Ocean Protocol:

A Decentralized Substrate for AI Data & Services

Technical Whitepaper

Ocean Protocol Foundation[1]

with BigchainDB GmbH[2] and DEX Pte. Ltd.[3]

Version 0.9.2

March 6, 2018

Abstract

This paper presents Ocean Protocol. Ocean is a decentralized protocol and network of artificial intelligence (AI) data/services. It incentivizes for a vast supply of relevant AI data/services. This network helps to power AI data/service marketplaces, as well as public commons data. The heart of Ocean's network is a new construction called Proofed Curation Markets (PCMs). PCMs bridge predicted relevance with actual relevance of each AI service, by combining curation markets with cryptographic proofs (e.g. proof of data availability).

[1] oceanprotocol.com
[2] bigchaindb.com
[3] dex.sg

Contents

1. Introduction

Modern society runs on data [Economist2017]. Modern artificial intelligence (AI) extracts value from data. More data means more accurate AI models [Banko2001] [Halevy2009], which in turn means more benefits to society and business. The greatest beneficiaries are companies that have *both* vast data and internal AI expertise, like Google and Facebook. In contrast, AI startups have amazing algorithms but are starving for data; and typical enterprises are drowning in data but have less AI expertise. The power of both data and AI — and therefore society — is in the hands of few.

Our aim is to equalize the opportunity to access data, so that a much broader range of AI practitioners can create value from it, and in turn spread the power of data. We must also respect privacy needs, which implies we must include privacy-preserving compute. (On-premise) compute also mitigates *data gravity* when the data itself is too heavy to move, and reduces friction for large organizations looking to share their data.

To reduce this to a practical goal, our aim is to develop a protocol and network — a tokenized ecosystem — that incentivizes for making AI data and services available. This network can be used as a foundational substrate to power a new ecosystem of data marketplaces, and more broadly, data sharing for the public good.

The main goal is how to **incentivize towards a large supply of relevant AI data & services**. This is not easy, as there are several challenges:

- How do we (or the network) know what's relevant? Can we even deterministically judge this, or do we need some other means?
- We want to incentivize not only relevant *priced* data but also relevant *public* or *commons* data. The latter is harder because it is free by its nature.
- How do we include / incentivize not only data, but also AI *compute* services? How do we ensure that they can account for privacy? How do we include *decentralized* compute service providers? How do we *guarantee* that the service was actually provided?
- How might we incentivize referrals at both the actor and data levels, i.e. for actors to bring new actors into the system, and share the word about relevant data assets?
- What are the attack vectors and how do we address them? For example, spamming with low-quality data to get many rewards; or "data escapes" where one actor publishes the data held by a different rights-holder.

We have devised a design called **Ocean Protocol** that, we believe, meets these objectives.

The Ocean network is composed of data assets and services. Assets are in the form of data and algorithms. Services are processing and persistence which leverage assets. The assets and services are the commodities made available for consumption via the network, and are similar to those found in any mature data ecosystem.

Ocean has strong incentives to submit, refer, and make available (provably) quality AI data & services, via a new construction that we call a **Proofed Curation Market** (PCM). A PCM has two parts: *predicted popularity* of a dataset/service, and its *actual* popularity:

1. **Cryptographic Proof.** The *actual popularity* is the count of the number of times the dataset/service is delivered or made available. To avoid being gamed, it must be made available in a provable fashion using a cryptographic proof. For example, this may be proof of data availability or a zero-knowledge compute proof.

2. **Curation Market.** This is for *predicted popularity*, a proxy for relevance. The crowd knows much better than designers of Ocean whether a given dataset/service is relevant; so we harness the crowd via a curation market setting. This market can be thought of giving reputation to data/services where the actor must "put their money where their mouth is." They stake to buy "shares" (*drops*) in that dataset/service. The earlier that an actor stakes or bets on a given dataset/service, the more drops they get for amount staked, and in turn the higher the reward.

Only stakeholders provably making high-quality data/services available will be able to reap rewards. Block rewards for a given dataset/service are distributed based on amount of stake in that dataset/service, and its popularity. In other words, PCMs instantiate the goals of *verification* and *virality*.

To our knowledge, Ocean is the first system that explicitly incentivizes people to share their data/services, *independent* of whether it is free or priced. Whoever bets on the most popular data/service (and makes it available) wins the most rewards.

Ocean Tokens are the main tokens of the network, the unit for buying/selling services and for block rewards. We denote Ocean Tokens as "Ọ". We also need Ocean Tokens to measure stake in *each* given dataset/service. For this, we use *drops*. Drops are derivative tokens of Ocean tokens denoted in "Ḍ". For example, 100 drops of stake in dataset X is "100 ḌX". Drops relate to Ocean Tokens via curation markets' bonding curves.

Ocean is a work in progress. Therefore this document should be taken as a *current* conception of what we are targeting with Ocean, with some description of the how. As we continue to develop the technology, we anticipate that there could be changes to the "what" or the "how" of the technology, from the building blocks to specific parameters. So please treat these as initial suggestions and options rather than final choices. When Ocean's public network is live, it should be considered as-is, and the reader should not infer any guarantee that particular functionality described in this whitepaper will be delivered.

The rest of this paper is organized as follows. Sections 2-4 provide context with use cases, stakeholders, and data ecosystem respectively. Sections 5 and 6 describe system structure and behavior. Section 7 describes the heart of Ocean's token design: Proofed Curation Markets and associated block rewards. Section 8 elaborates on curation markets; section 9 on identity and other core blocks. Section 10 elaborates on proofs of service delivery. Section 11 covers outstanding concerns. Section 12 concludes. The appendices discuss FAQs and concerns, design tradeoffs, and descriptions of some verifiable computational & data integrity services.

2. Use Cases

These use cases and others guide our design.

2.1. Proprietary Data: Autonomous Vehicles

A leading use case for proprietary data is autonomous (self-driving) vehicles.

The RAND Corporation calculated that 500 billion to 1 trillion miles driven are needed to get AI models accurate enough for production deployment of self-driving cars [Kalra2016]. Our collaborators at Toyota Research Institute (TRI) saw that it would be prohibitively expensive for each automaker to generate that much data on its own. Why not pool the data, via a data marketplace? With them, we built such a prototype [BigchainDB2017].

Then the challenge is, a single data marketplace may itself be centralized: we arrive at another data silo. We need a substrate that enables *many* data marketplaces to emerge. This is a key goal of Ocean Protocol. Critical new benefits emerge: higher liquidity for each marketplace, and organizations are directly incentivized to pool data rather than silo it.

Self-driving car training data illustrates how not all data is fungible: a mile driven in a blizzard is worth more than a mile driven on an empty, sunny desert highway. But one mile in the blizzard is fungible with other miles in blizzards. The system must account for both fungible and non-fungible data.

2.2. Regulated Data: Medical Research

This is a leading use case for data that must follow data protection regulations in support of privacy; and therefore it will need privacy-preserving AI compute services.

DEX Pte. Ltd. ("DEX") is working with ConnectedLife [ConnectedLife2018], Medical Researchers at the National Neuroscience Institute of Singapore, Specialist Professionals and Hospital Groups in Singapore, Germany, and elsewhere towards an objective measurement of the symptoms of Parkinson's Disease. The goal is to build subject specific and generalized models based on patient bio-medical and free-living sensor data. However, ethical and national personal data protection laws prevent patient data from being copied and shared without considerable transformation of the data taking place and thereby removing much of the value and potential impact in-terms of patient data driven applications. A data marketplace makes it easier to connect the data suppliers; and it must be decentralized to avoid the siloing issue. This provides us with an excellent use case for privacy-preserving compute.

2.3. Global Data Commons

Our vision is to grow a massive set of data assets, all free for the planet to use. We've seen glimpses of the power of this. For example, ImageNet is an open dataset with over 10 million tagged images—much larger than previous open image datasets. It has allowed AI researchers to train image classifiers with radically less error than before, for dozens of computer vision applications [ImageNet2018]. There are several other open data efforts; unfortunately each is siloed with little incentive to create more current, valuable data/information and share among them. Directly incentivizing data sharing can address this.

3. Stakeholders

Understanding network stakeholders is a precursor to system design. Table 1 outlines the stakeholders participating in the network. There are stakeholders beyond, from developers to auditors, but that is outside the scope of this paper.

Table 1: Key stakeholders in Ocean ecosystem

Stakeholder	What value they can provide	What they might get in return
Data/service provider, data custodian, data owner	Data/service (market's supply)	Ocean Tokens for making available / providing service
Data/service referrers, curators. Includes exchanges and other application-layer providers.	Data/service (via a provider etc), curation	Ocean Tokens for curating

Data/service verifier. Includes resolution of linked proofs on other chains	Data/service (via a provider etc), verification	Ocean Tokens for verification
Data/service consumer	Ocean Tokens	Data/service (market's demand)
Keepers	Correctly run nodes in network	Ocean Tokens for chainkeeping

4. Ocean as a Data Ecosystem

4.1. Decentralized Data Hub

We can draw on the experience and technology of existing centralized data ecosystems, which power modern enterprise applications, large-scale machine learning, data analysis, and more. These ecosystems combine many technologies, such as: mainframe systems, operational data stores, enterprise service and data buses, data warehouses and data lakes, ETLs (Extract, Transform, Load tools) and ELTs, distributed and in-memory computing, APIs and web services, over-the-counter consumption tools and web applications.

Traditional **Enterprise Data Hubs (EDHs)** have the following capabilities:

1. **Source** – The exposure of available initial data assets.
2. **Ingestion** – Help onboard data assets into the ecosystem.
3. **Processing** – Transform, normalize, and consolidate assets, including cleansing, normalization, and consolidation.
4. **Persistence** – Store the assets for use.
5. **Consumption** – Use the assets.
6. **Discovery** - Finding the assets.
7. **Governance** – Implement the ecosystem's governing rules, including crypto-conditions.

Each service incorporates one or more of these capabilities. For instance, an ETL service incorporates ingestion and processing, while a Spark distribution [Spark2018] incorporates processing and some finite in-memory persistence.

Ocean will support these capabilities, in a *decentralized* fashion. Therefore it is a **Decentralized Data Hub (DDH)**. Figure 1 illustrates. On top of these will be myriad marketplaces for data/services, both centralized and decentralized.

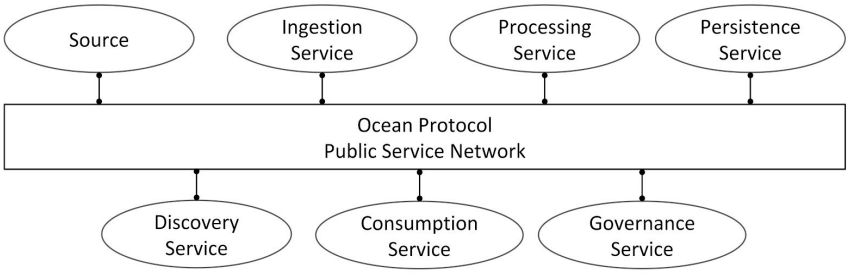

Figure 1: Ocean Protocol is a Decentralized Data Hub

4.2. Decentralized Data Pipeline

Orchestration of services in an EDH is handled by data pipelines. A data pipeline consists of control and data flows that manage system interactions across services. Ocean will facilitate such functionality between decentralized services.

Figure 2 shows an example of Ocean fulfilling some of these capabilities in a decentralized data pipeline going from left to right.

The **source** is a log stream or log files. These get **ingested** into a message queue. Then there is **processing**, which could be centralized (e.g. EC2 [Amazon2018c] or Lambda), decentralized (e.g. Golem [Golem2016], iExec [iExec2017]) and possibly with special capabilities like AI (e.g. SingularityNET [SingularityNET2017]) or privacy (e.g. Enigma [Zyskind2015]).

The next step is **persistence**, which could be blob stores or databases, and centralized or decentralized. For example, AWS S3 [Amazon2018b] is a centralized blob store, IPFS/Filecoin [IPFS2018] [Filecoin2017] and Ethereum Swarm [Trón2018] are decentralized blob stores, AWS Aurora [Amazon2018] and MongoDB Atlas Amazon Aurora [Amazon2018] and MongoDB Atlas [MongoDB2018] are centralized database services, and BigchainDB [BigchainDB2018] and OrbitDB [OrbitDB2018] are decentralized databases.

Finally, the data asset is consumed by a human looking at a dashboard, or by software in the form of Webhooks [Webhook2018], IFTTT or other callbacks technology.

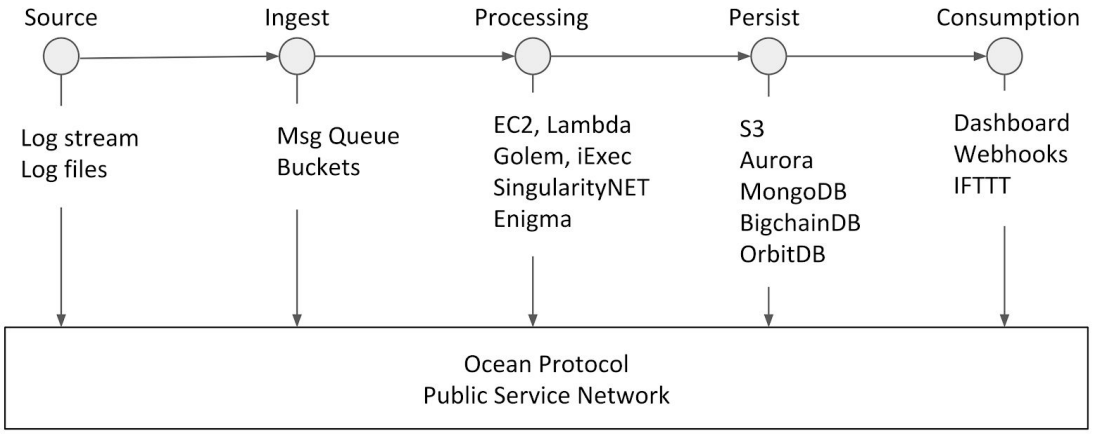

Figure 2: Ocean decentralized data pipeline

Figure 3 illustrates Ocean in a data pipeline incorporating privacy-preserving compute, using proxy re-encryption (e.g. with NuCypher [NuCypher2018]) and multi-party compute (MPC)(e.g. with Enigma [Zyskind2015]). It also shows the use of redundancy, which is a key feature of most distributed systems storage technologies.

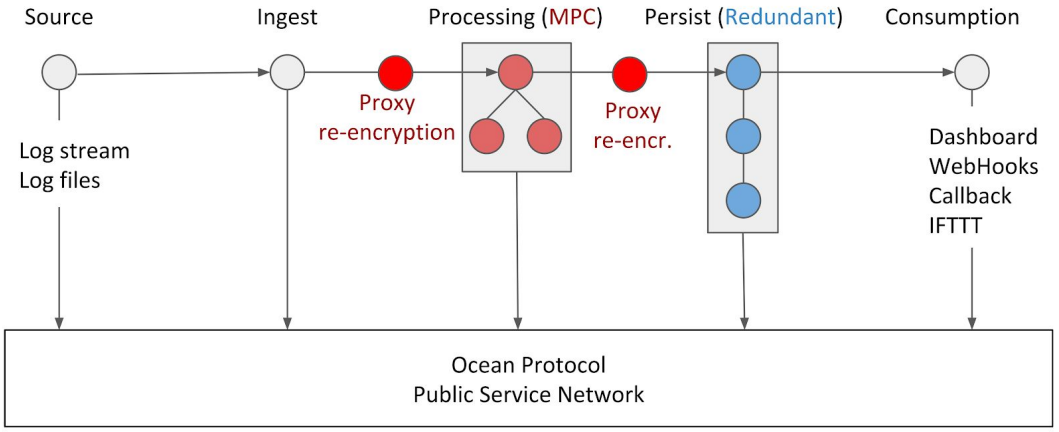

Figure 3: Ocean decentralized data pipeline with privacy and redundancy

4.3. Interservice Connections

To link to the decentralized services, we need **adapters** (bridges) for two things:

- **Value / liquidity** - to pay for the network services, we need to exchange Ocean Tokens to this network's native token. Infura exchange [Infura2018], and 2-way pegs like Interledger [Thomas2015] and Cosmos [Kwon2017] can help.
- **State / data** - there is a proof that a request that has been handled. If that proof has been verified then it's satisfied Ocean's needs. PolkaDot for state [Wood2016], Truebit for proofs [Teusch2017], and IPLD [IPLD2018] for content-addressed data can help.

