

Witnet: A Decentralized Oracle Network Protocol

[Working Draft // Early Request for Comments]*

ADÁN SÁNCHEZ DE PEDRO
STAMPERY, CTO
adan@stampery.com

DANIELE LEVI
STAMPERY, CEO
daniele@stampery.com

LUIS IVÁN CUENDE
ARAGON, PROJECT LEAD
luis@aragon.one

Version 0.1

November 27, 2017

Abstract

Witnet is a decentralized oracle network (DON) that connects smart contracts to the outer world. Generally speaking, it allows any piece of software to retrieve the contents published at any web address at a certain point in time, with complete and verifiable proof of its integrity and without blindly trusting any third party.

Witnet runs on a blockchain with a native protocol token (called Wit), which miners—called *witnesses*—earn by retrieving, attesting and delivering web contents for clients. On the other hand, clients spend Wit to pay witnesses for their Retrieve-Attest-Deliver (RAD) work. Witnesses also compete to mine blocks with considerable rewards, but Witnet mining power is proportional to their previous performance in terms of honesty and trustworthiness—this is, their reputation as witnesses. This creates a powerful incentive for witnesses to do their work honestly, protect their reputation and not to deceive the network.

The Witnet protocol is designed to assign the RAD tasks to witnesses in a way that mitigates most attack vectors to the greatest extent. At the same time, it includes a novel 'sharding' feature that (1) guarantees the efficiency and scalability of the network, (2) keeps the price of RAD tasks within reasonable bounds and (3) gives clients the freedom to adjust certainty and price by letting them choose how many witnesses will work on their RAD tasks.

When coupled with a Decentralized Storage Network (DSN), Witnet also gives us the possibility to build the Digital Knowledge Ark: a decentralized, immutable, censorship-resistant and eternal archive of humanity's most relevant digital data. A truth vault aimed to ensure that knowledge will remain democratic and verifiable forever and to prevent history from being written by the victors.

***Note:** Witnet is a work in progress. Active research is under way, and new versions of this paper will appear at <https://witnet.io>. For comments and suggestions, contact us at research@witnet.io.

Contents

Contents	2
List of Figures	4
1. Introduction	5
1.1 Motivation	5
1.2 Solution Overview	6
1.3 Key Components	7
2. Definition of a Decentralized Oracle Network	8
2.1 Retrieve-Attest-Deliver (RAD) Requests and Tasks	8
3. Witnet as a Decentralized Oracle Network	10
3.1 Setting	12
3.1.1 Participants	12
3.1.2 The Network, \mathcal{N}	13
3.1.3 The Ledger, \mathcal{L}	14
3.1.4 The Coin, Wit	15
3.1.5 The Headless Browser, \mathcal{B}	16
3.2 Retrieval and Attestation Capabilities	17
3.2.1 General Case	17
3.2.2 Specific Data Units Retrieval	19
3.2.3 Multiple Sources Fact Cross-checking	19
3.2.4 Future Facts	20
3.3 Retrieval and Attestation Limitations	21
3.3.1 Undecidability	21
3.3.2 Unverifiability	22
3.4 The Protocol	23
3.4.1 <i>Client</i> Cycle	23
3.4.2 <i>Witness</i> Cycle	24
3.4.3 <i>bridge</i> Cycle	25
3.4.4 <i>Network</i> Cycle	26
4. Reputation	28
4.1 Reputation Protocol	28
4.2 Truth-By-Consensus and SVD	31

5. Useful Work Consensus Algorithm	33
5.1 Reputation-Based Mining Protocol	34
5.2 Reputation-Based Task Assignment Protocol	36
6. Transactions	38
6.1 Transaction Properties	38
6.2 Types of Transactions and Standard Outputs	39
6.3 Fees	39
6.3.1 Miner fees	39
6.3.2 Witness fees	40
6.3.3 Bridge fees	41
6.4 Outputs and Scripting	41
6.4.1 MAST and Tail Call Execution	42
6.4.2 Covenants	42
6.4.3 Bundled Macros	43
7. Bridges and Smart Contracts	45
7.1 Bridges: Interacting with other Platforms	45
7.1.1 Ethereum Bridges	45
7.1.2 Decentralized Storage Networks (DSN) Bridges	46
7.1.3 Web Bridges	46
7.2 Native Smart Contracts	47
Appendices	48
A. The Digital Knowledge Ark	48
A.a Motivation	48
A.b Knowledge Commons and Commons-based Peer Production	49
A.c Preservation and its Principles	51
A.d Using Witnet to Agree on Facts to be Preserved	51
A.e Persisting Facts and Contents in Perpetuity	52
References	54

List of Figures

1	Network Participants Subsets	13
2	Checkpoints and Epochs	14
3	Directed Acyclic Graph	14
4	Controlled Supply Generation Algorithm	16
5	Coin Issuance Rate	16
6	Total Coin Supply	16
7	Truth-By-Consensus	18
8	Retrieval Paths	19
9	Content Retrieval Flow	20
10	Reputation Demurrage Function	29
11	Reputation Demurrage Simulation	30
12	Miner Influence Calculation	33
13	Block Mining Calculation	34
14	Backup Block Mining Calculation	35
15	Task Assignment Calculation	36
16	Witness Fees Calculation	40
17	M_REQUEST Macro	43
18	Witnet Smart Contract Example	47

1. INTRODUCTION

1.1. Motivation

OVER the last years, Bitcoin, Ethereum and other blockchain networks have proven the undeniable utility of decentralized transaction ledgers. These public ledgers process sophisticated smart contract applications that give digital money the capability to be programmed and incorporate logic. Those smart contracts, in their various forms, have the potential to allow for a more decentralized economy where individuals and companies worldwide can transact freely and safely without the need of intermediaries or trusted third parties.

However, current smart contract solutions are self-contained in their supporting blockchains and have very little to no capability to interact with other blockchains, the Internet and the rest of the world. They were built deliberately detached from the outer world for a good reason: they need to be deterministic¹, while events in the real world are highly undeterministic if not completely random.

The most widely known way to feed outer information into smart contracts is using an oracle. An oracle is a trusted entity which signs claims about the state of the world. As signatures can be verified deterministically, oracles allow smart contracts to react to events happening in the outside world. But given that this approach puts trust in a single attesting third party (the oracle), it can not be considered trustless² nor tamper-resistant and therefore it leaves space to contestation and repudiation. It is not that the honesty and transparency of anyone is called into doubt. The problem here is that such a single source of truth also represents a single point of failure that introduces the chance for an external malicious actor to rewrite or delete facts by breaking into a single system or network.

There is no way for smart contracts to deliver in their economic decentralization promises until we build some kind of decentralized oracle that does not rely on blind trust but on some digital equivalent of *the wisdom of the crowd*.

¹In computer science, a deterministic algorithm is the one which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.

²In this work, like in most blockchain literature so far, the term *trustless* has a very different meaning than the one in the dictionary. Instead of "*not worthy of trust*", it means "*that operates without relying on blind trust in third parties*".

1.2. Solution Overview

UNLIKE the oracles that have traditionally been described, the one we are proposing here is not based on trust. Its claims can be considered to be true not because of the authority of the proposed network as a whole or that of any of its members in particular; but because the claims themselves are built by comparing and combining a number of likely conflicting claims coming from a plurality of anonymous players. Those players (the miners) have divergent interests but are strongly incentivized to tell the truth, which guarantees that they will not collude to deceive the system or feed any false claim into it.

The proposed network can also be considered to work in a similar way to a decentralized prediction market³ in the sense that the outcome of the attestations comes from the tally of the claims that a plurality of volunteer peers who are unknown to each other (the miners) have "voted for".

Of course, the so-called miners are not actual human beings sitting in front of a computer, fulfilling assignments coming from an Internet overlord that commands them to use their web browser to navigate to a certain website and take a snapshot or copy some text that they must report. Instead, the miners are computers running a software that automatically receive and execute a series of tasks without the owner of the computer having to actively do anything else than install it and configure how much of the available CPU and bandwidth will be devoted to perform the tasks.

Summarizing:

- The Witnet protocol is a Decentralized Oracle Network (DON) built on a blockchain with a native token. Clients spend tokens to get web contents retrieved, attested and delivered back to them; while a special kind of participants called *witnesses* earn tokens for fulfilling such work.
- Witnesses are assigned tasks based on their previous performance, which is measured by reputation points. The more reputation a witness has, the more likely it will be assigned a task and the more its claims (solutions to assigned tasks) will be taken into account.
- Each witness earns reputation points when its claims match the claims brought by a majority of its peers. On the contrary, reputation points are deducted when its claims contradict the majority.
- Finally, witnesses can also participate in the creation of new blocks for the underlying blockchain.

The likelihood that a witness will become a miner for the next block is proportional to its

³Prediction markets are exchange-traded markets created for the purpose of trading the outcome of events. They are based on the idea that market prices can indicate what the crowd thinks the probability of the event is. Notable examples of decentralized prediction markets are Augur[1], Gnosis[2] and Delphi[3].

current reputation.

1.3. Key Components

The Witnet protocol builds upon a series of novel components:

1. **Decentralized Oracle Network (DON):** We provide an abstraction for network of independent providers to offer Retrieve-Attest-Deliver services. Later, we present the Witnet protocol as an incentivized, auditable and verifiable DON construction.
2. **Reputation-Based Mining Protocol:** We show how to construct a useful Proof-of-Work that can be used in consensus protocols. Miners do not need to spend wasteful computation to mine blocks, but instead must fulfill task assignments.
3. **Reputation-Based Task Assignment Protocol:** We put forward an algorithm that is analogous to the Reputation-Based Mining Protocol and lets the network assign tasks to miners in a decentralized, fair, uniform, unpredictable yet deterministic way.
4. **Bridge nodes:** We describe a special kind of network participant that focus in the 'Deliver' part of the Retrieve-Attest-Deliver (RAD) tasks by allowing Witnet to interact with other blockchains.

2. DEFINITION OF A DECENTRALIZED ORACLE NETWORK

We outline a Decentralized Oracle Network (DON) as:

- a computer network made up of nodes (computers running a specific software),
- which communicate and operate as peers in compliance with an agreed protocol,
- to acquire knowledge of information that is external to the network,
- verify and agree on the veracity of the acquired information,
- and supply such verified information to other applications or networks that may need it.

In other words, a Decentralized Oracle Network is a *peer-to-peer* (P2P) network capable of processing Retrieve-Attest-Deliver requests.

2.1. Retrieve-Attest-Deliver (RAD) Requests and Tasks

Retrieve-Attest-Deliver (RAD) requests are the primary and most important element in a DON. As their name suggests, they contain specific information on how a piece of information must be retrieved, attested and delivered.

Going back to the outline of DON, the three elements of a RAD request can be easily defined:

- **Retrieve:** to acquire knowledge of information that is external to the network.
- **Attest:** to verify and agree on the veracity of the retrieved information.
- **Deliver:** to supply such attested information to the creator of the RAD request.

When a DON participant sends a RAD request to the network, the rest of participants must process the request by (1) retrieving the information, (2) check that they all have verbatim copies of the information, and (3) make it available for the the requester.

When a RAD request is assigned to a DON participant, it is called a RAD task.

In order to incentivize people to run nodes in a DON, the network can implement some form of native token that the participant sending a RAD request can use to reward the rest of participants for their work. This is quite analogous to the way in which miners are rewarded in Bitcoin for their work of including transactions into blocks[4]. When a DON participant is assigned a RAD task, it will get rewarded with a fraction of the tokens attached to the attestation request if and only if its claim matches the claims brought by a majority of the rest of participants.

To reduce the computational, energetic and monetary cost of performing RAD tasks, a DON can include some kind of sharding feature that makes it possible to assign a RAD task only to a fraction of the participants.

Given that smart contracts can rely on a DON to decide the outcome of business transactions, it is to be expected that the DON participants may have possible conflicts of interest when performing RAD tasks. In anticipation to this, a DON can implement a reputation scoring system that give participants different chances of being assigned tasks depending on their past performance in terms of honesty.

In addition, if a DON participant decides to tamper with the RAD tasks and bring false, biased or completely made up claims⁴, the chances are that such claims will contradict those brought by a majority of protocol-abiding participants, and therefore it will miss the opportunity to collect the token rewards.

⁴To accomplish such a deception, the deceiver would need to modify the code of the DON software he is running in such a way that human intervention would be possible on his own nodes, bypassing the automated nature of the software and breaking the consensus rules of the protocol implicit in the network. The main two reasons for this misbehavior could be conflict of interest (bringing false claims as a response to RAD requests whose outcome could have a potential off-the-network impact on his income) and a mix of sluggishness and greed (based on the assumption that performing RAD tasks entails some kind of marginal cost, a DON participant may choose to not actually perform them and just make the claims up).

3. WITNET AS A DECENTRALIZED ORACLE NETWORK

The network we are proposing in this work not only satisfies all requirements of a Decentralized Oracle Network (DON) but also applies a series of novel algorithms and strategies to guarantee the quality of the result of RAD requests and reduce if not completely eliminate the chance of manipulation by detecting and penalizing collusion.

