible and interchangeable, much like real-world currency. The ERC721 [18] token standard is used to represent *non-fungible* tokens that denote ownership of a unique, non-divisible asset or collectible, or even track negative occurrences such as loans. Every token is represented by a 256-bit identifier and its ownership can be transfered from address to address.

Token specifications determine only the minimum functionality that needs to be implemented. They can be easily extended by implementing more functions in order to cover more specific needs. Compliance with standards allows tokens to automatically be supported by various exchanges, other smart contracts and applications. Furthermore, it allows popular token specifications to be implemented as libraries, facilitating the deployment of such tokens. Both ERC20 and ERC721 specifications have been implemented as pluggable contract libraries by OpenZeppelin [35] and are publicly audited, "battle-tested" and kept up to date with the latest Solidity changes, which is crucial in such a rapidly evolving ecosystem.

Tokens may be created out of thin air in a process called *minting* and can then be exchanged for Ether through crowdsale contracts. Crowdsales can be programmed to have a certain duration, maximum supply amount or even form bonding curves so that the rates scale with the current supply, encouraging the contribution of early

investors. Furthermore, token sales are commonly used as a fundraising mechanism. An *Initial Coin Offering* (ICO) is often used in order to fund the development of a new project and attract investors, usually after the release of the project's whitepaper.

2.2.4 Oracles

Smart contracts have no access to external data since the EVM in which they are executed is completely isolated from the real world. However, they may often require data from external systems. Use cases include true random number generation, identification and authorization through decentralized identifiers (DIDs), access to online information such as market data, IoT sensor data such as temperature, or any events and triggers that happen off-chain but need to be acknowledged on-chain.

This data can be introduced to a smart contract via an *oracle*, a third-party agent which seeks, verifies and provides external information and events which originate from outside of the blockchain and are used in smart contracts. Usually, a smart contract is created by an oracle which obtains data from an external source and stores the information in its contract storage. Other smart contracts may access this information through the oracle contract. In addition, oracles can be part of multi-signature contracts which require, among others, the signature of an oracle in order to trigger a specific action, such as the release of funds.

Off-chain occurrences cannot be verified by the Ethereum network, thus depending on knowledge originating from oracles constitutes an inevitable security risk. For instance, let us assume the existence of an oracle that generates random numbers for a contract that implements a game of roulette. There is a high incentive to compromise or bribe the oracle if the financial reward for doing so will cover the cost of the attack. For this reason, oracles are, ideally, systems that rely on decentralized principles for increased security and transparency.

2.3 Decentralized Applications

Armed with publicly verifiable, trustless and guaranteed correct program execution provided by the Ethereum blockchain, smart contracts can be used as a building block to develop decentralized applications. These applications, also known as dApps, use smart contracts to apply their business logic and implement a web interface that interacts with the blockchain using the `web3` stack. Optionally, they may also use a decentralized storage service, a decentralized messaging protocol or both. Web3 or "The Third Age of the Internet" [1] is the term used to describe the shift of focus of web applications towards more decentralized protocols, which exploit the autonomy and censorship-resistance provided by blockchain technologies.

The degree of decentralization of an application can vary. Some applications choose to utilize the blockchain only for a small part of their business logic, while others attempt to achieve complete decentralization by not only storing their data in decentralized file systems, but also serving their web interface through them.

2.3.1 Decentralized file services

Since the blockchain is an append-only data structure, storing data on it will eventually cause it to grow insurmountably, increasing synchronization time as well as block propagation time, thus reducing the overall performance of the network. As a result, there is an incentive to limit the amount of data stored inside contracts. This is achieved by applying prohibitively large gas fees for storing data. For reference, the

`ADD` opcode, which performs an addition operation, consumes only 3 gas units, while the `SSTORE` opcode, which saves a word (256 bits) to the contract storage consumes 20000 gas units [52].

Consequently, applications which use large amounts of data are forced to use contract storage in conjunction with an off-chain database to fulfill their storage needs. Contracts cannot access this storage, thus the application needs to sensibly split its data into contract and external storage.

Using any standard, centralized database to store external data is certainly possible. However, this approach suffers from the same issue that the application is trying to solve by using the blockchain: centralization. Public and decentralized file hosting services are able to meet most of the storage needs of a dApp without compromising decentralization and transparency.

The InterPlanetary File System (IPFS) [6] is a public, peer-to-peer global file hosting service. In the IPFS protocol, an uploaded file is split into small chunks of 256Kb which are then hashed and stored on multiple different nodes on the network. The hashes are stored in a distributed hash table (DHT), which is used to retrieve a file whenever it is requested. Storage provided by IPFS is content-addressable, meaning that a file is identified and accessed solely by its hash.

Additionally, the more popular a file is, the faster it can be accessed, since it is replicated on multiple nodes. Conversely, a file that is not regularly accessed is in danger of being garbage collected from the network. At least one node has to be online and *pin* the file in order to guarantee availability. Pinning ensures that the file is always kept locally on the node and is never garbage collected.

Due to the above, IPFS is currently not able to ensure data availability. However, there has been work in creating incentives for monetizing permanent storage. Ethereum Swarm [44], which functions similarly to IPFS, is part of the `web3` stack and works as an incentive layer for sharing permanent storage by rewarding Swarm nodes who offer their storage for use by other clients. Among others, Storj chain [14], Sia [42] and Filecoin [39] offer the same service, but are built on top of their own dedicated blockchain and use their own native currency for reward distribution.

A welcome property of content-addressability is the fact that immutability is guaranteed. Any changes to the file would result in it having a different hash. A commonly used pattern in the development of decentralized applications is to store large data on IPFS, obtain its unique content-addressed hash and permanently store it on the chain.

Chapter 3

Related Work

3.1 Blockchain voting

This section will review numerous protocols and applications that attempt the decentralization of the voting procedure using the blockchain. Emphasis should be given in ensuring privacy of the vote during vote casting in order to eliminate bias.

[53] proposes a voting protocol where the creator of the poll has the capability to grant a user the right to vote by registering their wallet address. The voters have a specific deadline before which they are obligated to declare their vote, and only one vote per user is allowed for each poll.

[13] presents a voting application designed for high-level educational instructions. Every user that can be identified with a valid identification code and email address that belong to the institution has the ability to create polls or participate in them. This application combines Ethereum smart contracts and Paillier Homomorphic Encryption in order to ensure the privacy of the vote.

An attempt to enforce a single vote per user per poll is presented in [25]. It suggests the incorporation of an Ethereum light client in a mobile application and the registration of each voter in the system by making use of the MSISDN (phone number) of the user's SIM card.

It is noted that none of the above solutions are not completely decentralized since they rely on a central authority which the potential voters have to register to. Consequently, the maximum number of voters is known before the start of the poll and voters potentially compromise their anonymity [25, 13] since they are uniquely identified through personal information. Furthermore, in [25, 53] there is no attempt to achieve privacy of the vote. Since submitted votes are visible to anyone on the chain, in order to guarantee confidentiality and non-repudiation, the votes need to be submitted to the contract in a hidden but verifiable format, as shown in [13] or by using a commitment scheme.

3.1.1 Commitment schemes

Commitment schemes are valuable for temporarily hiding data in trustless environments. They are used in a plethora of applications on the blockchain, such as voting, blind auctions and creating randomly generated numbers for multiplayer games.

An example implementation of a commitment scheme is demonstrated in [40], which implements an oracle for random number generation by using the input of multiple participants. Each participant must send the hash of a secret number along with a pledge. After the submission phase is over, users reveal their secret number. In the final phase, the contract computes the random number by combining all valid inputs and returns the pledge to each participant that revealed a number whose hash matches their previously committed hash.