Figure 4 illustrates.

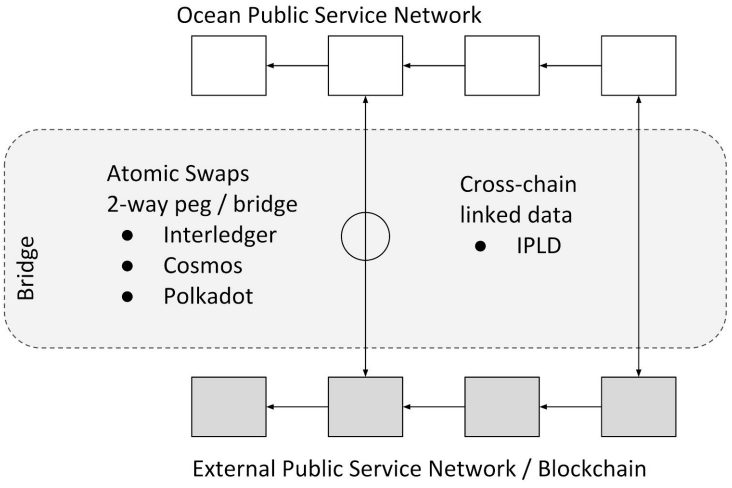

Figure 4: Connecting Ocean with other decentralized service networks

5. System Architecture

Figure 5 shows the overall architecture. At the top are stakeholders: data/service providers (including data custodians and owners) and verifiers, data/service consumers (most notably, AI experts), data/service marketplaces, and data commons interfaces.

Providers. These actors have AI data or services that they make available in a cryptographically provable fashion. Services may include: data itself, storage (centralized or decentralized), compute

(centralized or decentralized, privacy-preserving or not), and more.

Marketplaces. Data/service marketplaces are typically how providers and consumers interact with Ocean network, for convenience. Each marketplace is expected to facilitate:

1. Discovery - The ability to identify, promote, and curate assets or services within an ecosystem;
2. Transactability - The ability to reach transactional agreement between ecosystem stakeholders, facilitated by Ocean Tokens; and
3. Verification - The ability to verify that transactions were sufficiently completed.

To catalyze marketplaces for the community, we are building a reference data marketplace with a permissive open source license [OceanMkt2018].

While marketplaces will have their own approaches to pricing, but for discoverability, liquidity, and clearing, Ocean itself will store the pricing information.

Data commons interfaces. Side-by-side with data marketplaces that serve priced data are interfaces for data commons, for *free* or commons data. These interfaces might be webpages, software libraries, and so forth.

Keepers. The Ocean network itself is composed of a set of Ocean keeper nodes[4]. Keepers collectively maintain the network. Anyone can run an Ocean keeper node; it's permissionless. Participation is open and anonymous. Of course, just as in Bitcoin, higher layers like marketplaces and service providers can include their own identity and permissioning requirements.

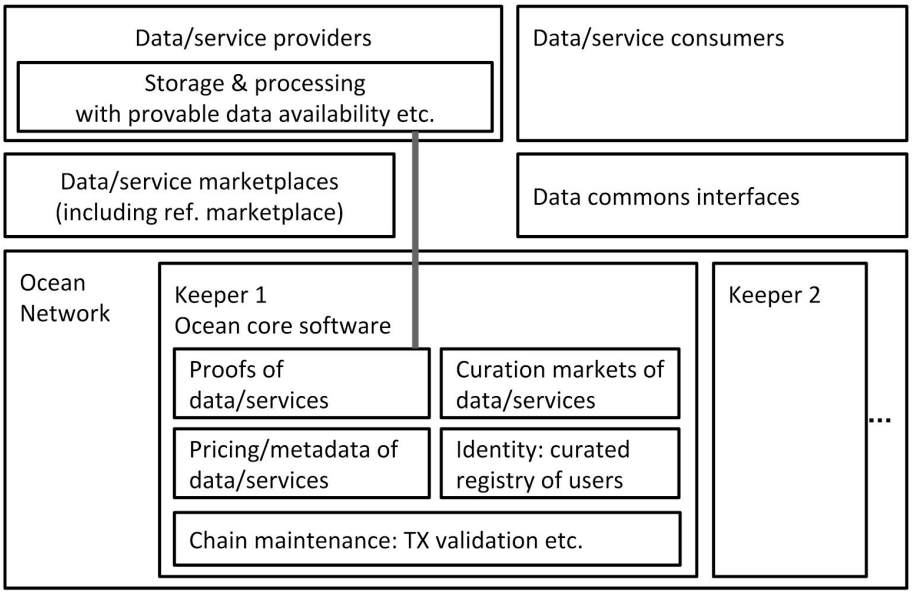

Figure 5: Ocean architecture

Each keeper node runs Ocean core software that speaks the Ocean protocol. When we say "core

[4] We prefer "keeper" [Zurrer2017] over "miner" as mining implies a proof of work; Ocean is not constrained to simply proofs of work.

software" we mean any correct implementation of the protocol. It has these key parts:

- **Proofs of data/services.** Tie-in to the storage & processing with provable data availability, etc. This is making sure that the keeper actually made a data/service available like they claimed they did. It's accomplished via cryptographic proofs, which we elaborate on later. Data blobs themselves may be stored on-premise, on the centralized cloud, or on the decentralized cloud. On-premise storage may pair with on-premise processing; in which case only the result of the processing is made available to the data/service consumer.
- **Curation markets.** This is a list of available data/services, with reputation for each in the form of a curation market. Curation markets combined with cryptographic proofs gives a new construction - Proofed Curation Markets - which bind predicted and actual popularity.
- **Pricing/metadata.** This is how much the provider asks for access of the data/service (fixed price, auction, etc) or whether it is free; in addition to other metadata. IP rights information is stored using [COALAIP2018], a blockchain-ready IP protocol. The Ocean network itself does little with this information; its goal is to make the information available to higher-level marketplaces, enabling discovery across all marketplaces.
- **Identity.** This is a whitelist of good actors, implemented as a Token Curated Registry of users [Goldin2017][adChain2017]. New members join with stake; if they act maliciously (as voted by the list) they lose stake and are removed. This whitelist is needed to avoid particular attack vectors, as elaborated in the section on identity.
- **Chain maintenance.** Because the Ocean network is a blockchain, it needs maintenance logic like validating transactions, storing Ocean Tokens, storing metadata (with links to services/assets), and more.

6. System Behavior

6.1. Overview

Figure 6 shows an overview of the system behavior. The stakeholders are grouped as follows.

- Top: **Client** is the data/service consumer
- Middle: **Services** include data/service providers/owners, referrers/curators, and verifiers.
- Bottom: **Keepers** are keepers (miners) of the blockchain network.

Each arrow has a label that describes a particular action between two stakeholders in the system. Let us go through these, left to right. The **publish** function is when an actor onboards a new dataset or service into the network. Publishing includes providing the dataset/service's metadata and making it globally accessible[5].

After the dataset/service is onboarded, actors can play **curator** in order to **stake**, thus indicating their confidence in the relevance of a dataset/service. Staking also provides a **signal** to help clients **query** and **discover** relevant services, typically inside **markets**.

When a client discovers a service they want to consume, they agree upon a **contract** by virtue of a market function. The **service** provider gives **access** to the service, providing a **proof** it was fulfilled.

[5] Albeit potentially with permissioning to reconcile privacy needs.

Finally, the service contract is enforced by the **verifier** doing the cryptographic **verification**. Block rewards are then distributed appropriately.

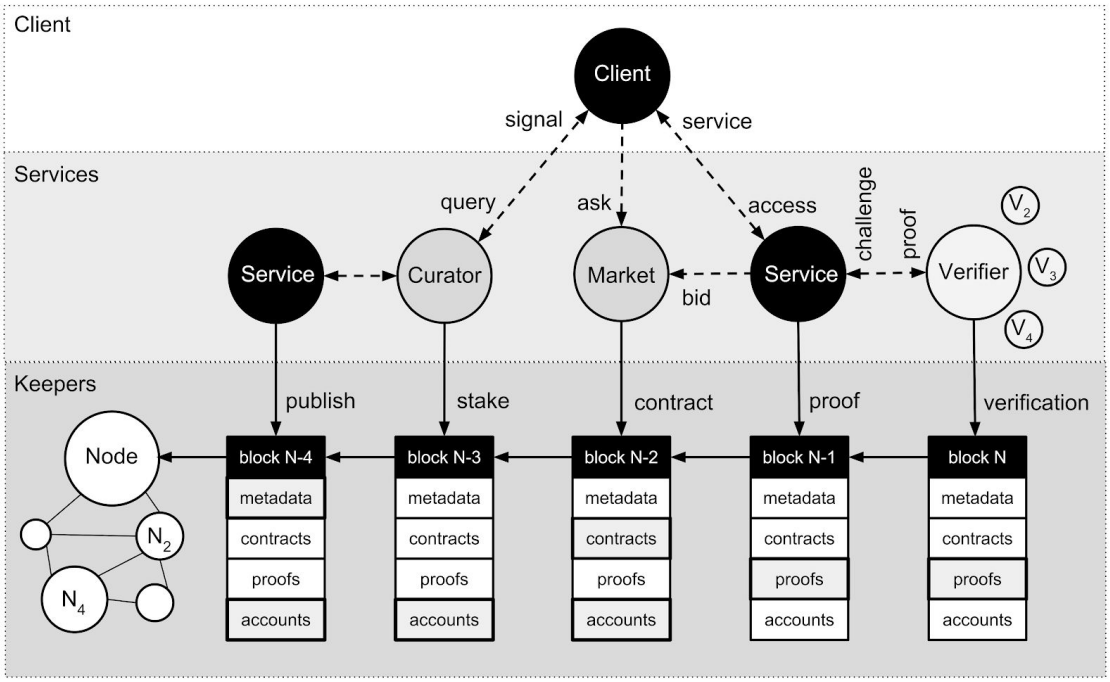

Figure 6: Overview of protocol functions

6.2. Service Delivery Protocol

Let's drill into the core function: setting up and delivering a service[6]. Here, we use smart contracts as programmable **service agreements**. Figure 7 depicts the protocol. It proceeds as follows[7].

1. **Contract Setup** - The service agrees and both parties come to a service agreement that is programmable and enforceable
    a. The client discovers a service that it wants to consume.
    b. The service provider (or client) sets up a service agreement including conditions related to settlement (fees and warranties), as well as resolution (execute and abort conditions).
    c. The client agrees with the contract by locking up Ocean Tokens in the escrow function of the contract.
    d. (Optionally) A set of verifiers with the capability to verify the service integrity proofs are chosen and allocate resources.
    e. The contract is digitally signed and deployed on-chain.
2. **Access Control and Consumption** - The client requests access to the service and consumes the service
    a. The client connects to the service provider and authenticates.
    b. The service provider authorizes the client based on the on-chain contract.

[6] For the data side, we treat data availability as a service.
[7] This is one possible order of events. In other cases, consumption may happen after verification, for example to avoid accidental consumption of bogus data.

c. Upon successful authorization, the client is granted access and consumes the service.

3. **Verification and Settlement** - The service consumption is verified and the contract is settled
   a. A pre-appointed verifier and/or the client challenges the service to provide proofs that the requested service is delivered according to integrity specifications.
   b. The service accepts the challenge, computes the proof and stores this on-chain with a reference to the contract.
   c. The verifiers validate the proof and optionally send out new challenges to the provider.
   d. Once enough proofs are provided, the contract goes into settlement:
      i. Upon correct verification of the proofs, the transfer of funds from client to provider is finalized; or,
      ii. Upon provable error or timeout, the funds are rolled back and optionally the provider is penalized by slashing stake.
   e. Block rewards are dispensed accordingly.

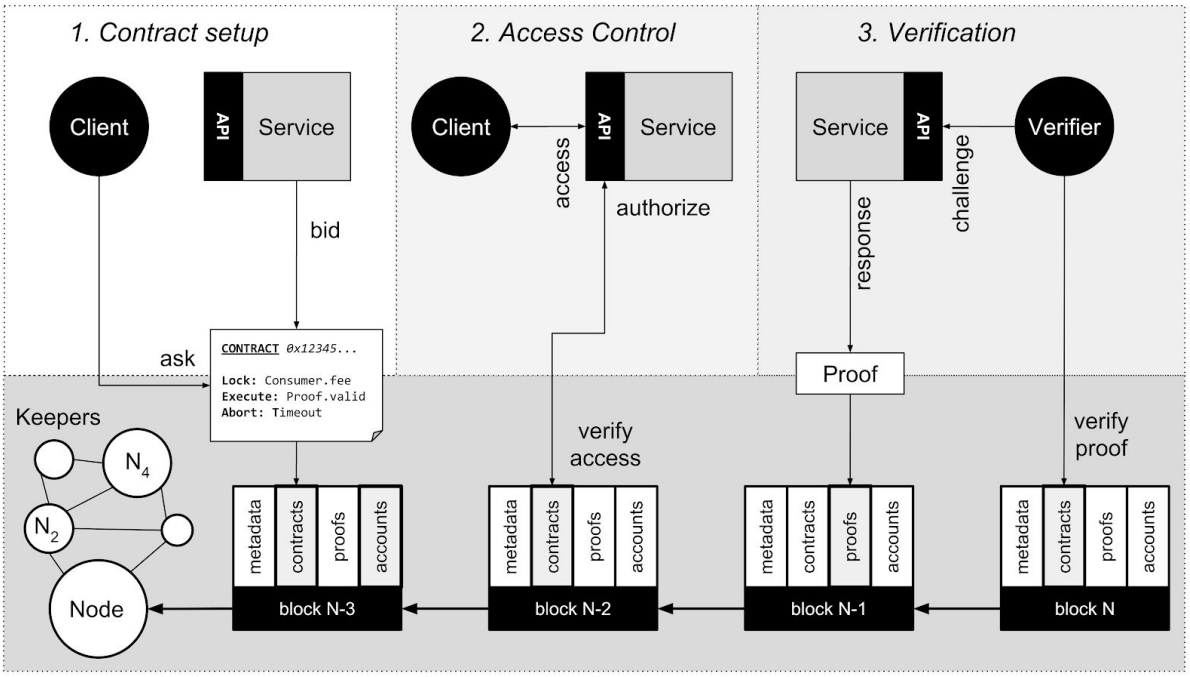

Figure 7: Service delivery protocol

6.3. Service Agreements

A marketplace on Ocean brings together multiple service consumers and providers. All participants in the transaction are exposed to certain risks - the provider may not get paid and the consumer may not get the expected service. However, if a description of the service is encoded in a programmable service agreement on a public trust layer, it can be enforced. Within the service agreement, the payment, settlement, and service parameters are specified. Parameters include access tokens, timeouts, service-specific proofs, and verifications. Figure 8 helps illustrate this.

Lock: Ocean Protocol employs a two-phase contract, similar to a hold or escrow. First, funds are locked up. This means that the funds are pending until a resolution of the contract has been met. Second, the service is executed or aborted. If executed, funds are released to the provider. If

aborted, funds are returned to the consumer. Additionally, warrantees or stake can be provided from the service provider side to further enforce the trust.

Resolve: Keepers determine whether a service agreement is executed or aborted. Keepers use verification of service proofs to resolve the contract. If no verified proofs can be provided within a specific timeout (measured in blocks), then the abort condition can be triggered and funds refunded, optionally with a forfeit of any warrantee or stake of the provider. Of course, some fees are due for the keepers and verifiers to run the contract and verify the proofs.

```
Lock:
     Consumer.fee // pay for the service
     Provider.warranty // put some stake

Execute:
     proof.is_valid() // verify service proof

Abort:
     Timeout // in blocks
```

Figure 8: Pseudocode for a service contract

The contracts are programmable and allow consumers and providers to express complex scenarios. However, complex contracts require more execution time on the keeper nodes as well as potentially more dedicated verifiers for specific proofs. For example, a service that wants to provide a combination of computation and storage would require proofs for both compute and storage with their own verification or challenge-response mechanism.