Apart from the incentives that the use of token rewards create for participants to behave and not to feed false claims into the DON, we are applying a series of techniques that are heavily influenced by:

- Sztorc consensus algorithm, originally presented in the Truthcoin whitepaper[5].
- Augur consensus algorithm, which in turn is based on Sztorc's and presented in the Augur whitepaper[1].
- SchellingCoin conceptual protocol by Buterin[6].
- Classic literature on focal points and spontaneous answers in absence of communication. Best example is T. Schelling's *"The Strategy of Conflict"*[7].
- Studies on Reputation-Based voting and how they are affected by collusion attacks, like Bendahmane et.al.[8], Watanabe et.al.[9], Damiani et.al.[10] and Xiong et.al.[11].
- Studies on collusion detection and design of collusion-resistant protocols and networks, like Porter[12], Lepinski et.al.[13, 14] and Shareef[15].
- "Maximum independent set" algorithms for introducing diversity of interests in witnesses selection, like Araujo et.al.[16].

This kind of measures seriously increase the risk inherent to lying because they affect negatively to the cheaters not only by depriving them from short and mid term rewards, but also by denying them the chance to have a say in future tasks which may be more relevant to their interest than the one that caused them to be penalized in the first place.

For this type of incentives and measures to work as intended, it is often necessary that participants of the network (1) have no way to identify each other, (2) have no way to communicate with each other and (3) can not prove the content of their claims to others before the reputation and reward redistribution is made.

As explained by Buterin[6], it could be relatively easy for a single entity controlling something near a 49% of the DON participants to pre-announce that it will vote for some false claim, and others will also go with that claim out of fear that everyone else will, and if they don not, they will be left out.

Indeed, with a slightly more twisted scheme, the malicious entity would not even need to participate in the DON at all. The effect is just the same if the entity pre-announces the vote for some false claim, promises a bribe to whoever votes for the same, such bribe is greater than the reward they get from telling the truth, and a majority of participants take the bribe.

However, our protocol renders this kind of gambits completely useless by not giving participants the chance to reveal or prove the actual value of the claims they vote for.

Even if a participant accepts a bribe, it can still tell the truth to the DON, lie to the briber and earn both the reward and the bribe. This is the most profitable of its choices, and therefore the most likely to occur.

For its part, the briber gets zero guarantee that the participants will hold true to their word. Indeed, as the participants are incentivized to act for the good of the DON, tell the truth and deceive the briber, the most likely outcome is that none of the bribed participants will actually vote for the false claim. In a nutshell, any kind of bribe attempt will lead the briber to pour its own resources down the drain.

Witnet, being a tokenized DON, implements also a public ledger that keeps a historic record of all the transactions happening in the network.

Witnet transactions are very similar to Bitcoin transactions in the sense that they unlock tokens from the outputs of previous transactions, aggregate their value, and finally redistribute and lock the value into a new set of outputs.

Transaction outputs use a stack-based scripting language to establish their own spending conditions. This language (explained in more detail in section 6.4) is heavily inspired by Bitcoin's *Script*, although notable differences exist.

Just like in many other public blockchains, transactions are periodically aggregated into blocks by miners. However, unlike in most of them, the requisite for a Witnet miner to be entitled to "*find a block*" (create a new block that is valid according to the protocol) does not depend on the miner solving a mathematical puzzle. Instead, block miners are chosen by a deterministic algorithm that assigns such roles to participants based on their reputation. The higher their reputation, the more likely they will be elected as block miners. As block miners can collect transaction fees, all participants are incentivized to abide by the protocol, honestly perform the RAD tasks they are assigned and look after their reputation.

In Witnet, the time between creation of new blocks does not depend on a probabilistic process (the time spent by the fastest miner on solving the *proof-of-work* challenge). Instead, it is a deterministic routine—blocks are created periodically, regardless of them being totally full or completely empty.

To successfully bootstrap the Witnet DON and encourage its early adoption, block miners will be also rewarded with a certain amount of new tokens that are freshly created in every block. These *block rewards* are the only way for issuance of new Wit tokens, and their amount will decay over time according to the decreasing-supply algorithm described in section 3.1.4 on page 15.

Witnet focuses on retrieval, attestation and delivery of web contents. Any content that can be publicly accessed through HTTP or HTTPS can also be retrieved, attested and delivered by Witnet. However, other protocols may also be added in the future (FTP, FTPS, SFTP, TFTP, WebDav, BitTorrent, IPFS, etc).

A considerable part of the modern web is no longer static. When you visit a website using your favorite browser, what you see is not only one styled HTML document but the result of many client-side JavaScript computations that alter the document in many ways and even load other documents and asynchronous content in the background. Because of this dynamism, the software in charge of performing web contents retrieval needs to be capable of interpreting websites exactly in the same way that the typical web browser would. At the same time, to cover some very interesting use cases that we explain in section 3.2, we need such software to be capable of running predefined scripts inside the same web context.

With regards to all these considerations, Witnet miners will use a scriptable *headless browser* to perform the web contents retrieval. A headless browser is a web browser without a graphical user interface. It provides automated control of a website in an environment similar to popular web browsers, but are executed and controlled programmatically by other software. In our case, miners launch instances of the headless browser, make them navigate to the URL specified in the RAD request, run a series of computations (also specified by the request) and simply close the browser when finished.

3.1. Setting

3.1.1 Participants

Any user can participate in Witnet by running a network node that can act as a *client*, a *witness*, and/or a *bridge*.

- *Clients* pay tokens to get web contents retrieved, attested and delivered by *miners* through the DON. The due amount of fee to be paid will be discussed in detail later in section 6.3— for now, suffice it to say that it depends on the complexity of the Retrieve task, the desired number of *witnesses* to hire for the task and the usage of *bridges*.
- *Witnesses* earn tokens in exchange of fulfilling the Retrieve and Attest part of the RAD tasks.

Witnessing nodes are also eligible to mine new blocks, and in doing so they hence receive the mining reward for creating a block and transaction fees for the transactions included in the block.

- *Bridges* are nodes who specialize in fulfilling the *Deliver* part of the the RAD tasks. They are connected to other blockchains and earn tokens by (1) watching other blockchains in search for potential RAD requests to be introduced into Witnet, and (2) replicating the result of the RAD tasks into other blockchains upon request of the *clients*.

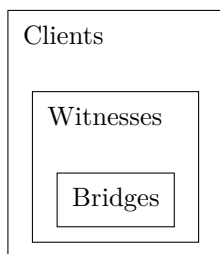


Figure 1: Network participants subsets

Bridges are a subset of the witnesses, which are in turn a subset of all the network participants (the clients).

All witnesses are clients, but not every client needs to be a witness. All bridges are witnesses, but not every witness needs to be a bridge.

3.1.2 The Network, \mathcal{N}

We personify all the users that run Witnet nodes as one single abstract entity: *the network*. The Network acts as an intermediary that runs the *Manage* protocol at every new block in the Witnet blockchain.

When we refer to network participants as *witnesses*, *miners*, or simply *the network*, we are not meaning actual human beings sitting in front of a computer, fulfilling assignments coming from an Internet overlord that commands them to use their web browser to navigate to a certain website and take a snapshot or copy some text that they must report back. Instead, the participants are computers running a *headless browser* software that automatically receives and executes the RAD tasks without the owner of the computer having to actively do anything else than install such software and configure how much of the available CPU and Internet connection’s bandwidth will be devoted to the RAD tasks.

3.1.3 The Ledger, \mathcal{L}

Our protocol is applied on top of an append-only ledger. For generality, we refer to this as *the ledger*, \mathcal{L}

Checkpoints are created at regular intervals, and the time between checkpoints is referred to as *epoch*. At any given epoch t , all users have access to \mathcal{L}_t , a snapshot of the ledger at the preceding checkpoint t . Epoch t begins at checkpoint t and finishes at checkpoint $t + 1$. This is, checkpoint t closes epoch $t - 1$ and opens epoch t .

The duration of each Witnet epoch (and thus the time elapsed between checkpoints) is fixed at 90 seconds.

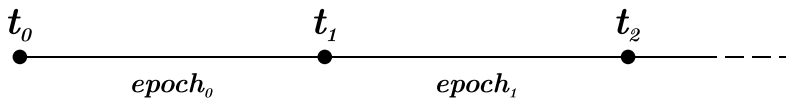


Figure 2: Checkpoints and epochs. Each epoch's number matches the preceding checkpoint.

There is not a minimum or maximum number of transactions that need to be included in a block. Miners can choose to include as many or as few transactions to the blocks they mine, although they are highly incentivized to include as many as possible in order to collect their fees. Empty blocks (those which include 0 transactions) can also exist.

Unlike Bitcoin's or Ethereum's blockchains, Witnet's ledger is not a linear chain but a more general form of Directed Acyclic Graph⁵ (DAG). This means that, for each epoch, more than one block can exist at the same time.

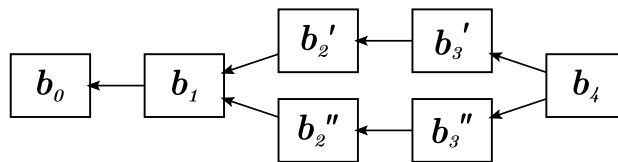


Figure 3: Directed Acyclic Graph. Our protocol uses a type of DAG that allows merging forked yet valid chains without endangering double-spend protection.

For every epoch, a new subset of the network witnesses are pseudo-randomly elected to be the "block miners" by the *Reputation-Based Mining Protocol* described in section 5.1. Each block miner has the exclusive power to mine (produce and broadcast) exactly one block during its one-epoch "term of office".

⁵In a DAG, every block is linked to any number of blocks, but there is no way to keep following the links from one block through other blocks back to the starting block. This is, no loops are allowed.

Blocks are identified not only for their hash (which is derived from its contents) but also for their checkpoint (which is equivalent to Bitcoin's block height). Two or more blocks for the same epoch can coexist as long as (1) they are correctly formed, and (2) the miners provide valid proofs of their epoch leadership inside the blocks' headers.

Given two or more blocks for the same checkpoint t , they will contain a similar (if not the same) set of transactions that were broadcast to the network during epoch $t - 1$. Transaction duplicity is addressed by our protocol in the most straightforward way possible:

- Transactions exist in the ledger as independent objects, identified by their hash.
- Blocks do not contain transactions per se but *pointers to transactions*.
- Two or more blocks for the same checkpoint can contain pointers to the same transaction.
- Two or more blocks for different checkpoints can NOT contain pointers to the same transaction. This is, a transaction can not be included into blocks for different checkpoints. The only valid transaction pointer will be the one in the block with the lowest checkpoint.

The double-spend⁶ problem is addressed in a similar way:

- Just like in Bitcoin, Witnet transactions unlock existing UTXO⁷s, redistribute the unlocked coins and lock them in a new set of UTXOs.
- The value of Witnet outputs is not expressed as an absolute number of coins but as a percentage of the sum of the values of the inputs.
- A number n of transactions spending the same UTXO with value v in the same block or in two different blocks for the same checkpoint are perfectly valid, but the value is equally split among their coincident inputs. This is, each of the n transactions will be able to spend at most $\frac{v}{n}$ coins.
- Witnet transactions have a series of special properties that make them trivial to be efficiently processed in parallel. Those properties are put forward in section 6.1 on page 38.

3.1.4 The Coin, Wit

Wit is Witnet's native token. Its generation algorithm defines, in advance, how new coins (token units) will be created and at what rate. Any coin that is generated by a malicious miner that does not follow the rules will be rejected by the network and thus is worthless.

Wits are created every time a miner publishes a new block. This is called the "*block reward*". In the event that two or more blocks were added to the ledger at the same time, the amount of coins

⁶Double-spending is an error in a digital cash scheme in which the same single digital token is spent more than once. (Wikipedia)

⁷Unspent Transaction Output.

created would be divided between the miners who mined them.

The number of wits generated per block starts at 500 and is set to decrease geometrically, with a 50% reduction every 1,750,000 blocks, or approximately 5 years. Each of these periodic reductions is known as *halving*. The result is that the number of wits ever created by the issuance mechanism will never exceed 2,500,000,000.

$$\sum_{i=0}^{\infty} \frac{500 \cdot 1750000}{2^i} \approx 2500000000$$

Figure 4: Controlled supply generation algorithm. As time passes, the issuance rate is periodically halved and the amount of coins ever issued by the mining rewards mechanism approaches the 2,500,000,000 limit. In this model, i represents the number of halving events ever passed.

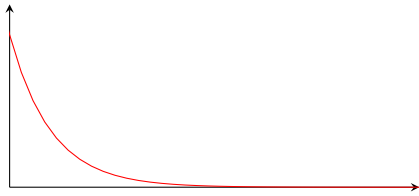


Figure 5: Coin issuance rate (inflation) will rapidly decay over time. Note that after only a few reward ages, block rewards will be marginal.

