Ocean Protocol:
A Decentralized Substrate for AI Data & Services
Technical Whitepaper

Ocean Protocol Foundation[1]
with BigchainDB GmbH[2] and Newton Circus (DEX Pte. Ltd.)[3]

Version 2019-FEB-19

Abstract

This paper presents Ocean Protocol. Ocean is a decentralized protocol and network of artificial intelligence (AI) data/services. It aims to spread the benefits of AI, by unlocking data while preserving privacy. It helps power marketplaces to buy/sell AI data & services, software to publish and access commons data, and AI/data science tools to consume data. Ocean does decentralized orchestration: at its core are decentralized service agreements and decentralized access control, which execute on decentralized virtual machines. This allows connection to, monetization of, and curation of arbitrary data services. On that, Ocean adds network rewards to incentivize data sharing, including privacy-preserving data commons.

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

Contents

1. Overview

1.1. 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 spread the benefits of AI by equalizing the opportunity to access data. To reduce this to a practical goal, our aim is to develop a protocol and network—a tokenized ecosystem. 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.

This is not easy, as there are several challenges:

- How do we make data available for use by AI, without losing control of the data (*data escapes*)? More generally, how do we resolve the tension between wanting more data for better AI models, with the individual right to privacy?
- How do we incentivize putting data in the commons? This is actually a re-statement of the classic paradox "information wants to be *free,* and *expensive*." The physics of bits make it nearly free to copy and spread. Yet the right information in the right place changes your life. [Information2019]. Can we address this? Can it reward relevant data over spam?
- How can the system not only self-sustain but actually improve over the decades, without guidance by a centralized actor?
- How do we spread the benefits and control of this ecosystem?

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

To answer the first two questions above, Ocean has functionality to reconcile spread of data with privacy and the commons. It has infrastructure for service agreements and access control, which allows compute to be brought to the data. It has cryptographic proof-of-service and network rewards to incentivize supply of relevant AI data & services.

For the last two questions, the Ocean design uses on-chain bounties to ensure long-term sustainability and improvement as a public network. Finally, Ocean will ensure a spread of benefits and control by going permissionless.

The next four subsections provide a brief introduction to these four solutions. Later sections will provide greater detail for each.

1.1.1. Service Agreements & Access Control

This addresses: "How do we make data available for use by AI, without losing control of the data (*data escapes*)?" and related questions.

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 made available for consumption via the network.

At the heart of this are decentralized **Service Execution Agreements** (SEAs) and decentralized access control, which together power data service supply chains. This allows connection to, monetization of, and curation of arbitrary data services. This allows **compute to be brought to the data**, thereby solving the concern of data escapes since sensitive data never needs to leave the premises (just the results of computation).

This baseline functionality will coincide with the deployment of the initial physical network. Ocean's main target users—AI researchers and data scientists—expect reasonable network performance and usage costs. For this reason, Ocean will initially ship as a Proof-of-Authority (PoA) network called "Pacific", where each node is a Parity Ethereum client. We refer to nodes and node operators as "keepers" [Zurrer2017], to highlight that they are different than traditional blockchain "miners". The keepers community will collectively govern adding/removing keepers and smart contract upgrades.

1.1.2. Proof-of-Service, and Incentives

This addresses: "How do we incentivize putting data in the commons?" and related questions.

On top of the SEAs infrastructure, Ocean adds network rewards to incentivize data sharing, which will lead to a commons. Specifically, Ocean incentivizes participants to submit, refer, and make available (provably) quality AI data & services, via a new construction that we call a Curated Proofs Market (CPM). A CPM 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.
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 power of the crowd via a curation market. 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" (aka *drops*) in that dataset/service. The earlier that an actor stakes or bets on a given dataset/service, the more drops they get for the amount staked, and in turn the higher the reward.

To avoid people gaming the reward system, only stakeholders provably making high-quality data/services available will be able to reap rewards. Network rewards for a given dataset/service are distributed based on amount of stake in that dataset/service, and its actual popularity. In other words, CPMs instantiate the goals of *verification* and *virality*.

To the best of 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.

1.1.3. On-Chain Bounties

This addresses the question "How can the system not only self-sustain but actually improve over the decades, without guidance by a centralized actor?"

Ocean will be a utility network to serve the public at large. Therefore it must be self-sustaining; and the technology and ecosystem need steady improvements, by the community and for the

community. To this end, the Ocean network will have an on-chain bounties system where a portion of the network rewards are for technology improvement and ecosystem development. We elaborate on this in a later section.

1.1.4. Permissionless

This addresses the question "How do we spread the benefits - and control - of this ecosystem in a way that balances needs among various stakeholders and towards all of society?". Here, Ocean will go from PoA to permissionless, while maintaining performance. Technology options include ETH2.0, Parity Substrate, and Cosmos SDK. We elaborate on this later.

1.2. Ocean Tokens

Ocean Tokens are the main tokens of the network. We denote Ocean Tokens as "Ọ" or with ticker symbol OCEAN. They are used in several ways.

First, Ocean Tokens are used as a **unit of exchange** for buying and selling data/services. A marketplace would price data/services in Ocean Tokens (OCEAN), or any other currency of the marketplace's or vendor's choice, such as USD, EUR, ETH, or DAI. It the latter, the marketplace would use a crypto exchange to convert just-in-time to OCEAN. Therefore the Ocean network would only see OCEAN. We explicitly chose a one-token design over two-token design for simplicity, and to help equalize the access to upside opportunities of owning assets. The appendix has details.

Second, Ocean Tokens are used for **staking**. This includes staking in *each* given dataset/service, and introduce a long tail of additional tokens called *drops*. Drops are derivative tokens of Ocean tokens denoted in "Ḍ". Each dataset would have its own derivative token. For example, 100 drops of stake in dataset X is "100 ḌX". Drops relate to Ocean Tokens via curation markets' bonding curves, which determine the exchange rates between them for different dataset/services.

Finally, Ocean Tokens are used in dispensing **network rewards**, according to Ocean's inflation schedule.

1.3. Paper Organization

The rest of this paper is organized as follows.

Section 2 provides context with discussion on use cases, stakeholders, and data ecosystem.

Sections 3 and 4 describe Ocean core architecture and behavior, respectively.

Sections 5-7 describe core components of Ocean for Ocean V1 (Q1 2019). These core components enable the construction of data marketplaces and data commons, and integration to data science tools.

- Section 5 - Identity & metadata
- Section 6 - Service execution agreements & access control
- Section 7 - Initial PoA network deployment

Sections 8-12 describe components beyond V1 that realize the full vision of Ocean. The roadmap [OceanBlog_Roadmap2019] has schedule details.

- Section 8 - Proofs-of-service
- Section 9 - Incentives
- Section 10 - On-chain bounties
- Section 11 - Clans
- Section 12 - Permissionless network deployment

Section 13 describes outstanding concerns. Section 14 concludes. Section 15 has references.

The Appendices give information: the components of secret store and computing services; how the token design addresses key goals; FAQs and concerns, including arbitration, IP rights, and data escapes; and data pricing strategies.

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 the current 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. This also goes for future updates to the network.

2. Context

2.1. Use Cases

These use cases and others guide our design.

2.1.1. Proprietary Data: Autonomous Vehicles

A leading use case for proprietary data is autonomous vehicles (AVs).

The RAND Corporation calculated that 500 billion to 1 trillion miles driven are needed to get AI models accurate enough for production deployment of AVs [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 this goal in our minds, we built such a prototype together with TRI [BigchainDB_TRI_2017].

However, a single data marketplace may itself be centralized, which means 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.

AV 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.

The TRI collaboration helped to spark the creation of MOBI [Mobi2019], a blockchain consortium of automakers representing 80% of world auto production. We collaborate with MOBI towards an AV data marketplace, and more.

2.1.2. 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.

Munich-based ConnectedLife [ConnectedLife2018], along with medical researchers at the National Neuroscience Institute of Singapore, specialist professionals and hospital groups in Singapore, Germany, and elsewhere are working 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 concerns and 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.

2.1.3. 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. A second problem is that there are costs to hosting data even if it is free to consume; so commons datasets can go offline if funding is not secured. Ocean's network rewards are a new way to pay for this cost, and therefore solve the issue.

2.2. Stakeholders

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

Table 1: Key stakeholders in Ocean ecosystem

Stakeholder Role	What value they can provide	What they might get in return
Data/service provider, data custodian, data owner	Data/service. Suppliers to the market.	Ocean Tokens for making available / providing service
Data/service curators	Signal the relative value of data/service	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, e.g. data scientist	Ocean Tokens, signals to curators	Data/service (market's demand)
Keepers	Correctly run nodes in network	Ocean Tokens for chainkeeping
Marketplaces, data commons	(Optional) Connect other actors. Run metadata store, secret store	Transaction fees

2.3. On Pricing

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 without needing to pay.

Priced Fungible Data/Services. Some data is exchangeable with decent liquidity, for example Electrocardiogram (ECG) data for 100 people in Singapore is the roughly 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; for example a single dataset with an exclusive license. When that's the case, pricing options include fixed price, auction, and royalties. Each has pros and cons.

The Appendix elaborates on possible pricing strategies.

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 (fixed, royalties, auction, etc.).

2.4. Ocean is Decentralized Orchestration

In the world of big data, there are frameworks for large compute jobs (or streams) like MapReduce, Hadoop, Spark, and Flink. The user configures a directed acyclic graph (DAG) of compute and storage. Often those DAGs are a simple compute pipeline. The framework orchestrates the work. Figure 1 illustrates.

Since AI loves data, AI people use these frameworks in the course of building models (or AI-tuned variants like TensorFlow).

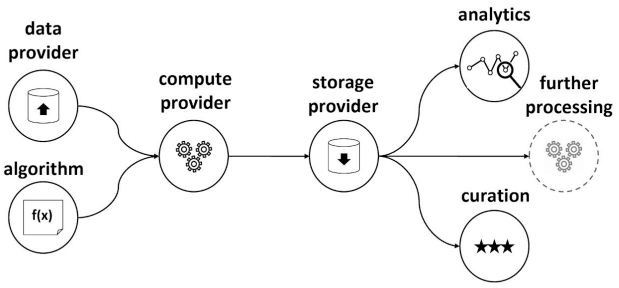

Figure 1: An example compute DAG. Orchestration frameworks execute compute DAGs.

Ocean does *decentralized orchestration*: it orchestrates the execution of compute DAGs in a decentralized setting. In Ocean, the compute DAG is specified by a Service Execution Agreement (SEA). A SEA is the decentralized equivalent of Service Level Agreements (SLAs) found in big data environments.

Figure 2 shows the mechanics of Ocean orchestration. A SEA Contract handles each step in the DAG compute; and compose to one higher level SEA for the whole DAG. All the SEAs have guaranteed execution due to running on the blockchain-based Ocean network (bottom). Once you deploy them, they simply go; a single entity can't intervene and stop them (unless it's within the definition of a SEA).

The decentralized setting means that SEAs unlock new capabilities, like privacy-preserving compute. Arbitrary forms of compute can be brought to the data. Therefore private data—the most valuable data—never needs to leave the premises, while at the same time value can be extracted from it.

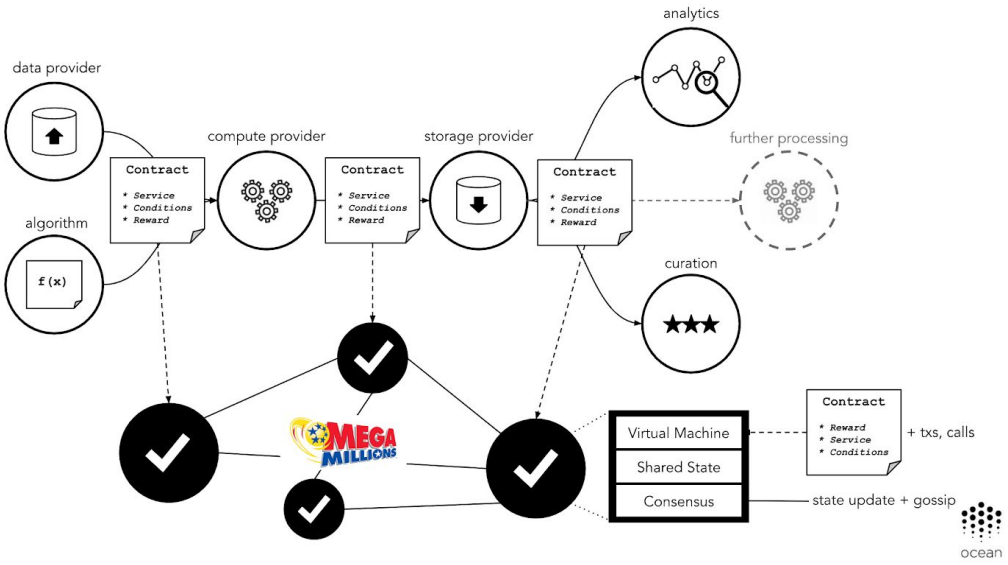

Figure 2: The mechanics of Ocean orchestration

We envision that Ocean will have adapters to leverage existing front and back-end components of existing orchestration frameworks. For example, people might specify a SEA within an Apache Spark context; but then the SEA will be executed within Ocean.

Template SEAs make commonly-repeated tasks easier to deploy.

2.5. Ocean as Inter-Service Network

In executing compute DAGs, each step might have one of many providers. There will be providers for data, algorithms, compute, storage, etc. Each of these may be centralized behind a firewall, centralized on the cloud (e.g. AWS S3 or EC2), or fully decentralized as their own networks (e.g. FileCoin or Golem, Enigma).

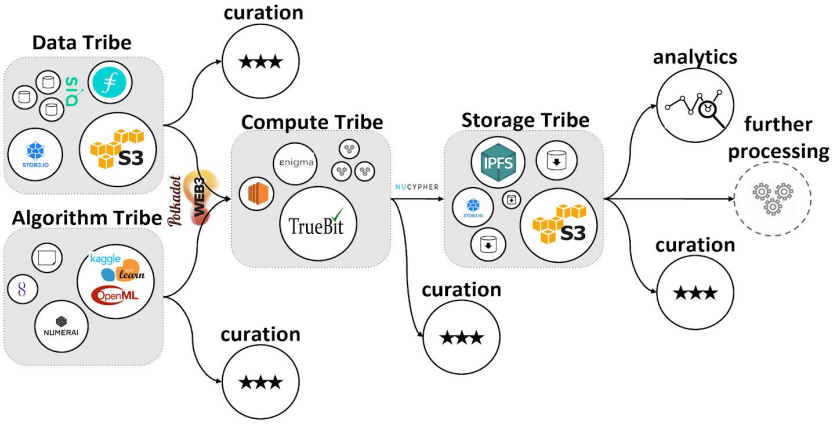

Figure 3: Ocean acts as inter-service network[4]

[4] In this image, logos shown are example services that could be integrated. It does not imply that the services are currently integrated, nor does it imply a business relationship.

Ocean doesn't provide these services itself, it simply connects them. This makes it an *inter-service network*, specifically for compute DAGs on big data / AI. Figure 3 illustrates.

2.6. Decentralized Federated Learning

Ocean and SEAs enable *decentralized* federated learning. In traditional federated learning [Konečný2016], the model is learned across many private data silos - a great starting point for data privacy. The challenge is that a centralized entity orchestrates the learning and stores the final model. This opens up privacy attack vectors. For example, being the man in the middle of orchestration means it can get signals about information flow. Furthermore, fully controlling the final model is dangerous if that model can be reverse-engineered to get PII. It also means that the orchestration task can be halted midway by a single controlling entity, for example a government overstepping.

Decentralized federated learning learns also across many private data silos, with the benefits for doing so. In addition, no single entity controls the execution of the federated learning task; and if desired, control of the final model is also decentralized. This addresses privacy attack vectors, spreads control, and guarantees execution of a SEA.

OpenMined [OpenMined2019]is a related effort that focuses solely on decentralized federated learning. It is complementary to Ocean; we look forward to when it plugs into Ocean as a service.

3. Ocean Core Architecture

3.1. Introduction

Ocean's Decentralized Service Execution Agreements (SEAs) and decentralized access control combine to power data service supply chains. This allows **compute to be brought to the data**, thereby solving the concern of data escapes since sensitive data never needs to leave the premises. This allows connection to, monetization of, and curation of arbitrary data services.

This section describes the architecture and components of these foundational elements of Ocean. They're described in a more precise software form than later sections because they've already been built and are running on a testnet.

This section provides a high-level overview. Further details will be found in the Ocean software documentation [OceanDocs2019] and Ocean Ocean Enhancement Proposals (OEPs) [OEPs_2019].

3.2. Architecture Overview

The Ocean Protocol network architecture is implemented based on OEP3 [OEP3_2019].

Figure 4 shows the following components (from top to bottom):

- **Frontend (Tier 3)** - Application implemented using HTML + JavaScript + CSS, running in the client side (user's browser).
- **Data Science Tools (Tier 3)** - Applications executed by data scientists, typically getting access to the Ocean data and executing algorithms on top of that data.
- **Aquarius (Tier 2)** - Backend application providing metadata storage and management services. Typically executed by marketplaces.
- **Brizo (Tier 2)** - Backend application providing access control services, i.e. consumer access to publisher data or compute services. Typically executed by publishers.
- **Keeper Contracts (Tier 1)** - Solidity smart contracts running on a decentralized Ethereum Virtual Machine (EVM).