6.4. Access Control

The Ocean Service Agreements are a starting point for a consumer to gain access to a service. Authorisation of the requestor to access a service occurs by verifying that the requestor has been granted a capability such as HTTP access (GET, PUT, ...), a specific range of queries (SQL, noSQL), or even secure modes of interaction (MPC, TEE, sHSM, ...). The capabilities are stored in the service agreement and these are queried via a call from the service upon verification of the authorization request [Smolenski2017].

7. Core Token Design: Proofed Curation Markets

7.1. Introduction

The previous sections provided context to Ocean, and gave high level overview of Ocean's behavior and structure. This section describes the core token design. At its heart is a block rewards function (objective function) implemented by a **Proofed Curation Market.**

Recall that Ocean's main goal is: **maximize the supply of relevant AI data & services**. This drives the beating heart of Ocean - the **block reward function**. It acts similarly to an **objective function** in the optimization literature, for example as used in [McConaghy2009]. Optimization can be framed as a subset of mechanism design [Evans2017]. This objective function is what Ocean is optimizing towards, by incentivizing actors in its ecosystem to contribute to. Bitcoin rewards contribution to hash rate with Bitcoin tokens; Ocean rewards contribution to relevant data/services with Ocean Tokens.

Besides the main goal, we had several **ancillary questions / goals** that guided us in token design. Early designs did not meet them. As we will see later, the chosen design does. They are as follows:

- For priced data, is there incentive for supplying more? Referring? Good spam prevention?
- For free data, is there incentive for supplying more? Referring? Good spam prevention?
- Does it support compute services, including privacy-preserving compute? Do they have incentives for supplying more, and for referring?
- Does the token give higher marginal value to users of the network versus external investors?
- Are people incentivized to run keepers?
- Is it simple?
- Is onboarding low-friction?

7.2. Block Rewards to Incentivize Relevant Data/Services & Make It Available

Block rewards are the key tool to incentivize desired behavior, i.e. to "get people to do stuff" [McConaghy2018]. Ocean emits Ocean Tokens as block rewards.

We want Ocean to have strong incentives to submit, refer, and make available quality data/services. To accomplish this, we introduce Proofed Curation Markets, which combine (a) cryptographic proofs that the data/service was made available, with (b) Curation Markets [Rouviere2017] for reputation of data/services. It uses stake as a measure of the belief of the future popularity of the data/services, where popularity is measured by number of times that service is made available. Block rewards for a dataset/service are a function of how much an actor has staked in that dataset/service, the dataset/services's actual (proofed) popularity, and the actor's serve-versus-use ratio.

We now elaborate. First we describe an ideal token allocation approach; then we describe a practical implementation.

Here is the ideal allocation approach, i.e. the approach assuming no computational constraints. R_{ij} is the block rewards for actor *i* on dataset/service *j*, *before* being normalized across all actors and datasets/services. The actual block rewards received are normalized: $R_{ij,norm}$.

$$R_{ij} = log10(S_{ij}) * log10(D_j) * R_i$$

$$R_{ij,norm} = \frac{R_{ij}}{\sum_i \sum_j R_{ij}} * T$$

where

- S_{ij} = actor *i*'s stake in dataset/service *j*, measured in *drops*.
- D_j = number of deliveries of dataset/service *j* in the block interval
- R_i = global ratio for actor *i* serving up vs. accessing a dataset; details are below
- T = total Ocean Tokens given during the block interval according to the overall token reward schedule (see Appendix)

The first term in R_{ij} is $log10(S_{ij})$. It reflects the actor's belief in the popularity of the dataset/service, measured in drops. If the actor that posts the data/service believes that it will be popular, then they can stake even more than the minimum posting amount, curation-market style, and receive more drops. Additionally, others that believe in the future popularity of the data/service can stake whatever they would like, curation-market style, and receive drops. These mechanics incentivize participants to submit relevant datasets/services, and gives them an opportunity to make money too. We use log10 on curation market stake (drops) to level the playing field with respect to

token whales; and so that token whales are incentivized to make a greater number of datasets/services available. This has theoretical roots in Kelly Betting: applying the log is the optimal strategy for an individual to maximize their utility in betting [KellyCriterion2017] [Simmons2017].

A later section elaborates on curation markets including stake in drops; and another section on how we manage identities to prevent stake whales from creating multiple accounts).

The second term, $log10(D_j)$, reflects the popularity of the dataset/service; that is, how many times it has been (provably) used in the time interval. We use log10 to incentivize actors to stake and make a greater number of datasets/services available.

The first and second term can be summarized as a binding of *predicted* popularity * *actual* popularity. This is the core mechanic of a Proofed Curation Market.

The third term, R_i, is to mitigate one particular attack vector for data (it's excluded for services). "Sybil downloading" where actors download files repeatedly to increase their block rewards (more on this later). It uses a tit-for-tat approach like BitTorrent by measuring how much data an actor has served up, versus how much is accessed, as follows:

$$R_i = \{min(B_{served},\ B_{downloaded}),\ 1.0)\ if\ all\ data\ assets\ served;\ 0.0\ otherwise\}$$

where

- B_{served} = total number of bits that the actor served (made available) across all data assets they have staked
- $B_{downloaded}$ = total number of bits that the actor accessed, from any data assets

If an actor has staked on a data asset and they want to get rewarded, then they must run a keeper node that makes that data asset available. If they don't make it available when asked (or fail on other keeper functionality), they will lose their stake in that data asset. It's ok if they retrieve it last-minute from S3 or another miner; it's more reward as a CDN (content delivery network) [CDN2018] as opposed to proof of storage like Filecoin [Filecoin2017].

For an early staker in a data/service that has since had more stake, they can subsequently pull out their stake at a profit, curation-market style.

It's worth emphasizing: when we say "stake" for that dataset/service, we mean the amount it's worth in terms of the derivative token for that dataset/service, called "drops". A later section elaborates.

7.3. Block Rewards: Practical Implementation

To implement the block rewards as described above has complexity and high compute cost because, for each block rewards cycle, we need to compute the amount of stake for each dataset/service made available by each actor, and we'd need a transaction to *each* actor to reward their effort.

We can address these issues by giving keepers the same *expected* value of block reward (though higher variance), with less computation using a Bitcoin-style strategy (called "probabilistic micro-payments" in [Salomon2017]). In Bitcoin, every ten minutes, tokens (Bitcoins) are awarded to a *single* keeper (miner) where the probability of reward is proportional to value added (miner hash rate), compared to the rest of the network's value added (network hash rate = network difficulty). Network difficulty is updated every two weeks.

Ocean is similar. Rather than rewarding at fixed time intervals, every time a keeper makes a dataset/service available to a consumer, Ocean randomly chooses whether to give block rewards. The amount awarded is based on the value added by the keeper R_{ij} and total network value added. $R_{difficulty}$ is the network difficulty; it gets updated every two weeks (20160 minutes)[8], i.e. the difficulty interval. R_{recent} is the value added since the last difficulty update.

At network launch, $R_{difficulty} = 0.$ At the beginning of each difficulty interval, $R_{recent} = 0$.

Here's what happens when actor *i* makes a dataset/service *j* available to a consumer.

1. Compute value added:

$$R_{ij} = log10(S_{ij}) * log10(D_j) * R_i$$ [9]

2. Update total recent network value added:

$$R_{recent} = R_{recent} + R_{ij}$$

3. Compute the probability of getting a block reward, *P*. If we wanted one reward on average every two weeks, it would be (1). But let's have rewards every 1 minute on average. 20160 minutes is two weeks. So, we add in the factor (20160 minutes)/(1 minute). The result is (2).

$$(1) \quad P = \frac{R_{ij}}{R_{difficulty}}$$

$$(2) \quad P = \frac{R_{ij} * 20160/1}{R_{difficulty}}$$

4. Compute whether actor *i* gets the reward:

u ~ U[0,1], i.e. draw a random real number between 0.0 and 1.0, using e.g. [Randao2018][Syta2017]

If $u \geq P$ then actor *i* will get the reward

5. If the actor *i* is to get the reward, then compute $reward$ and give it, via a transaction with output to actor i. Since step 3 has a bias to reward more often using the factor (20160/1), here we need to divide the amount awarded by that same factor. We arrive at *F*, the fraction of rewards for this action in this difficulty interval. To compute $reward$, we scale *F* by $T_{difficulty}$, where $T_{difficulty}$ is the total Ocean Tokens given during the two week difficulty period according to the overall token reward schedule (see Appendix).

$$F = \frac{R_{ij}}{R_{difficulty} * 20160/1}$$

$$reward = F * T_{difficulty}$$

Once every difficulty interval (two weeks), the difficulty will be updated with R_{recent}. The change is limited to 0.5x to 2.0x of the previous difficulty value.

[8] This parameter, like many parameters in Ocean, are subject to change.

[9] We actually wrap each log() expression with a max to avoid negative values. E.g. max(0, log(S_{ij}))

$$R_{difficulty} = max(0.5 * R_{difficulty},\ min(2.0 * R_{difficulty}),\ R_{recent})$$

7.4. Separation of Roles vs. One "Unified" Keeper

In designing the system, we wanted to incentivize the stakeholder roles of data/service provider, referrer, validator, and keeper. Some initial designs gave a percentage of block rewards to each role based on their respective actions. But this opens up attack vectors, such as keepers taking all the rewards for themselves. Our solution was to explicitly couple all the roles into one: if you've staked (provider or referrer) then the only way to get block rewards is to run a keeper node.

If we discover alternatives to overcome the security concerns, the final implementation may separate these into distinct roles.

8. Curation Markets Details

8.1. Introduction

Recall that Ocean's objective function (block reward function) is to maximize the supply of relevant AI data/services. Ocean uses curation markets [Rouviere2017] to signal how relevant an AI dataset or service might be. Curation markets leverage the wisdom of the crowd: people stake on datasets/services they believe in. In other words, they put their money where their mouth is. In traditional curation markets, the main action for actors is stake and un-stake as a means of signaling. Ocean builds on this by binding those staking actions with actual work of making a service available - Proofed Curation Markets. This section elaborates on curation markets.

Each dataset/service has its own curation market, which in turn has its own token called **drops,** and a **bonding curve** that relates drops to Ocean Tokens "Ọ".

8.2. Tokens for Services: Drops

Let's elaborate on drops first. Recall that drops are derivative tokens of Ocean Tokens denoted in "Ḍ" that measure stake for *each* dataset/service. For example, 100 drops of stake in dataset X is "100 ḌX". Users can get value from drops in two ways:

1. *Block rewards.* People earn Ocean Tokens if they bet on an AI dataset / service and make it available when asked.
2. *Un-staking.* One can un-stake in order to convert from Ḍ in a service back to Ocean Tokens.

Drops are a measure of a user's attention: if a user cares about dataset X, the user will stake on dataset X to get drops of X; that is, ḌX. Because there is scarcity of Ocean Tokens, there is scarcity of drops, which mirrors a user's scarcity of attention. In short, ḌX are a proxy for mindshare in X.

Because each dataset/service has its own token, a user of Ocean will likely hold not just Ocean Tokens in their crypto wallet; they may also hold ḌX, ḌY, or in general a variety of drops for the datasets and compute services that they've staked.

8.3. Bonding Curves

A **bonding curve** relates a token's drops "Ḍ" to Ocean Tokens "Ọ" for a given dataset/service. Figure 9 shows a bonding curve for dataset X. It relates the **price in Ọ to buy more drops of X** (y-axis) as a function of the **current supply of drops** (x-axis). As people stake more interest in X, its ḌX supply goes up according to the bonding curve.

Bonding curves can take whatever shape the creator wishes. But to reward early adopters, a bonding curve typically makes it more expensive to buy ḌX as more people stake in it; this is the positive slope in the curve.

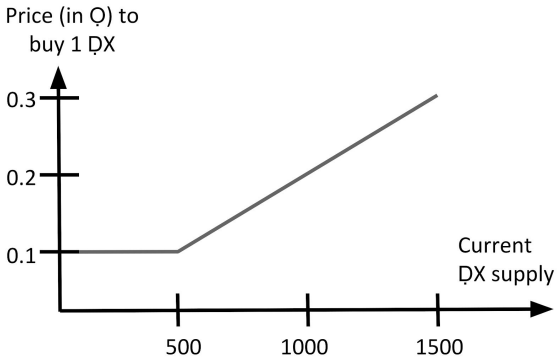

Figure 9: Bonding curve for ḌX

A new curation market is initialized each time a new dataset or service is made available. With this, the actor has already staked Ọ in order to have the dataset or service vetted. A later section describes vetting. Once vetted, this stake goes into the curation market, in return for drops as to a measure stake. Figure 10 illustrates. We're at the far left of the bonding curve because 0 ḌX have been generated. There, each ḌX costs 0.1 Ọ. If the initial user staked 50 Ọ, she would gain 50 Ọ/ 0.1 Ọ/ḌX = 500 ḌX. The supply for ḌX increases from 0 to 500.

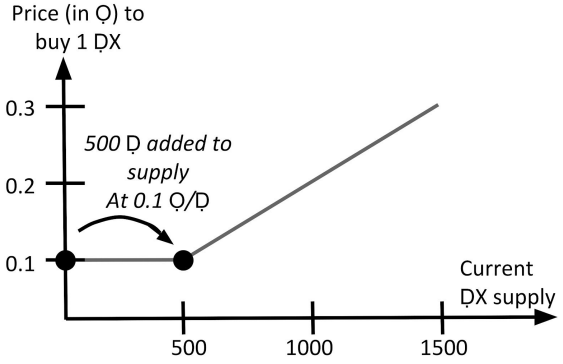

Figure 10: increasing supply to 500 ḌX

From here on, anyone can stake further in X. Let's say a user wants to purchase 500 ḌX by staking more Ọ tokens. This would make the supply go from 500 ḌX to 1000 ḌX. In that range, the price is (0.1 + 0.2 Ọ/ḌX)/2 = 0.15 Ọ/ḌX. The user would get 500 ḌX for a total cost of 500 ḌX * 0.15 ḌX/Ọ = 75 Ọ. Figure 11 illustrates.

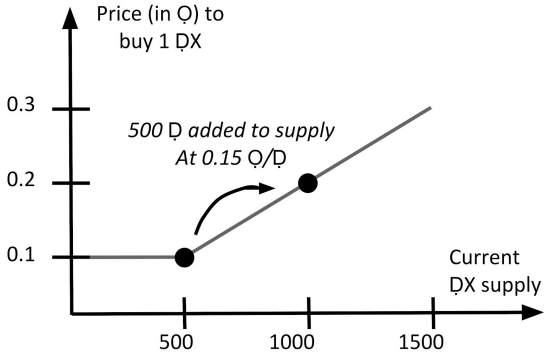

Figure 11: increasing supply to 1000 Ḍ X

8.4. Un-Staking

A user can sell some of their Ḍ in a service at any time, in return for Ọ. This is the exact backwards action compared to staking. The supply of Ḍ goes down correspondingly.

The ability to un-stake for Ọ leads to the possibility for pump-and-dump behavior. In a later section, we discuss this further, and how to mitigate it.

8.5. Convergence to Relevant Data/Services

One can ask: how does the token design lead to a large supply of relevant data/services?

Overall, each actor has "holdings" in terms of stake (belief) of the relative value of different datasets/services. If an actor is early to understand the value of a dataset/service, they will get high relative rewards. This implicitly incentivizes referrals: I will refer you to datasets/services that I have staked in, because then I get more block reward.

Actors get rewarded the most if they stake large amounts on popular datasets/services - the first and second terms in the block rewards function, respectively. Put another way, they must predict that a dataset/service will be popular, then see its actual measured popularity (as a proxy for relevance). Just one alone is not enough. Over time, this causes convergence towards relevant datasets/services.

9. Core Blocks: Identity, IP, Pricing, Governance

9.1. Identity: Token Curated Registry of Users

There may be a need for some form of identity for several reasons. First, we want to incentivize good actors to be in the system, and bad actors to leave. We want stake whales to have diminishing returns in stake on a dataset/service; to accomplish this the block rewards function applies log10() to stake in a dataset/service; in turn, this is only possible if we have a reasonable handle on identity so that a stake whale doesn't replicate themselves several times. On the flip side, we do not want the core of Ocean network to be tied to any jurisdiction's laws or processes; therefore identity defined by something like KYC (Know-Your-Customer) like in banking would be too heavy. We need an approach that strikes a balance.