[30] presents a secure and self-tallying voting procedure, meaning that eventually the votes are publicly accessible and anyone may compute the tally of the election. A voter registers for an election by submitting a deposit and a zero knowledge proof for authentication purposes. The voting procedure is then performed using a standard commitment scheme for privacy preservation. An administrator orchestrates the poll by triggering each next phase after all members have participated in the current one. Revealing a valid vote returns the deposit to the user. The deposit may also be returned if one or more voters do not commit their vote before the commit phase ends. After the last vote has been revealed, the administrator calls a contract method which tallies the votes.

The main drawback of this method is that an administrator entity is required to advance through the poll phases, which sacrifices autonomy. Furthermore, it requires knowledge of the number and identities of voters beforehand and it relies on all users cooperating in order to advance to the next phase.

For this thesis, more flexibility is required regarding the number of voters in each poll. New voters may join the platform at any time, and voters may abstain from voting on some polls. More importantly, regarding the Ethereum network, there is virtually no cost for generating new accounts and attempting to impersonate multiple voters. In these circumstances, when voter registration and identification is not an option, the result of the poll should not be computed by tallying individual votes, but by tallying voting power.

3.1.2 Token-weighted voting

Using tokens as a voting right is justified by the assumption that the benefit of the platform is of direct interest to its stakeholders which have made financial investments in it by purchasing and holding tokens. Consequently, the larger their investment, the heavier their vote should be. Variations of token-weighted voting are used by decentralized autonomous organizations (DAOs) to implement self-governance.

An implementation of token-weighted voting is presented in [36]. The votes are concealed through a commitment scheme in order to ensure confidentiality. Users can decide how many tokens they wish to lock in a specific poll and they may lock their tokens in multiple polls simultaneously. The amount of tokens they lock in a poll is used to calculate their weight for the poll in question. They may only retrieve their locked tokens after the end of the voting process, in order to prevent double-voting using the same tokens in different polls.

[32] proposes a blockchain governance method which allows users to vote with a token locking mechanism. In order for a proposal to be approved, first it needs to receive enough approval votes within a certain time window. If this is achieved, a new voting round starts where opponents of the proposal have to counter the approval using the same voting method. This *ping-pong* procedure continues until one side is not able to gather the needed votes to satisfy the threshold, which resolves the issue.

Kleros [26] is a decentralized dispute resolution system. A smart contract on Ethereum may set Kleros as an arbitrator in order to resolve disputes. Token holders register as jurors by locking their tokens in a smart contract. The locked tokens serve as a stake. Jurors are selected randomly, while their chance to be chosen scales with the amount of tokens they lock. Their weight in the vote is proportional to the number of times they are drawn as a juror for the dispute in question and by extension it is proportional to their amount of locked tokens. The voting procedure is performed via a standard commitment scheme to prevent juror bias. Jurors whose

decision agrees with the majority are rewarded with a part of the lost tokens in the stake pool as arbitration fees, relative to their weight.

3.2 Token-curated registries

In order to compensate for the transaction fees associated with storing data on the blockchain, a user has to be incentivized to publish their content. Furthermore, the need for a certain content evaluation system becomes apparent considering that it can be used not only for determining the publisher's reward, but also filtering out low value content. In this section, there will be a brief presentation of decentralized content sharing platforms and the methods they employ to ensure that the conditions mentioned above are met.

Steemit [43] is a decentralized news and blogging service, analogous to Reddit [41], a popular social news aggregation platform. Voting is a fundamental part of the platform. It contributes to publisher's revenue by rewarding posts based on its amount of *upvotes*, while voters are encouraged to curate content by gaining a percentage of the total amount earned by the post. The biggest contributors are also the ones to receive the largest profit. Various techniques are employed to mitigate Sybil attacks. Among them is the use of a special resource named *Steem Power* (SP) which can be acquired by committing an amount of the network's native, tradable coin (STEEM) to a vesting schedule for a period of 13 weeks. This resource is used to calculate a user's *Voting Power*, which essentially represents a vote's influence. Attempting to divide vesting tokens into multiple accounts will also divide their influence and thus leave the net influence intact. Voting Power becomes increasingly weaker and increasingly slower to replenish with every vote. Lastly, all accounts need to maintain a minimum balance in order to be allowed to transact in the platform.

[28] implements a decentralized content sharing platform built on top of the Ethereum Blockchain. Users may interact with the platform by using an inflationary token named *Primas*. For instance, publishers are initially required to register by permanently locking a number of Primas tokens, while publishing content requires a small amount of tokens to be locked for seven days. In addition, if a publisher desires, they may monetize their content by requiring token payments for its reproduction. Plagiarism is punished by token deductions. Users with similar interests may participate in self-governing groups whose members are responsible for evaluating the content published in the group and are subsequently rewarded for their contributions in tokens. Finally, the publishers' revenue is based on the content quality of the submitted post which is in turn measured by factors such as number of likes, reviews, recommendations and reproductions.

[49] presents a decentralized paper publication platform which allows anyone to upload and share their work in a network of researchers. Each publication creates a transaction which stores metadata such as title, keywords, publisher's address, publication transaction hashes of all cited papers and timestamp of the submission. The paper itself is uploaded to IPFS. Authors are charged with a certain amount of currency in order to submit a paper. This amount is allocated to the paper's reviewers and to authors of all cited papers. After a fixed interval of time, the authors gain a reward determined by the score of the paper's reviews. From then on, the paper may only generate revenue every time it is cited. Reviewers of a paper submit a score and optionally a comment which is also stored on IPFS. Comments can be rated by

readers the same way papers are rated by reviewers. Highly rated comments are rewarded.

Lunyr [38] attempted to develop a crowdsourcing based decentralized encyclopedia, similarly to Wikipedia. Contribution to the network may be fulfilled either by submitting or reviewing content. Users that wish to submit are required to first review other submitted articles. They are also obligated to provide sources in order to prove the credibility of the information they are attempting to publish. Before an article can be successfully inserted in the platform, it must be peer-reviewed and approved by multiple users. It implements three different tokens in order to incentivize participation. LUN, a standard ERC20 token, is the reward for contributions and can be used to purchase advertising. Contribution tokens (CBN) are gained through successfully submitting or peer reviewing content and are used to determine the amount of LUN tokens that will be given as a reward to the user at the end of each two-week reward cycle. Finally, Honor (HNR) is used for governance purposes such as creating or voting in polls that concern malicious content which the reviewing system might have mistakenly approved.

3.3 Decentralized data markets

An alternative way of evaluating submitted documents is to monetize them, thereby forming a case law marketplace where document submitters have ownership of their documents and are able to sell them to any interested parties. A considerable amount of inspiration for this concept can be drawn from decentralized data markets. Areas of interest include the methods used for secure storage, monetization and exchange data between two or more parties, proof that the data provided by the seller is valid and the design of a transparent and efficient purchase procedure for the end-user. All these methods should ideally not compromise decentralization.

Datapace [15] is a decentralized application built on a private blockchain and based on the Hyperledger Fabric framework. It aims to create a marketplace for IoT sensor data. Buyers, sellers and validators may participate in the platform. Validators comprise a closed consortium which validates transactions using the PBFT consensus algorithm. Buyers may browse the marketplace for IoT data streams offered by sellers and receive a temporary URL to the data upon purchase. Sellers offer their collected data for sale. Datapace provides specialized hardware for sale in order to ensure that the data source is valid. Sellers who purchase the hardware are flagged as trusted, which increases their visibility in the market. Honest behavior is also incentivized due to their initial financial investment in the network.