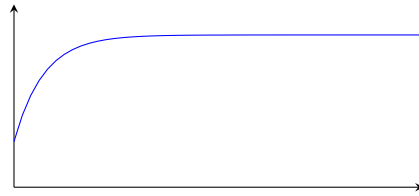


Figure 6: The total number of coins in circulation will rapidly grow at the beginning but then start to slow down after only a few reward ages.

The inflation rate steadily trends downwards. The block reward given to miners is made up of newly-created wits plus transaction fees. As inflation tends to zero over time, miners will obtain an income only from transaction fees, which will provide an incentive to keep mining to make transactions irreversible.

3.1.5 The Headless Browser, \mathcal{B}

As mentioned earlier in the introduction of section 3, witnesses use a scriptable *headless browser* to perform the web contents retrieval.

A headless browser is a web browser without a graphical user interface. It provides automated control of a website in an environment similar to popular web browsers, but are executed and controlled programmatically by other software.

In our case, witnessing nodes launch instances of the headless browser, make them navigate to the URL specified in the RAD request, run a series of computations (also specified by the request)

and simply close the browser when finished.

RAD requests can contain a small script that tells the headless browser which specific data units to pick from the visited website. That is the case for *Specific Data Units Retrieval*, explained in section 3.2.2 on page 19.

One RAD request can cause more than one instance of the headless browser to be spawned in parallel. That is the case for *Multiple Sources Fact Cross-Checking*, explained in section 3.2.3 on page 19.

Using a headless browser also makes the witnesses indistinguishable from a human being using a regular web browser to navigate a website, so it prevents an hypothetical kind of attack in which the administrator of a website—aware of his site being used to influence the outcome a smart contract—may tamper with the attestations by presenting fake or contradicting information to miners or even blocking them altogether.

3.2. Retrieval and Attestation Capabilities

3.2.1 General Case

At their simplest, these are the main steps of Witnet’s RAD flow:

1. Alice wants to get a web content retrieved, attested and delivered to her.
2. Alice prepares a RAD request that will be sent to the network. She needs to attach a certain amount of tokens to the request. This amount will depend on (1) the complexity of the retrieval, (2) the level of certainty that she expects for the attestation (the number of witnesses that will be employed), and (3) usage of `Retrieve` clauses in the request. RAD fees are further discussed in section 6.3.
3. Alice encodes the RAD request as a Witnet transaction and send it to the network.
4. The task of solving Alice’s request is assigned to a subset of all the witnesses. These are elected by consensus thanks to the *Reputation-Based Task Assignment Protocol* described in section 5.2. Each of these miners will:
 - (a) Retrieve the web contents that are specified in the request using the headless browser as described in section 3.1.5. The resulting value is what we call a *claim*.
 - (b) Calculate a *nonced hash* of the claim. This hash will be different for each witness and will be derived from the claim itself, the miner’s public key and the hash of the latest block.
 - (c) Send the hash of the claim to the network as a commitment to publish the actual claim when the rest of designated witnesses have also made their own commitments. The

witnesses also use these commitment transactions as a pledge for reserving for themselves a share of the tokens attached to the RAD request.

5. Once all the designated witnesses have made their pledges, the block miner(s) will divide the reward tokens that are attached to the RAD request among all the witnesses who made a valid pledge for such request.
6. The witnesses who made their pledges will start to reveal the actual retrieved contents. They will do so by sending reveal-redeem transactions that spend their commitment transactions and send the reward tokens to their own wallet.
7. The block miner(s) for the next block will have to compare the claims coming from different witnesses and choose which is the winning claim (nominally, *"the truth"*) by applying the *Truth-By-Consensus* algorithm explained in section 4.2.
 - The witnesses who told the truth will earn reputation points and the transactions redeeming their share of the reward will be accepted and included into next block.
 - On the contrary, those witnesses who lied will lose a fraction of their reputation points and they will not be able to redeem their share of the reward.
8. At this point, the result of the attestation will be public and available to Alice as well as to any other participant of the network.
9. If a Deliver clause was specified in the request, bridges⁸ capable of performing such type of tasks will come into play and do their job (e.g.: *"Send the result as a parameter to the callback() function in the Ethereum smart contract with address 0xe711fA745e..."*).

$$\left. \begin{array}{l} Claim_{witness1} \\ Claim_{witness2} \\ Claim_{witness3} \end{array} \right\} \rightarrow \text{Truth-By-Consensus} \rightarrow \mathbf{Result}$$

Figure 7: Truth-By-Consensus. For each RAD request, the block miner(s) compare the claims coming from all the designated witnesses, weight the values depending on each one's reputation and compute the result using the *Truth-By-Consensus* algorithm as described in section 4.2

Section 3.4 on page 23 defines the client and miner cycles in detail.

There exist some use cases described hereafter that despite of introducing some additional complexity are addressed by the proposed network without any problems.

⁸See section 3.1.1 on page 12.

3.2.2 Specific Data Units Retrieval

For example, Alice might be interested in the acquisition and attestation of a single data unit out of a whole web page.

For these cases, Witnet provides a flow-based, tacit, point-free scripting language that will let the client to (1) indicate the specific element to be retrieved, attested and delivered, and (2) apply basic transformations on it.

For the sake of ease, such language can be implemented as a *domain specific language* (DSL) on top of the Javascript programming language, which is already known by most smart contracts developers and web developers.

The combination of one URL and one of these scripts is called a *retrieval path*.

The *RAD cost* mentioned earlier in this chapter is directly affected by the computational complexity of the retrieval path being included in the request, as well as by the data size of the returned value. RAD fees are further discussed in section 6.3.

$$Acquisition + Normalization = RetrievalPath$$

Figure 8: Retrieval paths. Each retrieval path is composed of one acquisition (URL) and one normalization script.

3.2.3 Multiple Sources Fact Cross-checking

Alice may also want to cross-check some fact by requesting the retrieval and attestation of data units representing the same reality but published by different entities across a number of web sites.

Our protocol covers this case as well by allowing the client to include multiple retrieval paths in a single attestation request. In addition, the scripting language introduced in section 3.2.2 includes normalization and aggregation methods in a MapReduce style that enable the merging of data coming from a plurality of sources despite of the slight format differences that might exist between them.

Generally speaking, a RAD request must contain one or more retrieval paths and the definition of one *path aggregation function*.

For every retrieval path, a headless browser instance is launched, the corresponding URL is loaded and the matching normalization script is run. Once the normalization scripts from all the retrieval paths have finished running, an additional instance of the headless browser is created and pointed to a clean context in which the aggregation function is run over a list containing the results

of each retrieval path.

The whole retrieval flow performed by each miner in a multiple sources fact cross-checking RAD request is depicted in Figure 9.

$$\left. \begin{aligned} Acquisition_a + Normalization_a &= RetrievalPath_a \\ Acquisition_b + Normalization_b &= RetrievalPath_b \\ Acquisition_c + Normalization_c &= RetrievalPath_c \end{aligned} \right\} \rightarrow Aggregation \rightarrow \mathbf{Claim}$$

Figure 9: Content retrieval flow. Each designated witness brings a single claim that is derived from the aggregation of the values that result after applying normalization methods on the acquired web contents.

3.2.4 Future Facts

It is also conceivable that Alice may want to retrieve and attest a verified piece of data whose value is unknown, uncertain or impossible to resolve at the time of formulation of a RAD request.

Example 1

A smart contract is set to have a different outcome depending on the answer to the question "How much will 1 bitcoin be worth in 2 years from now?".

Such outcome can not be trivially resolved today because of indetermination: the answer is yet unknown and will only be known once the date stated by the question itself has come.

The inability to evaluate this kind of questions comes from the statements having a verification precondition—either expressed or implied—that must be met before the claim itself can be evaluated and verified. In the Example 1, the precondition is "*Has it been 2 years now since the request was formulated?*".

Although the kind of functionality implemented by a DON is commonly known as *oracle*, Witnet is radically different to the oracles from the classical antiquity in the sense that we make an important distinction between predictions or beliefs (claims that *may* hold true but are impossible to verify) and truth (claims that are verifiable at this time).

As we the ordinary mortals have no means to verify a claim whose veracity is not yet determinable or to foresee the future answer to simple questions like the one in Example 1, the only thing we can do is waiting for such claims to eventually become verifiable.

For these cases, the proposed network includes a *time lock* feature that, upon request, will keep a RAD request unresolved until a certain point in the future. Once that point in time has passed, *witnesses* are strongly incentivized to immediately resolve the request in order to harvest its fees. In anticipation that witness fees and block miner fees may grow over time, a *Replace-By-Fee* feature will allow a client to "republish" its request using an increased reward as long as the retrieval paths remain the same.

3.3. Retrieval and Attestation Limitations

3.3.1 Undecidability

There exist claims and statements whose truth or falsehood are neither provable nor refutable. These are called *undecidable statements*.

Undecidable statements and undecidable problems have been object of abundant study by many authors during the last century, notably Gödel[17], Church[18], Turing[19], Rosser[20], Rice[21], Kleene[22] and Conway[23].

The concept of undecidable statements is itself based in the notion of *decidability*. In logic, decidability is the question of the existence of an effective method to determine the truth or falsehood of a statement.

- A statement whose truth or falsehood can be evaluated in the present is decidable.
- A statement whose truth or falsehood can not be evaluated in the present but will become assessable in the future is also decidable.
- However, a statement whose truth or falsehood can not be evaluated to a correct (determined and well-formed) value after finite though possibly long time, is considered undecidable.

The problem behind establishing the decidability of a certain statement is quite equivalent to the halting problem in computability theory: determining, from a description of a computer program and an input, whether the program will eventually finish running and return a valid output or whether on the contrary it will continue to run forever.

Just like in the halting problem, which was proved undecidable by Alan Turing in 1936[19], the assessability of a certain statement is itself undecidable. Although common sense can tell us in most cases whether a statement is assessable at the moment of formulation or whether it will be assessable in the near future, there still exist statements whose assessability can not be predicted or assured to the 100%.

Example 2

A smart contract is set to have a different outcome depending on the answer to the question "Will 1 bitcoin ever be worth \$50K?".

Such outcome can not be trivially resolved today not only because of indetermination—the answer is yet unknown—but also because of undecidability, as we can not foresee when will the question itself be answerable or whether it will ever have an answer whatsoever.

Like with the question in the Example 1 in section 3.2.4, the inability to evaluate this kind of questions also comes from the statements having a verification precondition—either expressed or implied—that must be met before the statement itself can be evaluated and verified. In the Example 2, the precondition is *"Has 1 bitcoin ever been worth \$50K?"*⁹.

For these cases, the Witnet protocol specifies a special type of RAD requests that remain resolved indefinitely. This is, no miner is able to pledge a solution for it.

However, if an undecidable request eventually becomes decidable, the client can produce a transaction that will "relaunch" the request, but this time as a regular RAD request so that miners can start pledging solutions to it as soon as the next epoch starts.

3.3.2 Unverifiability

As seen in sections 3.2.4 and 3.3.1, not all claims are verifiable, being indetermination and undecidability the two main reasons for such lack thereof.

Verifiability or provability is the capability of a certain statement to be demonstrated, verified, confirmed, substantiated or logically proved.

Douglas R. Hofstadter, in his 1979 Pulitzer-winning book *"Gödel, Echer, Bach: An Eternal Golden Braid"*, states that *"provability is a weaker notion than truth"*[24]. We often can not prove things that we know are true.

As regards Witnet, the only possible truth is the verifiable one. Indeed, verifiability is hard-coded into the design of the network. The mere act of performing distributed retrieval of data is in itself a form of verification, specially if several web sources are queried via multiple acquisition paths as described in sections 3.2.2 and 3.2.3 on page 19. In the same manner, the fact that a final

⁹Note that in the given example the check in the precondition is the same as the question itself but in past tense. This is a special case in which the statement can never be evaluated to a **false** value: before the precondition is met the result is undetermined, and after it is met the result is always **true**.

agreed claim emerges from the aggregation of all the claims brought by a plurality of miners proves itself that the claim was verifiable at the time of the attestation.

There exist, of course, realities and facts that, despite of their truthness, can not be processed as true statements by Witnet because of their lack of verifiability. It is not that the network will fail to resolve a request expressing a question whose answer is unverifiable. As a matter of fact, such type of questions is just impossible to translate into *RAD* requests. This is because these requests—and more specifically *retrieval paths* as described in section 3.2.2 on page 19—are explicit about how claims are extracted and derived from available information online. That is why the only impossible attestation is the one with retrieval paths pointing to nonexistent web contents or applying transformations on the retrieved data in such a way that different paths from a single request result in contradicting claims. In other words, the success of a *RAD* request depends exclusively on its own design, so it is the responsibility of the client to only include valid and provable retrieval paths.

3.4. The Protocol

In this section, we give an overview of the Witnet DON by describing the operations performed by the clients, the Network and the different types of miners.