Our solution is to maintain a **whitelist of good actors** using economic incentives. Each actor needs to be incentivized to have good (or at least not bad) behavior. This starts with requiring skin in the game, e.g. via staking. "Good" means accredited as non-fraudulent by Ocean Token holders. "Actors" means all stakeholders except data/service consumers. This therefore includes not just

keepers, but also providers, referrers, verifiers, and consumers of data/services. We don't include consumers because there's no need to whitelist them for most cases, and we want to minimize onboarding friction wherever we can. Marketplaces and other higher-level services can add further whitelisting (including KYC) for stricter data requirements, such as making medical data available to research scientists.

We realize these goals using a **Token Curated Registry (TCR)** [Goldin2017] of actors. In TCRs, existing actors are economically incentivized to only add good actors and to keep out bad actors. If an actor in the registry is found to be acting bad, the contention mechanism could be invoked and the misbehaving actor's stake is lost.

A prospective new actor can enter this TCR whitelist one of two ways:

1. **Staking themselves**, like in [Goldin2017]. This is useful for actors who are new to the system, and don't know others, so are motivated to undergo a vetting process that we elaborate below.
2. **Risk-staking by others.** That is, others vouch for them. This is inspired by the OpenBazaar "trust is risk" proposal [Zindros2017]. This is useful for actors who do know others in the system who are willing to vouch for them, and can therefore start participating in the system immediately.

Let's elaborate.

Staking themselves. A new actor can stake a proposal to join. There is a vetting period in which challengers can come forward with stake. If there are no challengers, the new actor is in. If there is a challenger and the majority vote "ok", then the challenger loses tokens and the new actor is in. If the majority votes "not ok", the new actor loses their staked tokens. The actor must wait for this process to complete, to participate in the system.

Risk-staking others (referrals). An existing actor can risk-stake on a new actor to help get the new actor into the system. The "risk" part means the new actor can take the existing actor's stake anytime for themselves, but they aren't expected to, because they stand more to gain by participating in the ecosystem. The existing actor is awarded R_{rs} times the new actor's block rewards in betting on high quality datasets/services. R_{rs} is a parameter to be set, at a value about 0.1. This reward amount is not taken from the new actor's block rewards, it is extra rewards (a positive sum game).

The existing actor must always stake more than the total block reward that the new actor receives. If the new actor acts badly (e.g. loses their stake for not making dataset/service available), then the existing actor loses their stake in the new actor.

If the existing actor un-stakes in the new actor such that their total stake goes below the registry threshold, then they are removed from the registry. So, awards and punishments are based on the new actor acting well and badly, respectively. Risk-staking leads to an emergent web-of-trust. Ocean incentivizes for this to continue indefinitely, by continuing the stream of referral block rewards.

9.2. IP Attribution & Provenance: COALA IP

Ocean will use the COALA IP [COALAIP2018] protocol for specifying IP rights on data. COALA IP is a schema for the shape of JSON-style transactions going onto the blockchain. All COALA IP transactions are signed. Data is content-addressable, using IPLD [IPLD2018]. COALA IP builds on the lineage of semantic web [W3C2018]. At the marketplace level, metadata specified in COALA IP and fills out a templated legal contract that is then hashed to the blockchain, Ricardian-contract style [Grigg2004].

We started building ascribe [McConaghy2015] in 2013 as a service for IP on the blockchain, focusing on digital art. Users could claim copyright, specify a limited number of editions, and transfer rights of an edition to other users. We found that our initial IP protocol was not flexible enough. So, we co-developed COALA IP protocol with collaborators from Protocol Labs, COALA, Ujo Music, and more.

COALA IP is general enough to handle music, movies, 3D designs and more, as well as for data itself. It allows for fractional ownership, licensing different rights to different parties in different jurisdictions, time limits (e.g. "you can use this data for the next week"), derivative rights (e.g. cleaning up a dataset), and more.

COALA IP models transfer of rights from one holder to others. Because COALA IP transactions are *signed* and go to an *immutable* blockchain, and the supply of data and compute itself is cryptographically proven, Ocean ends up holding an immutable record of data and compute history, i.e. provenance. Such *data and compute provenance* has wide applicability to practical AI and data management problems. COALA IP also naturally supports remix provenance, e.g. going from a messy dataset to a cleaned one, then a normalized one.

9.3. Vetting IP Rights: TCR

A data provider should only post data if they are the rights holder, they have a license to post the data, or the data is public domain. Of course this is difficult to perform automatically and accurately. Consequently, Ocean discourages abuse via a Token Curated Registry (TCR), as follows.

When the provider posts the data, they must stake a minimum count of tokens for a minimum time period. Anyone can challenge the publisher's claim during that period, with stake. There is then a vote, where "yes" means "data is not junk and rights are ok".

- If the majority votes "yes", the challenger loses the staked tokens. The data becomes available in the network.
- If the majority votes "no", then the poster loses their staked tokens (on this data asset) and gets removed from the actors registry. Removal from the actors registry is a serious consequence, but we believe it's a critical step in order to maintain an ecosystem of good (non-infringing) actors. In this case, the challenger gets some of the provider's staked tokens as this incentivizes them to post challenges in the first place.

During this challenge time period, the data itself may get served up, but all block rewards are held in escrow by the network until the appropriate rights holder is identified. In doing this, we keep the friction to publishing data low, but ensure that the appropriate rights holders get the rewards. This mechanism is similar to that of SoundExchange in the music industry [SoundExchange2018].

9.4. 3rd Party Arbitration

In Ocean, people make claims about having particular IP rights. Of course, people can lie. Ocean's main tool to address this is via staking, as described. Another *possible* tool is to use plug-in 3rd-party arbitration, such as Mattereum [Mattereum2018], which could draw on the full force of the law. However, using that would bind the network to particular jurisdictions and laws. This is not desirable, as it would compromise the **borderless** nature of Ocean. Therefore our current approach is to make it easy to have plug-ins like Mattereum at higher levels (e.g. marketplaces), but for the core of the Ocean network to rely on staking.

9.5. Pricing: Basics

Marketplaces will have their own approaches to pricing, but for discoverability, liquidity, and clearing, Ocean itself will store the pricing information. We envision the following.

Free Data. We want to encourage a growing data commons for the world, where anyone can download commons data for free.

Priced Fungible Data/Services. Some data is exchangeable with decent liquidity, for example Electrocardiogram (ECG) data for 100 people in Singapore is the same as 100 people in Germany. *Exchanges* are a low-friction way to handle fungible data and services, as they let the market determine the price in an automated way.

Priced Non-Fungible Data/Services. Some data or services are not easily exchangeable. Then, pricing here may include fixed price, auction, and royalties. Each has pros and cons. Fixed price is simple. Auction pricing finds price at the cost of more complexity of implementation and user experience. Royalties could prove to be very useful when it's possible to compute the final value provided, so that royalties can propagate backwards. Ideally we compute relative impact by each dataset, e.g. like in [McConaghy2008], then pay proportionally.

For any pricing that is more complex than "fixed price", Ocean network will most likely need to have smart contracts holding the service contract. Ocean will provide schemas for the more common pricing approaches.

The Ocean Token is used as a currency for buying and selling. Data/services are priced in currency of the vendor's choice (e.g. USD, EUR, ETH) then converted just-in-time to a token price, according to crypto exchange rates. Conversion would happen ideally via a decentralized exchange, though in the near term centralized exchanges may be needed due to software maturity. Golem [Golem2016] and other emerging tokenized ecosystems work similarly, using e.g. the exchange capabilities of Infura infrastructure [Infura2018].

9.6. Pricing: Reputation and Staking

As mentioned earlier, we use a curation market to incentivize data supply or referral. We don't *require* this for priced data, as traditional supply/demand data is a sufficient signal to set pricing, and price is a proxy for data reputation. However, a curation market plus block rewards catalyzes more data to be added more quickly. Furthermore, by having a further reputation signal, it helps users discover quality data assets and choose among them, just as Amazon's star-based reputation system is a signal beyond simple pricing.

However, we do need to discourage bad acting. The starting point is the data/services registry. But going one step further, we use staking. In order to sell a dataset/service for amount *x*, the vendor must stake amount *x*. To sell it twice, they must stake 2x. The stake will get locked up, and cannot be re-used during that period.

9.7. Governance: Fixing Bugs, Protocol Updates

This section addresses how the codebase gets updated. We need to handle simple non-controversial bug fixes to larger, possibly controversial protocol updates; with shades of gray in between. Governance options range from fully on-chain to hard forks; with shades of gray in between here too. We are sympathetic to the full range of options.

We see pragmatic solutions emerging such as ZeppelinOS [Zeppelin_os2018], Aragon [Aragon2018] and Colony [Colony2017]. For example, ZeppelinOS provides a systematic means for tokenholders to agree on updates to smart contract code. We expect these to mature further as we develop towards

the Ocean production net. At the time of this writing, we expect that Ocean will employ one or a combination of these tools and processes.

10. Cryptographic Proofs: Service Integrity and Verifiability Framework

10.1. Introduction

Ocean Protocol is an open network that uses public trust rather than institutional trust to ensure and verify service integrity. In a permissionless setting, there is no reliance on trusted third parties to verify that a requested resource has been correctly delivered. Rather, participants can encode service contracts in a machine readable format that can be executed and verified by the network. Keepers ensure that resources specified in the service contract have been provided according to the service contract conditions before releasing funds.

For network keepers to perform their function, they need the ability to verify that datasets are correctly stored, verify that datasets have not been tampered with, and prove that computations are correctly executed. As a rule, network keepers need transparency to perform these functions. But giving network keepers visibility into the data and computation algorithms may conflict with privacy. One could replicate the computation. But this is prohibitively expensive for big data services and does not guarantee that the errors are uncorrelated. More advanced setups require trusted execution environments or cryptographic protocols.

Since the Ocean Protocol network embodies a variety of services within a broad range of needs, including transparency, privacy and complexity, a versatile framework is proposed to enforce trust.

We'll first describe a model for each service and keeper node in the system and then proceed to lay out the verifiability options for each class of nodes.

10.2. Actor Model for Services

We start with the actor model [Hewitt1973] which describes the behavior of a system when there is no consensus upfront. This model has actors, behaviors and messages. Intelligent agents in the system are reactive and communicate through messages. If an agent doesn't understand a message, the message is discarded. If the message corresponds to a contract or protocol operation that the agent understands, then it can respond by making internal decisions, changing state, sending more messages or creating more actors. Figure 12 illustrates.

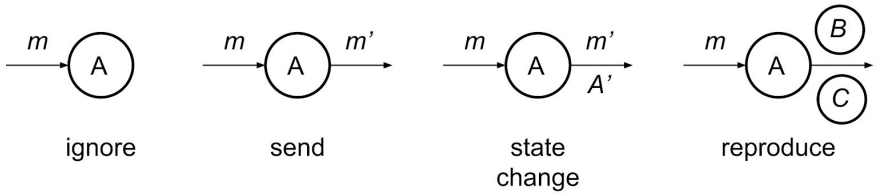

Figure 12: Actor responses on messages

The actor model follows Amdahl's Law [Rodgers1985], which states that the scale of a system is limited by the amount of shared state. Shared state requires consensus. A pure actor model has no shared state between the agents. In Ocean, the shared state is kept by a network of keepers that coordinate with each other under Byzantine agreement.

The behavior of actors can be modeled using automata or dynamical system theory. In a broad context, actors can be described as Turing machines with finite memory. This allows us to describe off-chain data services as well as on-chain smart contracts and transactions. As implementation patterns become more complex, such as in deep learning deployments or the vast amounts of data in data warehouses, so do the corresponding behaviours in the ecosystem.

If we want to port this model to our public, trustless Ocean network, there are many ways to enhance trust between actors, including modern cryptography, replication and consensus, provers and verifiers, claims, reputation, stake, curation, attention, governance, secure hardware and so on.

One can abstract the behavior of off-chain and on-chain services as a combination of procedures (computation) and state (storage). Depending on the service type, clients send a request through an API call to a typical web service or send or multicast transactions to a sufficient number of nodes of a decentralized network. Figure 13 illustrates.

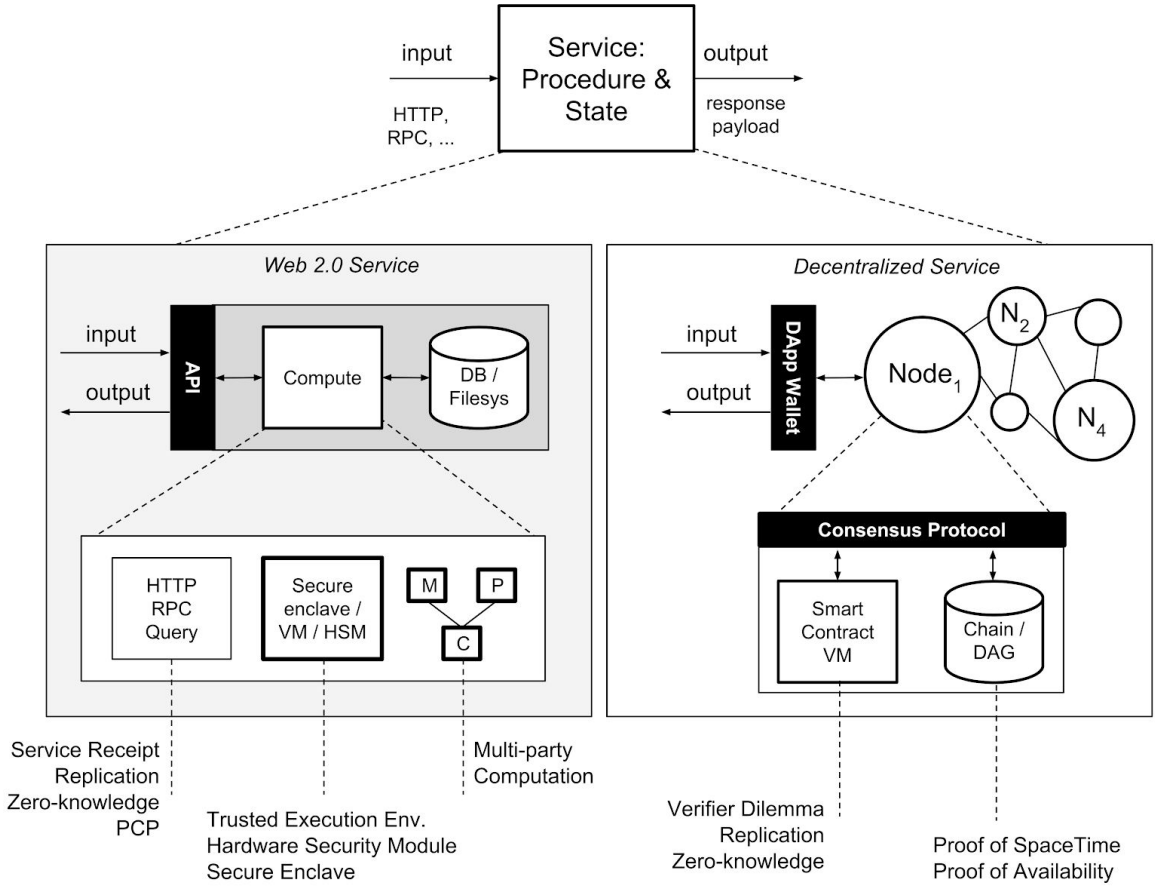

Figure 13: Left: a web2.0 service with various privacy measures. Right: decentralized services as replicated nodes with a consensus protocol. On the bottom, multiple options for service integrity verification are given.

Replicated state machines that implement a blockchain protocol require determinism for consistent state transitions, but this can't be guaranteed with off-chain oracles. Both the amount of replication (by consensus power) and transparency (in structure, code, state, history, random seeds, etc.) of a protocol impact the verifiability of the outputs produced by an agent in the system. Instead,

depending on the type of service and the level of transparency, the integrity of a service can be verified by the client or external verification services.