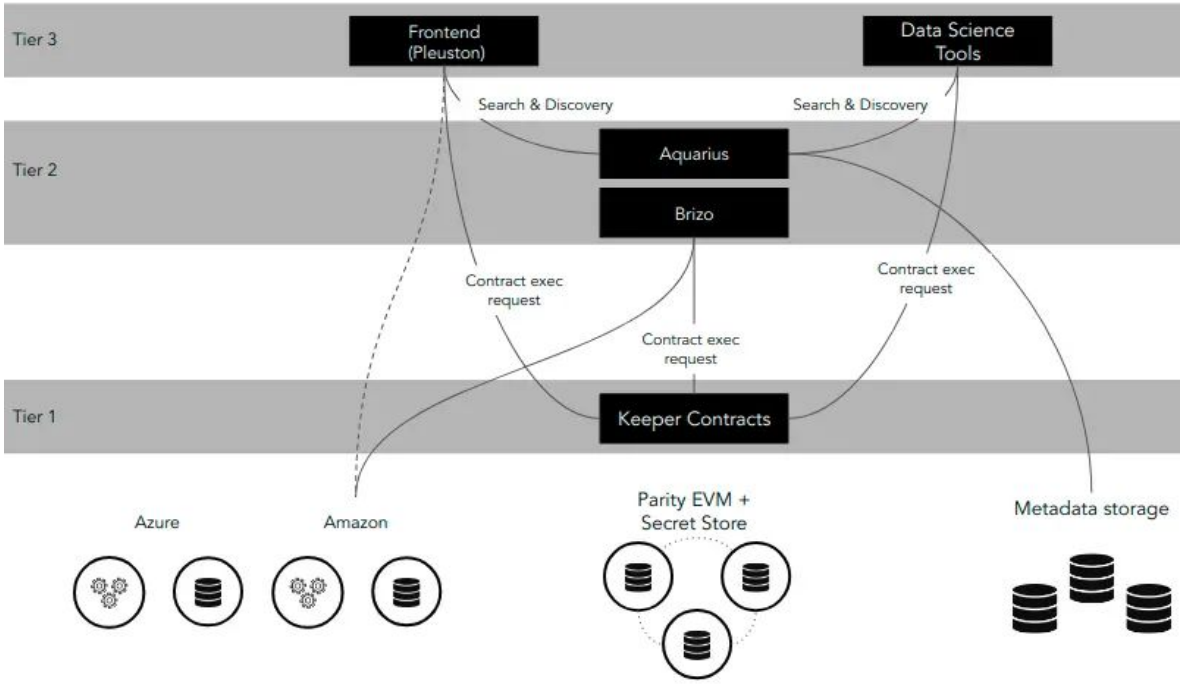

Figure 4: Ocean Protocol network architecture

Critically, neither metadata nor data is stored on the blockchain network (Ocean keeper network). Marketplaces store the searchable metadata. Publishers are incentivized to register their work on several marketplaces.

Ocean architecture has similarity to that of the Domain Name System (DNS). In the DNS, a global singleton "thin" lower-level store holds a mapping from domain name (e.g. amazon.com) to IP address (e.g. 12.34.56.78). One level up is a diverse set of marketplaces like GoDaddy and Gandi.net, where "thick" metadata including personally identifiable information (PII) is held, such as customer credit cards and contact information. Ocean has a global singleton "thin" lower-level blockchain store holding registered Decentralized Identifiers (DIDs) and associated permissions and service agreements. One level up is a diverse set of marketplaces for data and compute services, where "thick" metadata including PII is held.

Just as GoDaddy, Gandi.net and the like are each incentivized to hold a large number of domain addresses and act accordingly, marketplaces in Ocean will have the same incentives, which means that there will probably be many Ocean marketplaces having large supplies of data and services.

We now elaborate on each Ocean component, for Tiers 3, 2 and 1 respectively.

3.3. Tier 3 - Application Layer

3.3.1. Pleuston Frontend : Marketplace Template

Pleuston [OceanPleuston2019] implements a template for a marketplace, which includes example usage of lower-tier components. It helps demonstrate some of the capabilities in Ocean. It is not a final product, but can be used as reference to implement further marketplaces using the existing code. The Pleuston application implements the following high-level functionality:

- Publishing - Allows the user to publish new assets to the network.
- Consuming - Allows the user to list and consume published assets.
- Marketplace - Enables complex interactions and advanced capabilities like:
    - Publishing of Assets existing in cloud providers
    - Search and discovery of assets
    - Filtering
    - User registration using specific KYC processes (to be added)

Marketplaces communicate with the following external components:

- **Smart Contracts** enable interaction with the Ocean Keeper Contracts that provide the market business logic. This integration is implemented using the Ethereum Javascript API (web3.js) [Web3js2019].
- **Aquarius** and **Brizo**, for metadata management and access to assets (details are below). This communication is implemented with HTTP APIs.

The frontend application will subscribe to the EVM transaction log and listen to the events emitted from the smart contracts, enabling the receipt of asynchronous messages. This will facilitate the triggering of automatic actions when some events are raised (e.g. the request of an asset is triggered automatically when the purchase has been confirmed).

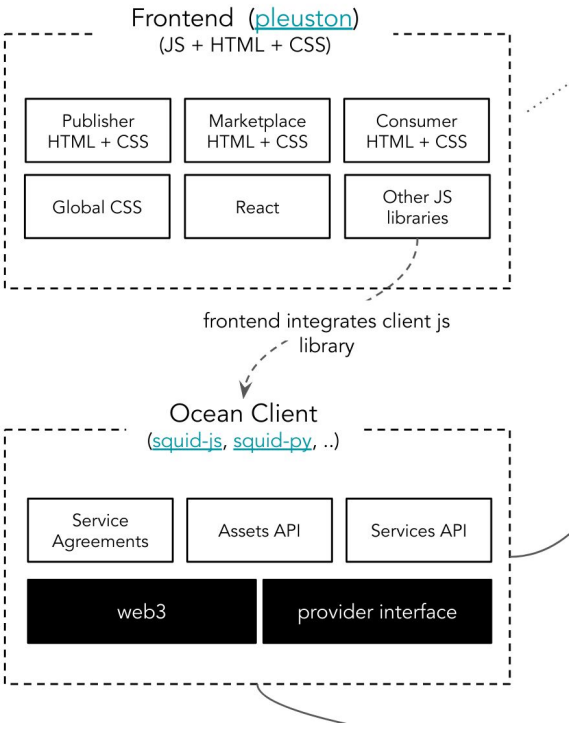

Figure 5: Frontend High-Level Architecture

The Squid library shown in Figure 5 encapsulates the logic to deal with the Ocean components (such as Keeper nodes and Aquarius nodes). Squid libraries (written in various programming languages) are part of Tier 2. [OceanDocs2019] has further documentation on setting up marketplaces.

3.3.2. Data Science Tools

Data science tools are the interface to Ocean used by AI researchers and data scientists. Typically written in Python, those tools and libraries expose a high-level API allowing one to integrate Ocean capabilities in various computation pipelines.

Ocean Manta Ray notebooks provide a guided tour of Ocean Protocol in an interactive Jupyter Notebook environment. More information is at [OceanDataScience2019].

3.4. Tier 2 - Protocol Layer

Includes all the high-level libraries to interact with Ocean Protocol, and the enabler interacting with the Keeper Contracts.

3.4.1. Squid Libraries : Main API to Ocean

Squid is a High Level specification API abstracting the interaction with the most relevant Ocean Protocol components. It allows one to use Ocean capabilities without worrying about the details of the underlying Keeper Contracts or metadata storage systems.

The complete specification of the Squid API is on GitHub [SquidApi2019]. This can be considered the formal specification to interface with the Ocean ecosystem.

Initially the Squid API is implemented in JavaScript [SquidJs2019] (used in Pleuston), in Python [SquidPy2019] (used in the data science tools), and in Java.

3.4.2. Aquarius : Metadata Management

Aquarius [OceanAquarius2019] is a Python application running in the backend that enables metadata management. It abstracts access to different metadata stores, allowing Aquarius to integrate different metadata repositories. The OceanDB plugin system can integrate different data stores (e.g. Elasticsearch, MongoDB, BigchainDB) implementing the OceanDB interfaces.

Figure 6 shows the high-level architecture of Aquarius.

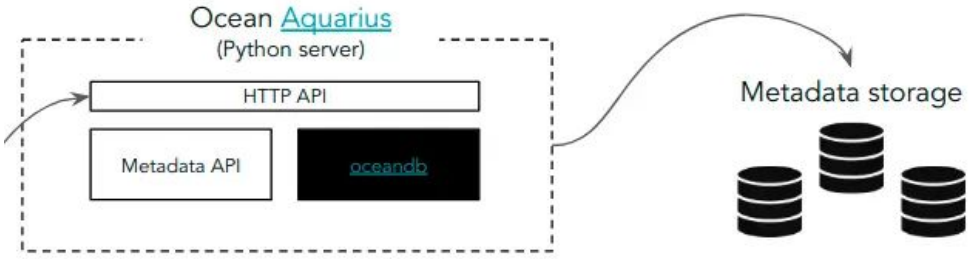

Figure 6: Aquarius High-Level Architecture

3.4.3. Brizo : Capabilities for Publishers / Service Providers

Brizo is a component providing capabilities for publishers. Brizo interacts with the publisher's cloud and/or on-premise infrastructure. Figure 7 illustrates.

The most basic scenario for a publisher is to provide access to the Assets the publisher owns or manages. In addition to this, other extended services could also be offered, e.g.

- Computing on top of the data without moving the data
- Storage services for newly derived assets

- Gathering of Service Proofs - Enables different kinds of service proofs from different providers. For example - allowing the retrieval of receipts from cloud providers to validate service delivery.
- On-Chain Access Control - Brizo is in charge of the on-chain validation, controlling the assets or services that a consumer is entitled to get access to. This happens by integrating with the Keeper from the Brizo side.

The Appendix contains more details on how Brizo enables publishers to provide computing services and related services.

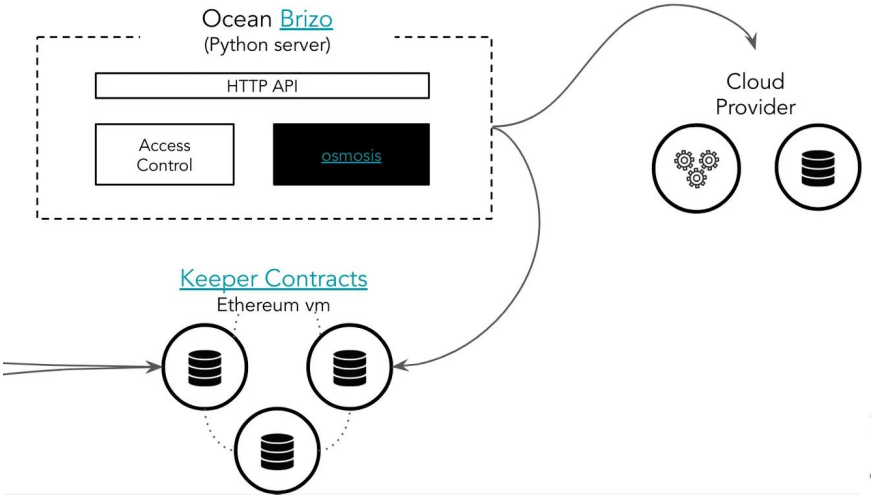

Figure 7: Brizo High-Level Architecture

3.5. Tier 1 - Decentralized VM Layer

The Keeper Contracts are Solidity smart contracts deployed and running in the decentralized Ethereum Virtual Machine (EVM).

4. Ocean Core Behavior

4.1. Asset Registration

Figure 8 describes how a publisher registers an asset. With the help of Squid, a DID (Decentralized Identifier) is generated, giving the asset a unique, cryptographically secure ID. The DID, URL and threshold ("document") are encrypted and stored with the help of Parity Secret Store (the Appendix has details). Then, a DDO (DID Descriptor Object) is created from the DID, asset's metadata, public key, encrypted URL, and list of services (data or compute to be provided).

The publisher now holds a DID with an associated DDO. She publishes this pair to a marketplace running Aquarius, which is storing its own database (an OceanDB instance) to persist the pair, searchable via the metadata. The publisher registers that DID on the Ocean keeper network as well.

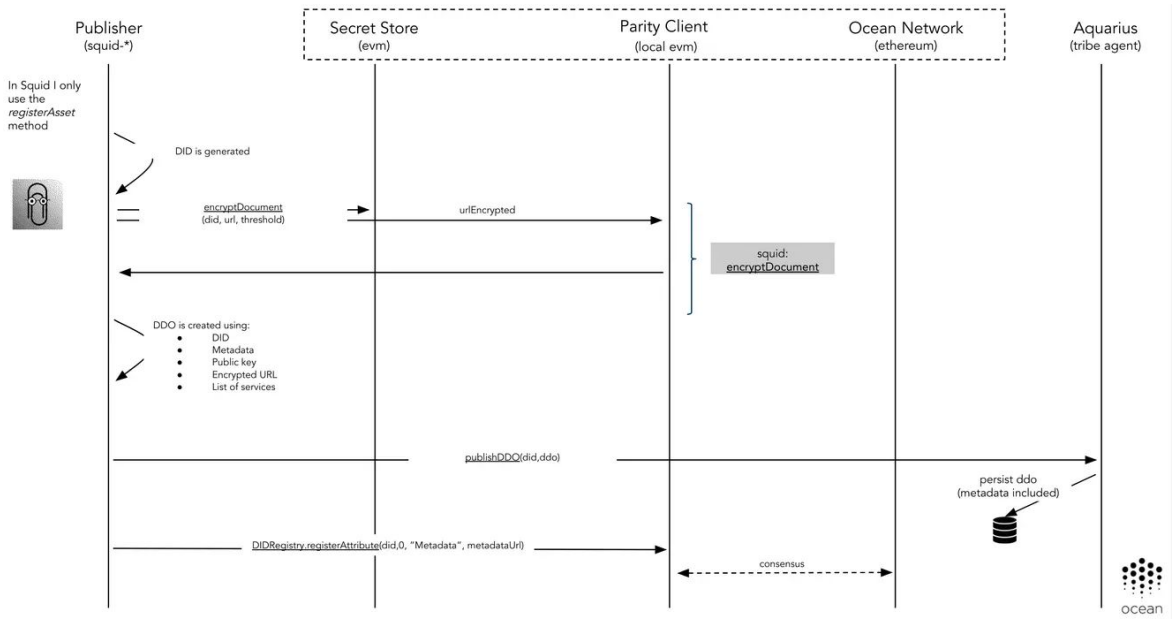

Figure 8: Asset Registration Flow

4.2. Asset Consumption

We now describe how an asset registered on the Ocean network is consumed, in Figure 9. First, a consumer (via Squid) conducts search on a marketplace's metadata store (via Aquarius interface). He/she finds a service offering (SEA) for data or compute that she likes. She digitally signs the SEA. In the next few steps, a service provider running Brizo will execute the agreement so that consumer can access (via on-chain access control) and consume the asset after sending the payment to Keeper smart contract. Note that on-chain access control helps resolve the DID into the DDO, which includes the metadata for the asset corresponding to the same DID.

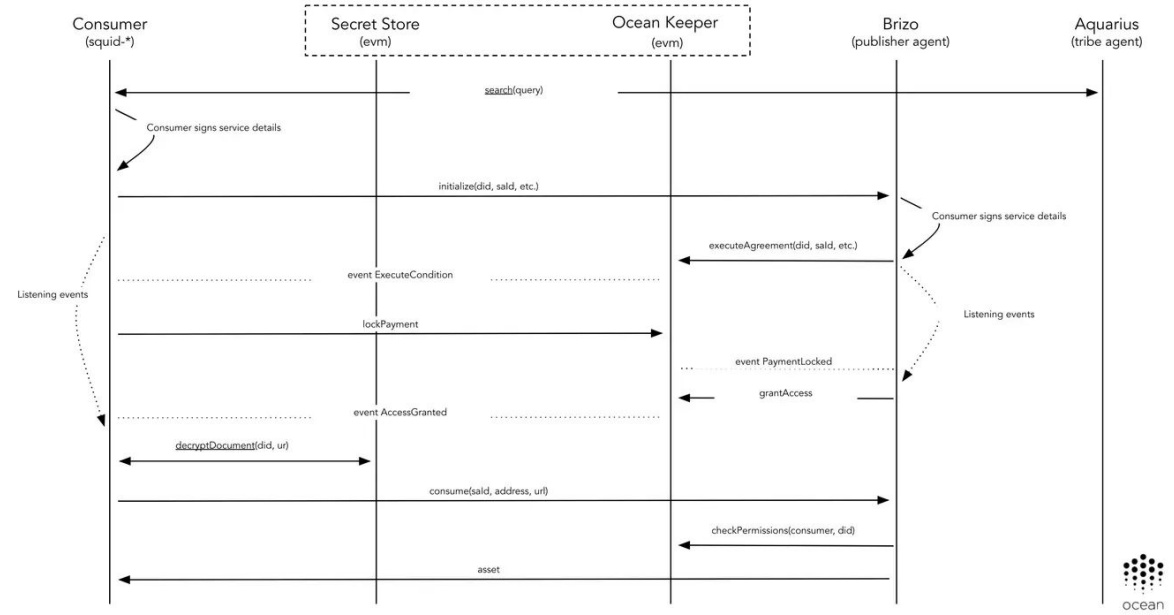

Figure 9: Asset Consumption Flow

We emphasize that Ocean's access control is on-chain. For simplicity, access control is implemented using the Service Execution Agreements (SEAs) as building blocks. OEP11 has a precise specification of on-chain access control with SEAs [OEP11_2019].

