Synamint Protocol

Todd Chapman
todd@snickerdoodlelabs.io

Lucas Novak
lucas@snickerdoodlelabs.io

Varun Parthasarathy
varun@snickerdoodlelabs.io

October 17, 2023

Contents

1 Abstract

The Synamint Protocol aims to enable an open, equitable, and consent-driven federated data market in which individuals will collect, own, and authorize the use of their own data. Data is most valuable in the aggregate so that it can be leveraged to build predictive models; this results in a data market where centralized aggregators hold the only economically valuable role. In this structure, individual users who actually generate data are often left with little to no benefit from the value of their data.

As more and more privacy violations and data breaches continue to occur on a larger scale, it has become apparent that the centralization of data poses a serious threat to both the individual and the community as a whole as it results in a single point of failure for malicious actors to focus their efforts. Technology companies, like Apple and Google, have produced software designed to provide more privacy forward data collection through federated learning and multi-party computation models optimized to run on hardware devices that these companies produce. However, these solutions are neither auditable nor open, and therefore still leave the end user removed from the value they generate. In addition, existing decentralized solutions often focus primarily on the sale of aggregated data sets. The protocol described in this whitepaper aims to enable the extraction and collection of new data in real time via edge computing techniques leveraging a permissionless blockchain as an orchestration layer and control plane.

This protocol will alleviate these concerns by turning user data into an individualized asset that can be leveraged without the need for an initial consolidation step. User-controlled localized "Data Wallets" will provide an easily usable form factor to the end user, while a business-oriented "Insights Service" will provide analytical and visualization services to entities who wish to extract intelligence from the data itself. By providing consent-driven ownership and granular access control to the end-user, we allow them to gain greater sovereignty over their digital identity, while freeing businesses from the overhead of regulatory compliance with regard to user data. Our solution will be permissionless, community-governed, and self-sovereign, and will provide an alternative to the data economy of today. Internet users are already recognizing the need for decentralization in the information economy, as demonstrated by the fact that the Web3 user base is growing at a rate comparable to the internet in the 1990s. By properly incentivizing these users, allowing businesses to reach more users with less risk, and creating an open protocol for decentralized infrastructure providers, the protocol will facilitate a data economy that benefits all.

2 Introduction

The current data economy operates an a massive scale and is trending to continue to increase year-over-year [1][2][3]. In much the same way many corporations are valued more for their financial power and assets than their businesses (airlines, fast food, etc.), information businesses have come to be valued primarily for the data they have accumulated. From location history to search queries, user data has become a key asset for evaluating Big Tech and information businesses [4][5]. In any economy, value generation is driven by the extraction and utilization of resources. In an information economy, the primary resource is data.

In the second internet age ("Web2"), the data economy has been largely characterized by consolidation [6][7]. Much like the post-industrial resource economies of the United States in the early 20th century, the digital information space has been heavily dominated by a small number of large corporate entities focused on monetizing business intelligence that can be extracted from large data sets [1]. Large corporations collect, store, and manage immense amounts of data within consolidated, permissioned, data lakes, and use this information advantage to wield immense market power and maintain dominance [8]. Unlike physical resources, however, digital data is infinitely copy-able, usable, and transferable. This presents a number of challenges within the realms of security, privacy, and overall economic equity.

Despite the rapid increase in collection capabilities and the immense value of actionable information embedded within the data, the role of the individual within the traditional data economy has been largely minimized [8][2]. While new technologies have led to an explosion in the amount of data that can be collected and organized, the originator of said data has limited control and visibility into its collection, its usage, or the distribution of its value [9]. Users are not equitably rewarded for their role within the data economy, which is demonstrated by the surplus-value of the entities who dominate the data markets today [8][3][1]. In addition, collection, management, and storage are all often third-party activities, leading to de-facto ownership through the consolidation of roles.

In addition, the lack of transparency and accountability once the data has been extracted has led to

widespread concerns surrounding privacy and security within the existing data economy [9][6]. Large companies like Facebook can lose track of where the data of their users is stored [10]. Without strong and auditable security and privacy controls to protect user data, users are at risk and could be exposed to preventable data breaches, fraud, identity theft, and malicious data usage. The Synamint Protocol, which will be referred to as "the Protocol", aims to reorient the data economy towards a more user-driven model. By putting the user in control of the data they generate, the protocol will allow individuals to properly consent to usage, claim equitable compensation for their participation in the data economy, and gain transparency into how their data is leveraged. Additionally, businesses and other data-consuming entities will benefit through implicit regulatory compliance and increased competition in and access to the data market.

Recent innovation in the fields of distributed computing and cryptography have led to the emergence of large-scale permissionless, trustless, and highly available decentralized networks [11]. This has led to utility most notably in the form of new digital asset classes and markets, but has also resulted in new forms of digital identity, signature schemes, distributed governance, and auditable computation [12]. A token economy will serve as the compensation mechanism for actors within the user data economy and provide governance functionality, while smart contracts and non-fungible tokens will serve as the backbone of auditable and transparent consent mechanisms for data sharing. However, in general, an increase in transparency typically results in a decrease in privacy, making it critical that data be stored in a private and secure fashion. Thus, the Protocol will serve as a decentralized data control and coordination plane to facilitate the sharing of data, while individual users and the entities they engage with will collectively form the data delivery plane it administers.