3.4.1 *Client Cycle*

Clients performing *RAD* requests run the following protocols in addition to the ones described in the *network Cycle* in section 3.4.4 on page 26.

1. **RAD-Post:** *Client requests the retrieval, attestation and delivery of web contents.*

Clients can request the retrieval, attestation and delivery of web contents by paying witnesses in *Wit* tokens.

A client initiates the *RAD-Post* protocol by submitting a *RAD client request* transaction to the network. This type of transaction is further described in section 6.2 on page 39.

Clients can decide the amount of miners that will be assigned to each *RAD* task by specifying a replication factor in the request. The minimum replication factor is 2. Higher redundancy results in a higher certainty and confidence of the attestation. A replication factor around 6 should be more than enough for most cases while keeping costs under control¹⁰.

The *RAD client request* transaction must pay a miner fee appropriate to its own size when

¹⁰A maximum replication factor shall also be imposed in order to avoid abuse of the DON to conduct (expensive) DDoS-attack on websites. In the same manner, the network should only allow a limited number of *RAD* requests that included equivalent retrieval paths and are set to be resolved at the same checkpoint.

serialized. This is, the sum of the values of its inputs must exceed that of the outputs in a sufficient amount to make it attractive for block miners to include it into a block.

2. **RAD-Get:** *Client reads the result of RAD requests.*

In the same moment that block miners apply the Truth-By-Consensus algorithm on the claims brought by witnesses, a final "*verdict*" pointing to the retrieved and attested content emerges. The reveal-redeem transactions containing the value of the retrieved and attested content (*the solution*) are immediately included into the block being mined.

As any client can see and read the ledger at any time, as soon as the block containing the solution is broadcast to the network, it can be easily traced back to the related RAD-Post transaction and matched with the original request.

Therefore, the RAD-Get protocol can be run locally—thus with no transaction cost—by any client with an up-to-date copy of the Witnet ledger.

3.4.2 Witness Cycle

Witnesses perform the following protocols in addition to the ones described in the *network* cycle in section 3.4.4 on page 26.

1. **Receive RAD requests:** *Witnesses read RAD-Post requests from the blockchain.*

From the moment that a RAD-Post gets broadcast to the network, all witnesses have all the information they need to start working on the task by executing its Retrieve part.

However, if the request has a precondition (see *time locked* and *undecidable* requests in sections 3.2.4 and 3.3.1), it must be kept unresolved until its precondition is met.

Any RAD request without a precondition can be worked on immediately, although miners should refrain from doing so until they know if they have been designated as witnesses for such task. Otherwise, they may be spending resources in exchange of no reward at all.

2. **Discover Retrieve tasks assignments:** *Witnesses apply the task assignment protocol to discover task assignments.*

At any moment, a witness can locally run the *Reputation-Based Task Assignment Protocol* described in section 5.2 on all the RAD requests broadcast during the current epoch to figure out which RAD tasks it has been designated for.

If there were any pending RAD requests with a time lock precondition expiring in the last checkpoint (see section 3.2.4), they can also be checked for assignment using the same task assignment protocol.

3. **Retrieve and commit-pledge:** *Witnesses perform the retrieve part of their assigned RAD tasks and pledge to publish the results in the future.*

At this point, each witness should start fulfilling their assigned RAD tasks using their bundled headless browser as described in section 3.1.5. For every RAD task, each designated miner will come up with one and only one claim.

Once they have their claims, witnesses can compose their **commit-pledge** transactions and broadcast them right away. In these transactions, witnesses must also include all information necessary to prove their task assignment to the rest of the network.

4. **Reveal-redeem:** *Witnesses reveal the results of their assigned RAD tasks and redeem their reward.*

Once the **commit-pledge** transactions have been included in a block, each of the witnesses who made the pledges must reveal their claims and provide everything necessary to prove that (1) the nonced hashes that they committed in their respective **commit-pledge** transactions were actually derived from the revealed claims, and (2) they are the authors of the pledges they are trying to redeem.

3.4.3 *bridge* Cycle

Bridge nodes (as defined in section 7.1) are in charge of delivering the claims that result from Request-Attest tasks to other blockchains. Bridges also monitor those other blockchains in search for potential RAD requests to be introduced into Witnet.

They perform the following protocols in addition to the ones described in the *witness* cycle in section 3.4.2 on the previous page and the *network* cycle in section 3.4.4 on the next page.

1. **Discover Deliver task assignments:** *Bridges apply the task assignment protocol to discover Deliver task assignments.*

At any moment, a bridge node can locally run the *Reputation-Based Task Assignment Protocol* described in section 5.2 on all the RAD requests resolved during the current epoch to figure out which Delivery tasks it has been assigned to.

2. **Deliver attested claims:** *Bridges perform delivery of attested claims to other blockchains.*

Bridges are in charge of reporting the results of RAD tasks to other blockchains. In exchange of performing this work and spending their own time and coins in sending their reports, they will be awarded with Wit tokens that were allocated for such purpose in the request transaction.

3. **Discover outer RAD requests:** *Bridges discover RAD requests posted in other blockchains.*

Bridges monitor other blockchains in search of RAD requests codified inside transactions broadcasted by unknown clients in those blockchains. When bridges find one of these transactions, they read its payload and convert it into a valid Witnet RAD request.

4. **RAD-Post:** *Bridges post RAD requests in behalf of users of other blockchains.*

Bridges must broadcast the RAD request they have built in behalf of the unknown client from the outer blockchain. In exchange of performing this work and spending their own Wits, bridges will need to be rewarded by the unknown clients in whatever form they accept (likely in the native tokens of the outer blockchain).

3.4.4 Network Cycle

For every Witnet epoch, the network (all participants) must:

1. **Receive and validate last epoch's block(s):**

At the start of each epoch, the network checks that all the blocks received for the last epoch contain a valid proof of leadership and a commitment (signed reference) to one or more blocks from the previous block.

Each participant of the network must check if the transactions included by the miners into the blocks have been previously received and checked for validity. If not, they must request their peers for the newly discovered transactions.

The network must then create a Merkle tree with all the transactions included in each of the blocks and check if the root of the resulting tree matches the root used in the block header.

2. **Receive and validate incoming transactions:**

During each epoch, all the transactions broadcast by the clients are sent to each of the participants of the network. Upon reception of a new transaction, the network must check its validity (correctness of their inputs and success of redeem scripts) and store it for posterior inclusion into blocks.

3. **Check for epoch leadership:**

At every epoch, all network participants (clients, witnesses and bridges) can use the *Reputation-Based Mining Protocol* from section 5.1 to discover if they have been elected block miners for the current epoch. If so, they will be implicitly authorized by the network to produce and broadcast exactly one block once the epoch has finished.

4. **Mine a block:**

At each epoch checkpoint, every block miner has the power to mine (produce and broadcast) one block.

Block miners can include as many transactions as they want into a single block, with the only limitation being keeping the size of the block under 1MB. Predictably, block miners will prioritize small transactions paying high rewards over bigger transactions paying lower

rewards¹¹.

A block miner who created a new block needs to prove its right to do so to the rest of the network by including a proof in the block header. These "proofs of leadership" are described in detail in section 5.1.

Each of the block miners who created a new block must broadcast their blocks to the rest of the network during the next epoch. Otherwise, their blocks may be rejected by the network.

¹¹Note that *small* and *big* are not referring to the amount of tokens being transferred but to the size in bytes of the transaction when serialized as described in section ??.

4. REPUTATION

A key feature of Witnet is *reputation points*. The total amount of reputation points in the system is a fixed quantity, determined upon the launch of Witnet. Holding reputation points entitles a Witnet witness to be elected for performing RAD tasks and more generally any network participant to mine new blocks. The higher the reputation score a participant has, the more likely it will be able to collect token rewards from performing RAD tasks and mining new blocks.

Witnet reputation points work in a similar way to Augur's *REP*[1], Truthcoin's *Votecoins*[5] and, to a lesser extent, Filecoin's *Power*[25]—they are gained and lost depending on how reliably their owner votes with the consensus.

Reputation points clearly define and limit the fuzzy concept of "reputation". The existence of a fixed amount of total reputation points provides Sybil attack immunity¹² and at the same time gives the network an effective way to penalize miners for laziness.

Reputation points are affected by *demurrage*¹³: network participants lose reputation if they hoard their points instead of using them to become witnesses and have a say in the outcome of RAD requests. In that sense, it can be said that reputation points are a liability as well as an asset, because their owners are obliged to put them to proper use or lose them altogether if they do not. On the contrary, Wit—Witnet's native token—is not affected by this demurrage policy.

4.1. Reputation Protocol

- Every possible public key has its own reputation score¹⁴.
- At checkpoint 0 of the ledger, all public keys will have equal reputation scores of 1.
- The sum of the scores of all the public keys is fixed and will equal 2^{256} points at all times.
- New reputation points can not be created after checkpoint 0.
- Reputation points can not be destroyed and they never leave the network.
- Reputation points are earned by witnesses when they agree with a majority of the rest of the designated witnesses on the resulting claims of the RAD tasks they get assigned.

¹²The proposed network is designed to not depend in any way on the number of participants in the network. All of its implicit economic models work just the same regardless of whether there is a single large actor holding a big part of the reputation or a million small actors holding the same amount of reputation points.

¹³Demurrage is the cost associated with owning or holding currency over a given period. It is sometimes referred to as a carrying cost of money. For commodity money such as gold, demurrage is the cost of storing and securing the gold. For paper currency, it can take the form of a periodic tax, such as a stamp tax, on currency holdings.—Wikipedia – Demurrage (currency).

¹⁴Witnet uses ECDSA over the `secp256k1` curve as its main signature algorithm. This setup can accommodate up to 2^{256} different public keys.

- Reputation points are lost by witnesses when they contradict or fail to agree with a majority of the rest of the miners on the resulting claims of the RAD tasks they get assigned.
- At every epoch checkpoint, reputation points are lost by all the network participants at once as a form of demurrage. Those points are rewarded to the honest miners that fulfilled such epoch’s RAD tasks.
- At every epoch, the sum of reputation points earned by honest miners equals the sum of points deducted from dishonest ones plus the sum of points deducted from every participant by the demurrage system.
- If during a certain epoch every witness is honest, none of them will lose reputation. However, demurrage will still apply to all participants, and the deducted points will be distributed evenly among the honest witnesses and bridges that fulfilled such epoch’s RAD tasks.

In Witnet, being elected for block mining or designated for fulfilling RAD tasks works just like a lottery in which reputation points are the lottery tickets. The more reputation a participant owns, the bigger its chances to win the right to collect block and tasks rewards.

To ensure that the most reputable participants also bear a greater liability, the reputation demurrage function has been modeled in such a way that it applies a deduction on each reputation score in proportion to the score itself. It causes the reputation score of the most reputable participants to decay rapidly while the score of the smallest participants is left almost intact.

This type of progressive demurrage can also be seen as a measure to fight concentration and favor redistribution of reputation points, just like wealth taxes take a larger percentage from high-income earners than they do from low-income individuals.

Figure 10 depicts the function that governs Witnet’s reputation demurrage system. The decay rate (\mathcal{D}) must lie in the unit interval $[0, 1]$, so $0 \leq \mathcal{D} \leq 1$. In the proposed function, this rate is a parameter that can be adjusted to make the reputation system more or less progressive. The closer that \mathcal{D} gets to 0, the more that the most reputable participants will be urged to participate in the system. On the contrary, the closer \mathcal{D} gets to 1, the more that they will be able to remain idle without a profound negative impact to their potential income.

$$\mathcal{R}_{epoch_t} = \mathcal{R}_{epoch_{t-1}} \cdot \mathcal{D}^{\log_{10}(\mathcal{R}_{epoch_{t-1}})}$$

Figure 10: Reputation demurrage function. *The logarithmic exponentiation of the decay rate (\mathcal{D}) causes the score of the participants with the biggest reputation stake to decay more rapidly.*

We are proposing an initial decay rate of $\mathcal{D} = 0.99$. Although this rate could seem too con-

servative at first sight, it applies a reasonable decay to participants of all sizes, as depicted in figure 11.

	Epochs								
	0	1	25	50	75	100	125	150	175
Scores	1	1	1	1	1	1	1	1	1
	10	9.90	7.87	6.36	5.25	4.42	3.79	3.30	2.91
	100	98.01	62.06	40.46	27.58	19.56	14.37	10.90	8.51
	1000	970.29	480.99	257.42	144.85	86.51	54.50	36.01	24.84
	10000	9605.26	3851.53	1637.54	760.72	382.61	206.63	118.95	72.50

	Epochs												
	200	225	250	275	300	325	350	375	400	425	450	475	500
	1	1	1	1	1	1	1	1	1	1	1	1	1
	2.61	2.36	2.16	1.99	1.85	1.74	1.64	1.56	1.49	1.43	1.37	1.33	1.29
	6.82	5.59	4.67	3.98	3.45	3.03	2.70	2.44	2.22	2.04	1.90	1.77	1.67
	17.81	13.21	10.11	7.96	6.42	5.29	4.45	3.81	3.32	2.93	2.62	2.37	2.17
	46.52	31.25	21.87	15.89	11.93	9.23	7.33	5.96	4.95	4.20	3.61	3.16	2.81

Figure 11: Reputation demurrage simulation. In this simulation with $\mathcal{D} = 0.99$, 5 participants remain idle (not accepting RAD tasks) for 500 epochs (12.5 hours). At epoch₀ the differences among each participant's scores is $\times 10$. At epoch₂₀₀, demurrage has reduced such differences to less than $\times 3$, and at epoch₅₀₀ their scores have nearly converged. Over the total period, the score for each of the participants was divided by: 1, 8, 60, 461 and 3559 respectively.