5. Identity & Metadata

5.1. DIDs & DDOs

Each Ocean user, asset, and group of users (clan) gets an identity in the form of a DID (Decentralized Identifier) and associated DDOs (DID Descriptor Objects). This improves discoverability and usability of data assets.

A DID has its own associated DDO, which includes claims and metadata. Some of that metadata points to specific asset resources. Metadata is stored in OceanDB that is running by marketplaces. With large scale of metadata, discoverability is a indispensable functionality to search for needed assets.

In addition, Keeper Contracts' service execution agreements (SEAs) grant access to specific assets for specific users or clans. Figure 10 illustrates. OEP7 has details [OEP7_2019].

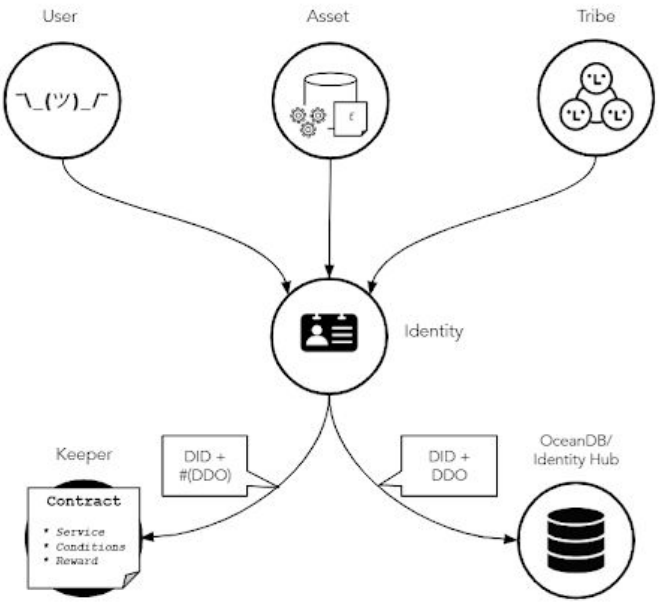

Figure 10: Each user, asset and clan gets a DID (identity).

5.2. Assets Metadata Ontology

Asset metadata is one part of Ocean DDOs. This is a JSON object with information about the Asset. Ocean's Assets Metadata *Ontology* is the schema for the Asset Metadata, and is specified in OEP8 [OEP8_2019]. It's based on the public DataSet schema from schema.org [SchemaOrg2019]. OEP8 specifies the common attributes that must be included in any Asset Metadata stored in the Ocean Network, such as name, dateCreated, author, license, price, files (to URLs), file checksums, tags, and more. In addition, OEP8 recommends some additional attributes for discoverability and normalizes these attributes for curation purpose, which serve a common structure for sorting and filtering on DDOs.

Life Cycle of Metadata. Metadata is first created by the publisher of the asset. During the publishing process, the publisher provides the file URLs as plaintext, which will be encrypted by metadata store (Aquarius) in the backend and stored as encrypted URLs.

6. Service Execution Agreements (SEAs)

6.1. Introduction

This section introduces SEAs [DeJonghe2018]. They are the fundamental building blocks for provable provenance, dispute resolution, reward mechanics and so on. It consists of various service conditions in the Ocean network and fulfills specific data services. SEAs specify the compute DAG that Ocean orchestrates.

6.2. Unhappy Paths in Service Execution

A lot can go wrong when dealing with data services, for example:

- Services might not exist, although the consumer paid for them.
- Services might be delivered correctly, however the consumer refuses or forgets to reward the service provider (SP).
- SPs might not grant access to rightful users or do grant access to non-rightful users.
- The service didn't meet the functional requirements or performance expectations of the consumer.
- Service response or logs got "lost" in the network or translation.

6.3. Traditional Service Level Agreements

In our physical world, Service-level Agreements (SLAs) are key to defining the relationship between an end-user client and their service providers (SPs). SLAs are commitments among the multiple involved parties. Each SLA explicitly states the service that the end-user expects to receive from the SP and clarify the performance metrics used to gauge the service quality. In case of any disagreements around the delivered service, all involved parties must turn to SLA to resolve the dispute.

6.4. Introducing SEAs

Ocean Service Execution Agreements (SEAs) bring the idea of SLAs to blockchains. In surviving unhappy paths, SEAs can take advantage of not only legal agreements, but also add layers of mathematical security (encryption, signatures, hashes, cryptographic proofs & attestations) and automation.

SEAs are building blocks for following services:

- Decentralized access control
- Dispute resolution
- Provenance of service consumption
- Network rewards and incentives

SEAs are composed of: a DID, a *serviceAgreementId*, referencing the unique agreement between the consumer and publisher; a *serviceDefinitionId* referencing the service the consumer wishes to use; and a *templateId*, referencing a service agreement template from which the unique service agreement is derived. With these components, the transaction can be clearly defined and executed.

SEAs are implemented as smart contracts running on Ocean Keepers. A SEA operates as a decentralized escrow, holding payments. The contract can resolve (i.e. the contract "executes" or

"aborts" the payments) when sufficient conditions in the contract are fulfilled (inspired by the Interledger Protocol [Thomas2015]).

6.5. Web2 & Web3 Service Agreements

Distributed service networks add a layer of *redundancy* at the cost of replication. As the trust level decreases, for example in the case of (anonymous) permission-less networks, one can explore *economic and reputational incentives* that use the underlying token. The more critical a service is within a supply chain, the more stringent the levels of guarantees that need to be enforced. Figure 11 illustrates.

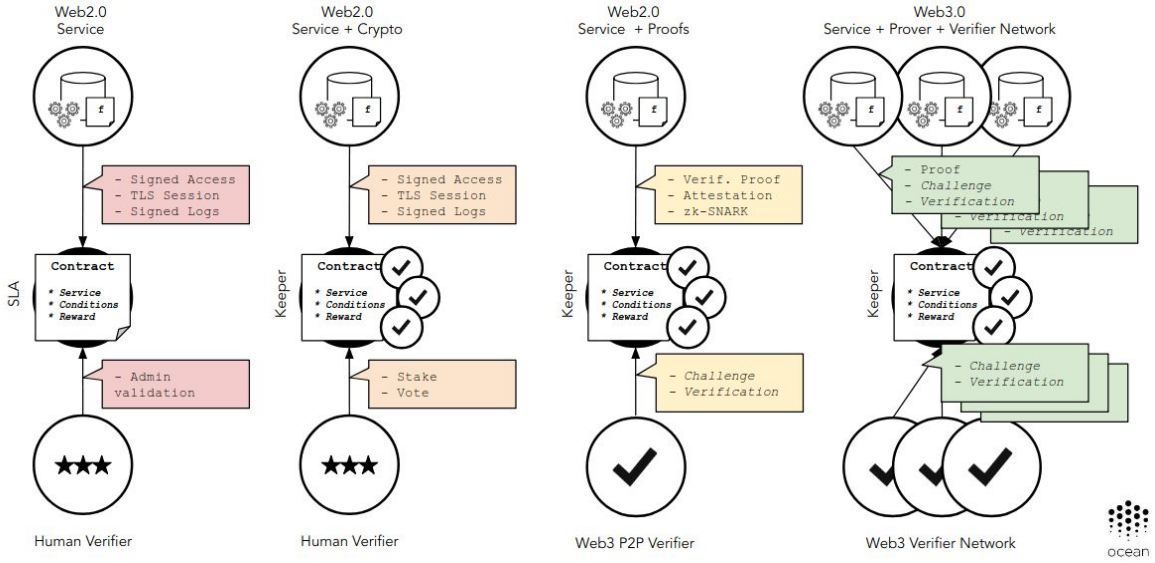

Figure 11: Levels of trust increase with more cryptographic verification (going left to right)

6.6. SEA Templates

Ocean provides reference implementation of SEA templates for certain functionalities, such as data access or on-premise compute. User can create their own SEAs, on top of new conditions such as:

- Minimum accuracy requirement of a trained model is satisfied, which triggers the release of payment to the model creator.
- Consumers hold certain credentials or affiliation of a certain institute.
- Release partial payments to the service provider depending on the percentage of fulfilled services.

Once the new template is created, the user can create a new service agreement by instantiating the new *templateId* and submitting it to the network. Once the service agreement has been signed by both a service provider and a consumer, it will be initialized on-chain, and voilà, therefore, a new customized service agreement is established by both parties, which can be used to fulfill more complex services.

6.7. SEAs Example on Providing Data

Figure 12 illustrates one scenario. The Provider is willing to deliver a data service after a consumer ensures a payment will follow upon delivery. The consumer in turn will only pay if granted access to the service and the performance of the service is verified by a Verifier. The Verifier (or network of

Verifiers) also demands to see a reward locked in escrow (for her) before she puts in any verification work.

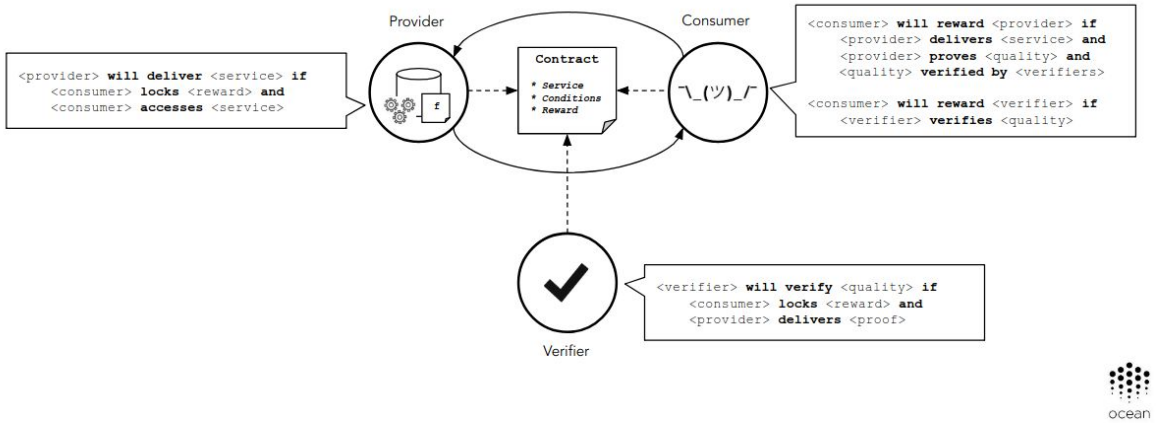

Figure 12: Three parties engaging in a service agreement.

6.8. Anatomy of a modular SEA

In Ocean Protocol, SEAs have a modular design to give flexibility for onboarding and consuming a variety of web2.0 (cloud/on-premise) and web3.0 services. We identify three main parts in a SEA: service identifier, conditions & fulfillments, and reward logic. Figure 13 illustrates. We now describe each in more detail.

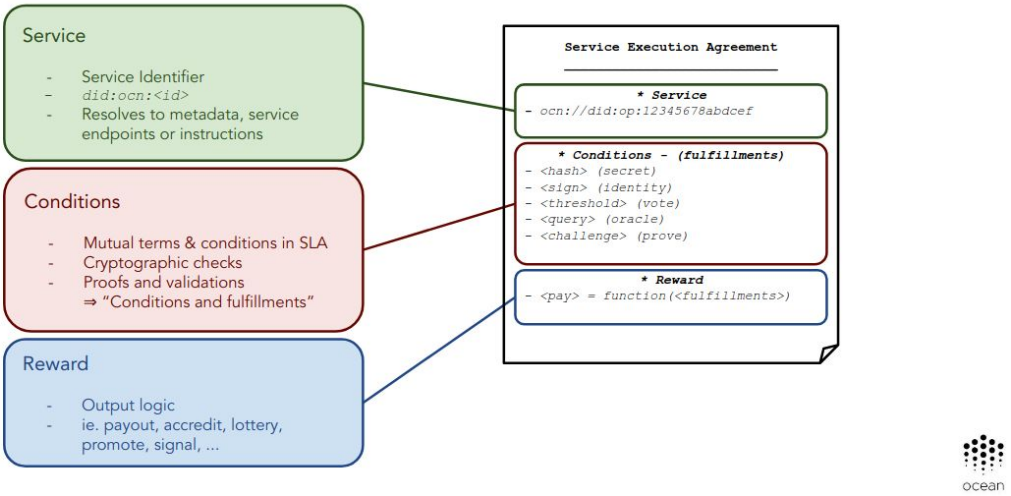

Figure 13: Components of a Service Execution Agreement.

6.8.1. Service Identifier

This is a cryptographic ID that uniquely identifies the service to be consumed. We use DIDs here for that purpose.

6.8.2. Conditions and Fulfillments

In an imperfect world, we deal with off-chain, on-chain, side-chain and other-chain services and events. These services can either execute correctly, fulfill partially, or even fail. At some point in time

the Ocean SEAs will be interested in knowing the status of these services to settle or resolve disputes.

Hence we introduce conditions and fulfillments: cryptographic and non-cryptographic conditions that can be fulfilled. Each condition has a validation function that returns a state: “True”, “False“ or “Unknown”. “Unknown” value implies that a condition has not been provably fulfilled. All conditions have the same initial state of “Unknown”. The validation logic is executed on-chain. Conditions can serve as the inputs to a SEA.

Conditions allow us to flexibly encode Proof-of-Service into SEAs. Figure 14 illustrates. Conditions are the challenges to be solved and fulfillments are the solutions. Reward logic distributes the rewards depending on the fulfilled conditions.

Conditions can vary from simple cryptographic challenges (e.g. provide a pre-image which hashes to a value with 14 trailing zeros, or prove that you have the private key corresponding to a public key), to more complex ones (e.g. STARKs, compute attestations, proof-of-spacetime, proof-of-retrievability) and to more subjective ones (e.g. m-of-n signatures in a voting or curation scenario, stake/slash).

When validation events occur in non-Ocean networks, conditions can be simply fed into oracles or bridge contracts to resolve the dispute.

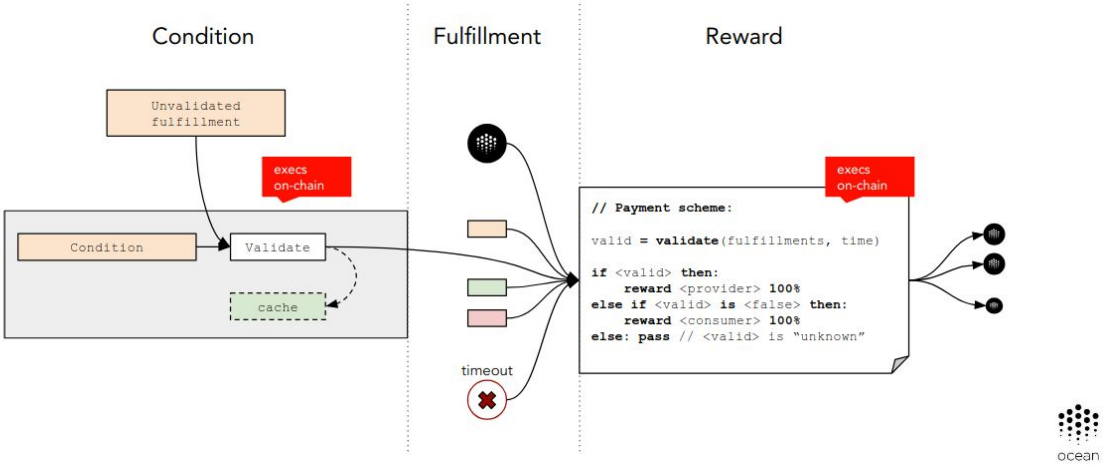

Figure 14: From conditions over fulfillments to rewards. Conditions are the challenges to be solved and fulfillments are the solutions (green: fulfilled/valid, orange:unfulfilled/unknown, red:invalid). Reward logic distributes the rewards (outputs) depending on the fulfilled conditions.

The actual implementation of conditions and fulfillments is a variant of the crypto-conditions Internet-Draft (of IETF), initiated by the Interledger protocol [Thomas2015]. Each condition/fulfillment is a cryptographic challenge/proof pair such as:

- *Hash / pre-image:* find a pre-image that computes to a given hash. The computation of the hash of the pre-image happens on-chain. This condition is useful for a party to prove that they have knowledge of a secret (without revealing that secret).
- *Public key + message / signature:* sign a given message with a private key corresponding to a given public key. The verification of the signature happens on-chain. Useful for a party to prove that they have the private key corresponding to the given public key, and the message wasn't altered in transit.

- *m-of-n threshold:* validates the proof to be "True" if m out of n conditions have been correctly fulfilled. Useful for multi-party dispute resolution such as voting.
- *Query/resolve:* link to a publicly available state value (that is recorded with a timestamp) and resolve/compare that value upon validation. The query is executed on-chain, hence it's limited to GET operations within the state context of the chain (e.g. contractAddress.getValue). Useful to bridge services and oraclize off-chain values.

Conditions can be combined to build more complex logic to express:

- *Payment conditions:* the amount of token submitted to the contract equals the predefined asset price.
- *Access control:* an access-control secret has been communicated with the consumer. [OceanBlog_AccessControl2018] has details.
- *Verified compute:* a network of Verifiers agrees and verifies that a service was correctly delivered or not.