[24] allows its users to monetize data collected by telemetry in a secure and decentralized manner. Users are able to set fees for the data they own and submit to the network. The data is anonymized, encrypted and finally stored in BigchainDB [7]. The data release is enacted off-chain with a key exchange between the data provider and the data consumer. Buyers are responsible for validating the data after the purchase and a reputation system is planned to facilitate the procedure.

Chapter 4

Design

4.1 System overview

This thesis designs *AnyCase*, a token-curated registry (TCR) for case law. The fundamental component of the platform is a set of Ethereum smart contracts which implement the core protocol for document submission, validation and taxonomy, as well as a tokenized economy. The platform also consists of a web application which interacts with the Ethereum blockchain and allows participation through the user's web browser.

Documents may be accessed by data consumers, such as legal professionals or casual researchers, without cost. However, participating in the platform either as a submitter or as a curator requires the use and management of *LAW*, the token which functions as the platform's currency and consequently, as an incentive mechanism for positive contribution to the network.

Through the platform, a user is able to upload a document to a public, decentralized and content-addressed file service such as IPFS and then submit its hash on the chain along with a deposit in LAW. Document submission triggers the creation of polls which determine its validity as a court decision, its category of law and juristic importance.

Using a commitment scheme to prevent bias, curators are able to stake LAW and submit their vote on the chain. The curators whose vote was coherent with the majority are rewarded by sharing the staked tokens of the unsuccessful voters. If a document is deemed valid by the majority, the submitter also receives part of the stake pool.

The platform incentivizes users to act as curators without relying on a centralized authority for tallying the votes or directing the voting procedure, essentially leading to the establishment of a public, decentralized and autonomous platform dedicated to case law. Furthermore, it allows both humans and machines to participate in the network. Documents and their metadata are stored in a format which is easy to parse and consume, while access to the results of the polls is publicly available via the contract's storage. Thus, the platform provides labeled data which can be used to train machine learning models. This facilitates and encourages participation by those who wish to use their machine learning model in order to earn tokens.

4.2 Token engineering

4.2.1 LAW

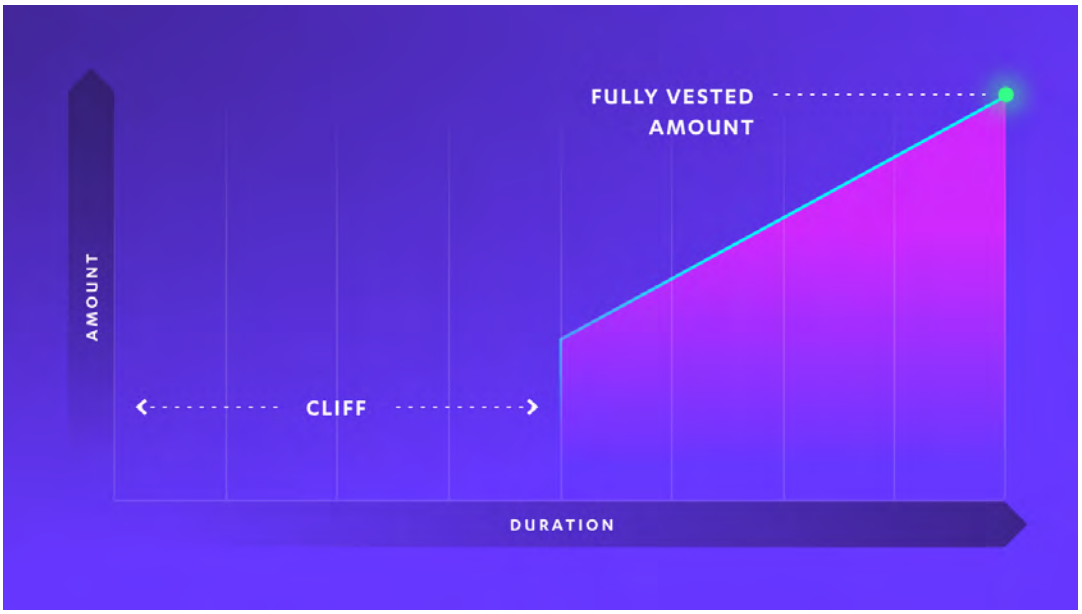
The platform's medium of currency is LAW, an ERC20 compatible token with additional features. The token implements the ERC20 specification which is supported by the majority of exchanges and wallets, while it adds support for the management

of token vesting schedules. Vesting schedules are used to turn LAW into *VLAW*, a vested token whose utility will be analyzed in detail in the next section.

Users may gain LAW by purchasing it from the crowdsale contract in exchange for Ether, by engaging in trading with other token holders or as a reward for contributing positively to the platform after successfully participating in a poll. In addition, actions that involve voting or inserting content in the platform require an amount of LAW to be deposited in the contract as a stake.

4.2.2 VLAW

A vesting schedule is a method of gaining ownership of assets, such as company shares or benefits, usually after an extended period of time. Its main purpose is to encourage loyalty. For instance, an employee may earn rights to retirement benefits based on how long they have been employed in a company. A *graded vesting* schedule allows gradually gaining assets within the schedule's duration, while in *cliff vesting*, the assets are released after the duration of the cliff has passed. During the cliff, the stakeholder has no ownership of the assets.



Source: https://medium.com/cardstack/building-a-token-vesting-contract-b368a954f99

FIGURE 4.1: Example of a token vesting schedule using a combination of cliff vesting and graded vesting.

In AnyCase, vesting schedules allow a user to turn LAW into *Vested LAW* (VLAW) by locking it in a contract for a long period of time. The amount of VLAW gained from a vesting schedule increases linearly with time until the end of the vesting period, when the full amount of VLAW has been acquired. Furthermore, a cliff is put into place to prevent gaining any vested tokens for a small period of time immediately after the start of the schedule. A user may revoke a vesting schedule at any time, but their LAW will be locked for a short period of time before being released.

In the platform, VLAW is an equity token which represents voting power and its main purpose is to add sybil resistance to the platform. Since voting power is equal to a user's vested balance, there is no difference, for instance, between a user having 10 VLAW and 10 users having 1 VLAW each, rendering sybil attacks impossible.

Transferring VLAW means transferring voting power. If users were able to transfer voting power instantly between accounts, double-voting would be an issue. The

vesting cliff prevents this, due to the cliff duration being longer than the duration of the poll's commit phase. This disallows voting more than once in the same poll using the same tokens, since, by the time the vested tokens have been transferred to another account, the voting process of the poll will have come to an end.

Additionally, the release duration should be long enough to prevent a user from intentionally vandalizing the platform and instantly selling their tokens. This ensures that the user has to wait until their actions actually affect the value of the platform and by extension, the token, before being able to sell, encouraging positive behavior. Finally, since VLAW may only be gained through a long-term financial investment, it is a good indication of faith and loyalty to the platform, making it a suitable criterion for determining influence.

4.3 Platform rules

4.3.1 Rulebook

The rules that define the platform need to be stated in detail and stored publicly in order to be easily accessible by all participants. As mentioned in 2.3.1, contract storage is not the solution, as there is a steep cost required to store large amounts of data on-chain. Instead, the rulebook is converted to an easily parsable format such as JSON and uploaded to a public, decentralized and content-addressed file service. The service provides the hash of the file in return, which is then permanently stored on-chain during the deployment of the contract.