Options to verify integrity include signed receipts, replicated execution, trusted execution or prover-verifier setups. For the latter, a prover P can efficiently convince the verifier V of a mathematical assertion. In most cases, the prover is the service provider and the verifier is the service consumer. This allows a verifier to check if a computation has been executed correctly without having to re-execute the computation itself. Similarly, one can prove that data has been stored correctly without having to store the data itself. Ocean will provide an open-ended framework to encompass a multitude of service proofs. Additionally, 3rd-party verifiers can be assigned to verify a specific service at a certain cost.

We now discuss service integrity further. First we'll define service integrity as a superset of data and computational integrity. Next we'll give some background on specific verification implementations that facilitate the service verification framework.

10.3. Service Integrity

Eliminating trusted third parties raises the need for more privacy and service integrity. Corresponding to the decomposition of a service in procedure and state, service integrity is based on two components: computation and data integrity. We use the following definitions.

- The **time** T denotes a monotone sequence of timestamps representing events, epochs, blockheights and so on. The set of **inputs** are denoted by I and the **outputs** by O.
- We model the space of latent variables (**state** space) by S. These variables capture (in-)finite memory, stack depth and the **rank** N equals the minimal number of states required to implement the algorithm or storage. The rank N of a system is a measure of the complexity of a system. It follows that a deep neural net has a higher rank than a simple look up in a key-value store.
- We also define a **state transition function** : $C \ : S \times T \times I \rightarrow S \times O$ that acts on the input, changes the state and signals an output [Teusch2017]. C can be a smart contract, more generic Turing machines or behavior of an actor, for example. Figure 14 illustrates. On-chain smart contracts have a deterministic transition function while off-chain oracles are more black-box and probabilistic.

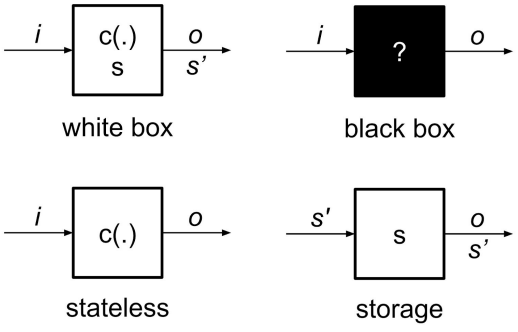

Figure 14: Possible system representations for actors in the system

- **Service integrity** implies that a reported response o of a service is correct with respect to a request i where service is defined by the tuple (C, S). This allows an independent verifier V to assert whether the following statement is $true$ for an unobserved service provider:

$$\tau_{(C,S,i,o,T)} := \text{“}o \text{ is the response of service } (C,\ S) \text{ on input } i \text{ after } T \text{ steps”}$$

As service provisioning is typically a combination of computation and state (data), we decompose the integrity definition. As mentioned above, computational integrity requires a variety of techniques and approaches to satisfy specific compute delegation requirements, depending on the task type and the level of transparency and privacy required. Data integrity is correlated to data availability, data consistency and the extent of data propagation in the network. For big data systems, verifications need to be performed exponentially faster than the data size or the complexity of the actual computation:

$$rank\ proof \sim O(log(max(rank\ C,\ rank\ S)))$$

Figure 15 shows types of service integrity, grouped according to data vs. computation and sub-groups.

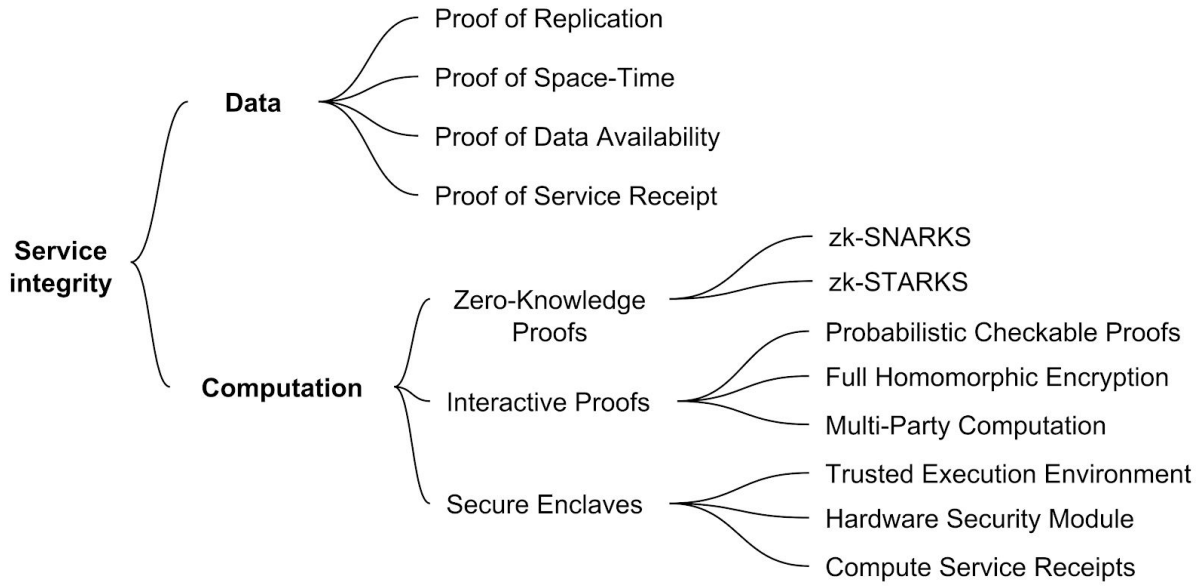

Figure 15: types of service integrity

Verifiers will audit proofs using one of the schemes in the figure; subsequent sections have details on these proofs. The level of replication of the service verification can be set by the parties that go into a service agreement. It's possible that only the consumer verifies the service proofs delivered by the provider. Alternatively, verification services can be provisioned randomly or predetermined to further enhance the security and integrity of the provided service. Of course, more replication leads to more computational cost and needs to be accounted for in the service agreement.

10.4. Service Integrity: Computational Integrity

In on-premise computation, the data consumer needs a provably correct model execution on the purchased data. Hence, a service needs to provide sufficient proof to convince a verifier that the code C actually ran on the dataset S.

Computational integrity implies that a reported response o of a computation C is correct with respect to a request i and dataset S such that $o = C(S)$, ensuring that a prover P correctly reports the output rather than a more favorable output to the prover.

At a high level, the computational integrity is represented by two parties where there are verifiers and provers. Let us illustrate in Figure 16. A verifier V is simply able to send a task or a function C and input i to a prover P. P will execute the computation on behalf of V then return the output o along with a short proof. Computational integrity is defined by correctness, soundness and zero knowledge, where correctness means that P can convince V concerning a true statement and soundness means that P cannot convince V of any false statement.

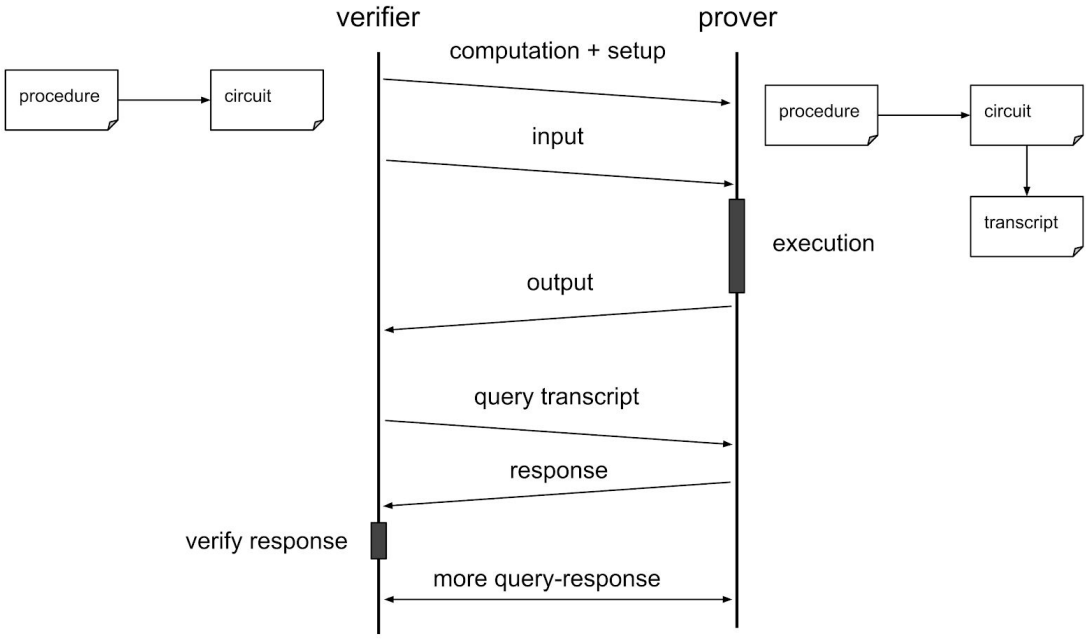

Figure 16: Computational integrity framework

Each proof system usually relies on assumptions. Assumptions mean that the prover may have a huge computation power which guarantees that the protocol will execute any task or the prover cannot solve certain problems. Also the verifier might have access to all inputs (like public blockchains) or not (such as confidential transactions). Moreover, assumptions can include replication of computation (for example proof of work), executing tasks on a trusted hardware like (Trusted execution environment or "TEE" protocol), using multi-party computation (MPC) where no single entity has the whole secret or the toxic waste, attestation, or auditing. There are multiple factors for selecting the suitable protocol or proof system including the functionality of the protocol, the implementation complexity, the public verifiability, applicability of zero-knowledge, the number of required messages to be transferred between prover and verifier, etc.

The appendix elaborates on some popular computational integrity approaches.

10.5. Service Integrity: Data Integrity

Data availability proofs are key cryptographic primitives in Ocean Protocol. We need to be able to prove that an actor made the correct file available, versus an incorrect one. Put another way, how do we tell if the data asset just made available is the same as the one that was initially uploaded?

For clients (verifiers) to reliably retrieve a data object, a storage service (prover) is required to provide a concise proof that data was made available and can be recovered in its entirety. Early work introduces a cryptographic building block known as a proof of retrievability (POR). A POR enables a user (verifier) to determine that an archive (prover) "possesses" a file or data object S. These proofs rely on efficient hash functions while ensuring that the memory and computational requirements for the verifier are independent of the (potentially large) size of the file S. In other words:

- **Data integrity** requires that no bounded prover P can convince clients V to accept altered or falsified data $S' \neq S$ after a recovery or GET operation [Filecoin2017].
- **Data availability**: if most clients with access permissions to the datum S can see S, then S is available.

For Ocean Protocol both data integrity and availability are important design constraints. Popular datasets should become more available by referral while respecting ownership attribution.

The appendix elaborates on some data integrity & availability approaches.

11. Outstanding Concerns

We believe this system design is a reasonable first cut. However, we still have concerns. The biggest include:

- **Complexity.** While the core is basically cryptographic proofs with curation markets, there are many building blocks around it: the actors registry, staking IP claims, etc. These each add complexity.
- **Decentralization substrate: scale, ecosystem.** We currently do not see any decentralization substrate that has both *scalability* and a mature developer *ecosystem*. However, there are several options, and these will likely evolve as we move towards deployment.
- **Data availability proofs - maturity issues.** Filecoin's Proof-of-Space-Time is not available yet. Teusch's data availability proof is expensive. Our challenge-response proof needs vetting.
- **Compute proof issues.** We assume that it won't be straightforward when linking to compute proofs and especially to related decentralized compute networks. Another issue will be linking full data-compute flows. All of these will take time and effort.
- **Concerns described elsewhere.** The appendix describes more specific concerns. We believe we have reasonable answers to each concern. However, these answers may not be perfect or have unforeseen issues.

In addressing these concerns and others that appear, it may turn out that the final designs will be quite different than the ones in this document. As engineers, we are perfectly comfortable with this.

12. Conclusion

This paper presented Ocean Protocol: a protocol and network for AI data and services. Ocean incentivizes relevant *priced* data, relevant *public or commons* data, as well as privacy-preserving AI

compute services. Ocean's core mechanic is Proofed Curation Markets, which combine cryptographic proofs with curation markets; binding actual and predicted popularity of data/services.

13. References

[adChain2017] adChain team, "Introducing the adChain Registry!", May 31, 2017, https://medium.com/@AdChain/introducing-the-adchain-registry-cc5b8b831a7e

[Amazon2018] "Amazon Aurora: MySQL and PostgreSQL Compatible Relational Database Built for the Cloud", Amazon.com, last accessed Feb. 13, 2018, https://aws.amazon.com/rds/aurora/

[Amazon2018b] "Amazon S3: Object Storage Built to Store and Retrieve Any Amount of Data from Anywhere", Amazon.com, last accessed Feb. 13, 2018, https://aws.amazon.com/s3/

[Amazon2018c] "Amazon EC2: Secure and Resizable Compute Capacity in the cloud. Launch Applications when Needed without Upfront Commitments", Amazon.com, last accessed Feb. 19, 2018, https://aws.amazon.com/ec2

[Aragon2018] "Aragon: unstoppable organizations", homepage, last accessed Feb. 19, 2018, https://aragon.one/

[Banko2001] M. Banko and E. Brill, "Scaling to Very Very Large Corpora for Natural Language Disambiguation", Proc. Annual Meeting on Association for Computational Linguistics, July, 2001, http://www.aclweb.org/anthology/P01-1005

[BigchainDB2018] BigchainDB homepage, last accessed Feb. 13, 2018 https://www.bigchaindb.com/

[BigchainDB2017] BigchainDB team, "BigchainDB and TRI Announce Decentralized Data Exchange for Sharing Autonomous Vehicle Data", May, 2017, https://blog.bigchaindb.com/bigchaindb-and-tri-announce-decentralized-data-exchange-for-sharing-autonomous-vehicle-data-61982d2b90de

[Bitansky2012] Nir Bitansky et al., "From Extractable Collision Resistance to Succinct Non-interactive arguments of Knowledge, and Back Again", Proc. Innovations in Theoretical Computer Science Conference, ACM, 2012, https://dl.acm.org/citation.cfm?id=2090263

[Cachin2017] Christian Cachin and Marko Vukolić, "Blockchain Consensus Protocols in the Wild", 2017, https://arxiv.org/abs/1707.01873

[COALAIP2018] "A blockchain-ready, Community-driven Protocol for Intellectual Property Licensing", COALA IP homepage, last accessed Feb. 13, 2018, https://www.coalaip.org/

[Colony2017] Alex Rea et al., "COLONY Technical Whitepaper", Sept 20, 2017, http://swarm-gateways.net/bzz:/whitepaper.joincolony.eth/

[ConnectedLife2018] ConnectedLife homepage, last accessed Feb. 13, 2018, https://connectedlife.io

[CDN2018] "Content delivery network", Wikipedia, last accessed Feb. 13, 2018 , https://en.wikipedia.org/wiki/Content_delivery_network

[Cormode2012] Graham Cormode, Michael Mitzenmacher, and Justin Thaler, "Practical Verified Computation with Streaming Interactive Proofs", Proc. Innovations in Theoretical Computer Science Conference, ACM, 2012, https://dl.acm.org/citation.cfm?id=2090245

[Crain2017] Tyler Crain et al., “(Leader/Randomization/Signature)-Free Byzantine Consensus for Consortium Blockchains”, May 2017, https://arxiv.org/abs/1702.03068 (“Red Belly” blockchain)

[Crevier1993] Daniel Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. Basic Books, 1993: 148–150.