In Witnet, reputation has the following properties:

- *Public*: By reading the blockchain, anyone can calculate the reputation of each participant at any point in time.
- *Verifiable*: Reputation is earned and lost by performing RAD tasks whose outcome is publicly available. By reading the blockchain, anyone can verify the outcome of those tasks and check if the reputation claimed by a participant is correct.
- *Fluid*: At any point in time, participants can earn reputation points by becoming witnesses and performing RAD tasks honestly; or lose them if they tamper with those tasks or simply ignore them. In this way, reputation points are always flowing from those who do not contribute to the system toward those who do.

4.2. Truth-By-Consensus and SVD

Truth-By-Consensus is Witnet's protocol for comparing and finding an "agreed truth" among a number of potentially conflicting claims brought by independent participants of the network.

Truth-by-consensus is roughly the same algorithm as Truthcoin's[5], which in turn is based on *Singular Value Decomposition* (SVD). This algorithm is in a way analogous to the statistical technique of Principal Component Analysis (PCA), although it introduces weighting in order to take reputation into account.

The purpose of SVD is to analyze a matrix containing all the claims brought during one epoch and reveal and sort its effects by influence, detecting and dropping outliers and collusion (coordination) in the process.

To measure coordination, SVD uses the first score from a weighted principal components analysis. This column represents the degree to which each miner's claims differ from those of a theoretical maximally representative of the covariance across all miners and claims.

For each epoch, reputation points are redistributed among all the witnesses who were designated for fulfilling RAD tasks and brought their claims in a timely manner. As said before, if the claims are 100% unanimous, reputation scores do not change (apart from the effect of demurrage).

Reputation redistribution could become a rather expensive operation in terms of computational complexity as the number of network participants increase. To relieve this complexity, this protocol can be implemented in such a way that scores get only updated once every few epochs. However, for security reasons, this "lazy reputation recomputation period" should be kept as low as possible as otherwise we would be deferring the punishment to liars and such measures would lose part of their efficacy. Please note that this recomputation period must form part of the network consensus as reputation score has important implications to the Useful Work Consensus algorithm in section 5.

With the view to ease the computational cost of reputation recomputation, the network shall keep a list of all participants whose reputation is different to the initial neutral value of 1: the *engaged set*. To limit the size of such set, all reputation scores with a value just above 1 can be immediately assimilated to 1, and the difference added to the reputation redistribution. As demurrage does not affect those miners with a neutral reputation, at any epoch checkpoint, only the reputation of the participants in the engaged set will need to be recomputed.

Truth-by-consensus ensures that across a number of epochs, the network participants have a strong incentive to become witnesses, perform their assigned RAD tasks honestly and bring true claims: revenue maximization. All participants are incentivized to get and keep a high reputation as their potential income depends heavily on their score.

As introduced earlier in section 3, Witnet provides a strong incentive for witnesses to keep their claims secret until all of them have revealed their commitments, or even lie to each other or to a theoretical briber. This "double-agent incentive"—as named by Sztorc[5]—guarantees that in the event that someone was trying to coordinate witnesses outside of the network, witnesses would prefer to lie to the coordinator and still perform their tasks honestly.

As long as $>50\%$ of the witnesses are honest, cartels and pools are heavily discouraged as each of the witnesses will want to minimize the number of fellow honest voters, as they all compete for a share of the same rewards and can thus be seen as rival.

5. USEFUL WORK CONSENSUS ALGORITHM

The Proof-Of-Work (PoW) systems used by most public blockchains has been proved to be a great measure to achieve decentralized consensus and ultimately tell who gets the right to have the last word on which transactions are written into a common ledger.

PoW puts a number of parties—the *miners*—to compete for being the first to solve a mathematical problem whose solution is a "golden ticket" which grants its bearer the right to (1) aggregate as many transactions as they can fit into a limited-size block, (2) collect the unassigned value—the *fees*—of those transactions, and (3) collect the block reward. Such mathematical problems are asymmetrical: they are extremely hard to solve, yet their solutions are easy to verify; and change every time a new valid block is published.

While everyone can produce and broadcast a block, the network will only accept the first one that contains (1) a reference to last known valid block, and (2) valid a solution to the current problem. In short, your potential income as a miner depends on your computing power, which in turn depends on your ability to invest in specialized mining hardware and the price of electricity in your area.

Given the nature of DONs, the chances of a certain miner to mine a block or to have a say in the outcome of a RAD request can not depend on its purchasing power or low price of electricity. Instead, those chances need to be related to reputation—past performance in terms of honesty—and, ultimately, randomness.

The consensus scheme used by Witnet is similar to those proposed by Micali[26], Bentov et.al.[27], Daian et.al.[28], and Benet et.al.[25]. In this scheme, the probability that the network elects a certain participant to create a new block or to fulfill a RAD task is proportional to its reputation score in relation to the total reputation points existing in the network.

Miner influence. In Witnet, the *influence* I_i^t of a participant \mathcal{M}_i at checkpoint t is the fraction of \mathcal{M}_i 's reputation score r_i^t over the total reputation points in the network $\sum_j r_j^t$.

$$I_i^t = \frac{r_i^t}{\sum_j r_j^t}$$

Figure 12: Miner influence calculation. *The influence of a participant in the network is proportional to its reputation stake. In that sense, reputation works as shares in a hypothetical "oracles corporation" formed by the set of all witnesses [5].*

One could directly substitute the $\sum_j r_j^t$ term in figure 12 on the previous page with the total supply of reputation points. However, this would imply assuming that all the possible key pairs correspond to witnesses who are actively competing for leadership of the current epoch. Such assumption would cause new participants to have a ridiculously low chance to be elected for leadership ($\frac{1}{2^{256}}$ for ECDSA).

Instead, the $\sum_j r_j^t$ term must be computed from the *engaged set* as defined in section 4.2. This is, $r_j^t > 1$. This way, the influence of a certain participant is always proportional to its own reputation in comparison to the summation of that of its active peers.

5.1. Reputation-Based Mining Protocol

Witnet’s Useful Work Consensus Algorithm aims to deterministically, unpredictably, and secretly elect a small set of miners at each epoch. Predictably, the number of elected miners per epoch is 1.

This a probabilistic consensus protocol, where each epoch introduces more certainty over previous blocks, eventually reaching enough certainty that the likelihood of a different history is sufficiently small.

A participant \mathcal{M}_i is a miner at epoch t if the following condition is met:

$$\mathcal{H}(\langle t \parallel \text{rand}(t) \rangle_{\mathcal{M}_i}) / 2^L \leq I_i^t$$

Where:

- $\text{rand}(t)$ is a public randomness that can be extracted from the blockchain at epoch t .
- $\langle t \parallel \text{rand}(t) \rangle_{\mathcal{M}_i}$ is a signature of message $t \parallel \text{rand}(t)$ produced with private key \mathcal{M}_i .
- \mathcal{H} is a deterministic, uniform and non-reversible hash function.
- L is the number of bits of the output size of the \mathcal{H} hash function.
- I_i^t is the reputation of participant \mathcal{M}_i at epoch t , calculated as in figure 12 on the previous page.

Figure 13: *Block mining calculation.*

Being this protocol probabilistic, the number of miners per epoch can be expected to be 1, but only *on average*. For some epochs there will be more than 1 miner, which is no problem at all for Witnet’s ledger because of its DAG architecture. On the contrary, for some other epochs, it could happen that none of the participants would be eligible for mining¹⁵. Although empty blocks are

¹⁵Or none of them realized their eligibility.

possible (as described in section 3.1.3 on page 14), this protocol provides a "backup" strategy that allows the network to avoid such empty blocks by letting a different subset of participants to take over the mining process if the former elected miners fail to produce and broadcast valid blocks.

A participant \mathcal{M}_i is a backup miner at epoch t if the following condition is met:

$$\mathcal{H}(\langle t \parallel \text{rand}(t) \rangle_{\mathcal{M}_i}) / 2^L \leq \mathcal{B} \cdot I_i^t$$

Where:

- $\text{rand}(t)$, $\langle \dots \rangle_{\mathcal{M}_i}$, \mathcal{H} , L and I_i^t preserve the same meaning and value as in figure 13 on the previous page.
- \mathcal{B} is the backup index. Note that figure 13 is equivalent to this equation when $\mathcal{B} = 1$.

Figure 14: Backup block mining calculation.

During each epoch, participants can claim their right to mine and produce blocks for any value of \mathcal{B} such that $\mathcal{B} \in \mathbb{N}$ and $\mathcal{B} \geq 1$. Nevertheless, only those valid blocks with the lowest \mathcal{B} value will be accepted and worked upon by the network. As producing a block is a trivial burden from a computational standpoint, it is to be expected that every participant will try to produce and broadcast valid blocks for different low values of \mathcal{B} , just in case no other participant succeeds to produce and broadcast a valid block for any lower value of \mathcal{B} .

Methods for extracting randomness from public blockchains—as needed by $\text{rand}(t)$ —have already been proposed by Bonneau et.al.[29]. Note that the value of the randomness $\text{rand}(t)$ is derived from the hash of the blocks for epoch $t - 1$ and therefore can not be known before epoch t .

This type of scheme provides three main properties, as pointed out by Benet et.al.[25]:

- *Fairness*: each participant has a fair chance to make a profit from their own work during each epoch, since signatures are deterministic and both t and $\text{rand}(t)$ are fixed. Assuming \mathcal{H} is a secure cryptographic hash function, then $\mathcal{H}(\langle t \parallel \text{rand}(t) \rangle_{\mathcal{M}_i}) / 2^L$ must be a real number uniformly chosen from the range $[0, 1]$. Hence, the probability for the equation to be true must be I_i^t , which equals to the participant's influence—its portion of reputation within the network. Because this probability is directly proportional to influence, this likelihood is preserved even under splitting or influence pooling.
- *Secrecy*: an efficient adversary that does not own \mathcal{M}_i 's secret key can compute the signature with negligible probability, given the assumptions of digital signatures.
- *Public verifiability*: a designated block miner can convince any wary verifier by showing t , $\text{rand}(t)$ and $\mathcal{H}(\langle t \parallel \text{rand}(t) \rangle_{\mathcal{M}_i}) / 2^L$. Given the previous point, no one can generate a proof

without having a winning secret key.

5.2. Reputation-Based Task Assignment Protocol

The protocol that grants witnesses the right—and responsibility—to perform Retrieve and Attest tasks is quite similar to the Reputation-Based Mining Protocol described earlier in this section. Witnesses get higher or lower chances to be designated for fulfilling those tasks depending on their reputation score. Assignment of Deliver tasks to bridges is also ruled by this protocol and done by the same means.

As introduced earlier in section 3.4.1, every RAD request must contain a *replication factor*, \mathcal{R} , that tells the network the minimum number of witnesses that must perform the RAD task and participate in the attestation.

A witness \mathcal{M}_i is elected for fulfilling a certain RAD task in epoch t if the following condition is met:

$$\mathcal{H}(\langle t \parallel \text{rand}(t) \parallel n \rangle_{\mathcal{M}_i}) / 2^L \leq \mathcal{R} \cdot I_i^t$$

Where:

- t is the epoch in which the *time lock* of the request expires. If there is no time lock, the epoch in which the RAD request was mined into a block is used instead.
- $\text{rand}(t)$, $\langle \dots \rangle_{\mathcal{M}_i}$, \mathcal{H} , L and I_i^t preserve the same meaning and value as in figure 13 on page 34.
- n is a flag that indicates the type of task (RA or D) and guarantees that no participant is assigned different types of tasks for the same request and epoch.
- \mathcal{R} is the replication factor.

Figure 15: *Task assignment calculation.*

Being this a probabilistic protocol, even if a certain replication factor is specified, it can not guarantee that the number of actual witnesses getting assigned the task will match such factor. For example, for $\mathcal{R} = 5$ the number of witness could well be 0 or 9, but the thing is that most times—and in average—it will be 5.

If the number of witnesses who have discovered their assignment to a certain task is equal or greater than such task's replication factor, all of the witnesses are accepted.