While there are multiple such services, which have been enumerated in 2.3.1, from this point on only IPFS will be mentioned as it was also used in the implementation of the platform. Using such a service allows storing data without relying on a centralized storage provider. The rulebook is then able to be accessed and used as a reference for appropriate behavior both by submitters and curators. Furthermore, content-addressability ensures that the rulebook has not been tampered with after its submission, as that would result in its hash being different than the one permanently stored on-chain.

4.3.2 Poll types

This section will analyze the three different types of polls that can be identified in the platform. First, the **validity poll** determines whether the document in question is valid case law or not. It begins immediately following the document's submission. The validity conditions are stated in the rulebook and should be taken into consideration by the document submitter in order to ensure that their document is accepted. If any rules are broken, an honest validator should vote the document down. Examples of possible validity conditions are stated below:

> "The document must be written in formal language and exhibit use of legal vocabulary."
>
> "The document must include a headnote that accurately summarizes the court decision."
>
> "The document's date must be in the format YYYY-MM-DD."

The **classification poll** places the court decision into a category of law. The poll begins accepting votes after the end of the validity poll, on the condition that the

document was deemed valid. Depending on the desired taxonomy, the available categories of may vary. Examples include high-level categories such as "Labour law" and "Property law", or categories specific to the decision content such as "homicide" and "arson". Generally, the number of available categories may range from 20 to 100. The winning category is the one which gathers the majority of votes.

The **importance poll** assesses the document's juristic value. It follows the case law ranking scheme adopted by the European Court of Human Rights [21], in which cases are placed in four categories according to their importance level. The highest ranked cases represent influential case law, while the lowest ranked cases are common and of little legal interest. Similarly to the classification poll, this poll is scheduled to start after the end of the document's validity poll.

4.4 Document submission

Through the web application, a user is able to author and submit a court decision to the platform, following the document validity conditions stated publicly in the rulebook. First, the document and its metadata such as date and decision number, are converted to JSON format and uploaded to IPFS. Next, the document submission to the contract is performed using a *deposit-refund* paradigm. The submitter sends a transaction to the contract which operates as follows:

1. It stores the IPFS hash of the uploaded document and the submitter's address in the contract in a new document entry.
2. It submits an amount of LAW as a deposit. The deposit amount is defined as a constant in the contract storage.
3. It initializes the three polls mentioned in 4.3.2.
4. It openly votes `TRUE` in the newly created validity poll using the submitter's current voting power.

The submitter is mainly interested in the validity poll, since their deposit and reward depend on its verdict. If the document is deemed valid, the deposit is returned and the submitter may collect part of the reward pool.

It is noted that it is impossible to prevent the document submitter from voting for themselves. Attempting to prohibit the submitter's address from voting in the poll would result in all submitters maintaining a second account just to vote for their own documents. The issue with allowing it is that the submitter's vote cannot be ambiguous, as there is no incentive for one to vote against their own document. This is also the reason why the vote is submitted openly, instead of through the commitment scheme. This adds bias to the poll, especially if the submitter is a high-influence user, since this could cause other voters to vote in agreement with the submitter in order to increase their chance of gaining the reward.

4.5 Voting process

Voting is the most fundamental component of the platform. Key elements of the voting process are its commitment scheme and deposit-refund paradigm. This section will analyze the voting process and its transactional flow in detail.

A standard commitment scheme is used during the voting procedure in order to ensure the confidentiality of the vote. Confidentiality prevents bias between voters

and constitutes a necessary property since voters are rewarded based on their vote and the outcome of the poll. Since transaction data in the Ethereum network is public, the vote needs to be concealed locally before sending the transaction. This prevents voters from monitoring contract transactions and copying the voting patterns of the platform's most powerful voters in order to increase their chance of gaining a reward.

The deadlines of the commit and reveal phases of a poll are initialized during its creation. Thus, the contract is able to compare the current block timestamp and the phase deadlines to determine the current phase. No authority is required to manually advance the poll through its phases. In addition, the duration of each phase is long enough to allow the participation of a large network population.

During the **commit phase**, voters send a transaction which performs the following actions:

1. It submits `hash(v, s)` to the contract, where `v` represents the vote as an index that points to the desired candidate of the poll and `s` is a locally and randomly generated value which is used as a salt. The salt is necessary since it adds entropy to the committed hash and prevents other voters from using a rainbow table to expose the vote.

2. It deposits an amount of LAW as a stake. This is a method of insurance against bad actors, since the participants who vote incorrectly are punished by losing their stake.

3. It records the voter's weight at the time of the vote by computing their vested balance (VLAW) based on the current block timestamp.

The deposit amount is small and constant for every user in order to curb the maximum amount of tokens that one can gain from a poll. It prevents wealthy and high-influence users from gaining too many tokens by staking considerable amounts of LAW.

During the **reveal phase**, voters send a transaction which performs the following actions:

1. It submits the vote `v` and salt `s` to the contract and verifies that the revealed values match the committed hash by performing the hash computation on-chain.

2. It refunds part of the voter's stake in order to discourage voters from abstaining from the revealing process.

3. It adds the voter's previously recorded weight to the tally of the voted candidate, thus officially recording the vote.

As the end of the reveal phase approaches, it becomes easier to predict the outcome, with the last voter essentially having knowledge of the outcome before other voters. Consequently, this would lead voters to wait until the last moment to reveal their vote, while voters who predict that their vote is unsuccessful may choose to abstain from revealing in order to avoid paying the transaction's gas cost. Simply put, it would result in voter bias and a skewed poll verdict. Providing compensation for revealing leads to voters being honest and revealing their vote regardless of the final verdict.

After the end of the reveal phase, the vote tally becomes immutable as the poll stops accepting reveal transactions. The winning candidate of the poll can be determined by finding the candidate which has accumulated the most weight during the reveal phase.

Users who voted for the winning candidate may send a transaction in order collect their stake and reward. The total reward pool consists of the staked LAW tokens of users who voted incorrectly, or did not reveal their vote.

Individual reward is proportionate to the amount of the voter's VLAW that was recorded on commit. This is because sharing the reward equally between winners would hurt the protocol's sybil resistance. A user could cheat by voting with multiple accounts and earning an equal amount of tokens in all of them. In total, their share of the reward would be larger than that of honest voters who only use one account.

Consequently, in order to calculate an individual reward, the total reward pool is multiplied by the user's recorded voting power and divided by the total voting power recorded in the poll. This not only prevents sybil attacks by taking into account the user's contribution to the outcome, but it also rewards high-influence voters with a larger share of the reward pool, which is a desirable result.

There is no deadline for collecting poll rewards. Collecting a poll reward finalizes the user's involvement with the poll and concludes the transactional flow of the voting process.

4.6 Analysis

4.6.1 Collective wisdom

The ground-truth regarding the validity, category and importance of a submitted document is unknown and the platform cannot rely on a centralized authority to provide it. Consequently there is a need to produce or at least approximate it by aggregating individual knowledge. The key element of determining the result of each poll is "coherence".

[31] presents consistency with other users as a parameter that effectively evaluates human taggers. Additionally, [12] applies this concept to a blockchain which implements a decentralized data feed. It incentivizes participants to perform the same task but rewards only those whose result agreed with the majority of voters. It relies on the theory that it is harder to coordinate and agree on a specific lie than simply being honest. [37], which implements a decentralized prediction market, also relies on this theory.