[DeJonghe2017] Dimitri De Jonghe, “Curated Governance with Stake Machines”, Dec. 4, 2017, https://medium.com/@DimitriDeJonghe/curated-governance-with-stake-machines-8ae290a709b4

[Economist2017] “The World’s Most Valuable Resource is No Longer Oil, But Data”, *The Economist*, May 6, 2017, https://www.economist.com/news/leaders/21721656-data-economy-demands-new-approach-antitrust-rules-worlds-most-valuable-resource

[EthSharding2018] “Sharding FAQ: On Sharding Blockchains”, last accessed Feb. 13, 2018, https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[Evans2017] Alex Evans, “A Crash Course in Mechanism Design for Cryptoeconomic Applications”, Oct 17, 2017, https://medium.com/blockchannel/a-crash-course-in-mechanism-design-for-cryptoeconomic-applications-a9f06ab6a976

[Filecoin2017] Protocol Labs, “Filecoin: A Decentralized Storage Network”, August 14, 2017, https://filecoin.io/filecoin.pdf

[FSM2018] “Finite State Machines”, Wikipedia, last accessed Feb. 20, 2018, https://en.wikipedia.org/wiki/Finite-state_machine

[Gennaro2013] Rosario Gennaro et al., "Quadratic Span Programs and Succinct NIZKs without PCPs", Proc. Annual Intl. Conf. on the Theory and Applications of Cryptographic Techniques, pp. 626-645. Springer, Berlin, Heidelberg, 2013, https://eprint.iacr.org/2012/215.pdf

[Goldin2017] Mike Goldin, “Token-Curated Registries 1.0”, medium.com, Sep. 14, 2017, https://medium.com/@ilovebagels/token-curated-registries-1-0-61a232f8dac7

[Golem2016] Golem Team, “The Golem Project: Crowdfunding Whitepaper”, Oct. 2016, http://golemproject.net/doc/DraftGolemProjectWhitepaper.pdf

[Grigg2004] Ian Grigg, “The Ricardian Contract”, Proc. IEEE Intl. Workshop on Electronic Contracting, 2004, http://ieeexplore.ieee.org/document/1319505/

[Halevy2009] Alon Halevy, Peter Norvig, and Fernando Pereira, “The Unreasonable Effectiveness of Data”, IEEE Intelligent Systems 24(2), March-April 2009, https://research.google.com/pubs/archive/35179.pdf

[Hanke2018] Timo Hanke, Mahnush Movahedi and Dominic Williams, “DFINITY Technology Overview Series: Consensus System”, Jan. 2018, https://dfinity.org/pdf-viewer/pdfs/viewer?file=../library/dfinity-consensus.pdf

[Hewitt1973] Carl Hewitt, Peter Bishop, and Richard Steiger, “A Universal Modular Actor Formalism for Artificial Intelligence", Proc. Intl. Joint Conf. on Artificial Intelligence, 1973

[iExec2017] iExec team, “The iEx.ec project: Blueprint For a Blockchain-based Fully Distributed Cloud Infrastructure”, March, 2017, https://iex.ec/app/uploads/2017/04/iExec-WPv2.0-English.pdf

[ImageNet2018] “ImageNet”, Wikipedia, last accessed Feb. 13, 2018, https://en.wikipedia.org/wiki/ImageNet

[Infura2018] “Infura: Scalable Blockchain Infrastructure”, Infura Homepage, last accessed Feb. 13, 2018, https://infura.io

[IPFS2018] “IPFS: IPFS is the Distributed Web”, IPFS homepage, last accessed Feb. 19, 2018, https://ipfs.io/

[IPLD2018] “IPLD: IPLD is the data model of the content-addressable web”, IPLD Homepage, last accessed Feb. 19, 2018, https://ipld.io/

[Kalra2016] Nidhi Kalra and Susan M. Paddock, “Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?”, RAND Corporation, Apr 12, 2016 https://www.rand.org/pubs/research_reports/RR1478.html

[KellyCriterion2017] “Kelly Criterion”, Wikipedia, last accessed Feb. 17, 2018, https://en.wikipedia.org/wiki/Kelly_criterion

[Kiayias2017] Aggelos Kiayias et al., “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”, Proc. Annual International Cryptology Conference, August 21, 2017, https://eprint.iacr.org/2016/889.pdf

[Kokoris2018] Eleftherios Kokoris-Kogias et al., “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding”, Proc. IEEE Symposium on Security & Privacy, 2018 (preprint in 2017), https://eprint.iacr.org/2017/406.pdf

[Kwon2017] Jae Kwon and Ethan Buchmann, “Cosmos: Network of Distributed Ledgers”, 2017, https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md

[Lamport1982] Leslie Lamport et al., “The Byzantine Generals Problem”, ACM Trans. Programming Languages and Systems 4(3), pp. 382-401, July 1982, http://research.microsoft.com/en-us/um/people/lamport/pubs/byz.pdf

[Mattereum2018], “Mattereum: Smart Contracts for the Real World”, Mattereum homepage, last accessed Feb. 13, 2018, https://mattereum.com/

[McConaghy2008] Trent McConaghy et al., “Automated Extraction of Expert Knowledge in Analog Topology Selection and Sizing”, Proc. Intern. Conference on Computer-Aided Design (ICCAD), Nov. 2008, http://trent.st/content/2008-ICCAD-cad_synthesis_insight.pdf

[McConaghy2009] Trent McConaghy and Georges G.E. Gielen, “Globally Reliable Variation-Aware Sizing of Analog Integrated Circuits via Response Surfaces and Structural Homotopy,” IEEE Trans. Computer-Aided Design 28(11), Nov. 2009, http://trent.st/content/2009-TCAD-sangria.pdf

[McConaghy2015] Trent McConaghy and David Holtzman, “Towards an Ownership Layer for the Internet,” ascribe whitepaper v1.03, June 24, 2015, http://trent.st/content/2015-06-24%20ascribe%20whitepaper.pdf

[McConaghy2016] Trent McConaghy, “The DCS Triangle: Decentralized, Consistent, Scalable”, July 10, 2016, https://blog.bigchaindb.com/the-dcs-triangle-5ce0e9e0f1dc

[McConaghy2018] Trent McConaghy, “Token Design as Optimization Design”, 9984 Blockchain Meetup, Feb. 7, 2018, Berlin, Germany, https://www.slideshare.net/TrentMcConaghy/token-design-as-optimization-design

[ModifiedGHOST2018] “Ethereum Whitepaper: Modified GHOST Implementation”, Ethereum Wiki, last accessed Feb. 13, 2018, https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation

[MongoDB2018] “MongoDB Atlas: Database as a Service”, last accessed Feb. 13, 2018, https://www.mongodb.com/cloud/atlas

[Moor2003] James Moor, ed. *The Turing Test: The Elusive Standard of Artificial Intelligence*. Vol. 30. Springer Science & Business Media, 2003

[Nakamoto2008] Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, Oct 31, 2008, https://bitcoin.org/bitcoin.pdf

[NuCypher2018] “NuCypher: Proxy Re-encryption for Distributed Systems”, NuCypher Homepage, last accessed Feb. 19, 2018, https://www.nucypher.com

[OceanMkt2018] Ocean Protocol team, “Ocean Protocol: Reference Marketplace Framework”, 2018, https://oceanprotocol.com/marketplace-framework.pdf

[OrbitDB2018] “OrbitDB: Peer-to-Peer Database for the Decentralized Web”, Github repository, last accessed Feb. 19, 2018, https://github.com/orbitdb/orbit-db

[Parno2013] Bryan Parno et al., “Pinocchio: Nearly Practical Verifiable Computation”, Security and Privacy (SP)”, Proc. IEEE Symposium on Security & Privacy, 2013, https://eprint.iacr.org/2013/279.pdf

[Poon2016] Joseph Poon and Thaddeus Dryja, “The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments”, January 14, 2016, https://lightning.network/lightning-network-paper.pdf

[Poon2017] Joseph Poon and Vitalik Buterin, “Plasma: Scalable Autonomous Smart Contracts”, August 11, 2017, https://plasma.io/plasma.pdf

[PoReq2018] Proof of Replication Technical Report (WIP) Protocol Labs, https://filecoin.io/proof-of-replication.pdf

[Raiden2018] “What is the Raiden Network?”, last accessed Feb. 13, 2018, https://raiden.network/101.html

[Randao2018] “RANDAO: A DAO working as RNG of Ethereum”, last accessed Feb. 17, 2018, https://github.com/randao/randao

[Rodgers1985] David P. Rodgers, “Improvements in Multiprocessor System Design”, ACM SIGARCH Computer Architecture News 13(3), pp. 225–231, https://dl.acm.org/citation.cfm?id=327215

[Rouviere2017] Simon de la Rouviere, “Introducing Curation Markets: Trade Popularity of Memes & Information (with code)!”, Medium, May 22, 2017, https://medium.com/@simondlr/introducing-curation-markets-trade-popularity-of-memes-information-with-code-70bf6fed9881

[Salomon2017] David L. Salomon et al, “Orchid: Enabling Decentralized Network Formation and Probabilistic Micro-Payments”, January 29, 2018 Version 0.9.1, https://orchidprotocol.com/whitepaper.pdf

[Sasson2014] Eli Ben Sasson et al., “Zerocash: Decentralized Anonymous Payments from Bitcoin", Proc. IEEE Symposium on Security & Privacy, 2014, http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf

[Schwartz2018]“Schwartz-Zippel Lemma: Probabilistic Polynomial Identity Testing”, Wikipedia, last accessed Feb. 13, 2018, https://en.wikipedia.org/wiki/Schwartz%E2%80%93Zippel_lemma

[Setty2012] Srinath Setty et al., “M. Making Argument Systems for Outsourced Computation Practical (sometimes)”, Proc. Network and Distributed System Security, 2012, http://www.cs.utexas.edu/users/richard/pepper-ndss12.pdf

[Setty2012b] Srinath Setty et al., "Taking Proof-Based Verified Computation a Few Steps Closer to Practicality", Proc. USENIX Security Symposium, 2012, https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final26_0.pdf

[Setty2013] Srinath Setty et al., “Resolving the Conflict between Generality and Plausibility in Verified Computation”, Proc. ACM European Conference on Computer Systems, 2013, https://dl.acm.org/citation.cfm?id=2465359

[Simmons2017] Andrew Simmons, “Prediction markets: How betting markets can be used to create a robust AI”, Sept. 13, 2017, http://dstil.ghost.io/prediction-markets/amp/

[SingularityNET2017] SingularityNET team, “SingularityNET: A Decentralized, Open Market and Inter-Network for AIs,” Dec. 2017, https://public.singularitynet.io/whitepaper.pdf

[Smolenski2017] Mike Smolenski, “Permissioned Blocks: A Protocol for Blockchain Privacy & Confidentiality”, Dec. 2017, https://github.com/autocontracts/permissioned-blocks/blob/master/whitepaper.md

[SoundExchange2018], SoundExchange homepage, last accessed Feb. 13, 2018, https://www.soundexchange.com

[Spark2018] “Apache Spark: Lightning-Fast Cluster Computing”, Apache Spark homepage, last accessed Feb. 13, 2018, https://spark.apache.org

[Syta2017] Ewa Syta et al., “Scalable Bias-Resistant Distributed Randomness”, Proc. IEEE Symposium on Security & Privacy, 2017, https://eprint.iacr.org/2016/1067.pdf

[Teusch2017] Jason Teutsch and Christian Reitwießner, “A Scalable Verification Solution for Blockchains”, November 16, 2017, https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf

[Teusch2017b] Jason Teutsch, “On decentralized oracles for data availability”, Dec. 25, 2017, http://people.cs.uchicago.edu/~teutsch/papers/decentralized_oracles.pdf

[Thaler2013] Justin Thaler, “Time-optimal Interactive Proofs for Circuit Evaluation", Proc. Advances in Cryptology, Springer, Berlin, Heidelberg, 2013, https://link.springer.com/chapter/10.1007/978-3-642-40084-1_5

[Thomas2015] Stefan Thomas and Evan Schwartz, “A Protocol for Interledger Payments”, Ripple Inc., 2015, https://interledger.org/interledger.pdf

[Trón2018] Viktor Trón et al, “Swarm Documentation”, Jan. 25, 2018, https://media.readthedocs.org/pdf/swarm-guide/latest/swarm-guide.pdf

[Vu2013] Victor Vu et al., “A Hybrid Architecture for Interactive Verifiable Computation,” Proc. IEEE Symposium on Security & Privacy, May 2013, http://ieeexplore.ieee.org/abstract/document/6547112/

[W3C2018] “Semantic Web”, W3C Homepage, last accessed Feb. 19, 2018, https://www.w3.org/standards/semanticweb/

[Webhook2018] “Webhook”, Wikipedia, last accessed accessed Feb. 19, 2018, https://en.wikipedia.org/wiki/Webhook

[Wood2016] Gavin Wood, “PolkaDot: Vision for a Heterogenous Multi-Chain Framework”, 2016, https://github.com/polkadot-io/polkadotpaper/raw/master/PolkaDotPaper.pdf

[Zeppelin_os2018] “zeppelin_os: Securely develop and manage any smart contract application”, homepage, last accessed Feb. 19, 2018, https://zeppelinos.org/

[Zindros2017] Dionysis Zindros, “Trust is Risk: A Decentralized Trust System”, Aug. 1, 2017, https://www.openbazaar.org/blog/trust-is-risk-a-decentralized-trust-system/

[Zurrer2017] Ryan Zurrer, “Keepers — Workers that Maintain Blockchain Networks” , Aug. 5, 2017, https://medium.com/@rzurrer/keepers-workers-that-maintain-blockchain-networks-a40182615b66

[Zyskind2015] Guy Zyskind, Oz Nathan, and Alex Pentland, “Enigma: Decentralized Computation Platform with Guaranteed Privacy”, arXiv (whitepaper), June, 2015, https://arxiv.org/abs/1506.03471

14. Appendix: Extended Functionality

This section describes Ocean functionality that will not be in the initial core, but when deployed will help make Ocean a much richer ecosystem for ontologies of knowledge and reputation.

14.1. Extended Functionality: Labels

If we want to organize data/services, we could “bucket” it into groups, or even organize it as a hierarchy. But what if the dataset/service belongs to multiple groups? Labels overcome this. While they are simple, they are highly useful. The power of labels and ontological mappings have long been recognized for their value in both AI and data domains.

Ocean’s will have a curation market for *labels*. Think of each label as having a registry of datasets/services that fit that label; but then add incentive to curate the labels more strongly (via curation market).

This enables users to flexibly choose labels for a variety of use cases, in an entirely market-based approach:

- Labels for **security and privacy** needs. For example: “to follow GDPR, this data cannot leave German soil”.
- Labels for **AI usage**. E.g. “this AI training data has binary outputs”.

- Labels for **data shape.** E.g. “this is streaming data”.
- Labels for **hierarchies of data**. E.g. “this is farm data & streaming data”

This, in turn, enhances Ocean’s scalability across many data assets, moving from a single global pool to a collection of local pools organized by the crowd. This is best exemplified by hierarchies of data, such as subreddits of subreddits for data. We can use labels similarly for services. Ultimately, an ontology of knowledge [W3C2018] about AI data and services emerges.

14.2. Extended Functionality: Stake Machines

Initially we will have a single Token Curated Registry (TCR) for “good” actors. But as Ocean grows, we want to make it possible for actors to get promoted to new responsibilities such as being able to curate a datasets with a particular label, or having greater governance responsibilities.

We can implement this with **stake machines** [DeJonghe2017]. Stake machines marry TCRs with Finite State Machines (FSMs). In a traditional Finite State Machine (FSM) [FSM2018], the machine can be in one of many discrete states (represented as nodes), and can transition from one state to another via a state transition function (represented as directed edges). A stake machine is an FSM where the state transition function is *staking-based* using TCR mechanics.

In Ocean, the state relates to the actor’s reputation. So, for an actor to transition from the initial “good actors” TCR to a next-stage TCR, that actor stakes to join the next-stage TCR and the actors already in the next-stage TCR have the opportunity to challenge the actor.

Traditional TCRs have binary state (in or out); stake machines can have >2 discrete states; and curation markets have continuous states.

15. Appendix: Addressing Key Goals in Token Design

The main goal of Ocean network is to deliver a large supply of relevant data/services: “commons” data, priced data, and AI compute services. As previously discussed, we developed a set of questions as key criteria to compare candidate designs against. Table 2 describes the question / criteria (left column) and how the token design addresses those criteria (right column).

Table 2: How Ocean Token design addresses key goals

Key Question	
For priced data, is there incentive for supplying more?	Block rewards are a function of stake in a dataset/service. Actors are incentivized to stake on relevant data/services as soon as possible because of curation market pricing. The most obvious way to get the best price then is to supply it.
For priced data, is there incentive for referring?	Curation markets incentivize data referrals, because they signal which is high quality data. If I’ve bet on a dataset, I’m incentivized to tell others about it, as they’ll download it or stake on it, both of which reward me.
For priced data, is there good spam prevention?	No one (or at least few) will download low-quality data, i.e. spam. Therefore no one is incentivized to stake on it in a curation market. Therefore, while it can exist in the system (and not hurt anyone) there is no incentive to stake on it.