It is inevitable that many new conditions will emerge in the ecosystem, hence careful auditing and curation are required to green-light conditions that can be safely added into a SEA, for example using an AragonDAO [Aragon2019].

6.8.3. Reward Logic

The outputs of a SEA are rewards that are typically distributed to agents that have fulfilled one or more conditions. Rewards can be issued in the form of payments, royalties, permissions, badges, reputation or lottery tickets for a network reward function, and so on. As multiple reward mechanics can be envisioned, they are consolidated in governed templates, similar to the library of conditions.

A basic reward structure implemented in Ocean Protocol is the Escrow or Hold token. Here tokens are locked up in an SEA to be either:

- *Executed* if all conditions are fulfilled before a timeout occurs. Executing the payment means that the locked tokens can be transferred to the receiving party.
- *Aborted* if not all conditions are fulfilled after the timeout occurs. Aborting the payment means that the locked tokens are (typically) returned to the issuing party.

Future functionality can include bounties, competitions, royalty schemes, and payment streams.

6.9. The Life of a Service Execution Agreement

Having all components of a SEA in place, one can start publishing services, attaching a service agreement, and interacting on marketplaces with consumers. Figure 15 illustrates.

6.9.1. Service Publication

A provider can provision services by defining metadata (see [OEP8_2019]) and API calls for access, consumption and monitoring (see [OEP11_2019]).

Next, the provider takes on the role as a publisher (or delegates that role) in a marketplace. A publisher chooses a SEA from the templates and includes it in the Service Identity Document before publishing in a marketplace (see "Publishing" section of [OEP11_2019]). Publishing methods include a public metadata store (Aquarius), Web APIs/forums and peer-to-peer messaging.

Once a service is published, it can be discovered by a consumer. This will enable both parties to instantiate a SEA by signing and executing the agreement (see "Consuming" section of [OEP11_2019]).

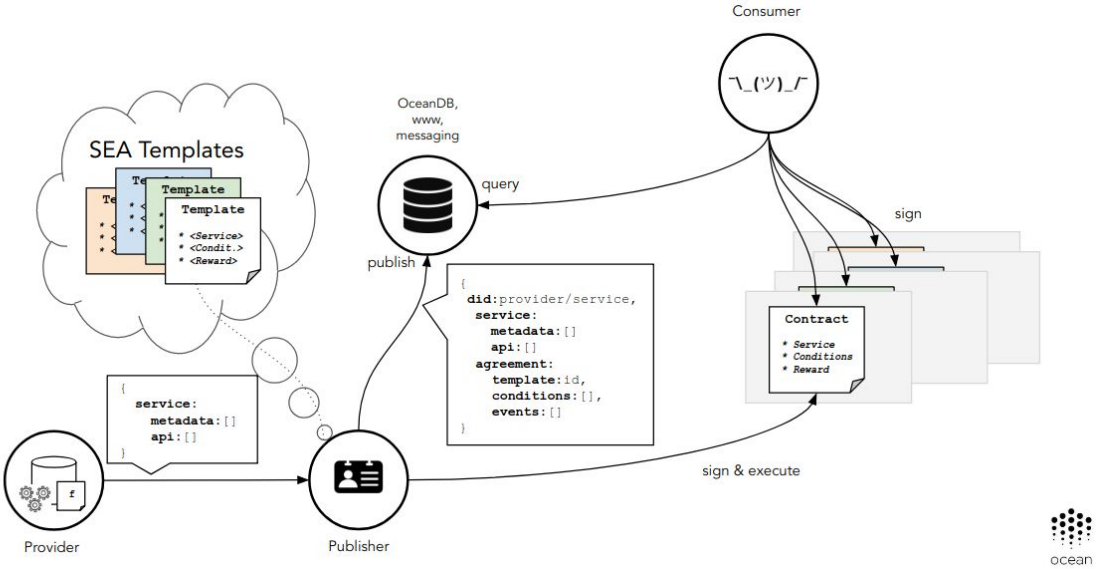

Figure 15: Ocean Protocol's Publishing Flow: From Resource to Service Execution Agreement

Now let's explore life cycles of a few example SEAs during their operation.

6.9.2. Access Control

One elementary SEA is provisioning access control to off-chain resources with escrow on consumer payment. Figure 16 illustrates.

Here are the relevant events for the SEA:

1. *Sign and Execute:* Both parties agree and an instance of the Access Control SEA is created.
2. *Payment:* Consumer locks up the required amount of tokens in the escrow smart contract.
3. *Access:* The service Provider grants access to the resource and reports this event on-chain.
4. *Rewards:* The escrow contract will either execute or abort, taking the access condition and timeout into consideration.

[OceanBlog_AcessControl2018] has details.

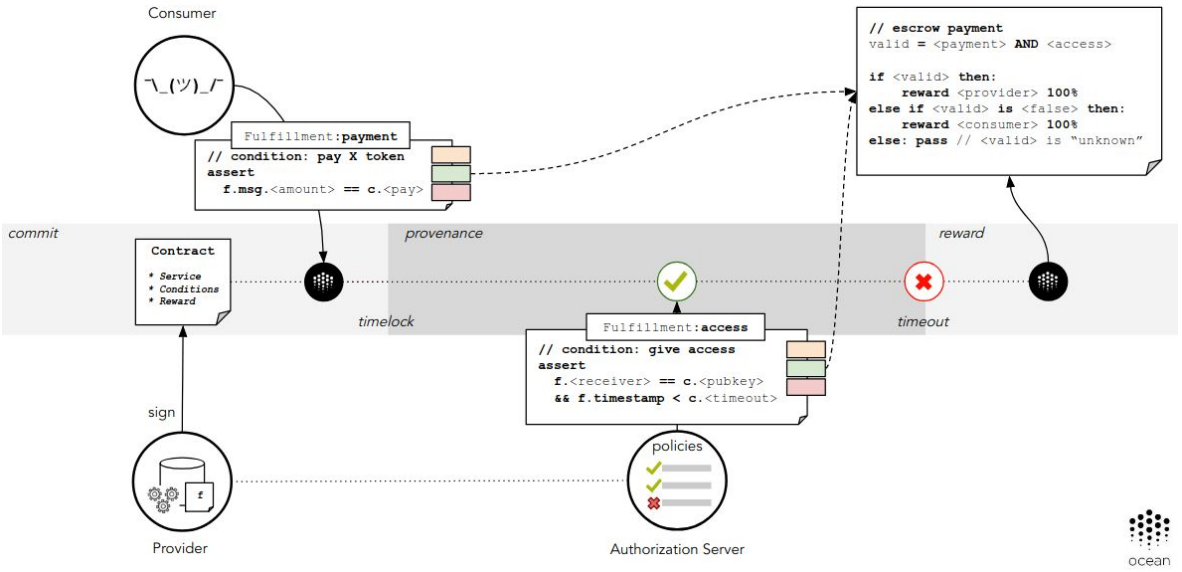

Figure 16: Life of a Simple Access Control SEA after publication

6.9.3. Verified Services

A more complex use case extends the above access control by adding verification events to the service. Here, the resource provider submits one or more service proofs or attestations to a Verifier network. Figure 17 illustrates.

The Verifier network is tasked with resolving disputes about the performance of the service (e.g. Truebit, fitchain, Enigma, Filecoin). Here, the SEAs are connected by either an oracle or bridge contract that can be queried using query conditions. Hence, the SEA will be able to access and resolve the dispute resolution outcome of the Verifier network.

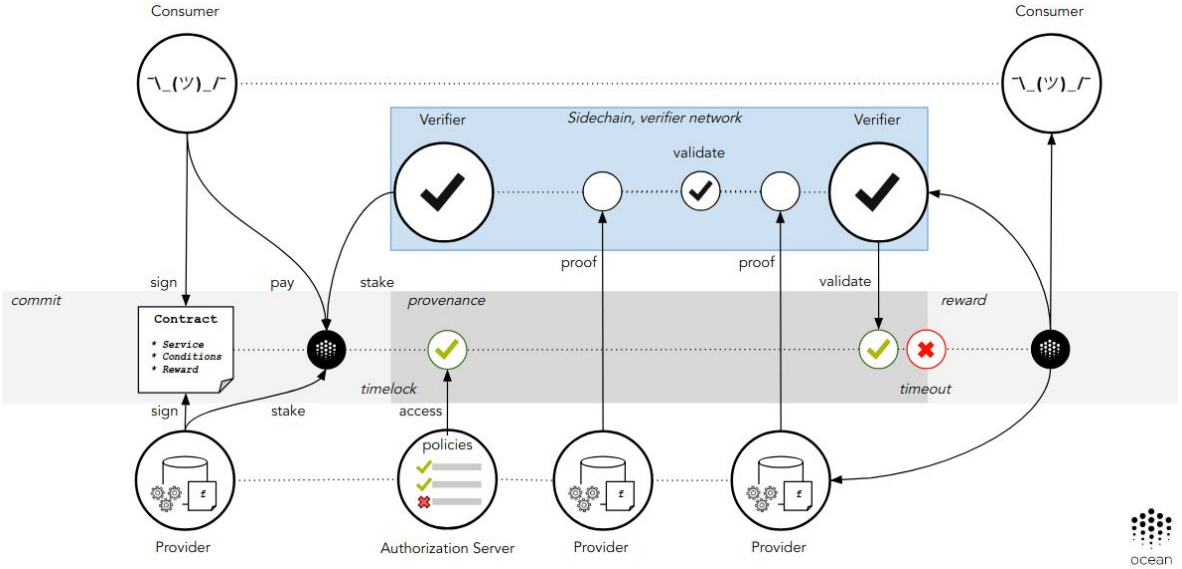

Figure 17: An external verifier network used for validating proofs-of-service and bridging the outcome of the dispute resolution cycle.

6.9.4. Combining Publishing, Consumption, and Validation

We've done the groundwork to put all events in the life of a SEA together, in Figure 18. Notice that multiple SEAs can easily be executed in parallel. Off-/side-chain resources and authorization servers simply need to listen for the predefined events emitted from the SEA.

7. Initial Network Deployment

7.1. Physical Network

Ocean V1 includes the launch of the initial physical network. Ocean's main target users are AI researchers and data scientists. They demand reasonable network performance and usage costs. For this reason, Ocean will initially ship as a Proof-of-Authority (PoA) network called "Pacific" running Parity Ethereum clients [McConaghy2019][5]. Ocean will become permissionless in a later milestone.

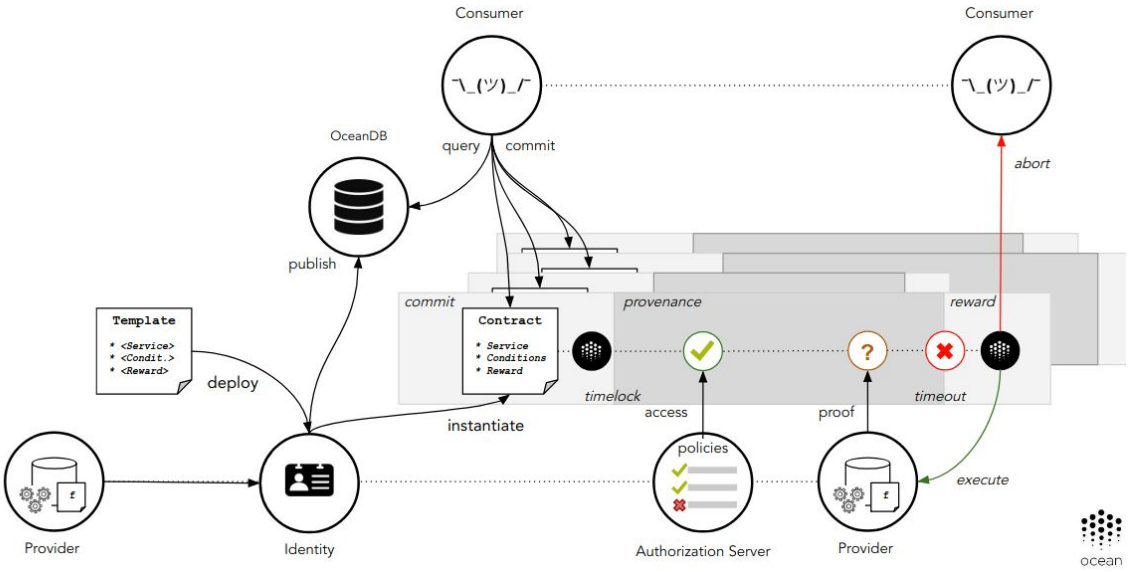

Figure 18: The life of an Ocean Protocol SEA from publishing to consumption and validation.

Using PoA raises the question: how do users on Pacific get Ocean tokens, and exchange with Ether? There's a good answer: use an ERC20-ERC20 TokenBridge [PoaNetwork2019] between Pacific and Ethereum mainnet. In such a bridge, users can move tokens back and forth at any time. Ocean users will use Pacific, except for using Ethereum mainnet to interface with token exchanges.

Launch will also include a Parity Secret Store cluster, to allow the sharing of secrets among parties. The Appendix has more information.

Launch may also include Commons Marketplace (Provider) with its own Aquarius (with OceanDB metadata store) and Brizo nodes (service provisioning). This will help demonstrate Ocean capabilities using free/open assets.

[5] To maximize clarity, Ocean is not shipping on the 3rd-party network called "POA Network" [PoaNetwork2019b]. Ocean is shipping on its own dedicated network.

7.2. Network Legals

If one is running software that is publicly accessible, whether a web server or a PoA node, then one is subject to laws of the jurisdiction that they operate in and potentially other jurisdictions. These include data processing regulations like the EU General Data Protection Regulations (GDPR).

Accordingly, node operators face liability around data protection in light of these laws. Then a question for node operators is: do we want their liability to be uncertain, or clearly delineated? We prefer the latter. We can take steps to make liability clear, even in a decentralized setting. We do this by creating a legal entity that interfaces with the legal world, with appropriate governance.

- We have chosen a Dutch Stichting, which is a nonprofit foundation with base in the Netherlands. It is European, yet has low latency and flexibility in changing its articles of association. These characteristics will be useful as we evolve Ocean governance to being permissionless.
- The Stichting lays out contracts for each node operator, including data processing agreements. This clarifies liability. We believe that node operator risk is low because there is little opportunity to have personally identifiable information (PII) on-chain: Ocean keepers do not store data or metadata, and even hyperlinks are encrypted via Parity Secret store.

7.3. Network Governance

The most important governance decisions are (a) adding and removing node operators, and (b) upgrading the smart contracts running on the nodes' virtual machines.

We make both decisions to be a function of node operators: one operator one vote. Call it "multisig". In practice, the Stichting makes the decisions, and it is democratically governed by node operators.

We aim for a minimum degree of evolvability in a "software upgrades" framing - **minimum viable governance**. The smart contract upgrades are handled by ZeppelinOS [Zeppelin2019].

Ocean will become permissionless in a later milestone.

8. Proofs of Service

8.1. Introduction

On top of the SEAs infrastructure, Ocean adds network rewards to incentivize data sharing. Ocean rewards participants for providing data and other services. To avoid being gamed, Ocean needs some proofs that verify the work to provide data/services was actually done: service integrity proofs.

Ocean needs service integrity proofs for both data and compute, for both Web2 and Web3 (blockchain) services. The first priority is Web2 data, which is the most mature and unlocks many use cases.

8.2. Service Integrity: Data Integrity

Introduction

We need to be able to prove that an actor made the correct file available, versus an incorrect one. Put in 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.

Web2 Data Services

Today, most data is not available in Web3 services. We don't expect there to be massive amounts of data available in Web3 services in the next few years, and data doesn't want to be moved. So we need to go where the data is. Data is in the cloud, or on-premise environments in the shape of Big Data Lakes, Data Warehouse or bespoke solutions. The two largest cloud providers are Amazon AWS and Azure. Ocean V1 supports both. In addition, Ocean V1 supports standard HTTP on-premise URLs. The next question is how to get a proof from these services.

Web2 Data Integrity

This is tricky in a Web2 environment. The best scenario, unless you want to move the data (not a good idea) is that you need to rely on the infrastructure provider (Amazon, Azure, etc). To mitigate this, we store a data integrity hash in an on-chain smart contract representing the ground-truth of multiple attributes in the dataset. To compose this hash we use the checksum information given by the cloud providers. It means if the owner of the file changes the content or deletes it, the same hash can't be computed by cloud providers again. The hash is calculated as:

1. Concatenation of all the DDO.services['AccessService'].metadata.files.checksum[*] attributes. Every file included in the asset can have a file checksum associated
2. Concatenating to the previous string the metadata attributes name, author and license
3. Concatenating to the previous string the DID (e.g. did:op:0ebed8226ada17fde24b6bf2b95d27f8f05fcce09139ff5cec31f6d81a7cd2ea)
4. Hashing the complete string generated using SHA3-256

Section "How to compute the integrity checksum" of [OEP7_2019] has details.

Web2-Web3 Data Integrity

Oracles bring Web2 data services (off-chain) closer to Web3 trust levels (on-chain). [Chainlink2019] requires multiple independent sources for the same data; the replication factor is up to the user. [Oraclize2019] uses TLSNotary proof [TlsNotary2019].

Web3 Data Integrity & Availability

There are several options here. None are fully mature, and some are still in early stages.

- **Proof-of-Space-Time (PoST)** proves that a data blob was stored for a time interval [Filecoin2017]. Ethereum Swarm [Trón2018] has similar aims, but emphasizes erasure coding.
- **Proof-of-Retrieval (PoR)** proves that data was retrieved, without worrying about storage [Filecoin2017]. PoR chunks data and pays for each piece in sequence, using state channels for efficiency.