In this platform, if the rulebook was removed, the voting game could still be played, but without significant collusion, there would be chaos concerning the results of each poll. The rulebook makes is easier for voters to coordinate on what to agree on in order to get rewarded. If voters assume that the majority of other voters are honest, they themselves will also vote honestly for fear of losing their deposit and reward.

Individual rewards are proportionate to the amount of disagreement, since, the more voters disagree, the higher the reward pool, due to more voters losing their stake. Thus, one could attempt to collude in order to increase their profits by convincing a small amount of voters to vote dishonestly. For instance, the colluder announces to a small number of voters that they will vote for `B` while planning to vote for `A`. The plan is to convince just the right amount of voters which will lead to an increased reward pool but not result in a different winning candidate.

Finally, an advantage of using this method to reward voters is that the rewards are provided by participants who lost. New tokens do not need to be minted to reward winners, instead they are exchanged between the poll participants. Minting new tokens, which, concerning real-world economy, could be considered similar to printing dollar bills, would lead to inflation which could significantly affect the value of the token.

4.6.2 Effects of token value on the protocol

Platform incentives are sensitive to the value of LAW. Due to the rewards scaling with VLAW, the income of a low-influence voter will be significantly lower compared to other voters. When the reward of winning the poll is so low that the total sum of gas costs spent for the participation exceeds the value of the reward converted to Ether, there is no incentive to participate. This may happen when the value of LAW drops excessively. Conversely, when the value of the token increases significantly, voters are reluctant to participate in fear of losing wealth.

These issues derive from the fact that the vote stake is set to a constant amount of LAW, which helps limit the reward pool. An approach to introduce reward flexibility would be to implement self-governance mechanisms in the form of general polls which affect the various parameters of the protocol, such as vote stake.

4.6.3 Effects of token accumulation

A voter or a colluding group of voters who owns more than 50% of the platform's VLAW can single-handedly determine the outcome of a poll. This could easily be the case with token exchanges, which could eventually accumulate a large amount of LAW tokens and vest them in order to gain a significant amount of voting power. This constitutes the main limitation of the presented concept.

It is impossible to limit the voting power threshold per voter in a poll, since they could simply split their VLAW to multiple accounts in order to bypass it. Instead, the long vesting schedule required to gain one's full value of LAW as well as the period where LAW tokens are locked before being released from a schedule, lead to a greatly reduced token liquidity. This prevents a high-influence user from voting and instantly releasing their LAW with the intention to sell.

Assuming that their intention is to attack the platform, their negative contribution would cause the platform to be considered unpredictable and unreliable, therefore decreasing the value of the token. Since the attacker is not able to instantly sell their tokens, their attack would harm their wealth. Consequently, the protocol assumes that the curators with the highest amount of influence in the network are honest and attempt to contribute positively to the network out of self-interest.

In order to mitigate the initial influence of users, a *user score* parameter could be introduced, which starts at 0 and increases with each positive contribution. Again, the way a contribution is counted should scale with a user's VLAW in order to avoid sybil attacks. Otherwise, a person which votes with two accounts earns two contributions instead of one.

The user score could represent a multiplier that is applied on a voter's VLAW with every vote, adjusting their power. In addition, the user score has to increase indefinitely. If a cap on the user score was applied, it could be easily bypassed by creating another account every time the cap is reached on the previous account.

However, the multiplier will eventually reach the threshold where it no longer reduces voting power. This would lead to the user gaining even more voting power,

which will in turn lead to an even higher user score. A VLAW-wealthy user could very quickly increase their user score due to the user score gains scaling with VLAW. Thus, the user score parameter would significantly decelerate the progress of low-influence users, while a high-influence user could bypass the handicap with limited contribution.

4.6.4 Reliability

While crowdsourcing image labeling or sentiment analysis could yield satisfactory results regardless of the voters' technical skills or specialties, the validation and classification of case law demands a extensive knowledge of legal terms. Thus, uncertified voters may not be considered adequate for the task since they cannot guarantee to have the knowledge to curate the submitted content.

Sacrificing decentralization is unavoidable, in present times, in order to gain authenticity and reliability. A semi-decentralized solution could rely on decentralized identifiers (DIDs) and blockchain certifications for legal knowledge provided by institutions in order to authenticate voters. Authenticated legal professionals could be provided with a user profile which starts with a high-influence. There has already been an initiative to store academic records on the blockchain [10, 3, 8].

4.6.5 Exploring the marketplace concept

Authoring and submitting a document requires a user to expend a lot of effort and the reward gained from the voting process might not be enough to justify that effort. The formation of a decentralized market for case law is a possible approach that could infuse the submitted documents with financial value.

The potential approaches to this concept presented in the literature [15, 24, 4] come with consequences to convenience, financial cost and decentralization. Furthermore, the concept faces a larger problem when it comes to preserving the security of the documents, since access to them by curators is necessary for their validation before being made available for purchase.

A method to mitigate the problem would be to require users to purchase the document before they are able to process it. This, however, would lead to users being reluctant about which documents they choose to process, preferring documents in which they have a direct interest, thus reducing the overall participation in the network. As a consequence, the amount of attention a document receives would be directly proportionate to the interest the network shows for it. This would lead to some documents not being processed correctly due to low participation.

Furthermore, piracy and collusion between buyers can also result in a loss of income for the submitter. Buyers may freely resell or share a document after its purchase. Additionally, they could collude by agreeing to share the cost of the document.

The data exchange between the two parties, *buyer* and *seller*, should be performed using a secret channel. However, since off-chain communication and data flow is not verifiable, disputes could arise. The seller could choose to deliberately send invalid data. In that case the buyer would have to initialize a dispute and the seller would have to prove their innocence. Furthermore, the buyer could receive the correct data but attempt to grieve the seller by falsely accusing them. Dispute resolution would then require the document to be revealed to the arbitrators in order to reach a decision.

In conclusion, the lack of verifiability of intra-user transactions, uncertainty concerning the real content of a document due to it being hidden before purchase, reliance on a reputation system to ease this uncertainty, constitute limitations which result in a complex and inefficient marketplace.

Chapter 5

Implementation

This section presents a proof-of-concept implementation of the protocol described in chapter 4. At the time of writing, the source code and documentation of both the smart contracts and the web interface are stored in a private GitHub repository which will be made public shortly after the publication of this thesis.

5.1 Smart contracts

Three smart contracts were written in Ethereum's Solidity smart contract programming language. The Truffle framework [45] was used during development in order to generate the project template, as well as to provide tools for project management such as building, unit testing and deploying the contracts to both private and public test networks. During early development, Ganache [22], a personal Ethereum blockchain simulator, was used to launch a local, private blockchain for testing contract interactions and monitoring transactions.

5.1.1 Main contract

The main contract stores most of the vital application data and implements the document submission and voting protocol. The main data structures used throughout the contract are `Document` and `Poll`.

The `Document` struct stores only the submitter address and IPFS hash of the uploaded document. Documents are stored in a global `documents` array, as the web application might need to access them sequentially, in order, for instance, to display the 10 most recent documents.

The `Poll` struct stores its poll type, the index of the document it is associated with, deadlines for the commit and reveal phases, and a helper `VoteData` struct which includes vote information such as committed hash, vote weight and revealed vote for every user. Each piece of information is stored in a `mapping`, a key-value table which maps addresses to information. The reason why a helper struct was used is merely due to a limitation in the Solidity language, which does not allow returning arrays of structs which contain `mapping` fields. Wrapping all the mapping fields in a separate struct bypasses this limitation.