In the same manner, if the number of witnesses who discovered their assignment to a certain task is less than such task's replication factor, all the witnesses are accepted, but the "assignment

contest" is not yet over. Instead, we must find the difference \mathcal{D} between the replication factor and the actual number of designated witnesses and wait for next epoch $t + 1$. As soon as the new epoch begins, a witness \mathcal{M}_i will be considered for "applying for the vacant positions" as soon as it can satisfy the condition from figure 15 with $t = t + 1$ and $\mathcal{R} = \mathcal{D}$. This process will be repeated until the number of witnesses who have committed to the task is equal or greater than \mathcal{R} .

6. TRANSACTIONS

6.1. Transaction Properties

Definition. *Transaction* : A Witnet transaction can be thought of as a pure function $f(s)$ such that, when applied on a valid ledger state s_1 , results in a different valid ledger state s_2 .

$$s_1 \neq s_2, s_1 \in S : f(s_1) = s_2 \in S$$

Definition. *Validity* : A transaction f is valid over state s_1 when s_1 belongs to its domain.

$$f : s_1 \rightarrow s_2$$

Definition. *Independence* : Two transactions f and g are independent over state s_1 when both of them are valid over such state.

$$f : s_1 \rightarrow s_2 \text{ and } g : s_1 \rightarrow s_3$$

Definition. *Dependence* : Transaction g depends on transaction f over s_1 when it is not valid over s_1 but it is valid over $f(s_1)$. That is, the domain of g is the codomain of f .

$$f : s_1 \rightarrow s_2 \text{ and } g : s_2 \rightarrow s_3$$

Definition. *Equivalence* : Two transactions f and g are equivalent over an initial state s_1 when the separate applications of each one of them over the initial state result in the same final state s_2 .

$$f(s_1) = s_2 \text{ and } g(s_1) = s_2$$

Theorem (Completeness). *For any initial state s_1 and final state s_2 , being both different valid states, there exist one and only one transaction f such that when applied on s_1 results on s_2 .*

$$s_1 \neq s_2, \forall s_1 \in S, \forall s_2 \in S : \exists! f \ni f(s_1) = s_2$$

Theorem (Composition). *For any finite set of transactions F , there exist one and only one transaction $g(s)$ that is equivalent to the composition of all the functions in the set.*

$$\forall F = \{f_1, \dots, f_{|F|}\} : \exists! g \ni (f_1 \circ \dots \circ f_{|F|})(s) = g(s)$$

Theorem. *For any initial state s_1 and final state s_2 , being both valid states, there exist an infinite number of finite sets of transactions F such that, when all their member transactions f_n are composed and applied over s_1 , result in s_2 .*

$$s_1 \neq s_2, \forall s_1 \in S, \forall s_2 \in S : \exists^\infty F \ni \prod_{n=1}^{|F|} f_n(s_1) = (f_1 \circ \dots \circ f_{|F|})(s_1) = s_2$$

Theorem (Commutativity). *For any set of transactions F , composition of all its member transactions $f_n \in F$ is a commutative operation if and only if all of its members are independent to each other.*

$$\forall F = \{f_1, \dots, f_{|F|}\} : (f_1 \circ \dots \circ f_{|F|})(s) = (f_{|F|} \circ \dots \circ f_1)(s)$$

Put more simply:

$$f_1(f_2(s)) = f_2(f_1(s))$$

6.2. Types of Transactions and Standard Outputs

There exist at least 4 types of "standard" transactions:

- **Value transfer transactions (VTT).** Roughly equivalent to Bitcoin's P2PKH and P2SH.
- **Client RAD request transactions.** Codified RAD requests. They contain the retrieval paths, the aggregator function and optionally one or many `deliver` clauses.
- **Witness commit-pledge transactions.** Used by witnesses to (1) commit the results of their retrieval tasks without revealing the actual claims, and (2) pledge their portion of the witness reward.
- **Witness reveal-redeem transactions.** Used by witnesses to (1) reveal the actual claims that they committed in their commit-pledge transactions, and (2) redeem their portion of the witness reward.

When sent over the network, all of these types of transactions are encoded using the same serialization format. Also, all the types of transactions use the same output scripting language described in section 6.4 on page 41.

6.3. Fees

6.3.1 Miner fees

Miner fees work just like in other public blockchain protocols: being block space a very limited resource, transactors need to compete for it. Thus, their only means for persuading block miners to include their transactions before others' is to pay a miner fee higher than the rest's.

As seen with Bitcoin, it is to be expected that as long as the block size limit is not reached, the miner fees will be kept low. Then, as blocks start to be full, a transaction backlog is formed and fees start to raise so that superfluous (or even *spam-ish*) transactions are disincentivized and no longer made, which translates to more available disk space and eventually lower fees. Over time,

this cycle ends up striking a balance in which (1) blocks are full, and (2) miner fees only grow at the same pace that demand for block space does.

Since in Witnet’s case miners do not need to perform any extremely expensive operation to mine blocks, they are not urged to arbitrarily reject low-fee or even zero-fee transactions. As mentioned earlier, it is obvious that they will always try to prioritize best-paying transactions. However, as long as all the pending transactions fit in a single block, there is no reason for a block miner to not include them all.

This perfectly matches Satoshi’s vision on miner fees: *The fee the market would settle on should be minimal. If a node requires a higher fee, that node would be passing up all transactions with lower fees. It could do more volume and probably make more money by processing as many paying transactions as it can. The transition is not controlled by some human in charge of the system though, just individuals reacting on their own to market forces*[30].

It is worth remembering that mining is initially subsidized by the Wit generation algorithm as described in section 3.1.4. This is, block miners do not only obtain tokens from the miner fees attached to the transactions they process, but also get a fixed amount for every block they mine—the block reward. As the inflation will decay over time due to block reward being halved periodically, mining will be less subsidized on the long run and miner fees will gain importance.

6.3.2 Witness fees

As mentioned earlier in this work, witness fees calculation is a function of:

- Number of **acquisition paths**, as defined in section 3.2.3.
- Upper bound of **computational complexity** of each of the normalization scripts in the acquisition paths.
- The **replication factor**, \mathcal{R} , required by the client.

This calculation tells the client how expensive will a RAD request for the network to fulfill. As the computational power of the network for every epoch is rather limited, this function offers an absolute metric that is similar to that of transaction size in comparison to block size.

$$fee \propto \mathcal{R} \cdot \sum_{n=0}^{|paths|} \mathcal{O}(paths_n)$$

Figure 16: *Witness fees are proportional to the required replication factor and the total computational complexity of the acquisition paths ($\mathcal{O}(\dots)$).*

The actual cost for witnesses to perform the tasks they get assigned is marginal and virtually negligible when compared to the rewards they get in exchange. The only significant costs involved will be the miner fees they will need to pay to broadcast their commit-pledge plus reveal-redeem transactions and eventually collect their rewards.

For each RAD request, each witness' reward will always equal the request's witness fee divided by the number of witnesses who commit their solutions to the request, which in turn shall be equal or greater than the replication factor.

For these reasons, as long as (1) a witness has spare computational power and bandwidth, and (2) the witness reward is above a certain threshold—greater than twice the miner fees—, there is no reason why a witness node would opt to ignore a single task it got designated for, all the more since such laziness is heavily penalized by the reputation demurrage mechanism introduced in section 4.1.

It is also worth stating that Witnet's reputation scheme somehow turns miner fees into bonuses for which only honest witnesses are eligible. This means that the profit that witnesses receive in exchange for their work does not only come directly from the witness fees that clients pay, but also indirectly from the earned reputation points, which eventually give them a higher chance to mine a block and collect block rewards, including miner fees. Theoretically, this fact could even lead some witnesses to accept tasks at a loss in the expectation that long term block rewards could compensate and exceed the loss.

6.3.3 Bridge fees

Bridge fees follow the same criteria as witness fees.

6.4. Outputs and Scripting

Witnet is conceived to use a scripting language to lock tokens in outputs, akin to Bitcoin's script language[31]. This language—called WitScript—is used to specify under which conditions the tokens locked in a transaction output can be redeemed, and in most cases, by whom. These scripts embedded in transactions is what we commonly call *smart contracts*.

In the light of the successful activation of segregated witnesses (*segwit*) in the Bitcoin protocol and the countless benefits it supposed for the health of the network, we consider that such set of features should be available in the Witnet protocol from day one.

Adopting *segwit* from the very beginning opens the door to supporting multiple smart contract languages that go beyond WitScript, like EVM[32], Michelson[33] or Simplicity[34]. While these languages are still low-level, they are closer to be general-purpose and can be used as compilation

targets for popular high-level smart contract languages, such as Solidity[35] or Ivy[36].

Using different *segwit* version numbers also leaves space for a variety of signature schemes that can offer greater flexibility, scalability and privacy than ECDSA, as is the case with Schnorr signatures[37].

6.4.1 MAST and Tail Call Execution

To provide smart contracts with greater flexibility, WitScript shall support *Merkalized Abstract Syntax Trees* (MAST). This feature, as proposed by Robin et.al[38], Lau[39, 40] and Friedenbach et.al.[41, 42, 43] enables decomposition of complex branching scripts into a set of non-branching flat execution pathways, committing to the entire set of possible pathways, and then revealing only the path used at spend time.

One fundamental part of MAST is *Tail Call Execution*. It changes the behaviour of the output scripts interpreter in such a way that if the memory stack is not clean at the end of script execution, the remaining elements in the stack will (1) be treated as serialized scripts and inputs, (2) immediately executed in a secondary stack, and (3) finally replaced in the main stack by the result of the execution. Different types of *tail scripts* can exist to implement different behaviors.

6.4.2 Covenants

To enable and assure the commit-pledge and reveal-redeem transactions scheme described earlier in section 3.4.2 on page 24, WitScript implements a special type of tail script that, when used to lock an output, forces spending transactions to have only outputs that abide by a certain "template". This feature, called *covenants*[44][45], is a kind of "*forced script heirship*": inputs can cause outputs to inherit a certain policy—in form of script—that the spender can not help but honor.

Covenant clauses are copied in front of all the transaction outputs' redeem scripts in the same order as they appear in the inputs. A transaction spending outputs affected by covenants but missing to pass such clauses to its outputs is considered not valid.

Thanks to this feature, RAD requests can require spending transactions to be commit-pledge transactions that can only be spent by reveal-redeem transactions.

While incredibly useful for many use cases, covenants must be carefully implemented by user-facing client applications such as wallets. These applications must double-check that incoming value transactions are free from covenant clauses before showing them as valid value inputs and updating the user's balance accordingly. Otherwise, a malicious payer could induce a gullible recipient to believe that the value of a certain payment is freely spendable, while in fact it can only be spent in

a certain way predefined by the malicious payer.

6.4.3 Bundled Macros

RAD requests, commit-pledge and reveal-redeem transactions are significantly bigger in bytesize than regular value transfer transactions. In anticipation of a significant part of the limited block space being taken by those types of transactions, WitScript provides a series of macros that implement the most frequent transaction formats so that they take less space when stored or sent over the network.

Every macro is equivalent to a predefined WitScript code. For illustrative purposes, figure 3 depicts usage of a theoretical M_REQUEST macro and its equivalent WitScript program. Please note that this is just an example and the final M_REQUEST macro used in a first implementation of Witnet will not necessarily need to be exactly like this.

Example 3

```

<client_key_hash> <replication_factor> <rad_request_bin> M_REQUEST
-----
<rad_request_bin> OP_DEPLOY
OP_DUP OP_HASH160 <client_publicKeyHash> OP_EQUAL
OP_IF
  10 OP_CHECKLOCKTIMEVERIFY OP_ELSE
  OP_DUP <replication_factor> OP_CHECKMINERVERIFY
  <M_PLEDGE> <COVENANT_FLAG> # These are serialized for tail call execution
OP_ENDIF
OP_DROP OP_CHECKSIGVERIFY
    
```

Figure 17: *The M_REQUEST macro. It allows elected miners to pledge a fraction of the locked funds as long as all outputs in the pledging transaction start with the M_PLEDGE macro. If not pledged after 10 blocks, the original requester can apply for a refund.*

When the WitScript interpreter bumps into a macro, it will not just substitute it by its equivalent WitScript code. Instead, the interpreter will:

1. Pop as many elements from the stack as the number of parameters required by the macro.
2. Run a precompiled function that implements the very same logic than the macro being exe-

cuted, passing the popped elements as arguments to such function.

3. Push the return value of the function to the stack.
4. Keep interpreting the WitScript.

This macro optimization functionality is inspired in the concept of *"jets"* as introduced by O'Connor[34] and Yarvin et.al[46]. Macros not only save block space and bandwidth, but also can make WitScript run faster, as many resource-intensive parts such as signature verification can be delegated to precompiled, optimized and formally verified modules.

7. BRIDGES AND SMART CONTRACTS

7.1. Bridges: Interacting with other Platforms

Bridge nodes, as introduced before in section 1 on page 13, aim to connect Witnet to other platforms. Of special note is the connection of the Witnet network to other public smart contract platforms like Ethereum[32], Decentralized Storage Networks such as Filecoin[25] and directly to the web.

Bridges targetting different platforms also use different values for the n parameter in the Reputation-Based Task Assignment Protocol function as defined by figure 15 on page 36.