- **Proof-of-Replication (PoRep).** [PoRep2018] proves that data was replicated, using verifiable delay functions in a challenge-response setting.
- **Dedicated POW blockchain with full replication.** In [Teutsch2017b], each miner promises to serve up every dataset; and voting happens among all miners. Challenge-response could delay voting unless there was actually a challenge. But full replication means expensive storage.
- **Dedicated POW blockchain, with per-block sharding.** In [Arweave2019], each block points to a Merkleized data blob. Stakers can stake on any blocks. If asked to return data, they get a reward if they succeed and are slashed otherwise. This shards with natural bias to replicate the most popular content more. The chain breaks if data cannot be returned when asked; therefore the community is incentivized to always maintain every block's blob.
- **Provably Secure Certified Mail.** With literature going back two decades, [Pfitzmann2000][Schneier1998] this seems to be proof-of-data-availability in disguise, but we would need to investigate further to confirm.
- **Network of validators on hash of Merkle subtree.** This is early-stage research in Ocean combining blockchain replication and erasure coding. For example, Alice claims to have served up a dataset. The network randomly selects 100 validators, from the set of validators that have staked on that dataset. Each validator attempts to download a Merkle branch of the dataset from Joe (whole dataset not needed). Alice gets rewards if 51 validators agree that she served it up; she gets slashed if 51 agree that she didn't; otherwise the validators get slashed. A higher-efficiency variant is to have a challenge-response gate before resorting to a network of validators to verify.

8.3. 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 19. 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.

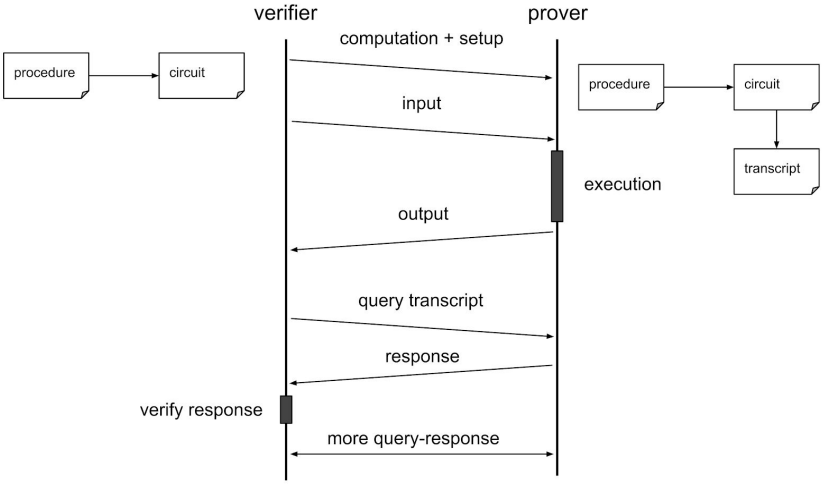

Figure 19: Computational integrity framework

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 (such as trusted execution environment or "TEE"), 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.

9. Incentives

9.1. Core Token Design: Curated Proofs Market

9.1.1. Introduction

This section addresses: "How do we incentivize putting data in the commons?" and related questions. Ocean adds *network rewards* to incentivize data sharing, which will lead to a commons. This section describes the core incentive model of Ocean. At its heart is a network rewards function (objective function) implemented by a **Curated Proofs Market.**

Recall that Ocean's main goal is: **maximize the supply of relevant AI data & services**. This drives Ocean's **network reward function**. It is akin to an **objective function** in the optimization literature, for example as used in [McConaghy2009]. This objective function is what Ocean is optimizing, by incentivizing actors in its ecosystem to contribute to it. 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?

9.1.2. Network Rewards to Incentivize Relevant Data/Services & Make It Available

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

We want Ocean to have strong incentives to submit, refer, and make available quality data/services. To accomplish this, we introduce a Curated Proofs Market, which combines (a) cryptographic proofs that the data/service was made available, with (b) Curation Markets [Rouviere2017] for reputation of data/services. It does curation on cryptographic proofs. 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. Network 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 network rewards for actor *i* on dataset/service *j*, *before* being normalized across all actors and datasets/services. The actual network rewards received are normalized: $R_{ij,norm}$.

$$R_{ij} = log_{10}(S_{ij}) * log_{10}(D_j) * R_i$$

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

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 $log_{10}(S_{ij})$. 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 $\log_{10}$ 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, $log_{10}(D_j)$, reflects the popularity of the dataset/service; that is, how many times it has been (provably) used in the time interval. We use $\log_{10}$ 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 Curated Proofs 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 network 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:

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

where

- $B_{served,i}$ = total number of bits that actor *i* served (made available) across all data assets they have staked
- $B_{downloaded,i}$ = total number of bits that actor *i* 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.

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

In designing the system, we want to incentivize the various stakeholder roles: data/service provider, curator, validator, and keeper. One possibility is to give percentage of network rewards to each role based on their respective actions. However, this raises the concern that keepers take all the rewards for themselves.

One solution is to explicitly couple all the roles into one: if you've staked (provider or curator) then the only way to get network rewards is to run a keeper node. However, this doesn't work in a PoA (Proof of Authority) setting with a limited number of keeper nodes, because it severely restricts who can be a commons service provider getting network rewards.

Another solution is to *disincentivize* keepers from taking all the rewards for themselves. For example, keepers may have stake slashed by the community if they act unfairly.

Another approach is to do nothing. If the community believes that keeper behavior is unfair, then they will simply leave the network, and the keepers are left with nothing. However, this would only work if the cost of leaving was lower than malfeasance on behalf of a keeper; that's probably too strong of an assumption.

We will defer the final choice until we have progressed farther in implementation.

9.1.4. Network Rewards: Towards Implementation

To implement the network rewards as described above has complexity and high compute cost because, for each network 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 network 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 network 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)[6], 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:

$$R_{ij} = log_{10}(S_{ij}) * log_{10}(D_j) * R_i$$ [7]

2. Update total recent network value added:

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

3. Compute the probability of getting a network 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

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

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

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.

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

9.1.5. Network Rewards: SEAs-Based Implementation

We aim to implement network rewards simply as special cases of Service Execution Agreements (SEAs). In Figure 20, the Ocean network token dispenser periodically mints new Ocean tokens that drip into the treasure chest (top middle). A Commons Provider stakes to provide a service (left). Every time she provides a service, she gets "raffle tickets" if that service fits within a commons template (bottom). A raffle occurs among raffle ticket holders, and the winner gets the rewards in the treasure chest (top right).

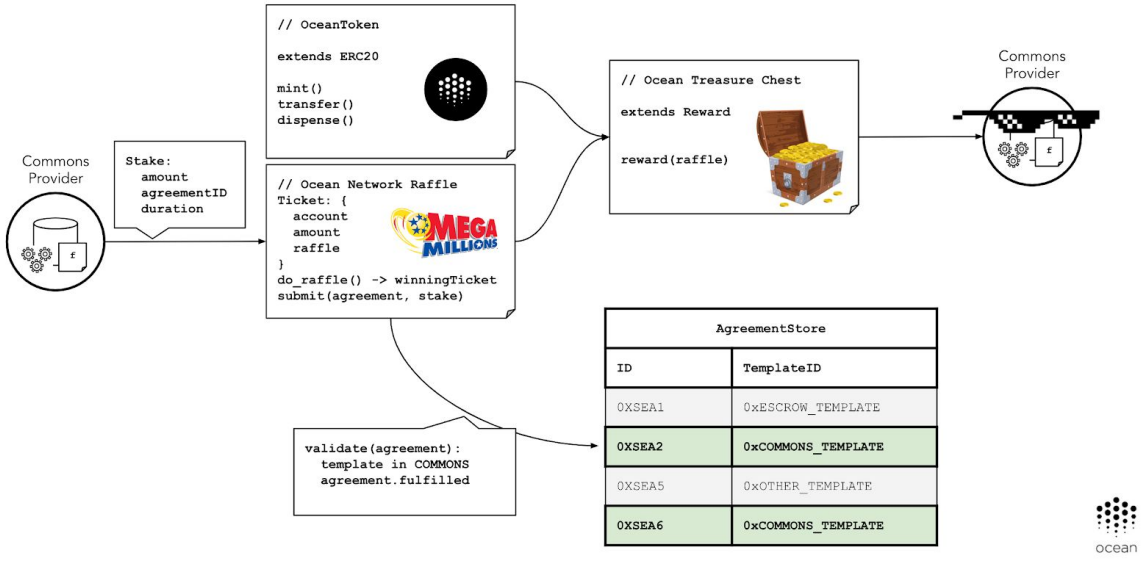

Figure 20: Network rewards by Commons Agreement Template

9.2. Curation Markets Details

9.2.1. Introduction

Recall that Ocean's objective function (network 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 - a Curated Proofs Market. 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".

9.2.2. 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. *Network rewards.* People earn Ocean Tokens if they bet on an AI dataset / service and make it available when asked (in the model where one unified keeper plays several roles).
2. *Un-staking.* One can un-stake 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.

9.2.3. Bonding Curves

A **bonding curve** relates a token's drops "Ḍ" to Ocean Tokens "Ọ" for a given dataset/service. Figure 21 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.

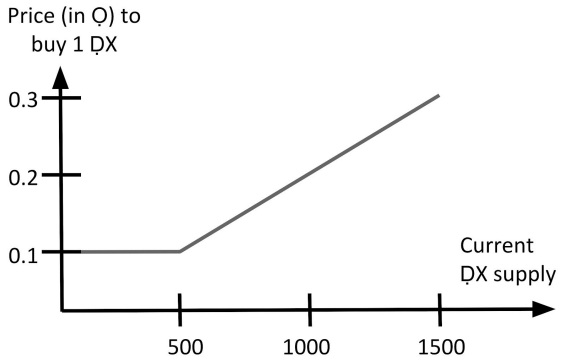

Figure 21: Bonding curve for ḌX

A new curation market is initialized each time a new dataset or service is made available. With this, the actor has already staked Ọ 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 22 illustrates. We're at the far left of the bonding curve because 0 ḌX have been

generated. There, each Ḍ̃X costs 0.1 Q̃. If the initial user staked 50 Q̃, she would gain 50 Q̃/ 0.1 Q̃/Ḍ̃X = 500 Ḍ̃X. The supply for Ḍ̃X increases from 0 to 500.

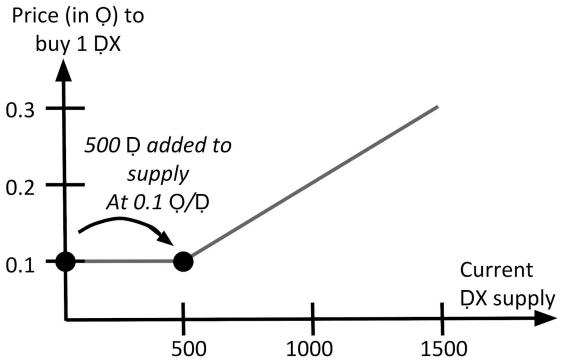

Figure 22: increasing supply to 500 Ḍ̃X

From here on, anyone can stake further in X. Let's say a user wants to purchase 500 Ḍ̃X by staking more Q̃ tokens. This would make the supply go from 500 Ḍ̃X to 1000 Ḍ̃X. In that range, the price is (0.1 + 0.2 Q̃/Ḍ̃X)/2 = 0.15 Q̃/Ḍ̃X. The user would get 500 Ḍ̃X for a total cost of 500 Ḍ̃X * 0.15 Ḍ̃X/Q̃ = 75 Q̃. Figure 23 illustrates.

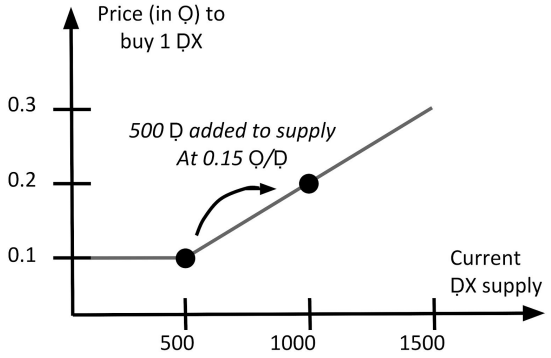

Figure 23: increasing supply to 1000 Ḍ̃X

9.2.4. Un-Staking

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

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

9.2.5. 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 network reward.

Actors get rewarded the most if they stake large amounts on popular datasets/services - the first and second terms in the network 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.

9.3. Network Rewards Schedule

This section describes Ocean's network rewards schedule.

9.3.1. Bitcoin Rewards Schedule

Bitcoin's network rewards schedule is:

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

where

- $F(H,\ t)$ is the fraction of all 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.
- Bitcoin uses a half-life of 4 years.

The Bitcoin reward schedule has several nice features, including a fixed supply of tokens, more benefits to early movers, and a track record of being live for a decade. For these reasons, Ocean uses Bitcoin's approach as a starting point.

9.3.2. Challenges for Ocean

The Ocean schedule / token design must resolve these additional challenges:

1. A means to pay for the development of the network
2. It can take several years for internal enterprise processes to prepare their data assets for sharing
3. Ocean V1 release does not have network rewards; only Ocean V2 release does.
4. Ocean will only become permissionless in V4. There needs to be strong incentive for the network to be permissionless.
5. How do we verify the incentives when the best verification is having the system live, because it means real actors with real skin in the game? This is a chicken-and-egg concern.

9.3.3. Addressing the Challenges

We address challenge (1) with pre-mining to build Ocean software (e.g. with developer bounties), incentivize the community, and more. 62.0% of Ocean Tokens remain for network rewards.

We address challenge (2) via a half-life of ten years, i.e. H =10. This gives entities wishing to share data breathing space.

We address challenge (3) by inserting a delay into the network rewards function: it only emits >0 tokens once the network rewards infrastructure of Ocean V2 is ready.

We address challenge (4) by only turning on full (100%) emission once the network is permissionless (scheduled for Ocean V4), or a maximum of 5 years, whichever is sooner.

We address challenge (5) by **three forms of verification: human, software, and economic**. Human verification is simply human-based vetting of the token design, including whitepaper reviews and security audits. Software verification is running software simulation and verification tools, as is done in engineering fields. Economic verification includes **ratcheting up the amount of rewards over time, until stability is reached**. Specifically for Ocean, we start with a 10% multiplier on rewards, then 25%

after 6 months, then 50% after another 6 months (capping at 50% until the next gate), and finally 100% when permissionless (or 5 years hit).

We considered having more ratchets based on milestones other than time. However, each non time-based milestone adds complexity, and must be measured with a relatively obvious threshold. For these reasons, we only have a single non-milestone ratchet after the initial network rewards: when the network becomes permissionless.

The "5 years hit" part for the final ratchet is just in case the "permissionless" measure is ineffective, or after 5 years the priority for being permissionless has diminished.

9.3.4. Ocean Network Reward Curves

Figure 24 shows how network rewards will be dispensed over the next 2.5 years. One can readily see the ratchet every 6 months.

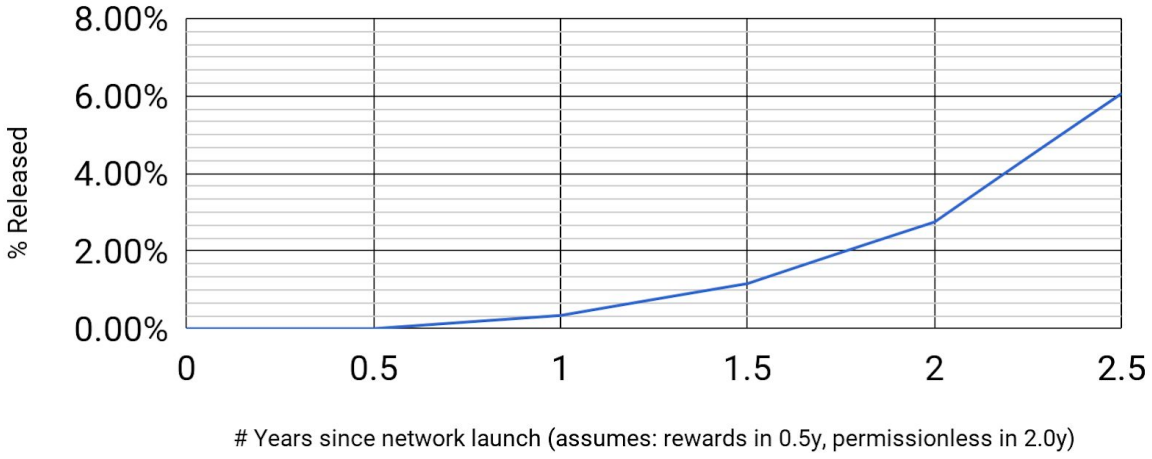

Figure 24: % network rewards tokens released over next 2.5 years

Figure 25 shows network rewards over 50 years. The curve is shaped much like Bitcoin's, except the first two years of ramp-up and the longer half-life.