Additionally, it is noted that, concerning the contract, the sole functional difference between the poll types is their number of candidates. Thus, the contract requires only the number of available candidates of each poll type to be recorded in its storage. This allows the contract to initialize the vote storage of each poll, represent each candidate as an index which corresponds to the desired candidate in the candidate array and finally, to verify whether a submitted vote is valid or not during the reveal phase. The contract is agnostic of the meaning and verbose description of

each candidate. These may be stored in the rulebook, which can be easily fetched from IPFS and displayed in the web application.

Similarly to `documents`, a global `polls` array is initialized. New documents and polls created by `addDocument()` are appended to their respective arrays. Thus, knowledge of the poll index and voter address allows $O(1)$ access to a voter's data for the poll in question.

The contract data could be linked in more complex ways in order to allow the contract to answer queries such as "documents which were uploaded by user 0x1234 between April and June", however that would increase storage needs and by extension, gas costs for the user. Generally, it is adequate to use arrays of `structs` for data that needs to be accessed sequentially and `mappings` for data that needs to be accessed by an arbitrary key. To answer more complex queries, an application which indexes and aggregates contract events is required, such as TheGraph [46].

The commitment scheme is implemented as described in 4.5. The functions `commitVote()` and `revealVote()` are used during the commitment scheme. The modifiers `canCommit` and `canReveal` are applied to the aforementioned functions in order to test if the conditions for a valid commit or reveal are satisfied. If that is not the case, the transaction is reverted. In addition, the modifier `withTokenDeposit()` is to applied to functions `addDocument()` and `commitVote()` since they require a user to stake tokens to proceed. Finally, `collectReward()` determines whether a user is eligible for collecting a reward for a specific poll, calculates the reward based on the method described in 4.5 and transfers the tokens to the transaction sender.

5.1.2 Token and crowdsale contracts

OpenZeppelin [35] provides contract templates, math libraries and token implementations for common token specifications. This work was based on the implementation of OpenZeppelin's ERC20 token. The full token name is "LAW Coin" and its symbol is "LAW". It is represented with 18 decimals in order to mimic the relationship between Ether and wei.

The token ties itself to the platform by allowing the address of the main contract to make transfers to itself through an additional helper function, `dappTransfer()`. Due to the specification of ERC20 tokens, in order for a contract to transfer tokens on behalf of a user, two transactions are required: an `approve()` transaction and then a `transfer()`. Thus, with `dappTransfer()`, the token contract is able to authorize the main contract to make transfers to itself on behalf of users using only one transaction.

The creation of a vesting schedule essentially determines the parameters that define the line seen in figure 4.1. These parameters are stored in a `VestingSchedule` data structure and each user is mapped to an array of such structures. In order to calculate the VLAW balance gained so far from a vesting schedule, we need to determine the value of the line function, given the current block timestamp. Consequently, the total vested balance of a user is the sum of their vesting schedule balances.

A loop is needed to compute the sum of balances. However, loops are not ideal due to gas costs. If the vesting schedule duration is short enough, it is possible to achieve lower gas consumption during transactions that need to compute a user's vested balance by occasionally restructuring the vesting schedules array in order to decrease its size. Completed vested schedules no longer need to be computed. During restructuring, it is possible to delete them from the array and keep their balance in a separate variable. Deletion is achieved by swapping the element to be

deleted with the last element and then decreasing the length of the array. This causes an amount of gas to be refunded to the transaction sender, since the transaction deletes data from a contract's storage.

Finally, the crowdsale contract was also based on the crowdsale implementation offered by OpenZeppelin. A standard non-timed crowdsale with a maximum total supply provided and a stable conversion rate of Ether to tokens was used and no additional business logic was implemented.

5.1.3 Time-dependent contract logic

The commitment scheme which is essential for ensuring a privacy-preserving voting procedure requires time-dependent logic in order to function successfully. The same applies to the token vesting schedule functionality implemented in the token contract. In Solidity, special global variables such as `block.timestamp` and `block.number` are made available for this purpose.

Relying on the timestamp of the block is a possible solution. However, its accuracy is unreliable as the miner is able to adjust it to some degree and is expected to do so if there is a large enough profit to be gained by this action. As a more secure solution, [1] recommends to use the block number in conjunction with the average block time of the chain in order to estimate time, since the block number is more difficult for a miner to manipulate.

The Ethereum network attempts to keep the block time at 15 seconds on average [20]. The block difficulty is adjusted accordingly to achieve this. However, the transition of Ethereum to a Proof-of-Stake network could decrease the block time, potentially rendering any estimations inaccurate. Furthermore, in this contract, no critical decision is made by relying on the block timestamp. Plenty of time is provided for all time-sensitive actions of the contract. Additionally, regarding vested balance which is also time-dependent, a deviation of several seconds would be negligible when considering the months or years required to complete a vesting schedule. Due to the aforementioned points, we opt for the timestamp method in order to enforce time constraints.

5.2 Web application

The web application provides users with an intuitive interface to the deployed contracts and their methods. Through the frontend of the application, it is possible to manage their tokens and vesting schedules, as well as participate in polls and submit documents. The frontend was developed as a Single-Page Application (SPA) using the ReactJS framework, meaning that all routing happens on the client side. Due to this fact, the frontend of the application can be hosted on IPFS, achieving, as a result, a higher degree of decentralization and potentially accelerated access as its popularity increases.

The backbone of the application is `web3.js`, a JavaScript library which allows interaction with an Ethereum provider through a HTTP or IPC connection. The provider may be either a local node running with `geth` or a remote node offered by various services such as Infura [19].

The MetaMask browser extension, a non-custodial wallet and Ethereum light-node, is used for interacting with the applications by signing and sending the user's transactions to the network. The application is tightly integrated with MetaMask as it is the least intrusive wallet solution for a new user and can be used with all major web browsers.

The user may navigate through the following pages:

1. The **upload** page, which allows a user to submit a document for validation. It provides forms for various metadata such as court decision number and headnote, which is a summary of the case separated from the document body. Documents are submitted to IPFS in JSON format, which is easy to parse and is widely used by most APIs. Finally, the `addDocument()` contract method is called in order to perform the actions mentioned in 4.4.

2. The **search** page, which allows a user to submit queries and browse through submitted documents. The search results display the headnote of the case as well as metadata such as category and importance level, if the respective document polls have ended. Rendering a selected document involves fetching it directly from IPFS and parsing the returned JSON.

3. The **contribute** page, which allows a user to participate in any of the three polls mentioned in 4.3.2. Upon choosing the desired poll type, all the open polls of such type are fetched and rendered as a list. Clicking on a poll fetches its data from the contract, including the document it is associated with. If the user is able to vote for the poll, the document is fetched directly from IPFS and displayed along with the poll's question. For instance, in the case of a validity poll, the validity conditions are displayed alongside the document and the user is instructed to tick them off individually. A cryptographically secure, randomly generated salt is created with `web3.utils.randomHex()` and finally, `commitVote()` is called. The vote and salt are stored in the user's `localStorage` on transaction confirmation in order to be revealed later on.

4. The **dashboard** page, which allows a user to manage their LAW and VLAW as well as post-commit voting procedures. It fetches the pending polls in which a user has committed their vote and, depending on their state, displays information about their reveal deadline and whether or not the user is eligible to collect their reward. Furthermore, a mechanism for exporting and importing the committed data is available. Its main use case is to back up locally stored vote data and import it to different browsers or computers.