7.1.1 Ethereum Bridges

Ethereum bridges are Witnet bridge nodes which also run an Ethereum full node, have full access to the Ethereum blockchain and have the capability to operate with `ether` and make contract calls. Ethereum bridges are in charge of two missions:

- **Requests introduction.** Ethereum bridges monitor the Ethereum blockchain in search for RAD requests codified inside regular Ethereum transactions. When they find one of these transactions, they read the payload and convert it into a valid Witnet RAD request that they must broadcast to the Witnet network. In this scheme, the bridge will act as an intermediate client between the actual client (who is an unknown Ethereum account or contract) and Witnet. In exchange of performing this work and spending their own Wits, bridges are rewarded by the Ethereum clients using `ether` (or any other Ethereum token).
- **Results reporting.** Ethereum bridges are also in charge of reporting the results of those RAD requests which specify Ethereum-targetted `Delivery` clauses to the Ethereum blockchain. In exchange of performing this work and spending their own `ether` by attaching gas to their report transactions, they are rewarded with Wit tokens that were allocated for such purpose in the request transaction.

Ethereum clients will need to make sure they attach enough value to their requests as to reward not only the bridge who will introduce the request into Witnet but also the one who will report the result back the client.

Ethereum bridges are expected to fulfill an important role for adoption of Witnet as they will ease the integration with the thriving Ethereum smart contract ecosystem. However, the economics behind this scheme are relatively fragile as significant fluctuations between the exchange rate of Wit and `ether` tokens may render it unsuitable for long-lived contracts. For this reason—in addition to

cost-effectiveness and lower latency—the preferred way to programmatically transfer value depending on the result of a RAD request will be using Witnet native smart contracts, as explained in section 7.2 on the following page.

7.1.2 Decentralized Storage Networks (DSN) Bridges

Decentralized Storage Networks (DSN) aggregate storage offered by multiple independent storage providers and self-coordinate to provide data storage and data retrieval to clients[25].

DSN bridges provide means for (1) storing the results of RAD requests into DSNs, as well as (2) ensure that those results are persisted in those DSNs in perpetuity.

This scheme and one of its most interesting use cases—ensuring that access to truth and human knowledge will remain democratic forever—are further explained in appendix A on page 48.

7.1.3 Web Bridges

Just like other types of bridges, they act as intermediaries who receive RAD requests from third parties and post them into Witnet.

Web bridges are specially convenient for integrating Witnet with traditional web technologies, as they provide:

- A public endpoint that third parties use to send their requests.
- A public endpoint that third parties use to read the result of their requests.
- A WebHook¹⁶ or EventSource[47] service that notifies clients when the results of their requests are ready.

These two components can be either APIs, user interfaces or both. Note that they are quite similar to block explorers, and it is to be expected that many web bridges will also act as block explorers and vice versa.

Participants running web bridges are free to charge their users for their services in whatever form they want.

¹⁶Webhooks are "user-defined HTTP callbacks". They are usually triggered by some event, such as pushing code to a repository or a comment being posted to a blog. When that event occurs, the source site makes an HTTP request to the URL configured for the webhook. — Wikipedia.

7.2. Native Smart Contracts

The core functionality of Witnet as a DON provides an efficient way to retrieve, attest and deliver verified web contents. This alone is enough to enable the creation of countless novel applications capable of reacting to the outer world without human intervention or relying on single, centralized—thus corruptible or hackable—sources of information.

However, an even broader spectrum of use cases and decentralized applications are made possible if the DON provides a smart contract language so that Wit tokens themselves are programmable and can react to the results of RAD requests.

Witnet implements a smart contract feature that inherits its approach from Bitcoin’s Script. Contracts in Witnet are stateless programs that regulate when, how and by whom certain funds can be spent.

Any WitScript used to lock a transaction output can read the result of a certain RAD request and push it into the execution stack by using the `OP_READ` operation code.

The example 4 depicts a smart contract that, in spite of being really simple, was impossible to implement in a trustless way until now.

Example 4

A transaction output is set to be spendable by one of two different parties, depending on a certain RAD request returning a result value above or under 50,000.

```

OP_HASH160 <RAD_request_id> OP_READ 50000 OP_GREATERTHANOREQUAL
OP_IF
    <participant_a_pubKeyHash>
OP_ELSE
    <participant_b_pubKeyHash>
OP_ENDIF
OP_EQUALVERIFY OP_CHECKSIG
        
```

Figure 18: *Witnet smart contract example.*

Appendices

A. THE DIGITAL KNOWLEDGE ARK

A.a. Motivation

"History is written by the victors."

Winston Churchill

CHURCHILL'S famous dictum may appear to no longer hold true in an age where the Internet wields enormous potential for any person to share their beliefs and opinions with the rest of the world. However, centralized systems for the archiving of human knowledge are still very vulnerable to manipulation or destruction by corrupt governments and other malicious actors who could greatly benefit from altering history.

As a society we have the responsibility to find a better way to preserve our cultural heritage from any odds that the future may hold. And we need it to be highly resilient, decentralized, self-governed and censorship-resistant, guaranteeing that knowledge will be accessible to everyone, everywhere, at any time, without discrimination of any type. Only in this way we will be able to ensure that access to human knowledge will remain democratic forever.

As explained in section 3.3.2 on page 22, verifiability is a weaker notion than truth. In this Information Age, verifiability is indeed very fragile because of the ephemerality inherent to digital media: one statement can be verifiable right now but lose its verifiability right after.

Example 5

Let us visualize how fragile verifiability is:

- 1. Open a random Wikipedia article.**
- 2. Scroll all the way down to the bottom of the page.**
- 3. Follow all the links in the *References* and *External links* sections.**

The chances are that at least one of the links is broken or does not point to the actual content it was supposed to.

In a similar way to Witnet¹⁷, Wikipedia has a strong policy stating that the only valid truth is the one you can verify[48]. Nevertheless, due to the ephemeral nature of the web, even a well docu-

¹⁷See subsection 3.3.2

mented Wikipedia article may be rendered questionable or even not suitable under the Wikipedia policies and guidelines if many of its sources disappeared from the web.

We are clearly in need of some infallible way to ensure that something that is verifiable today will still be verifiable tomorrow.

The current most popular approach to preserving the availability of some data is using the Wayback Machine service run by the Internet Archive initiative to request an on-demand snapshot of any web site in which the data is published. However, centralized solutions offer little to no guarantee of the very ingredients that make contents verifiable:

- Content integrity: equivalence between the content in the snapshot and the actual content published on the web at the time it was taken¹⁸.
- Custody integrity: equivalence between the original content in the snapshot and the one presented when retrieving such snapshot after some time¹⁹.

We would like to emphasize that our point here is not calling anyone's honesty into doubt. Our point is to stress the fact that such a single source of truth, no matter how reliable it is, also represents a single point of failure that introduces the chance for external malicious actors to rewrite or delete part of the history by breaking into a single system or network.

A.b. Knowledge Commons and Commons-based Peer Production

The commons is the cultural and natural resources accessible to all members of a society and held in common, not owned privately. Although the term originally referred to common land, nowadays it is taken to mean any shared and unregulated resource such as atmosphere, oceans, rivers, fish stocks, or even an office refrigerator.

Those resources identified as commons are often said to be vulnerable to social dilemmas[49] and governance problems that lead to competition for use, free riding, commodification, pollution, degradation, and ultimately non-sustainability[50]. These dilemmas are highlighted by the *the tragedy of the commons*.

The concept of *tragedy of the commons* was introduced by ecologist Garrett Hardin in a 1968

¹⁸Assuming that the trusted attesting third party will not misbehave, content integrity will still be broken if attestations are made while the systems or networks of the trusted attesting party are under control of an attacker (malware, DNS spoofing/poisoning, broken TLS encryption, etc). Only viable solution to this problem is performing the attestations by using several "witnesses" in a decentralized way, just like the Witnet miners do.

¹⁹Assuming that the trusted attesting third party will not misbehave, custody integrity can still be broken if the systems or networks of the trusted attesting party are under control of an attacker at any time after the attestation. Only viable solution to this problem is persisting the attested contents in Decentralized Storage Networks (DSN) based on public blockchains, which is the approach of the Digital Knowledge Ark we are proposing here.

article of the same name[51], inspired itself by an essay written in 1833 by the Victorian economist William Forster Lloyd[52].

The tragedy illustrates the argument that free access to a finite resource ultimately damage the resource through over-exploitation, temporarily or permanently. This occurs because the benefits of exploitation make some of the users want to maximize their own use while the costs of the exploitation are borne by everyone. This, in turn, causes demand for the resource to increase, which causes the problem to snowball until the resource collapses.

However, knowledge forms part of a different kind of commons: non-subtractible ones. Unlike environmental commons—e.g. a water spring—multiple users can access the same resource with no negative effect on its quality or quantity. When a teacher gives a lesson, knowledge is not split between the students but replicated across the minds of all of them.

While subtractible and non-subtractible commons are similar in their shared nature, there is a radical difference in the source of their value as resources. The value of subtractible resources is based on scarcity, whilst the value of non-subtractible resources is based on abundance.

Likewise, *preservation* of subtractible and non-subtractible commons involve very different actions. Preservation of a subtractible commons often means guaranteeing its availability by regulating access or imposing²⁰ reasonable use rules on the resources, effectively making it somewhat less open to the public. On the contrary, preservation of a non-subtractible commons means guaranteeing its availability by making it accessible to the greatest number of people and effectively making it more open to the public. Preservation is further discussed in section A.c on the following page.

The culture heritage mentioned in section A.a on page 48 is none other than the *knowledge commons*: the set of all knowledge and wisdom that our civilization has accumulated over the centuries and belongs to the whole of humanity. The knowledge commons is our legacy from the past, what we know today, and what we will pass on to future generations.

The proposed Digital Knowledge Ark aims to form part itself of the *digital commons* as defined by social researcher Mayo Fuster: "*information and knowledge resources that are collectively created and owned or shared between or among a community and that tend to be non-exclusive, that is, be (generally freely) available to third parties. Thus, they are oriented to favor use and reuse, rather than to exchange as a commodity. Additionally, the community of people building them can intervene in the governing of their interaction processes and of their shared resources*"[53].

The proposed Digital Knowledge Ark intends to leverage the current profusion of emerging blockchain technologies and projects to engage the people in the enrichment and preservation of the knowledge commons. It is conceived as a commons-based peer production initiative as defined

²⁰In many cases, *self-imposing*.

by professor Yochai Benkler: "*collaboration among large groups of individuals [...] who cooperate effectively to provide information, knowledge or cultural goods without relying on either market pricing or managerial hierarchies to coordinate their common enterprise.*"[54].

A.c. Preservation and its Principles

As mentioned earlier in section A.b, preservation of knowledge commons entails guaranteeing the availability of knowledge resources by making them accessible to the greatest number of people, effectively making those resources more open to the public.

There is one program from the Stanford University Libraries that we would like to acknowledge here for its notable contribution to defining the principles of long-term preservation of knowledge commons. Its name is pretty explicit about what is the cornerstone of its vision: "*Lots Of Copies Keep Stuff Safe (LOCKSS)*"[55].

The preservation principles of the LOCKSS program[56]—which we endorse and make them our own—focus on:

- Decentralized and distributed preservation.
- Preservation of original content.
- Perpetual, guaranteed and seamless access.
- Affordability and sustainability.

The Digital Knowledge Ark proposed here pays special attention to those very same principles, making the most of Decentralized Oracle Networks (DON), Decentralized Storage Networks (DSN) and other blockchain technologies to ensure their effective attainment.

A.d. Using Witnet to Agree on Facts to be Preserved

Anyone interested in storing information in the Digital Knowledge Ark shall:

1. Create a RAD request with one or more valid retrieval paths pointing to the source of such information.
2. Add a deliver clause to the request. This clause will ask for publication in a DSN.
3. Fund the transaction with an amount of Wit tokens enough to reward miners, witnesses and bridges. Fees have been discussed in section 6.3 on page 39.
4. Send the RAD request to the network, either directly as a client or through bridge nodes.

Provided that all these points are met, the information will be retrieved and attested by the Witnet DON, and bridge nodes will publish it into the DSN of choice.

As with bridges targetting different platforms, DSN bridge nodes use different values for the n parameter in the Reputation-Based Task Assignment Protocol function as defined by figure 15 on page 36.

A.e. Persisting Facts and Contents in Perpetuity

PERPETUAL storage is possible in one way or another in most existing public blockchains.

For example, Ethereum smart contract can be used to keep data inside their state, which can be simply organized in a key-value mapping that allows entering a new record by calling an external function. The contract must calculate the hash of the data, use the hash as the key for storing the data and return the hash to the sender. Then the sender or anyone else who knows the hash can retrieve the data by calling a constant function using the hash as its sole parameter. Note that although data retrieval is made through a constant function and therefore can be performed an unlimited number of times at zero gas cost for the requesting party, data storage has side-effects (modifies state). This implies that the cost of the storage request increases linearly with the size of the data to be stored. While it can be a very acceptable solution for storing small data units in perpetuity, at scale, storing bigger data units (even in the order of a few kilobytes) becomes really expensive and impractical in most cases.