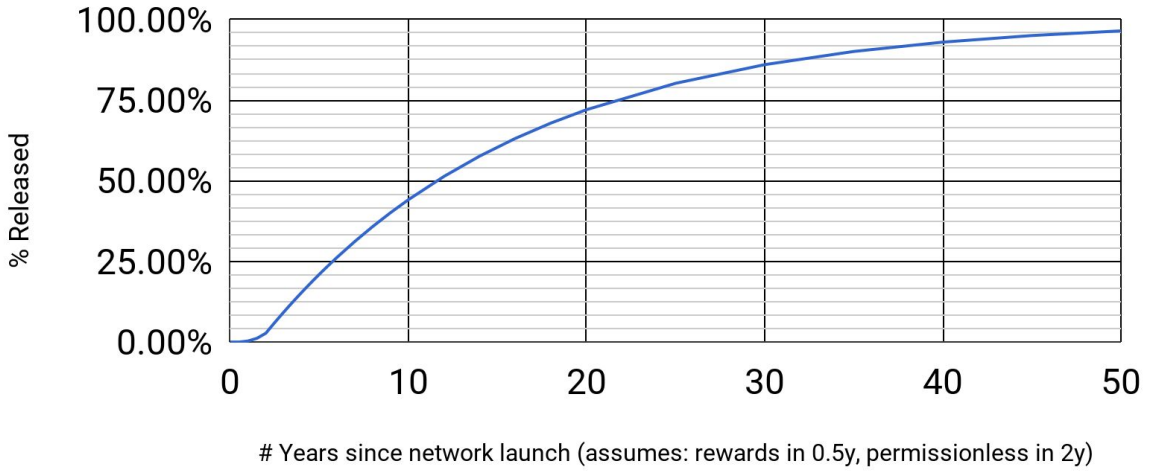

Figure 25: % network rewards tokens released over next 50 years

Table 2 shows network rewards going to 150 years. It takes 11.6 years to get to 50% dispensed; a 1.6

year lag compared to no ramp-up period.

Table 2: % network rewards released over next 150 years

# Years	0	0.5	1	1.5	2	5	10	11.6	25	50	100	150
% Released	0%	0%	0.3%	1.2%	2.8%	21.0%	44.1%	50.0%	80.3%	96.5%	99.9%	100%
% Left	100%	100%	99.7%	98.8%	97.2%	79.0%	55.9%	50.0%	19.7%	3.5%	0.1%	0%

9.3.5. Ocean Network Reward Equations

Here are the equations in detail. g(t) is the overall network rewards schedule. It's a piecewise model of four exponential curves.

$$g(t) = \begin{cases} 0 & t < T0 \\ g_1(t) & T0 \le t < T1 \\ g_2(t) & T1 \le t < T2 \\ g_3(t) & T2 \le t < T3 \\ g_4(t) & otherwise \end{cases}$$

Where g_i(t) are pieces of the piecewise model chosen depending on *t*, and *Gi* are the values of g_i(t) at the inflection points.

$$G1 = g_1(t = T1);\ G2 = g_2(t = T2);\ G3 = g_3(t = T3)$$

$$g_1(t) = M1 * f(t - T0)$$

$$g_2(t) = M2 * f(t - T0) - M2 * f(T1 - T0) + G1$$

$$g_3(t) = M3 * f(t - T0) - M3 * f(T2 - T0) + G2$$

$$g_4(t) = (1 - G3) * f(t - T3) + G3$$

And *f(t)* is the value of *F(H,t)* assuming a constant *H*. *F(H,t)* is the base exponential curve. The units of *H* and *t* are years.

$$f(t) = F(H, t)$$

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

The pieces of the model are a function of t, which are parameterized with *Ti*. The units of *Ti* are years.

$$T0 = < time\ interval\ after\ network\ launch\ for\ network\ rewards >$$

$$T1 = T0 + 0.5$$

$$T2 = T1 + 0.5$$

$$T3 = min(5,\ max(T2 + 0.5, < time\ interval\ after\ network\ launch\ for\ \ permissionless >))$$

The following parameter values are chosen: *H*=10 years; *M1*=0.10; *M2*=0.25; *M3*=0.5.

The curves shown earlier assume *T0*=0.5 and*T3*=2. These are assumptions are dependent on implementation timelines, and therefore could readily change.

10. On-Chain Bounties

10.1. Introduction

This section addresses the question "How can the system not only self-sustain but actually improve over the decades, without guidance by a centralized actor?" The Ocean answer to this is to have functionality of on-chain bounties.

10.2. Design

Ocean will be a utility network to serve the public at large. Therefore it must be self-sustaining; and the technology and ecosystem need steady improvements, by the community and for the community.

To this end, the Ocean network will have an on-chain bounties system where a portion of the network rewards are for (a) technology improvement and (b) ecosystem / community development.

The DASH network has an on-chain bounties system as part of its governance system [DashGovernance2019]. Its bounties have been live since September 2015 [DashWatch2019]. It has:

- An on-chain means for people to submit proposals;
- An on-chain means for the community to select proposals to fund, then fund them;
- An on-chain means to track progress and completion of each proposal;
- Off-chain interfaces to the above, such as Dash Central [DashCentral2019]; and
- Importantly, it enables on-chain *funding* for proposals. 10% of DASH's network rewards go towards its bounties system. Any allocated funds not spent are *burned*. This incentivizes a proposer to not just add >0 value, but to add >*x* value where *x* is the rise in the value of each DASH token due to the reduction in total DASH tokens.

DASH provides a good reference design for Ocean on-chain bounties. We aim to flesh out the Ocean design with learnings from DASH, other governance-oriented blockchains like [Decred2019] and [Tezos2019], and tooling like [Aragon2019] and [Zeppelin2019]. We hope to use off-the-shelf tooling / components where possible.

11. Clans

11.1. Introduction

In Ocean, a "clan" is a group of people, where membership rules for that group are governed by an AragonDAO [Aragon2019] .

Clans are motivated by the following use cases.

- **Contests, Hackathons, Impromptu Collaborations.** A group of hackers or data scientists self-organize to try to solve an AI problem, such as a Kaggle competition or a hackathon. They want to be able to easily access each others' data and compute services as they progress, especially if they are working remotely from each other.
- **Regulatory Sandboxes.** A government wants to give a means for organizations to run in a "monitored" regulatory environment that the government can observe. The organizations are vetted by the government and may have access to specially designated government compute services.
- **Enterprise data access.** An enterprise might make some of its data available to only its employees, but want to be able to use Ocean services available in the broader network.

- **Sharing autonomous driving data.** Individuals in each membership companies of Mobi [Mobi2019] need to access automotive data from any one of the Mobi member companies. It could be time-consuming and error-prone to specify permission for each member company individually. Furthermore, those permissions will fall out of date if Mobi members are added or removed; and updating the permissions one organization at a time could be time-consuming or error-prone. This involves two levels of permissions: access of member companies into Mobi, and access of individuals in each member company (enterprise).
- **Sharing medical data.** Researchers on European soil that wish to directly access German medical data need to demonstrate that they have been accredited by appropriate authorities. This will usually be through their hospital or university. There could be thousands of researcher accessing the data. As with automotive data, it will be time-consuming and error prone to specify and update permissions for each of these thousands of researchers. This may be two levels of permissions (hospital/university into EU authority; individual into hospital/university), or it may be among hospitals and universities in a more networked fashion.

11.2. Clans Design

Ocean's baseline permissioning system grants access to individual addresses. To address the given use cases, Ocean needs a means to grant permissions to a *group* of people. These people are grouped together via Ocean *clan* infrastructure.

A given clan needs membership rules: the process to select and add new members; and to remove existing members. Possibilities include:

- A higher-level authority chooses group members off-chain; and grants on-chain clan membership accordingly. An example is governments or enterprises.
- A group uses off-chain mechanisms (e.g. voting) to choose group members; and grants on-chain clan membership accordingly. An example is Mobi.
- A group uses *on-chain* mechanisms (e.g. voting) to choose group members. On-chain clan memberships then come directly. An example is a .

Voting can take many forms: one member one vote, one token one vote, one member one vote with minimum staking, etc. On-chain voting gives new options, such as a Token Curated Registry (TCR) [Goldin2017].

A member of a clan may be an individual, or another clan. The latter is needed to support nested clans, such as in the use cases of medical data and automotive data.

We aim to implement clans in Ocean using AragonDAOs.

12. Going Permissionless

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).

As we move from PoA to permissionless, our technology options include:

- Continuing to run a dedicated Ocean chain, but with permissionless consensus. Examples include Parity Substrate [ParitySubstrate2019] or Cosmos SDK [CosmosSdk2019] with a permissionless algorithm, or simply Ethereum (V1) code running on a separate network.

- Running on a 3rd party permissionless chain that has achieved both decentralized and scale, e.g. by improving the consensus protocol or sharding. Examples include ETH2.0, Dfinity [Hanke2018], Ouroboros [Kiayias2017], Red Belly [Crain2017], OmniLedger [Kokoris2018]
- Continuing to run a dedicated 100-node PoA Ocean chain, but giving better optionality for dissatisfied users to leave. This includes Plasma [Poon2017] and OST [Ost2019].
- Shifting some work to "Layer 2" state channels like Lightning [Poon2016] or Raiden [Raiden2018]

Factors influencing these choices include:

- The Ocean V1 keeper code is written in Solidity. The Ethereum ecosystem provides decent Solidity developer tools, as well as libraries / ERC standards. Ethereum has the biggest developer community and values well aligned with Ocean.
- Ethereum mainnet hosts many useful services that currently only run there, like decentralized exchanges. However "glue technology" like inter-chain networks can mitigate the effect of this. For example, Aragon is moving to run on Polkadot [Wood2016] in addition to Ethereum mainnet. Other technologies include Cosmos [Kwon2017], Interledger [Thomas2015], and TrueBit [Teutsch2017].

13. Outstanding Concerns

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

- **Usability.** SEAs are powerful but pretty complex for a typical data scientist to use. How do we make them more approachable? Templates will be part of the answer.
- **Complexity / Security.** There's a lot of infrastructure for SEAs, permissioning, validation proofs, curation markets, and all the interactions. Complexity exposes the size of the attack surface.
- **Incentives Fail.** We believe we have a reasonable token design to incentivize supply of AI data & services. However, we have not yet thoroughly verified it with software simulation. Nor have we verified it on a mainnet with economic skin in the game—when the real evaluation begins. We will only feel more comfortable about this once the system is live.
- **Verifiable data availability - maturity issues.** No technology has all of these characteristics: decentralized, fast, general, mature software, no Achilles heel (e.g. SGX hardware exploits). Web2 approaches rely on trusting Web2 cloud providers. Filecoin's Proof-of-Space-Time is not available yet. Teutsch's data availability proof is expensive. Our challenge-response proof needs vetting.
- **Verifiable compute - maturity & speed issues.** No technology is the combination of decentralized, fast, general, mature software, no Achilles heel. Web2 approaches rely on trusting Web2 cloud providers. Homomorphic encryption is not both general and fast. MPC is, but the software is not yet mature. Hardware-based trusted execution environments fail as soon as hardware-level vulnerabilities are found. Secure containers have big attack surfaces.
- **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.

14. Conclusion

This paper presented Ocean Protocol: a protocol and network for AI data and services. It aims to spread the benefits of AI, by unlocking data while preserving privacy. It helps power marketplaces to buy/sell AI data & services, software to publish and access commons data, and AI/data science tools to consume data. This paper described four challenges and a solution for each:

- Ocean helps make data available for use by AI without losing control of the data, via decentralized service agreements and permissioning that bring compute to the data.
- Ocean has network rewards to incentivize a supply of commons data and services.
- Ocean uses on-chain bounties to continuously fund improvements to the network and community.
- Ocean is permissionless with governance to balance stakeholder groups.

15. References

[Amazon2018] "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/

[Aragon2019] "Aragon: unstoppable organizations", homepage, last accessed Feb. 6, 2019, https://aragon.one/

[Arweave2019] "Arweave: Introducing the permaweb," last accessed Feb. 13, 2019, https://www.arweave.org/

[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

[BigchainDB_TRI_2017] 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

[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

[Chainlink2019] "Chainlink: Your smart contracts connected to real world data, events and payments", Homepage, last accessed Feb. 13, 2019, https://chain.link/

[CosmosSdk2019] "Cosmos SDK Documentation", homepage, https://cosmos.network/docs/

[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)

[DaoStack2019] "DaoStack: An operating system for collective intelligence." Homepage. Last accessed Feb. 13, 2019, https://daostack.io/

[DashCentral2019] "Dash Central - Masternode monitoring and budget voting." Homepage. Last accessed Feb. 6, 2019, https://www.dashcentral.org

[DashGovernance2019] "Dash Decentralized Governance System." Homepage. Last accessed Feb. 6, 2019, https://www.dash.org/governance/

[DashWatch2019] "Dash Watch: Discover more about Dash's funded proposals." Homepage. Last accessed Feb. 6, 2019, https://dashwatch.org/

[Decred2019] Decred - Autonomous Digital Currency. Homepage. Last accessed Feb. 6, 2019, https://decred.org

[DeJonghe2018] Dimitri de Jonghe, "Exploring the SEA: Service Execution Agreements", Ocean Protocol Blog, Nov. 30, 2018, https://blog.oceanprotocol.com/exploring-the-sea-service-execution-agreements-65f7523d85e2

[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

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

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

[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

[Hintjens2016] P. Hintjens, Social Architecture: Building Online Communities, 2016, https://www.amazon.com/Social-Architecture-Building-line-Communities/dp/1533112452

[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

[Information2019] “Information wants to be free”, Wikipedia, last accessed Feb. 15, 2019, https://en.wikipedia.org/wiki/Information_wants_to_be_free

[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

[Konečný2016] Jakub Konečný et al., “Federated Learning: Strategies for Improving Communication Efficiency”, Oct. 16, 2016, https://arxiv.org/abs/1610.05492

[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/

[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

[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

[McConaghy2019] Trent McConaghy, “Ocean on PoA vs. Ethereum Mainnet?”, Feb. 12, 2019, https://blog.oceanprotocol.com/ocean-on-poa-vs-ethereum-mainnet-decd0ac72c97

[Mobi2019] “Mobi - Mobility Open Blockchain Initiative”. Homepage. Last accessed Feb. 7, 2019, https://dlt.mobi/

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

[Monegro2018] Joel Monegro, “The Price and Value of Governance”, Video, recorded at Token Engineering Global Gathering, May 13, 2018

[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

[OceanAquarius2019] “Aquarius: provides an off-chain database store for metadata about data assets. It's part of the Ocean Protocol software stack”, last accessed Feb. 11, 2019, https://github.com/oceanprotocol/aquarius

[OceanBrizo2019] “Brizo: Helping to Publishers to expose their services”, last accessed Feb. 11, 2019, https://github.com/oceanprotocol/brizo

[OceanBlog_AccessControl2018] “Secure On-Chain Access Control for Ocean Protocol”, Ocean Protocol Blog, Dec. 5, 2018, https://blog.oceanprotocol.com/secure-on-chain-access-control-for-ocean-protocol-38dca0af820c

[OceanBlog_Roadmap2019] “Ocean Roadmap || Setting sail to 2021”, Ocean Protocol Blog, Feb. 13, 2019, https://blog.oceanprotocol.com/ocean-roadmap-setting-sail-to-2021-70c2545547a7

[OceanDocs2019] “Ocean Protocol Software Documentation”, last accessed Feb. 11, 2019, https://docs.oceanprotocol.com/

[OceanDataScience2019] “Manta Ray: Data Science powered by Ocean Protocol”, documentation, last accessed Feb. 11, 2019, https://datascience.oceanprotocol.com/

[OceanPleuston2019] “Pleuston: Web app for consumers to explore, download, and publish data assets”, github repository, last accessed Feb. 11, 2019, https://github.com/oceanprotocol/pleuston

[OEPs_2019] “Ocean Enhancement Proposals (OEPs),” last accessed Feb. 11, 2019, https://github.com/oceanprotocol/OEPs

[OEP3_2019] “Ocean Enhancement Proposal 3: Ocean Protocol Network Architecture,” last accessed Feb. 11, 2019, https://github.com/oceanprotocol/OEPs/tree/master/3

[OEP7_2019] “Ocean Enhancement Proposal 7: Decentralized Identifiers,” last accessed Feb. 11, 2019, https://github.com/oceanprotocol/OEPs/tree/master/7

[OEP8_2019] “Ocean Enhancement Proposal 8: Assets Metadata Ontology,” last accessed Feb. 11, 2019, https://github.com/oceanprotocol/OEPs/tree/master/8

[OEP11_2019] “Ocean Enhancement Proposal 11: On-Chain Access Control using Service Agreements,” last accessed Feb. 11, 2019, https://github.com/oceanprotocol/OEPs/tree/master/11

[OpenMined2019] “OpenMined: Building Safe Artificial Intelligence”, homepage, last accessed Feb. 19, 2019

[Oraclize2019] “Oraclize: Blockchain Oracle Service, Enabling Data-Rich Smart Contracts”, Homepage, last accessed Feb. 13, 2019, http://www.oraclize.it/

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

[Ost2019] “OST: Powering blockchain economies for forward thinking businesses”, Homepage, last accessed Feb. 13, 2019, https://ost.com/

[Ostrom1990] Elinor Ostrom, Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge, UK: Cambridge University Press, 1990.