For free data, is there incentive for supplying more? Referring? Good spam prevention?	Same as priced data.
Does it support compute services, including privacy-preserving compute? Do they have incentives for supplying more, and for referring?	Ocean's core construction is a Proofed Curation Market, binding cryptographic proofs with Curation Markets. For data, the main proof is for data availability. But this can be replaced with other proofs for compute, including privacy-preserving compute. In doing so, all the benefits of data supply and curation extend to compute services supply and curation.
Does the token give higher marginal value to users of the network versus external investors?	Yes. Token owners can use the tokens to stake in the system, and get block rewards based on the amount staked (and other factors). This means that by participating in the system, they're getting their tokens to "work" for them.
Are people incentivized to run keepers?	Yes. One only gets block rewards for data they've staked if they also make it available when requested; making data available is a key role of keepers.
Is it simple?	The system is conceptually simple: its simple block reward function is implemented as a binding of cryptographic proofs *curation market, to form a Proofed Curation Market. It adds ancillary affordances as needed, though those are changeable as new ideas emerge.
Is onboarding low-friction?	On-boarding for the actors registry, and for each dataset might have been high because in each case there is a token-curated registry that asks for staking and a vetting period. However, in each case we have explicitly added low-friction alternative paths, such as risk-staking.

.

16. Appendix: FAQs and Concerns

This section addresses frequently asked questions and concerns about Ocean.

16.1. Data Storage

Q: Where does data get stored?

A: Ocean itself does not store the data. Instead, it links to data that is stored, and provides mechanisms for access control. The most sensitive data (e.g. medical data) should be behind firewalls, on-premise. Via its services framework, Ocean can bring the compute to the data, using on-premise compute. Other data may be on a centralized cloud (e.g. S3 [Amazon2018b]) or decentralized cloud (e.g. Filecoin [Filecoin2017]). In both cases it should be encrypted.

This means that the Ocean blockchain does not store data itself. This also means that we can remove the data, if it's not on a decentralized and immutable substrate.

16.2. Data Protection Regulations

Q: What about data protection regulations, such as General Data Protection Regulations (GDPR) which come into effect in Europe in May 2018?

A: Higher-level marketplaces will provide the necessary resources around data protection laws, including GDPR. As part of the overall Ocean project, DEX is creating open-source software to support this, and the related compliance and legals. It will be used as a reference marketplace that other marketplaces can use as a starting point.

Ocean's labels registry will create a bridge from the network technology level with the higher level legals: people can curate labels on data assets such as "must meet GDPR".

16.3. Data Escapes

Concern: You're trying to sell data. But, someone else downloads it and then posts it for cheaper, or even for free. In fact the system incentivizes "free" block rewards, because there will be more downloads for something that's free, versus paid.

A: The only way to post is to be in the registry of "good" actors, which can only be there by staking themselves or by others risk-staking for them. If some actor is found to be using data that is not theirs, then the contention mechanism is invoked, and the actor's stake (or their voucher's stake) is lost. So, any gains an actor might have had for "escaping the data" evaporate because the actors collectively have an incentive to be good actors, and they have a mechanism to take away the stake of bad actors. If a data rightsholder is especially worried, they can also set permissions such that the data never leaves the premises; rather, compute must be done locally. They do this at the cost of virality due to the data's openness, of course.

16.4. Curation Clones

Concern: You've published a new unique dataset into the commons, and have kicked off a curation market. It's popular, so you earn many block rewards and many others stake on it too, raising the price of staking. Someone else sees the dataset's popularity, so they *re-publish* the dataset and started a brand new curation market, where they get significant stake in the market because they were the early adopter. Then, others repeat this. In a short time, we have 50 curation markets for the same dataset, hindering discoverability not to mention being unfair to the first publisher.

A: The main solution is the same as for the Data Escapes problem: in the registry of good actors, the contention mechanism would be invoked and the misbehaving actor's stake is lost. We can also leverage local compute.

A complementary tactic is in the design of the bonding curve: make the price to stake in the curation market *flat* to start with, rather than rising immediately. In this way the first 10 or 50 actors no longer have an incentive to start their own curation market; rather they are incentivized to grow the popularity of that collectively-owned curation market.

16.5. Elsa & Anna Attack

Concern: An actor stakes and publishes IP that they clearly don't own, such as the Disney movie "Frozen" featuring Elsa & Anna. As any parent of young girls would recognize, the asset quickly becomes extremely popular. In other words, Elsa & Anna have bombarded the network. Because the

actor has staked on the asset and served it up when requested, they would have quickly earned a tremendous amount of block reward tokens. Finally, the actor leaves the network, taking their earnings with them (and none to the rights holder).

A: What's the solution? Actually, the system would prevent the scenario above, as described in the section on vetting IP. In short: when an actor publishes IP, they must first get that IP vetted; and during that vetting process all block rewards are held in escrow by the network until the appropriate rights holder is identified.

16.6. Drops Supply When Stake is Lost

Q: When a misbehaving actor loses stake (drops in a dataset or service), where does that stake go?

A: We have a few options. We could burn the drops; redistribute them to other stakeholders in the market; or give the stake to the winning challenger. We choose the last option, because it maintains the same supply of drops which means simpler incentive dynamics; and it comes "out of the box" with TCR designs.

16.7. Sybil Downloads

Concern: An actor puts a high stake in one data asset, then downloads it many times themselves to get more mining rewards. This could be from their own single account, or from many accounts they create, or in a ring of the actor and their buddies. This is bad for a second reason: it's a giant waste of bandwidth. This issue is analogous to the "click fraud" problem in online ads.

A: We don't make rewards a function based only on the number of files accessed. Instead, we make it a function of the number of bits accessed versus registered and the price paid for the data.

16.8. Registry Scaling

Concern: Typical token curated-registries don't scale, with respect to the number of participants. That is, actors in tokenized registries have diminishing returns for letting more people in. Once they get to say 1000 people, it's really not worth the risk to let anyone else in, the system is rich enough. This is especially the case when there are rewards beyond just membership in the registry.

A: Have a have an additional mechanism to on-board: *risk-staking* (vouching) which has a direct incentive to refer others, because the vouching party can get some of their block rewards. Therefore, it keeps going and going. The new participants stay linked in a web-of-trust risk-staking framework.

16.9. Onboarding Friction

Concern: Typical tokenized registries need the user to stake in order to join; and must go through a vetting period of e.g. 28 days. That means they must go and purchase the network's tokens and then wait before even participating.

A: We address this via the risk-staking mechanism, where existing actors are incentivized to onboard new actors because they get block rewards for doing so (assuming the new actor stakes on high-quality data or services).

16.10. Sybil Referrals

Concern: A malicious actor has many tokens. They risk-stake these tokens to refer in thousands of people at once, and overtake the system by voting for a protocol update in their favor.

A: The risk for this attack is reduced with a large number of users in the system, because it will be too expensive to risk-stake so many people. The only concern is when there is a smaller number system users. However, because protocol governance is handed off gradually over time, by the time it's handed off to be exposed to this risk, there will either be a large number of users (then we're ok) or there won't be yet and additional constraints could be introduced (e.g. only one referral per actor per day). We can take actions to seed the network with a large number of relevant users, e.g. via airdrops.

16.11. Sybil Membership Applications

Concern: If there are lots of applications at once, existing holders get swamped. They don't have the bandwidth to properly review the applications.

A: The applicants need to stake tokens to apply. Token holders are incentivized to increase the number of actors in the system, since it will grow the value of the token; therefore they might go to great lengths to support growth. However, that might still have a breaking point. In this case, the community might decide, off-chain, to throttle all applications and publicly state "we can only take *x* applications per month, we will vote no to the rest".

16.12. Staking Vs. Liquidity

Concern: To make a sale, a vendor must stake 1x the sale amount for a fixed time period. This affects the overall velocity of tokens flowing through the system.

A: Our response has three parts.

"Work" mechanics. At first glance, Ocean may look like a proof-of-stake system. However, to get block rewards, keepers actually have to do work in some cryptographically provable way, such as making data or services available. This costs money / resources. Therefore people making lot of data available and using expensive bandwidth will need to periodically sell tokens to finance their operations. This ensures steady velocity of tokens. As an aside, this means that Ocean combines proof-of-stake mechanics with proof-of-work.

Security / simplicity. We are reluctant to compromise on the "1x" mechanic because it compromises security. A stake of <1x makes it easier for a bad actor to make money from acting badly. We could use insurance or another mechanic, but 1x staking is simple.

Emergent loans. If the 1x hinders data vendors, an external loans market could emerge. People could borrow money to be able to stake it; the interest rate could be based on the lender's calculations of risk.

16.13. Rich Get Richer

Concern: A long-standing concern with Proof-of-Stake (PoS) systems is that stakeholders get wealthier simply by having more tokens. As a result, many PoS systems have changed to where stake is needed simply to participate in the network; and perhaps higher staking gives a more active role like being a keeper node in Cosmos [Kwon2017].

A: "Rich get richer" is less of a concern for Ocean because of curation markets. Recall that stake in a data curation market is not "just" the amount you initially staked, but also how many tokens you would receive if you withdrew. Therefore, early adopters to a popular data or service asset will get

rewarded. For Ocean, it's not rich get richer, it's "curate data well" = "get richer". A secondary equalizing factor is using log10 on stake.

16.14. Pump-and-Dump on Drops

Concern: Recall that each AI dataset/service has its own token - its *drops* Ḍ, which are staked and un-staked in the context of its corresponding curation market. In this scenario, "pumping-and-dumping" is a concern. For instance, if someone stakes early in a curation market to get Ḍ, then promotes that dataset/service to others (pumps), and finally sells their Ḍ at a big profit and leaves everyone else hanging (dumps).

A: Overall, this may not be a significant concern because "active" Ḍ holders are actually earning Ocean Tokens Ọ by making that service available when asked; they are getting *positive marginal benefits* for staking Ḍ. If assuming an efficient market, over time we'd end up with only active Ḍ holders. That said, we might still see this type of behaviour from bad actors. Possible mitigations include:

- Have one bonding curve for buying Ḍ and a *second* one for selling Ḍ, where the sell price is lower than buy price.
- When selling, use a Dutch auction: the sell price is the *previous* buy price, not the current price.
- Have minimum staking periods. For example, the requirement to hold any Ḍ for at least a week.

In general, we simply want to add friction on the sell side in a way that good actors won't mind, and bad actors will leave. Overall, it's clear that if pumping-and-dumping becomes a real issue, we have tools to address it.

16.15. Block Rewards for On-Premise Data

Concern: If a data asset is on-premise, then only the actor storing that data asset can "keep" it and earn block rewards for making it available. Others who might believe that it's valuable may stake on it in a curation market (and sell that stake at a gain later); but they cannot make it available and therefore cannot get block rewards for it. This also means there is no automatic CDN functionality, so retrieving that data will become a bottleneck if it becomes popular.

A: The answer is twofold: privacy and markets.

Privacy. If the reason to store the data on-premise is privacy, then it should stay that way! Privacy trumps access convenience and CDN scaling.

Markets. If the actor storing that data asset sees that it becomes popular, they are incentivized to spread its usage more. The way to do that is to remove themselves as a bottleneck, by letting other actors store the data and make it available in CDN scaling fashion.

There's a variant of this concern when we bring in on-premise compute. With on-premise compute, *anyone* can play a keeper for data that is on-premise, where they play middleman between the party that's hosting the data on-premise and the buyer of the data. However the keeper won't be able to make the data available if the data host doesn't make it available. In this case, the discussion of the

previous section still holds: the host won't either make it available because of privacy, or they will make it available because of market forces.

17. Appendix: Decentralization, Consistency, and Scale

This section provides context on the constraints in design of the Ocean blockchain.

17.1. Decentralization and Fault Tolerance

There are three levels of fault tolerance in increasing levels: crash faults, Byzantine faults, and Sybil attacks. Ocean must handle them all. Let's elaborate.

Crash tolerance is when a network can tolerate crash faults, such as hard drives failing. This is acceptable in environments when one is trusting a single administrator of the system.

Byzantine fault tolerance (BFT). In a blockchain setting, each full node is effectively another sysadmin. We need to be tolerant to several of these sysadmins acting maliciously against the system; this is modeled as assuming they will attempt arbitrary behavior on the system. A BFT system handles such faults; typically f malicious nodes out of a population of $n \geq 3f + 1$ [Lamport1982]. In traditional BFT settings, the participants are all identified by their public keys. Each participant gets one vote whether a transaction or block of transactions enters the system. Each sysadmin has their list of allowed public keys, aka the "keyring". One needs *permission* to get on this list using some mechanism not built into the network, such as knowing the right person.

Sybil tolerance. Ocean needs to be *permissionless*, i.e. where anyone can run a node in the network without getting approval from some list. One cannot do this by simply making the keyring public and editable, because a malicious actor could come and replicate themselves many times, thereby swamping the votes and overtaking the system. This is a "Sybil attack"; what we call "attack of the clones". One solution is to change from "one actor one vote" to some other resource like "one electron one vote", as in Bitcoin's a Proof-of-Work setup [Nakamoto2008], assuming equal hashing efficiency. Or, traditional Proof of Stake setups give "one token, one vote". These Sybil-tolerant mechanisms can be framed as **power protocols** which capture *relative* power for each actor [Filecoin2017]. Depending on the consensus protocol, a system can tolerate up to 51% malicious power. Figure 17 illustrates.

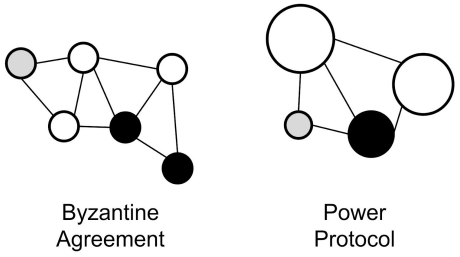

Figure 17: Modeling faults - Byzantine and Power protocols. Actors (defined by public keys) have equal votes in a Byzantine setting but not Power protocol setting.

17.2. Consistency / Finality

A practical definition for **consistency** in a blockchain setting is: *it's consistent if it prevents double spends*. That is, it won't let me successfully send the same tokens to two different addresses at once. If I attempt this, the transaction will not go through.

We can frame whether a transaction has gone through in a deterministic "sure" way, or a more probabilistic way - some protocols yield **finality** while others have **statistical convergence** under polynomial attacker assumptions. In this context, a blockchain is actually a directed acyclic graph (DAG), but for finality the algorithm must decide which fork of the directed acyclic graph is the main chain. This could be with the longest chain rule as in Bitcoin [Nakamoto2008], a weighted approach like modified GHOST protocol in Ethereum [ModifiedGHOST2018], or otherwise.

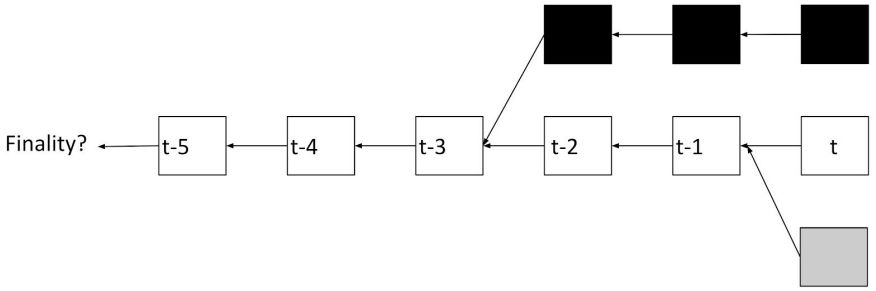

Figure 18: Finality in context of a blockchain - which fork is the main chain?

17.3. Scale and the DCS Triangle

It's well understood that Bitcoin and most other public blockchains have scaling issues. For example, the core Bitcoin network has <10 transactions per second. Systems so far have improved scaling by giving up consistency (e.g. [IPFS2018]) or loosening up decentralization (e.g. [BigchainDB2018]).

Overall, there is a tradeoff among **d**ecentralization, **c**onsistency, and **s**cale - the **DCS Triangle** [McConaghy2016]. However, we see this more as an engineering challenge than a fundamental constraint, and are hopeful about efforts to improve scaling while retaining sufficient consistency (preventing double spends) and decentralization (Sybil tolerance).