5. The **help** page, which provides information about the platform in a Frequently Asked Questions (FAQ) format.

5.3 Storage

5.3.1 IPFS limitations

The main disadvantage of content-addressed file services is their lack of support for sequential storage. Thus, files need to be downloaded individually in order to be processed in a meaningful way. Aggregation of data is not possible natively, thus, this functionality has to be implemented as a separate application on top of the file service.

In order to accommodate the needs of data consumers, it is necessary to maintain a server or server cluster that aggregates and indexes documents and accepts full-text search queries from users. This should constitute the only centralized component of the platform. It is noted that this functionality is only necessary to cover the needs of data consumers, who do not participate in the curation process, but are only interested in browsing through documents.

Additionally, IPFS cannot guarantee data availability, as non popular data is in danger of being garbage collected from the network. One or multiple IPFS nodes which pin newly added documents are required to ensure high availability and access speed for all documents.

5.3.2 Anonymization

While it has been noted that anonymization can make legal research harder [5, 47], countries which have adopted strict data protection laws are making efforts in order to remove identifying information such as names of physical persons occurring in the bodies of court decisions.

Consequently, depending on the platform's country of operation and the data protection and privacy regulations it is required to adhere to, document submitters might be forced to anonymize the data they provide. In such cases, anonymization should be added to the platform's document validity conditions. Furthermore, a method of de-personalization should be employed in order to preserve both the privacy of the involved parties and the usefulness of the data.

In order to protect the privacy of the involved parties, de-identification could be achieved by replacing victim and defendant names with initials such as `V1`, `V2` and `D1`, `D2` respectively. Genders of physical persons would have to be preserved as they may constitute a crucial parameter of the verdict, such as in cases concerning divorce and custody.

Chapter 6

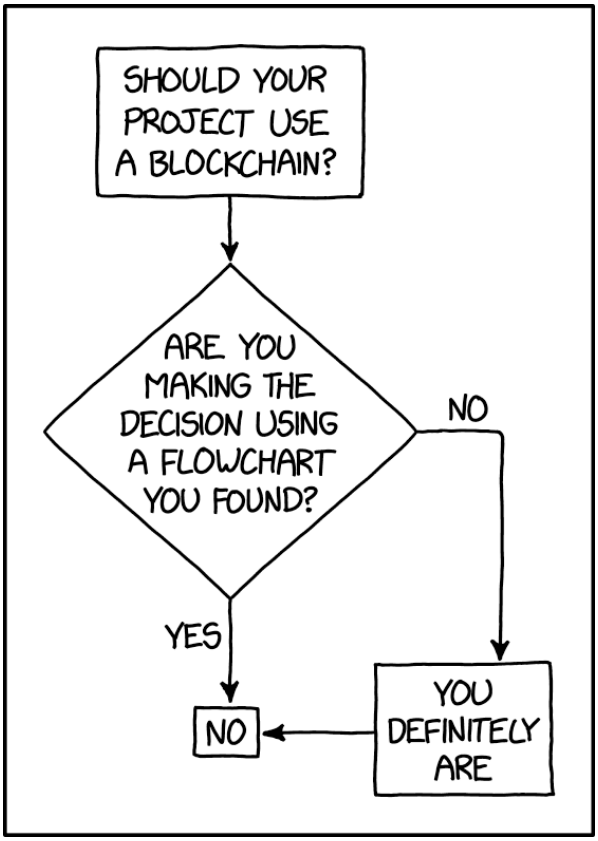
Synopsis and Prospects

This thesis has introduced AnyCase, a decentralized case law curation ecosystem powered by intrinsic financial incentives which generate wealth for its benefactors. The voting protocol presented herein is completely decentralized and autonomous since it requires no tallying authority or administrator entity to operate. The voting process itself is privacy-preserving and publicly verifiable, while it allows voter abstinence in all phases and does not compute the tally based on the number of individual voters, rendering it completely sybil-resistant. Lastly, no secret channels between participants are necessary during the voting process. Communication is only performed between the user and the smart contract through transactions.

Nevertheless, the presented concept has room for improvement. Due to the expertise needed to perform the tasks identified in the platform, it is necessary to raise the average skill level of participants and thus the reliability of the platform as a knowledge base. A semi-decentralized platform which identifies and authorizes skilled curators might result in higher quality validation and classification. Unfortunately, identity on the blockchain is one of the hardest problems to solve without sacrificing decentralization.

A more in depth financial analysis would be required to determine the solidity of the market this platform creates for the LAW token. Furthermore, self-governance mechanisms should be researched to increase autonomy and resistance to factors such as the tokens's market value and the value of Ether, which, for the time being, directly affect the incentives of protocol.

Close collaboration with legal professionals is crucial in order to identify the key elements which assist legal research through an online case law database. It is also vital to determine the pain points of user experience in order to design a practical and intuitive user interface for audiences unfamiliar with cryptocurrency and decentralized applications. Finally, it would be interesting to investigate the feasibility of using the platform as an oracle for a next-generation decentralized court.



Source: https://xkcd.com/2267/

FIGURE 6.1: *"Blockchains are like grappling hooks, in that it's extremely cool when you encounter a problem for which they're the right solution, but it happens way too rarely in real life."*

Bibliography

[1] A Antonopoulos and G Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. 2018. ISBN: 9781491971949. URL: https://github.com/ethereumbook/ethereumbook.

[2] *API Documentation | Caselaw Access Project*. URL: https://case.law/api/ (visited on 12/30/2019).

[3] Augusta Hightech Soft Solutions. *UAE University launches the blockchain records app to manage academic records*. 2019. URL: https://hackernoon.com/uae-university-launches-the-blockchain-records-app-to-manage-academic-records-2ff8ab7804a0 (visited on 02/25/2020).

[4] Prabal Banerjee and Sushmita Ruj. "Blockchain Enabled Data Marketplace - Design and Challenges". In: (2019). arXiv: 1811.11462. URL: http://arxiv.org/abs/1811.11462.

[5] Barnard, Catherine and Peers, Steve. *EU Law Analysis: Anonymity in CJEU cases: privacy at the expense of transparency?* 2018. URL: http://eulawanalysis.blogspot.com/2018/12/anonymity-in-cjeu-cases-privacy-at.html (visited on 02/10/2020).

[6] Juan Benet. "IPFS - Content Addressed, Versioned, P2P File System". In: *CoRR* abs/1407.3561 (2014). arXiv: 1407.3561. URL: http://arxiv.org/abs/1407.3561.

[7] *BigchainDB - The blockchain database*. URL: https://www.bigchaindb.com/ (visited on 09/09/2019).

[8] *Blockchain diplomas land in Virginia at ECPI | Higher Education*. 2018. URL: https://www.cryptoscoop.io/blockchain-diplomas-land-in-virginia-at-ecpi-higher-education/ (visited on 02/25/2020).

[9] *Bloomberg*. URL: https://www.bloomberg.com/company/ (visited on 06/03/2019).

[10] Busta, Hallie. *More colleges are using the blockchain for student records*. 2018. URL: https://www.educationdive.com/news/more-colleges-are-using-the-blockchain-for-student-records/542093/ (visited on 02/25/2020).

[11] Vitalik Buterin. "A Next-Generation Smart Contract and Decentralized Application Platform". In: January (2013). URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[12] Vitalik Buterin. *SchellingCoin: A Minimal-Trust Universal Data Feed*. URL: https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/ (visited on 02/23/2020).

[13] Gaby G. Dagher et al. "BroncoVote: Secure Voting System using Ethereum's Blockchain". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*. Icissp. 2018, pp. 96–107. ISBN: 9789897582820. DOI: 10.5220/0006609700960107.