[ParityEth2019] “Parity Ethereum: the fastest and most advanced Ethereum client”, last accessed Feb. 12, 2019, https://www.parity.io/ethereum/

[ParitySecret2019] “Parity Secret Store - Wiki” (Parity Secret Store Documentation), last accessed Feb. 12, 2019, https://wiki.parity.io/Secret-Store

[ParitySubstrate2019] “Parity Substrate: Build your own blockchains”, last accessed Feb. 19, 2019

[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

[Pfitzmann2000] Birgit Pfitzmann et al., “Provably Secure Certified Mail”, IBM Research Report RZ 3207 (#93253) 02/14/00, https://www.researchgate.net/publication/2645559_Provably_Secure_Certified_Mail

[PoaNetwork2019] “The TokenBridge Interoperability”, Homepage, https://poa.network/bridge

[PoaNetwork2019b] “POA Network: Public Ethereum sidechain with Proof of Autonomy consensus by independent validators”, Homepage, https://poa.network

[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

[PoRep2018] 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

[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

[SchemaOrg2019] “Schema.org DataSet schema”, last accessed Feb. 11, 2019, https://schema.org/Dataset

[Schneier1998] B. Schneier and J. Riordan, “A Certified Email Protocol”, IBM, 1998, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.6998

[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/

[SquidApi2019] “Squid API Specification”, last accessed Feb. 11, 2019, https://github.com/oceanprotocol/dev-ocean/blob/master/doc/architecture/squid.md

[SquidJs2019] “Squid-JS: JavaScript client library for Ocean Protocol”, last accessed Feb. 11, 2019, https://github.com/oceanprotocol/squid-js

[SquidPy2019] “Squid-PY: Python client library for Ocean Protocol”, last accessed Feb. 11, 2019, https://github.com/oceanprotocol/squid-py

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

[Teutsch2017] Jason Teutsch and Christian Reitwießner, “A Scalable Verification Solution for Blockchains”, November 16, 2017, https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf

[Teutsch2017b] Jason Teutsch, “On decentralized oracles for data availability”, Dec. 25, 2017, http://people.cs.uchicago.edu/~teutsch/papers/decentralized_oracles.pdf

[Tezos2019] Tezos. Homepage. Last accessed Feb. 6, 2019, https://tezos.com/

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

[TlsNotary2019] “TLSnotary - a mechanism for independently audited https sessions”, Sept 10, 2014, https://tlsnotary.org/TLSNotary.pdf

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

[Web3js2019] “web3.js - Ethereum JavaScript API”, last accessed Feb. 11, 2019, https://github.com/ethereum/web3.js/

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

[Zeppelin2019] “ZeppelinOS: Develop, deploy and operate any smart contract project securely”, homepage, last accessed Feb. 6, 2019, https://zeppelinos.org/

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

16. Appendix: Secret Store

16.1. Introduction

Ocean needs a secure, scalable way to share secrets among parties such as publishers, consumers, Providers, and Smart Contracts.

Parity Secret Store is a feature included in the Parity Ethereum client [ParityEth2019]. It allows users to store a fragmented ECDSA key on the blockchain, such that retrievals are controlled by a permissioned Smart Contract.

The Secret Store implements a threshold retrieval system, so individual Secret Store nodes are unable to reconstruct the keys to decrypt documents by themselves. A Secret Store node only saves

a portion of the ECDSA key. The decryption key can only be generated if a consensus is reached by an amount of Secret Store nodes bigger than the threshold that the publisher of the secret chooses.

From the Parity Secret Store documentation [ParitySecret2019]:

The Parity Secret Store is core technology that enables:

- *distributed elliptic curve (EC) key pair generation - key is generated by several parties using special cryptographic protocol, so that:*
  - *private key portion remains unknown to every single party;*
  - *public key portion could be computed on every party and could be safely exposed to external entities;*
  - *every party hold the 'share' of the private key;*
  - *any subset of t+1 parties could unite to restore the private portion of the key;*
  - *any subset of less than t+1 parties could not restore the private portion of the key;*
- *distributed key storage - private key shares are stored separately by every party and are never exposed neither to another parties, nor to external entities;*
- *threshold retrieval according to blockchain permissions - all operations that are requiring private key, require at least t+1 parties to agree on 'Permissioning contract' state.*

This last point can enable to Ocean Protocol to have a solid mechanism to distribute encrypted contents between parties (consumers and publishers), and these contents can only be decrypted after an on-chain authorization phase.

16.2. Architecture

The integration of the Secret could have the following characteristics:

- All the negotiation required to encrypt or decrypt a resource is happening without writing any information on-chain
- This integration only requires standard HTTP requests between the clients (consumer, publisher) and the Parity EVM & Parity Secret Store API's (no gas, quick)
- Requires the usage of the Parity EVM Secret Store API existing in the Parity Ethereum clients
- Requires the usage of the Parity Secret Store API. This software is part of the Parity Ethereum client
- It's based in the usage of a permissioned Secret Store cluster. This cluster would be deployed/governed by the Ocean Protocol Foundation at network launch as part of the base Ocean core capabilities. Additional parties (user/companies/organizations) could be added as members to decentralize the responsibility of running this infrastructure.
- Publishers and consumers could use their own Parity Secret Store cluster. The cluster to use is part of the Squid configuration used by the different users of Ocean.

Figure 26 shows the high-level architecture using secret store.

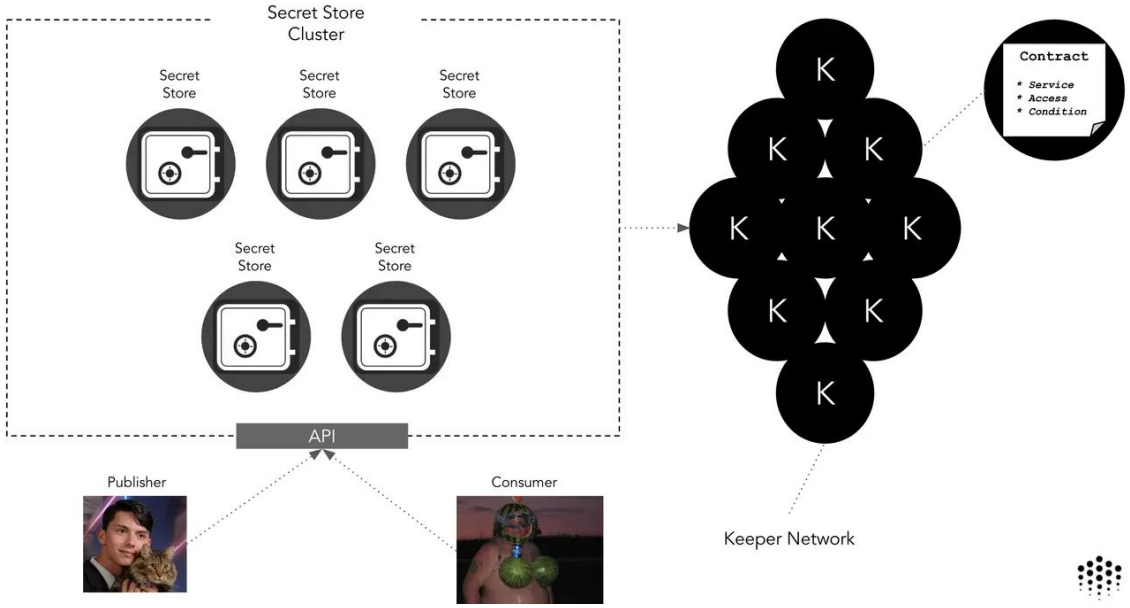

Figure 26: High-Level Architecture using Secret Store

16.3. Encryption

Figure 27 shows the standard Parity Secret Store publishing flow.

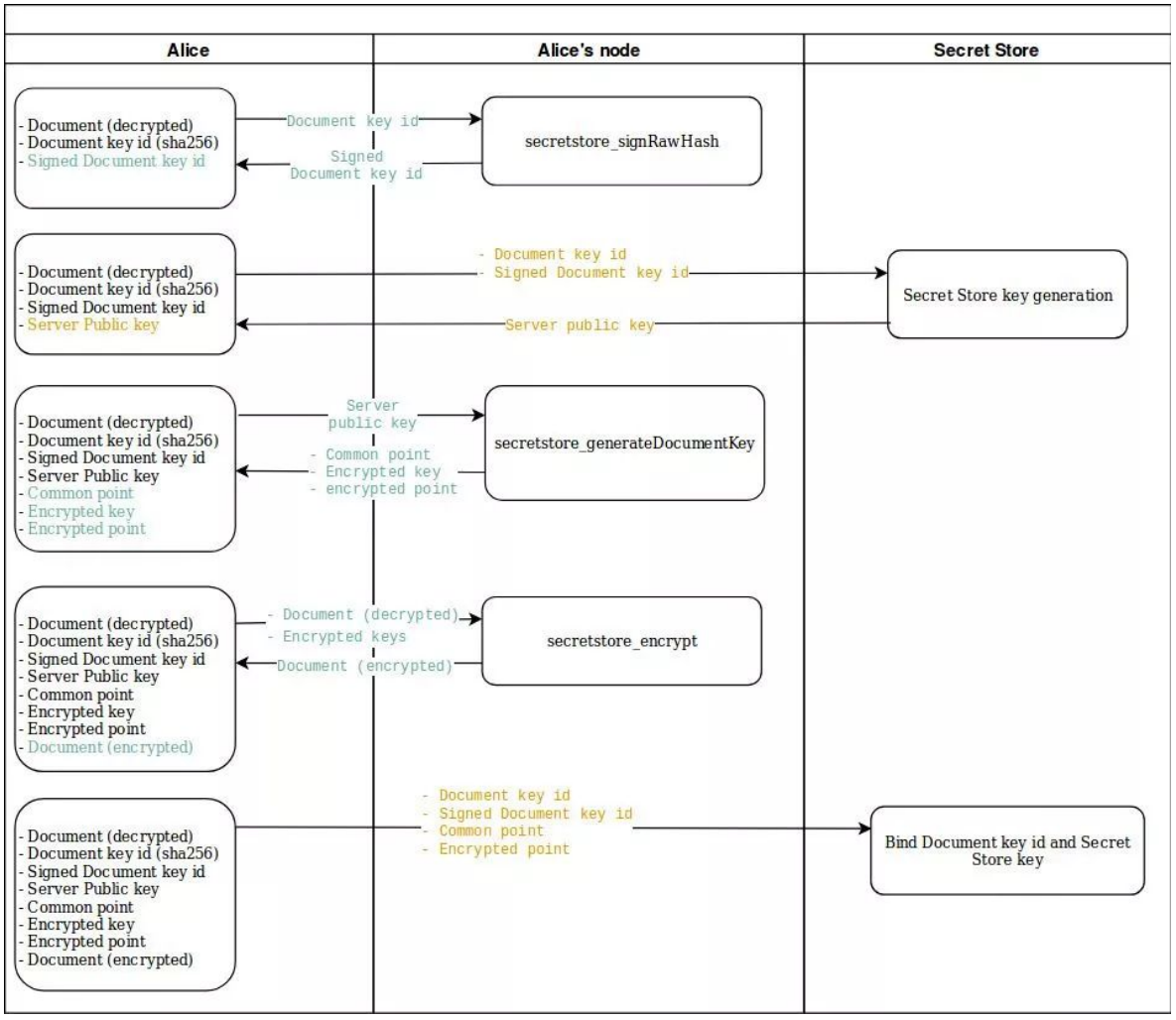

Figure 27: Secret Store Publishing Flow

Ocean implements this flow in various Ocean Protocol Secret Store clients (Javascript, Python, Java). It is abstracted as part of the encryptDocument Squid method. A publisher (such as Alice in the diagram above) can use encryptDocument to encrypt a document and to store the decryption key in the Secret Store cluster.

The type of document to encrypt is totally flexible. A document could be one or multiple URLs, access tokens to an external resource, etc. Typically in Ocean, during the Asset access phase, what we are decrypting is the URL to get access to an Asset that is stored in a cloud provider.

This could be extended in the future allowing configuration of access policies at runtime. This would allow the granting of access to specific IP addresses to get access to assets, after the authorization phase.

Figure 28 shows how a Secret Store integrates into the Ocean publishing flow.

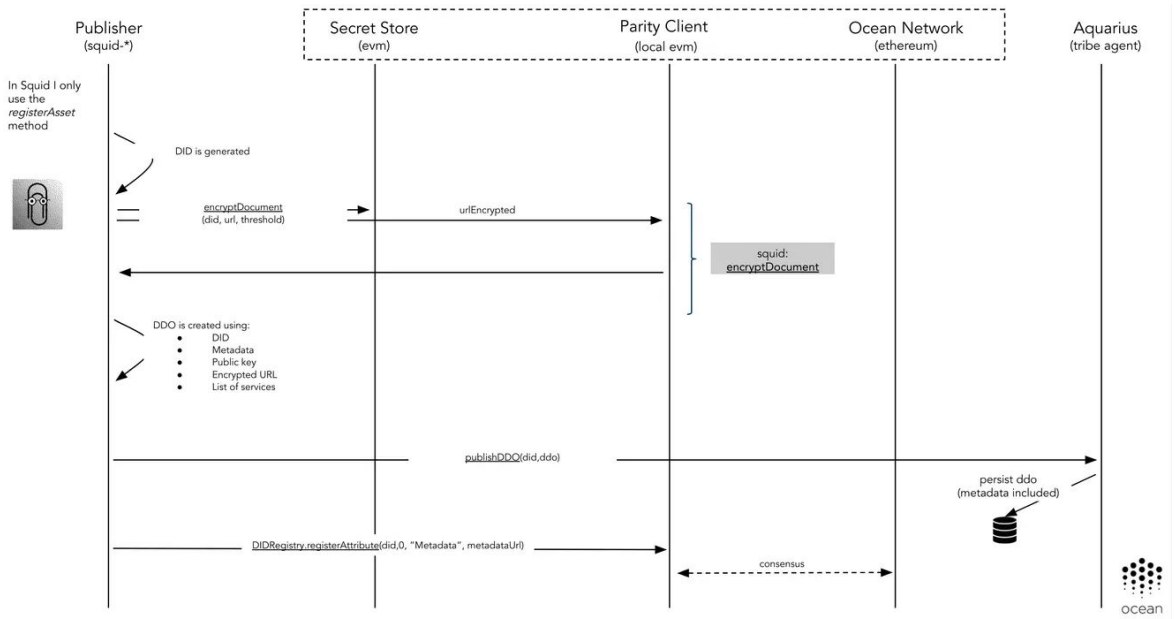

Figure 28: Ocean Secret Store Publishing Flow

As a result of this flow, the encrypted document can be shared with the potential consumers (e.g. as part of the metadata or via libp2p).

The action of granting permissions on-chain to a specific user is not part of this flow.

16.4. Decryption

Figure 29 shows the standard Parity Secret Store consuming flow.

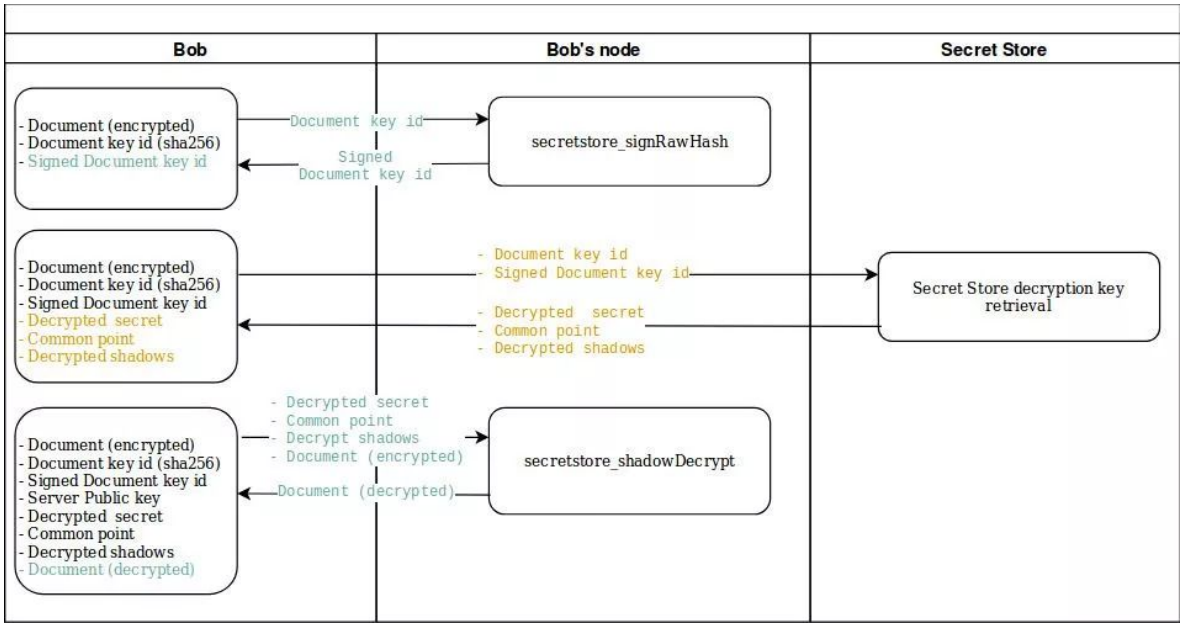

Figure 29: Standard Parity Secret Store Consuming Flow

This logic is encapsulated as part of the decryptDocument Squid method. This method allows a consumer, given a resource unique id and a encrypted document (shared for the publisher via metadata or libp2p), to decrypt this document using the Secret Store cluster capabilities.

Decryption only can be achieved if the Secret Store cluster achieves the quorum specified by the publisher during the publishing process with the **threshold** attribute.

The Secret Store checks the user's authorization permissions on-chain to determine if they are authorized to decrypt a document. The Secret Store won't allow decryption if the user doesn't have authorization.

Figure 30 illustrates how a Secret Store integrates into the Ocean asset consumption flow.