Scaling efforts include:

- **Improving the consensus protocol**. Examples: Ouroboros [Kiayias2017], Dfinity [Hanke2018], Red Belly [Crain2017], OmniLedger [Kokoris2018]
- **Sharding** so that each node only has a fraction of the compute or data workload. Example: [EthSharding2018]
- **Independent networks / chains** with "**glue**" connectors. Examples: Interledger [Thomas2015], Cosmos [Kwon2017], PolkaDot [Wood2016], Plasma [Poon2017], TrueBit [Teusch2017])
- **"Layer 2" payment channels.** Examples: Lightning [Poon2016], Raiden [Raiden2018]

18. Appendix: Computational Integrity

This section describes some popular approaches to computational integrity, as part of Ocean's overall services integrity framework.

18.1. Probabilistic Checkable Proofs (PCP)

PCP implies that a verifier can send any computation task C and input i to a prover and the prover outputs o. The proof runs in a randomized way through a set of interactions such that if o is correct the verifier will set to True else the verifier will reject all interactions but with bounded error. This error means that there is a small or negligible probability that the verifier could mistakenly view the wrong answer. What distinguishes the PCP protocols is the availability of compilers, as well as the

concrete efficiency of verifiers, but these systems are limited to small executions, and are extremely expensive. It might be useful only for special purpose applications.

In PCP frameworks, the computations are represented by Boolean circuits (AND, OR, and NOT gates). Because of the property of turing completeness in boolean circuits, it is easy to describe any deterministic computation/verification using boolean circuits. Thus the verifier will submit the circuit and inputs $i_1, i_2, \ldots, i_N$ to a prover. The prover executes the circuit $\widehat{C}$ and produces a "claim" or output o including the transcript which contains the assignment for each wire in the circuit after computation. The verifier can do the correctness check of the circuit output by executing any of the gates using the transcript and inputs. The prover, however, shouldn't send all transcripts to the verifier to satisfy the property of zero-knowledge. Therefore, the verifier queries particular locations in the prover's transcript. PCP can be classified into three approaches.

1. **Heuristic interactive based approach** uses higher random order of messages between verifier and prover where verifier sends a lot of queries to prover then gets the returned results. Therefore the verifier checks if there is any contradiction among queries results. Examples for interactive proofs such as CMT [Cormode2012], Allspice [Vu2013], and Thaler [Thaler2013].

2. **Commitment based approach** has two rounds. In the the first round, the verifier enforces the prover to commit particular transcript or proof. In the second round, the verifier queries different locations in the commited proof. Examples are Pepper [Setty2012], Ginger [Setty2012b], and Zaatar [Setty2013].,

3. **Encrypted queries approach** is a more theoretical approach where the verifier encrypts the queries beforehand, asks locations, retrieves prover's results and checks them using PCP. In the last few years there has been development and implementation activities such as Pinocchio [Parno2013].

18.2. Zero-Knowledge Proofs

Zero-knowledge based protocols preserve the privacy of inputs as well as providing transparency to multiple parties in the system. For instance, if a prover sends a transaction to a blockchain network, it might be better if the prover doesn't reveal any information about a particular transaction to the verifier, and also to keep the verification time small. Overall, the goal of zero-knowledge proofs is to provide the integrity, privacy, and succinctness (the proof can be verified in time t). These properties can be achieved using polynomials, pairing-based proof systems, and homomorphic encryption.

A zk-SNARK [Bitansky2012] is a variant of a zero-knowledge proof that enables a prover to succinctly convince any verifier of the validity of a given statement and achieve computational zero-knowledge without requiring interaction between the prover and any verifier. zk-SNARK stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge". It uses homomorphic encryption or hiding that concretely hide the actual data even after performing addition, or multiplication operations using the modulo and discrete logarithms. For instance, if "x, y" are two numbers, we can perform the addition and multiplication on modulo as follows:

$$A = x \bmod m \quad (1)$$
$$B = y \bmod m \quad (2)$$

$$A + B = (x + y) \bmod m$$

$$A \,.\, B = (x + y) \; mod \; m$$

The *discrete logarithm* term means that we need to apply the modulo operation on the logarithm i.e *X = log2 8* ***(mod 13)***. The word discrete refers to a discrete group {1,..,p-1} where p is a modulus and only an integer number. The primitive roots or generators for a modulus can generate all elements in the discrete group as shown below. If there is a loop where the remainder is repeated after a cycle, this group is called a cyclic group for a particular generator. For example, 3 and 5 are called **generators** for the group {1,...,7}, and is a cyclic group because it repeats itself every 7 steps, as shown below in the following table:

$$b^x \; mod \; 7 \; = \; remainder$$

Table 3: Generators

b	b^1 mod 7	b^2 mod 7	b^3 mod 7	b^4 mod 7	b^5 mod 7	b^6 mod 7	b^7 mod 7
1	1	1	1	1	1	1	**1**
2	2	4	1	2	4	1	**2**
3	**3**	**2**	**6**	**4**	**5**	**1**	**3**
4	4	2	1	4	2	1	**4**
5	**5**	**4**	**6**	**2**	**3**	**1**	**5**
6	6	1	6	1	1	1	**6**

If the modulus is at least greater than 300 digits, calculating the discrete logarithm will be considered as computationally expensive/hard to find the valid exponent. The cyclic group is the basis of discrete logarithm crypto systems where Z^*_p refers to cyclic group for modulus p. The * means that the cyclic group starts from 1 to p-1. Moreover, zk-SNARK relies on Polynomials to define the computations. Polynomials are sum of multiple terms with different exponents. For example, Lines are polynomials with degree (the largest exponent) 1. For two polynomials, they can intersect in less than or equal to *N* points, where *N* is the degree of polynomial. Polynomials have a property known as Schwartz-Zippel Lemma [Schwartz2018] in which "[f]or multiple polynomials of degree *N* they can agree on at most *N* points". This means that If the prover (Alice) is able to compute a puzzle in terms of a polynomial language, the verifier (Bob) will be convinced by evaluating a small, randomly chosen point in this polynomial.

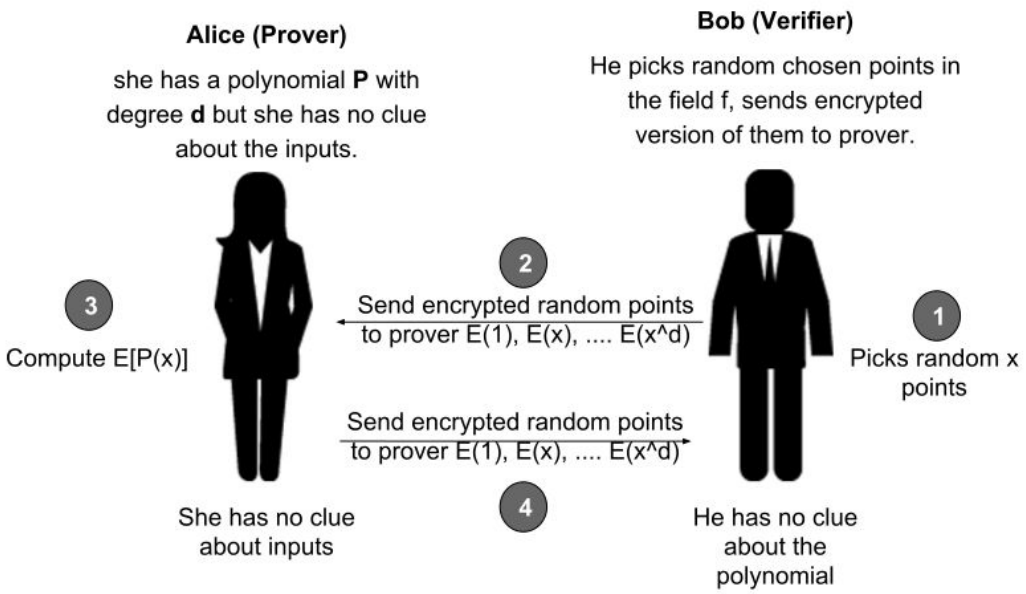

Figure 19: Using polynomials, homomorphic encryptions and pairing

As shown in Figure 19, the protocol preserves the blindness of computations, where the prover has no clue about the inputs and the verifier doesn't learn anything about the computation itself. But what could happen if Alice doesn't follow the protocol and she sends some cheated results. What assumptions can guarantee that she will send the right polynomial evaluation. The key point is to use pairing-based cryptography which assumes that Bob sends encoded, randomly chosen elements $((a_1,b_1), (a_2,b_2), ..., a_d,b_d)$ using α pair. For each coefficients tuple (a_i,b_i), the pairing between two elements should be computationally expensive in order to hide the prover's secret (polynomial). The role of Alice is to send her results encoded with the same α pair with small negligible error over Bob's choices.

The question now is how to convert a typical computation into a polynomial's world? The answer is using Quadratic Arithmetic Program (QAP) [Gennaro2013]. To keep the zero-knowledge maintained, the prover masks the polynomials using random shift in which the verifier accepts the statement without revealing any information about the computation/polynomial. Zcash [Sasson2014] uses zk-SNARK as a proof system for private transactions. The pairing, key generation, and polynomials setup is called the trusted setup because you have to trust the party who is generating those keys and hidden parameters, as well as destroying them after computation.

18.3. Multi-party Computation

Secure multiparty computation (MPC) addresses the problem of jointly computing a function among a set of mutually distrusted parties. It has a long history in the cryptographic literature, with its origins being found in literature in the mid 1980s. The basic scenario is that a group of parties wish to compute a given function on their private inputs, while still keeping their inputs private from each other. The goal is that the output of the protocol is just the value of the function, and nothing else is revealed. In particular, all that the parties can learn from one another is what they can learn from the output and their own input.

Informally speaking, the most basic properties that a multi-party computation protocol aims to ensure are:

- **Input privacy** - The information derived from the execution of the protocol should not allow any inference of the private data held by the parties, except for what is revealed by the prescribed output of the function; and,
- **Correctness** - Adversarially colluding parties willing to share information or deviate from the instructions during the protocol execution should not be able to force honest parties to output an incorrect result.

Secure multiparty computation can be leveraged to obtain a new paradigm of security: encryption of data while in use. For example, consider the case in which Alice holds an encryption key and Bob holds an encrypted database, and the parties wish to run an SQL query on the database without ever decrypting it. This exact problem can be cast as a two-input function, and thus can be securely computed. In fact, in this case input privacy means that the result of the SQL query is revealed and nothing else! Thus, SQL queries are computed while the database is encrypted, thereby keeping the database secure, even while it is being used.

19. Appendix: Data Integrity

This section describes some popular approaches to data integrity, as part of Ocean's overall services integrity framework.

19.1. Data Availability via Proof-of-Space-Time (PoST)

Filecoin [Filecoin2017] introduced the notion of PoST. We can use that directly in Ocean. However, that solution is overkill for our needs, as we do not need to prove that the data was stored uniquely over time. We just need to prove that the correct data was made available at this point in time. We do not care where it came from; unlike Filecoin, it's ok if it was served up from S3 copy at the last minute. That said, PoST is coming, and data on the Filecoin network is then a good fit for Ocean.

We can expect similar proofs coming from other decentralized storage networks, such as Ethereum Swarm [Trón2018].

19.2. Data Availability via Dedicated PoW Blockchain

Teusch recently introduced a new approach for data availability [Teusch2017b]. It uses a dedicated PoW blockchain where each keeper must actually promise to serve up each dataset. This means that each keeper is asked to perform a tremendous amount of work. However, it's a starting point; more efficient approaches are coming[10].

19.3. Data Availability via Challenge-Response

This is another possible approach that has potential to be more efficient. However, we add a disclaimer that this is a work-in-progress and will need much deeper vetting.

In this approach, the receiver can contend by querying more senders and penalizing the initial sender if the initial sender is not in the majority. The party found to be wrong loses stake.

Here is the protocol:

[10] Personal communication with J. Teusch, Jan. 2018

1. The receiver issues a “contention request” and posts stake.
2. The network (keeper nodes) randomly chooses two more providers of the data; each of which must make the data available.
3. The network computes a random seed S, and sends S to both initial provider, the two new providers, and to the receiver.
4. Each provider uses S to select a subset of the data, and compute the hash of that subset, and returns the result to the network.
5. From that the network determines whether the receiver was lying (or wrong), initial sender was lying, or other. Whoever is lying (or wrong) loses stake.
    a. Receiver was lying (or wrong) if the initial sender hash lines up with at least one other sender.
    b. Otherwise, initial sender was lying (or wrong) if hash of other two senders lines up, and does not line up with initial sender.
    c. Otherwise, one of the new providers was lying (or wrong); in this case the network randomly selects two more providers of the data, and re-initiates the process.
6. If there is still no resolution after 6 repeats, the contention stops and no one loses stake. This might happen, for example, if there are a small number of providers of that data asset.

This protocol is a bit like TrueBit’s challenge-response mechanism. However, it has fewer constraints because it does not need to find the instruction that went awry; therefore it has less complexity.

19.4. Data availability via Proof-of-Replication (PoRep)

Proof of Replication [PoRep2018] provides strong proof of retrievability and strong defense against data generation attack (A prover uses random seed to generate junk data rather than storing the actual data). The proof has two phases, the setup phase, and challenge phase. The setup phase returns a unique identifier for replica with an optional commitment. This phase includes slow encoding and fast decoding (verification) that is based on *verifiable delay functions*. The Challenge phase includes prove step and verification step. The prove step that takes challenge C and generates proof of storage R and the verification step takes the challenge, data identifier and commitment then outputs accept or reject.

What distinguishes this proof is that even if an adversary passes the verification step this implies that there is some entity that the adversary communicates with where he can extract the original data which implies the proof of retrievability. Moreover, PoRep preserves the proof of storage using prover state where two replicas can be stored in terms of snapshots. A verifier takes the prover state, splits it into two parts and from each of which the prover can get the original data. This is relying on the cryptography perspectives where each of data replica is XORed with the hash of the other replica then the whole data are encrypted and stored in one state. Also, the proof uses a strong encoding approach called Depth Robust graph which provides data availability in case of data deletion.

20. Appendix: Block Rewards Schedule

The overall supply of Ocean Tokens (Ọ) is fixed. 55% of Ocean Tokens are pre-mined, to build Ocean software (e.g. with developer bounties), incentivize the community, and more. The remaining 45% are for block rewards.

The block rewards schedule is:

$$F(H,\ t)\ =\ 1\ -\ (0.5^{\ t/H})$$

where

- $F(H,\ t)$ is the fraction of all block reward tokens that have been released after t years
- H is the half-life, in years. Half-life is the time taken for 50% of remaining supply to be released.

We use a half-life of ten years, i.e. H =10. This is longer than most comparable systems, because it can take several years for internal enterprise processes to prepare their data assets for sharing; we want to give them breathing space.

Figure 20 and Table 4 illustrate the percentage of mining tokens released over time.

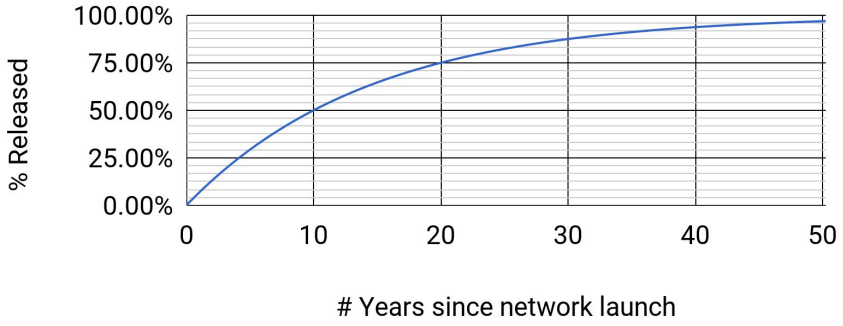

Figure 20: % mining tokens released over the next 50 years

Table 4: % mining tokens released over the next 150 years

# Years	0	1	2	5	10	25	50	100	150
% Released	0.0%	6.7%	12.9%	29.3%	50.0%	82.32%	96.88%	99.90%	99.9969%
% Left	100.0%	93.3%	87.1%	70.7%	50.0%	17.68%	3.13%	0.10%	0.0031%