Instead, archiving of big data units should be done by using decentralized storage solutions specially designed for storage of high volumes of data.

There are already a bunch of promising projects that could be used for that purpose. Among the ones that better fit the requirements of the Ark, five deserve special mention here:

- InterPlanetary File System (IPFS)[57]: a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files.
- FileCoin[25]: a distributed electronic currency whose nodes are incentivized to store as much of the entire network's data as they can, using the IPFS protocol.
- Sia[58]: a platform for decentralized storage in which peers can freely form blockchain-based storage contracts in a free and open market.
- Storj[59]: a peer-to-peer cloud storage network implementing client-side encryption without reliance on a third party storage provider.
- Swarm[60][61]: a decentralized and redundant store of Ethereum's public record, in particular to store and distribute dapp code and data as well as block chain data.

Interaction between the Witnet blockchain and these other networks will be made possible by DSN bridge nodes as introduced in section 7.1.2 on page 46.

None of the aforementioned systems offer perpetual storage per se. Instead, they allow for

establishing the duration of the storage contract. The more is paid, the longer the data will be persisted. The cost for storing a certain amount of data for a defined period of time is driven by supply and demand, and different nodes compete on factors like reliability and price.

Ensuring that a certain data unit is never deleted from the decentralized storage system of choice thus imply recurrent costs. In order to impede deletion of data included in the Ark, interested clients and DSN bridges shall maintain an index that will relate all the archived data to the address of their corresponding storage contract and its date of expiry.

The same clients that originally requested the retrieval of the information and the formalization of a storage contract can keep sending additional tokens to the storage contract to keep it in force. In addition, shall those indexes be publicly available, any other interested party could extend the storage contract by independently funding it.

In essence, as long as there are enough actors interested in maintaining humanity's most relevant digital data preserved in the Ark, we can be certain that access to our cultural heritage and legacy will remain democratic forever.

REFERENCES

- [1] J. Peterson and J. Krug, “Augur: a decentralized, open-source platform for prediction markets,” *CoRR*, vol. abs/1501.01042, 2015. <http://arxiv.org/abs/1501.01042>.
- [2] M. Köppelmann et. al., “Gnosis: Crowdsourced wisdom,” 2017. https://gnosis.pm/resources/default/pdf/gnosis_whitepaper.pdf.
- [3] Anonymous, “Delphi,” 2017. <https://delphi.markets/whitepaper.pdf>.
- [4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2009. <https://bitcoin.org/bitcoin.pdf>.
- [5] P. Sztorc, “Truthcoin: Peer-to-peer oracle system and prediction marketplace,” 2015. <http://www.truthcoin.info/papers/truthcoin-whitepaper.pdf>.
- [6] V. Buterin, “Schellingcoin: A minimal trust universal data feed,” 2014. <https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/>.
- [7] T. Schelling, *The Strategy of Conflict*. Harvard University Press, 1960.
- [8] A. Bendahmane, M. Essaïdi, A. E. Moussaoui, and A. Younes, “The effectiveness of reputation-based voting for collusion tolerance in large-scale grids,” *IEEE Transactions on Dependable and Secure Computing*, vol. 12, pp. 665–674, Nov 2015. <http://ieeexplore.ieee.org/document/6951404/>.
- [9] K. Watanabe, N. Funabiki, T. Nakanishi, and M. Fukushi, “Modeling and performance evaluation of colluding attack in volunteer computing systems,” in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. II, 2012. http://www.iaeng.org/publication/IMECS2012/IMECS2012_pp1658-1663.pdf.
- [10] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, “A reputation-based approach for choosing reliable resources in peer-to-peer networks,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS '02*, (New York, NY, USA), pp. 207–216, ACM, 2002. <http://doi.acm.org/10.1145/586110.586138>.
- [11] L. Xiong and L. Liu, “Peertrust: supporting reputation-based trust for peer-to-peer electronic communities,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 843–857, July 2004. <http://ieeexplore.ieee.org/abstract/document/1318566/>.
- [12] R. H. Porter, “Detecting collusion,” *Review of Industrial Organization*, vol. 26, no. 2, pp. 147–167, 2005. <http://www.jstor.org/stable/41799228>.

- [13] M. Lepinski, S. Micali, and A. Shelat, “Collusion-free protocols,” in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC ’05, (New York, NY, USA), pp. 543–552, ACM, 2005. <http://doi.acm.org/10.1145/1060590.1060671>.
- [14] M. Lepinski, *Steganography and collusion in cryptographic protocols*. PhD thesis, Massachusetts Institute of Technology, 2006. <http://hdl.handle.net/1721.1/38309>.
- [15] A. Shareef, “Collusion free protocol for rational secret sharing,” *IACR Eprint archive*, 2010. <https://eprint.iacr.org/2010/250>.
- [16] F. Araujo, J. Farinha, P. Domingues, G. C. Silaghi, and D. Kondo, “A maximum independent set approach for collusion detection in voting pools,” *Journal of Parallel and Distributed Computing*, 2011. <https://doi.org/10.1016/j.jpdc.2011.06.004>.
- [17] K. Gödel, *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Dover Publications, 1931.
- [18] A. Church, “An unsolvable problem of elementary number theory,” *American Journal of Mathematics*, vol. 58, no. 2, pp. 345–363, 1936. <http://www.jstor.org/stable/2371045>.
- [19] A. M. Turing, “On Computable Numbers With an Application to the Entscheidungsproblem,” in *Proceedings of the London Mathematical Society*, 1937.
- [20] B. Rosser, “Extensions of some theorems of gödel and church,” *The Journal of Symbolic Logic*, vol. 1, no. 3, pp. 87–91, 1936.
- [21] H. G. Rice, “Classes of recursively enumerable sets and their decision problems,” *Transactions of the American Mathematical Society*, vol. 74, no. 2, pp. 358–366, 1953. <http://www.jstor.org/stable/1990888>.
- [22] S. C. Kleene, *Mathematical Logic*. Dover Publications, 1967.
- [23] J. H. Conway, “On unsharable arithmetical problems,” *The American Mathematical Monthly*, vol. 120, no. 3, pp. 192–198, 2013. <http://www.jstor.org/stable/10.4169/amer.math.monthly.120.03.192>.
- [24] D. R. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*. New York, NY, USA: Basic Books, Inc., 1979.
- [25] J. Benet, N. Greco, D. Dalrymple, M. Zumwalt, E. Miyazono, and other contributors, “Filecoin: A decentralized storage network,” 2014-2017. <http://filecoin.io/filecoin.pdf>.
- [26] J. Chen and S. Micali, “Algorand: The efficient and democratic ledger,” 07 2016. <https://arxiv.org/pdf/1607.01341.pdf>.

- [27] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, “Proof of activity: Extending bitcoin’s proof of work via proof of stake [extended abstract],” *SIGMETRICS Perform. Eval. Rev.*, vol. 42, pp. 34–37, Dec. 2014. <https://eprint.iacr.org/2014/452.pdf>.
- [28] P. Daian, R. Pass, and E. Shi, “Snow white: Provably secure proofs of stake.” Cryptology ePrint Archive, Report 2016/919, 2016. <https://eprint.iacr.org/2016/919.pdf>.
- [29] J. Bonneau, J. Clark, and S. Goldfeder, “On bitcoin as a public randomness source,” *IACR Cryptology ePrint Archive*, vol. 2015, p. 1015, 2015. <https://eprint.iacr.org/2015/1015.pdf>.
- [30] S. Nakamoto, “Email from april 2009 to mike hearn,” 2009. <https://bitcointalk.org/index.php?topic=149668.msg1596879#msg1596879>.
- [31] Bitcoin Wiki contributors, “Bitcoin wiki - script.” <https://en.bitcoin.it/w/index.php?title=Script>.
- [32] G. Wood, “Ethereum: a secure, decentralised, generalised transaction ledger,” 2014. <http://gavwood.com/paper.pdf>.
- [33] L. M. G. (Pseudonym), “Michelson: the language of smart contracts in tezos,” 2017. <https://www.tezos.com/static/papers/language.pdf>.
- [34] R. O’Connor, “Simplicity: A new language for blockchains,” 2017. <https://blockstream.com/simplicity.pdf>.
- [35] Ethereum, “Solidity.” <https://media.readthedocs.org/pdf/solidity/develop/solidity.pdf>.
- [36] D. Robinson, O. Andreev, and T. Arcieri, “Ivy: A declarative predicate language for smart contracts,” 2017. <https://chain.com/docs/1.2/ivy-playground/docs>.
- [37] C. P. Schnorr, “Efficient signature generation by smart cards,” 1991. <http://www.mi.informatik.uni-frankfurt.de/research/papers/schnorr.smartcardsig.1991.ps>.
- [38] J. Rubin, M. Naik, and N. Subramanian, “Merkelized abstract syntax trees,” 2014. <http://www.mit.edu/~jlrubin/public/pdfs/858report.pdf>.
- [39] J. Lau, “Bip 0114: Merkelized abstract syntax tree,” 2016. <https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki>.
- [40] J. Lau, “Scripting system in merkelized abstract syntax tree,” 2016. <https://github.com/jl2012/bips/blob/mastopcodes/bip-mastopcodes.mediawiki>.
- [41] M. Friedenbach, K. Alm, and BtcDrak, “Bip 98: Fast merkle trees,” 2017. <https://gist.github.com/maaku/41b0054de0731321d23e9da90ba4ee0a>.

- [42] M. Friedenbach, “Bip 116: Merklebranchverify (consensus layer),” 2017. <https://gist.github.com/maaku/bcf63a208880bbf8135e453994c0e431>.
- [43] M. Friedenbach, “Bip 117: Tail call execution semantics (consensus layer),” 2017. <https://gist.github.com/maaku/f7b2e710c53f601279549aa74eeb5368>.
- [44] M. Möser, I. Eyal, and E. G. Sirer, “Bitcoin covenants,” 2016. <https://fc16.ifca.ai/bitcoin/papers/MES16.pdf>.
- [45] R. O’Connor and M. Piekarska, “Enhancing bitcoin transactions with covenants,” 2017. <https://fc17.ifca.ai/bitcoin/papers/bitcoin17-final28.pdf>.
- [46] C. Yarvin, P. Monk, A. Dyudin, and R. Pasco, “Urbit: A solid-state interpreter,” 2016. <http://media.urbit.org/whitepaper.pdf>.
- [47] Mozilla and individual contributors, “Server-sent events,” 2011. https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events.
- [48] Wikipedia contributors, “Verifiability, not truth — wikipedia, the free encyclopedia.” https://en.wikipedia.org/wiki/Wikipedia:Verifiability,_not_truth.
- [49] B. A. Huberman and R. M. Lukose, “Social dilemmas and internet congestion,” *Science*, vol. 277, no. 5325, pp. 535–537, 1997. https://www.researchgate.net/publication/317903463_Social_Dilemmas_and_Internet_Congestions.
- [50] C. Hess and E. Ostrom, *Understanding Knowledge as a Commons: From Theory to Practice*. The MIT Press, 2007.
- [51] G. Hardin, “The tragedy of the commons,” *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.
- [52] W. F. Lloyd, *Two lectures on the checks to population*. The University of Oxford, 1833.
- [53] M. Fuster, *Governance of online creation communities: Provision of platforms for participation for the building of digital commons*. PhD thesis, European University Institute, 2009. <https://goo.gl/aH7e6B>.
- [54] Y. Benkler, “Coase’s penguin, or, linux and *The Nature of the Firm*,” *The Yale Law Journal*, vol. 112, pp. 369–446, 2002. <http://www.yalelawjournal.org/article/coases-penguin-or-linux-and-the-nature-of-the-firm>.
- [55] L. O. C. K. S. S. program, “What is lockss?” <https://www.lockss.org/about/what-is-lockss/>.
- [56] L. O. C. K. S. S. program, “Preservation principles.” <https://www.lockss.org/about/principles/>.

- [57] J. Benet, “IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3),” 2015.
<https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf>.
- [58] D. Vorick and L. Champine, “Sia: Simple decentralized storage,” 2014.
<https://www.sia.tech/whitepaper.pdf>.
- [59] S. Wilkinson, T. Boshevski, J. Brandoff, J. Prestwich, G. Hall, P. Gerbes, P. Hutchins, and C. Pollard, “Storj: A peer-to-peer cloud storage network,” 2016.
<https://storj.io/storj.pdf>.
- [60] V. Trón, A. Fischer, D. A. Nagy, Z. Felföldi, and N. Johnson, “Swap, swear and swindle: Incentive system for swarm,” 2016.
<http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw^3.pdf>.
- [61] V. Trón, A. Fischer, and N. Johnson, “Smash-proof: auditable storage for swarm secured by masked audit secret hash,” 2016.
<http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/2/smash.pdf>.