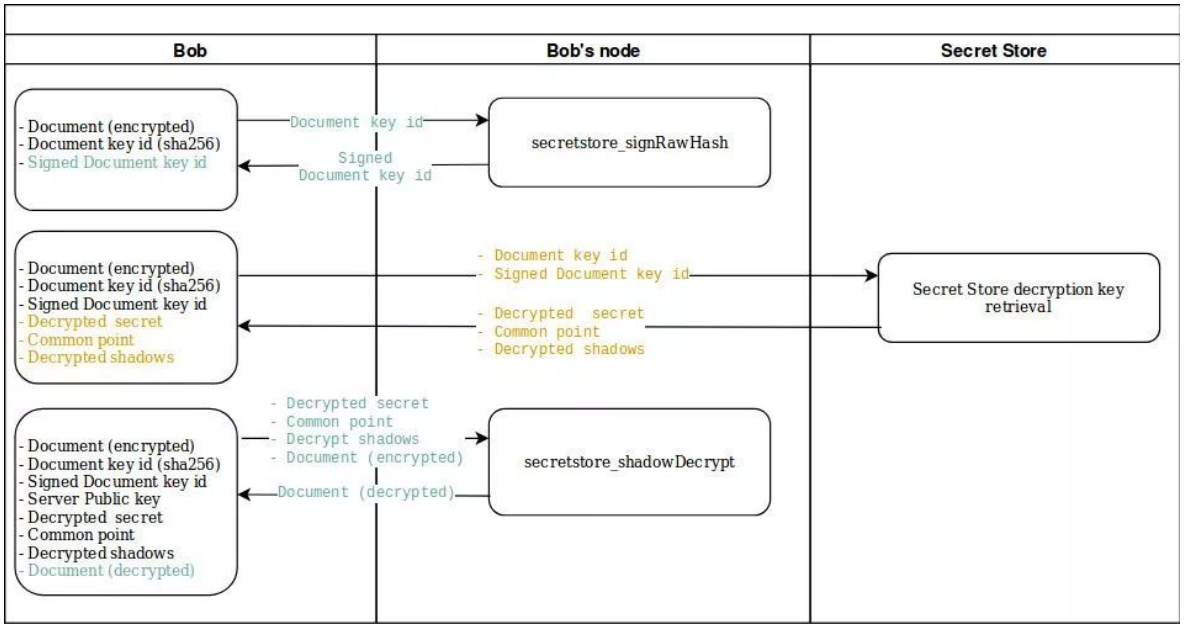

Figure 30: Ocean Secret Store Consuming Flow

16.5. 17.1.5. Authorization

OEP-11 [OEP11_2019] describes how understand how Service Agreements interact with a Secret Store to manage access control.

16.6. 17.1.6. Deployment

Secret Store functionality is provided by a permissioned cluster that will be executed as part of the Ocean Protocol basic infrastructure.

Nodes can only be added to this cluster by changing the configuration of the members of the cluster, so a third-party malicious user can't join the cluster without changing the configuration of the other nodes.

17. Appendix: Computing Services Details (Brizo)

This section describes how Ocean Protocol (with Brizo) enables publishers to provide computing services and related services.

17.1. Motivation

The most basic scenario for a publisher is to provide access to the datasets they own or manage. In addition to that, a publisher could offer other data-related services. Some possibilities are:

1. A service to execute some computation on top of their data. This has some benefits:
    - The data **never** leaves the publisher enclave.
    - It's not necessary to move the data; the algorithm is sent to the data.
    - Having only one copy of the data and not moving it makes it easier to be compliant with data protection regulations.
2. A service to store newly-derived datasets. As a result of the computation on existing datasets, a new dataset could be created. Publishers could offer a storage service to make use of their existing storage capabilities. This is optional; users could also download the newly-derived datasets.

17.2. Architecture / Enabling Publisher Services (Brizo)

The direct interaction with the infrastructure where the data resides requires the execution of a component handled by publishers.

This component will be in charge of interacting with users and managing the basics of a publisher's infrastructure to provide these additional services.

The business logic supporting these additional publisher capabilities is the responsibility of this new technical component.

The key component introduced to support these additional publisher services is named Brizo. Brizo is the technical component executed by the publishers, which provides extended data services. Brizo, as part of the publisher ecosystem, includes the credentials to interact with the infrastructure (initially in cloud providers, but it could be on-premise).

Because of these credentials, the execution of Brizo *should not* be delegated to a third-party.

Figure 31 illustrates how Brizo relates to Keeper Contracts and cloud providers.

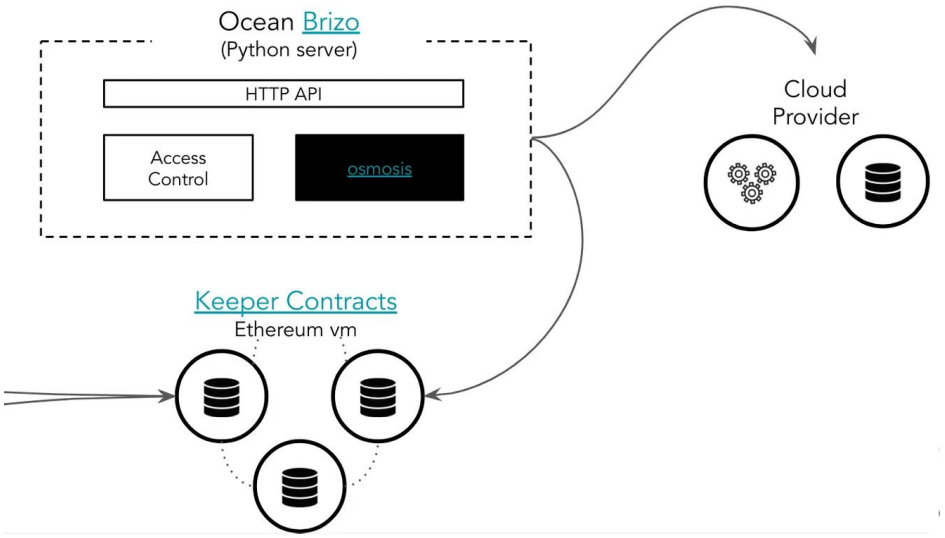

Figure 31: Brizo High-Level Architecture

17.3. Responsibilities

The main responsibilities of Brizo are:

- Expose an HTTP API allowing for the execution of some extended, data-related services (compute and/or storage).
- Authorize the user on-chain using the proper Service Agreement. That is, validate that the user requesting the service is allowed to use that service.
- Interact with the infrastructure (cloud/on-premise) using the publisher's credentials.
- Start/stop/execute computing instances with the algorithms provided by users.
- Retrieve the logs generated during executions.
- Register newly-derived assets arising from the executions (i.e. as new Ocean assets).
- Provide Proofs of Computation to the Ocean Network.

17.4. Flow

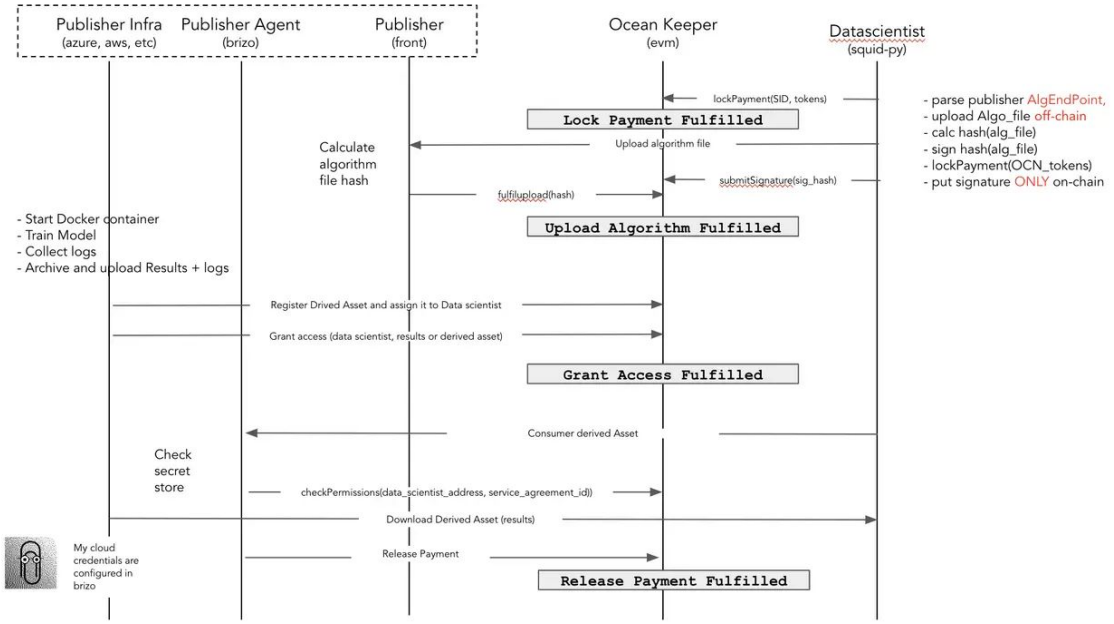

Figure 32: Sequence Diagram for computing services

Figure 32 shows the sequence diagram for computing services. It involves the following components/actors:

- Data Scientists - The end users who need to use some computing/storage services offered by the same publisher as the data publisher.
- Ocean Keeper - In charge of enforcing the Service Agreement by tracing conditions.
- Publisher Infrastructure - The storage and computing systems where the publisher's data resides (in a cloud provider or on-premise).
- Publisher Agent (Brizo) - Orchestrates/provides additional publisher services.
- Publisher - The actor running Brizo and other software. The publisher is often the same as the data owner.

Before the flow can begin, the following pre-conditions must be met:

- The Algorithm must be implemented in one of the languages/frameworks supported by the publisher.
- The Asset DDO must specify the Brizo endpoint exposed by the publisher.

- The Service Agreement template must already be predefined and whitelisted on-chain.

The following describes the steps in the flow.

1. Set Up the Service Agreement

- The game begins with a data scientist looking for a data asset and finding one where the data is attached to its own compute service. They pick the asset, sign the agreement, then send signature, templateId to the asset publisher.
- The publisher sets up a new Service Agreement instance on-chain, using the data scientist's signature.

2. Fulfill the Lock Payment Condition

- The data scientist listens for the ExecuteAgreement on-chain event, to lock the payment in the PaymentCondition.sol smart contract and fulfills the condition.

3. Fulfill the Upload Algorithm Condition

- The data scientist parses the DDO (using Squid) to see how the publisher exposes a computing service. The computing service defines how to upload an algorithm to the publisher side and how to consume an output.
- The data scientist uploads the algorithm off-chain directly to the data set publisher's algorithm endpoint.
- Once the algorithm files are uploaded, the data scientist calculates the files' hash, signs this hash and then submits ONLY the signature on-chain.
- The publisher receives the algorithm files, calculates the hash (algorithm files) and submits this hash to fulfill the uploadAlgorithm condition. The Keeper Contracts automatically verify that both parties see the same files using 1) the hash which is submitted by the publisher; 2) the signature submitted by the data scientist; and, 3) the data scientist's address.
- The Brizo software component is responsible for running the algorithm, attaching data assets, and running the container.

4. Fulfill the Grant Access Condition

- Meanwhile, the publisher starts computation, trains the model, collects logs and generates outputs (the derived assets). Finally, the publisher archives and uploads the outputs.
- The publisher registers the derived assets (outputs) as DIDs/DDOs and assigns ownership to the data scientist.
- The publisher grants access to the data scientist using the Secret Store and fulfills the grant access condition.

5. Release Payment

- Once the grant access condition is fulfilled (within a timeout), the publisher can release payment.
- The data scientist can consume the derived assets (outputs) by calling the Secret Store which in turn checks the permissions on-chain, and then downloads the outputs using the decryption keys.

6. Cancel Payment

The payment can be cancelled only if:

- The access to the derived assets (outputs) was not delivered within timeout period, or
- Payment was never locked.

7. Agreement Fulfillment

- The Service Agreement will be accepted as fulfilled only if the payment is released to the service provider.

18. Appendix: Addressing Key Goals in Incentive 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. Bringing compute to the data and incentives are two complementary ways to drive this. On the latter, as previously discussed, we developed ey criteria to compare candidate token incentive designs against. Table 3 describes the question / criteria (left column) and how the token design addresses those criteria (right column).

Table 3: How Ocean incentive design addresses key goals

Key Question	
For priced data, is there incentive for supplying more?	Network 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?	Few will download low-quality data, i.e. spam. Therefore even fewer are incentivized to stake on it in a curation market. So while it can exist in the system 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 Curated Proofs 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 network 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 network 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 network reward function is implemented as a binding of cryptographic proofs *curation market, to form a Curated Proofs 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 be 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.

19. Appendix: FAQs and Concerns

This section addresses frequently asked questions and concerns about Ocean.

19.1. 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 [Amazon2018]) 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 data providers can remove the data, if it's not on a decentralized and immutable substrate. Marketplaces would de-list the data accordingly. This is a highly useful feature for data protection regulations, hate speech regulations, and copyrighted materials.

19.2. Arbitration

Q1: What if I buy data, and the seller gives me garbage? Do I have recourse?

A1: Ocean SEAs allow for inclusion of traditional legally-binding contracts, which in turn often have clauses on arbitration, such as "if the parties disagree, the matter will be taken to arbitration court of jurisdiction x". It makes sense for data and services marketplaces to include such contracts (and corresponding clauses) in their terms of service, and in each specific SEA template that they use.

We anticipate that different marketplaces will gain specialties in serving different jurisdictions and data / service asset classes.

The legal contracts can be linked to the network as well, e.g. by putting it into IPFS and including the hash of the legal contract in the SEA instance [REF RICARDIAN CONTRACT GRIGG].

19.3. IP Rights Violation : Paid Data

Q: When someone registers a work in Ocean, they are claiming to have the appropriate IP rights to make that data available to others. What if they don't actually have those rights, and make money selling the data?

A: Given that it's paid data, then there will almost certainly be a legal contract (provided by the marketplace). This contract will include recourse to such violations, such as 3rd party arbitration.

19.4. IP Rights Violation : Free Data (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 network reward tokens. Finally, the actor leaves the network, taking their earnings with them (and none to the rights holder).

A: If it's commons data, there might not be a legal contract, just a SEA. The baseline recourse for this is similar to how modern websites handle copyrighted material today. With the USA's Digital Millennium Copyright Act, copyright holders have the onus to identify material that breaches copyright. If they identify that material, the host needs to remove it or de-index it. It's similar in Ocean: the marketplace will de-index the infringing work. Recall that the Ocean network itself stores no metadata or actual data, it's just a pass-through. Furthermore, Ocean has a 30 day delay before network rewards are given to service providers, to add friction to "hit and run" jobs.

19.5. 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" network rewards, because there will be more downloads for something that's free, versus paid.

A: 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.

19.6. 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 network 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 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.

19.7. 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.

19.8. 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 $\log_{10}$ on stake.

19.9. Pump-and-Dump on Drops

Concern: Recall that each AI dataset/service has its own token - its *drops* Ɖ, 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 Ɖ, then promotes that dataset/service to others (pumps), and finally sells their Ɖ at a big profit and leaves everyone else hanging (dumps).

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

- Have one bonding curve for buying Ɖ and a *second* one for selling Ɖ, 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 Ɖ 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.

19.10. Network 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 network 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 network 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 be an intermediary 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.

20. Appendix: One-Token Vs. Two-Token Systems

Recall that Ocean Tokens are used as the medium of exchange for buying/selling services, in addition to staking and network rewards.

We contemplated a two-token design, where the unit of exchange was the DAI stablecoin (or something similar). The reasoning is so that buyers and sellers are less subject to fluctuations in the value of OCEAN. However, that is less of an issue with just-in-time exchange.

The single-token approach has advantages. First, it's simpler. But the biggest advantage is to help create a data economy where the everyone has equal access to the benefits of assets which grow in wealth over time. Cash typically depreciates over time. Binding assets to cash helps ensure that those who rely on day-to-day cash have greater exposure to upside opportunities [Monegro2018].

21. Appendix: Data Pricing Strategies

Ocean Protocol enables many types of data pricing strategies. Here is a sampling.

Free market / fixed price. Here, the data provider fixes a price and let's the market feedback on it - buy it or not. "Here is my data, take it or leave it". Market feedback might influence pricing setting if provider is willing to listen. We believe other strategies will likely be more effective.

Auctions. People bid and the winning bidder gets the dataset/service. There are variants: English (lower bound on price that increases), Dutch (upper bound on price that decreases), Channel (both lower and upper bound on price).

Incentivized commons. "I put this data in the commons (free to consume), and when it gets used I get network rewards". This is built into Ocean's incentives system. A data provider could go further by mandating that network rewards be reinvested in the system.

Royalties. "You use my data and I get a % every time you create value." Simple but powerful, this is how music and many other industries work. We can bring this to AI data.

Prize. There's a prize to which competitor does the best according to some performance indicator. Mechanics are like Kaggle or the Netflix Prize. Here, there is no fixed finish line; the best result (or results) get the reward. Usually, this is time-bound.

Bounty. Whoever solves the problem first, or gets past some performance threshold first, wins the prize. There is a fixed finish line. The competition ends as soon as a competitor crosses the finish line.

For some pricing schemes, like fixed price and auctions, the data vendor may need to create digital scarcity to justify a price. This is possible, eg. by limiting access to a single dataset to 10 parties over a time interval.