[14] *Decentralized Cloud Storage - Storj*. URL: https://storj.io/ (visited on 08/28/2019).

[15] Drasko Draskovic and George Saleh. "Datapace - Decentralized Data Marketplace Based on Blockchain". In: (2017), pp. 1–16. URL: https://www.datapace.io/datapace{_}whitepaper.pdf.

[16] *Eigen Technologies: Unlock Your Data With Natural Language Processing*. URL: https://www.eigentech.com/ (visited on 05/28/2019).

[17] *EIP 20: ERC-20 Token Standard*. URL: https://eips.ethereum.org/EIPS/eip-20 (visited on 02/15/2020).

[18] *EIP 721: ERC-721 Non-Fungible Token Standard*. URL: https://eips.ethereum.org/EIPS/eip-721 (visited on 02/15/2020).

[19] *Ethereum API | IPFS API & gateway | ETH nodes as a service | Infura*. URL: https://infura.io/ (visited on 02/25/2020).

[20] *Ethereum Average Block Time Chart*. URL: https://etherscan.io/chart/blocktime (visited on 09/03/2019).

[21] *Explanation of Importance Level & Key cases*. URL: https://www.echr.coe.int/Documents/Explanation_Importance_Level_ENG.pdf (visited on 01/01/2020).

[22] *Ganache | Truffle Suite*. URL: https://www.trufflesuite.com/ganache (visited on 02/25/2020).

[23] *Google Scholar*. URL: https://scholar.google.com/ (visited on 12/30/2019).

[24] Roger Haenni. *Datum Network - The decentralized data marketplace - Whitepaper V15*. 2017. URL: https://datum.org/assets/Datum-WhitePaper.pdf.

[25] David Khoury et al. "Decentralized Voting Platform Based on Ethereum Blockchain". In: *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*. IEEE, Nov. 2018, pp. 1–6. ISBN: 978-1-5386-4500-0. DOI: 10.1109/IMCET.2018.8603050. URL: https://ieeexplore.ieee.org/document/8603050/.

[26] Lesaege, Clément and Ast, Federico and George, William. *Kleros - Short Paper v1.0.7*. 2019. URL: https://kleros.io/assets/whitepaper.pdf.

[27] *LexisNexis® Case Law | Comprehensive collection of case law*. URL: https://www.lexisnexis.com/en-us/products/caselaw.page (visited on 05/08/2019).

[28] Primas Lab Foundation Ltd. *Primas - Restoring Health to the Internet*. 2018. URL: https://primas.io/whitepaper/en (visited on 12/11/2019).

[29] *Luminance | Technology*. URL: https://www.luminance.com/technology.html (visited on 05/08/2019).

[30] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. "A Smart Contract for Boardroom Voting with Maximum Voter Privacy". In: *Financial Cryptography and Data Security*. Ed. by Aggelos Kiayias. Cham: Springer International Publishing, 2017, pp. 357–375. ISBN: 978-3-319-70972-7.

[31] Olena Medelyan, Eibe Frank, and Ian H. Witten. "Human-competitive tagging using automatic keyphrase extraction". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing Volume 3 - EMNLP '09*. Vol. 3. Morristown, NJ, USA: Association for Computational Linguistics, 2009, p. 1318. ISBN: 9781932432633. DOI: 10.3115/1699648.1699678. URL: http://portal.acm.org/citation.cfm?doid=1699648.1699678.

[32] Paul Merrill et al. "Ping-Pong Governance: Token Locking for Enabling Blockchain Self-governance". In: *Mathematical Research for Blockchain Economy*. Ed. by Panos Pardalos et al. Cham: Springer International Publishing, 2020, pp. 13–29. ISBN: 978-3-030-37110-4.

[33] Arthur S. Miller. "Statutory Language and the Purposive Use of Ambiguity". In: *Virginia Law Review* 42.1 (1956), pp. 23–39. ISSN: 00426601. URL: http://www.jstor.org/stable/1070047.

[34] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: October 2008 (2008), p. 9. ISSN: 09254560. DOI: 10.1007/s10838-008-9062-0. arXiv: 43543534534v343453. URL: https://bitcoin.org/bitcoin.pdf.

[35] *OpenZeppelin: Contracts*. URL: https://openzeppelin.com/contracts (visited on 02/14/2020).

[36] *Partial Lock Commit Reveal Voting System that utilizes ERC20 Tokens*. URL: https://github.com/ConsenSys/PLCRVoting (visited on 02/19/2020).

[37] Jack Peterson et al. "Augur: a decentralized oracle and prediction market platform". In: *arXiv preprint arXiv:1501.01042* (2015).

[38] Arnold Pham and Andrew Tran. *Lunyr - A decentralized world knowledge base on Ethereum driven by economic incentives*. 2017. URL: https://whitepaper.io/document/222/lunyr-whitepaper (visited on 12/30/2019).

[39] Protocol Labs. *Filecoin: A Decentralized Storage Network*. 2017. URL: https://filecoin.io/filecoin.pdf.

[40] randao.org. *Randao: Verifiable Random Number Generation*. URL: https://www.randao.org/whitepaper/Randao_v0.85_en.pdf (visited on 02/19/2020).

[41] *reddit: the front page of the internet*. URL: https://www.reddit.com/ (visited on 12/21/2019).

[42] *Sia*. URL: https://sia.tech/ (visited on 08/28/2019).

[43] Steemit. *Steem: An incentivized, blockchain-based, public content platform*. 2017. URL: https://steem.com/SteemWhitePaper.pdf (visited on 12/19/2019).

[44] *Swarm - Storage and Communication for a Sovereign Digital Society*. URL: https://swarm.ethereum.org/ (visited on 08/28/2019).

[45] *Sweet Tools for Smart Contracts | Truffle Suite*. URL: https://www.trufflesuite.com/ (visited on 02/25/2020).

[46] *The Graph - A Query Protocol for Blockchains*. URL: https://thegraph.com/ (visited on 02/21/2020).

[47] Marc Van Opijnen and Cristiana Santos. "On the Concept of Relevance in Legal Information Retrieval". In: *Artif. Intell. Law* 25.1 (Mar. 2017), 65–87. ISSN: 0924-8463. DOI: 10.1007/s10506-017-9195-8. URL: https://doi.org/10.1007/s10506-017-9195-8.

[48] Vessenes, Peter. *Deconstructing theDAO Attack: A Brief Code Tour*. 2016. URL: https://vessenes.com/deconstructing-thedao-attack-a-brief-code-tour/ (visited on 02/10/2020).

[49] Taotao Wang, Soung Chang Liew, and Shengli Zhang. "PubChain: A Decentralized Open-Access Publication Platform with Participants Incentivized by Blockchain Technology". In: *ArXiv* abs/1910.00580 (2019).

[50] *Westlaw UK | UK Legal Solutions | Thomson Reuters*. URL: https://legalsolutions.thomsonreuters.co.uk/en/products-services/westlaw-uk.html (visited on 05/08/2019).

[51] *Wikipedia Blocked in Turkey*. URL: https://turkeyblocks.org/2017/04/29/wikipedia-blocked-turkey/ (visited on 12/31/2019).

[52] Wood, Gavin. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. 2019. URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[53] Emre Yavuz et al. "Towards secure e-voting using ethereum blockchain". In: *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*. IEEE, Mar. 2018, pp. 1–7. ISBN: 978-1-5386-3449-3. DOI: 10.1109/ISDFS.2018.8355340. URL: https://ieeexplore.ieee.org/document/8355340/.