This paper will discuss the proposed protocol that allows individuals to securely and privately collect, manage, and store their data and allow interested parties to safely extract value from it. In addition, the protocol will equitably compensate all actors within the system, and seeks to minimize the monopolistic tendencies of the existing data economy. The Protocol aims to create a decentralized user data economy that is open and benefits all.

3 Background

This section will address our problem statement, the terminology we will be using throughout the paper, and other solutions that have been proposed or built to try and address similar problems.

3.1 Problem Statement

Individuals are constantly producing data from which actionable business intelligence is derived but these individuals do not own this value creation

Large companies are constantly monitoring their users for data while these users do not have a viable mechanism by which to control their data. This observation and collection of end-user data played a large factor in the shaping of the modern economy, even called surveillance capitalism by some. This has led to a variety of negative consequences, such as who gets the value and security, privacy, surveillance consequences, transparency, compensation, and consent.

The Protocol will help individuals control their own data as well as understand its use and derive value from it. This will increase the security and privacy of data and allow people to effectively monetize the data they generate. Additionally, it will make it simpler for companies interested in data to run analysis while respecting data privacy legislation as they can use the protocol as their data infrastructure.

3.2 Terminology

This section will define those terms and give context to why they are important.

3.2.1 Decentralization

Decentralization: When control over a system is held by a group rather than a single authority.

In order to prevent a single party from acquiring undue influence in the data economy, including Snickerdoodle Labs, the Protocol will be built on top of a decentralized blockchain data structure.

An important note is that decentralization by itself not the ultimate goal. Rather, the goal of the Protocol is to be permissionless, trustless, and available. Currently, only known viable way to achieve these properties is by designing the protocol to be inherently decentralized.

Permissionless Anyone should be able to interact with the protocol without the permission of a trusted third party. Snickerdoodle Labs must not be in a position to decide which individuals are able to collect and share their data, choose what businesses are able to request data, and what developers are able to build on top of the protocol.

The rules of the protocol will be determined by a Decentralized Autonomous Organization (DAO) which will provide a decentralized mechanism to manage the Protocol. See section 5.2.5 for more details about the Protocol DAO.

Trustless The operation of the system should not require trust in a particular, centralized third party in order for the protocol to function. Actors in the system must not need to rely on Snickerdoodle Labs or any other actors in order to own their own data or acquire insights.

It is worth noting that completely trustless systems do not exists, but there are varying levels of trustlessness. Ideally, one can trust many mathematicians and engineers that the math and systems built will force the system to behave in the correct way. In the worst case, a strong financial incentive can be relied on for the system to behave correctly. The Snickerdoodle Protocol will rely on both paradigms of trustlessness and aim to update the protocol to use stronger forms of trustlessness over time.

Availability

Actors in the system should be able to take feasible actions in the system in a reasonable amount of time.

3.2.2 Data Safety

Data Safety: Data is considered safe if it is securely stored and privately viewed.

When describing data, we often say that it is *safe* in order to encompass all aspects of data security and privacy. Safe data is data that is securely written, stored, transmitted, and accessed in a privacy-preserving manner. This means that the Snickerdoodle Protocol will have to have a strong sense of identity management that only allows authorized people are able to access and know about the data. We implement this identity management via the Snickerdoodle Data Wallet discussed in section 5.3.1.

3.2.3 Data Subscribers & Insights

Data Subscriber: A data subscriber is a data-consuming entity that pays for temporary access to data to gain insights

Insight: An insight is actionable intelligence gained from applying a function or algorithm to an appropriately structured data set.

In the modern data economy, all organizations need to make informed operational decisions. Data subscribers are interested primarily in the insights that data provides rather than the raw data itself. In a world where data is owned by individuals, organizations would not be able to store and own individual data forever, rather they would pay to be granted temporary access to that data to produce insights.

3.3 Other Solutions

There are a variety of different approaches and technologies that aim to allow users to own their own data. In this section, we discuss some of these approaches

3.3.1 Policy Solutions

Policy decisions such as GDPR in the European Union or the CCPA in California, attempt to regulate consolidated data warehouses and other types of centralized storage. These give individuals rights that allow them to control how their data is used. These laws are a positive step towards giving individuals ownership over their data. However, these laws can be hard for both end-user and application developers to interpret. Snickerdoodle Labs aims to address these problems by building a system that is compliant with these regulations by default, easy for developers to leverage for their applications, and easy for individuals to express their rights.

3.3.2 Data Sharing Techniques

There also exist a number of solutions that attempt to tackle the issues surrounding data-sharing. For example, perturbation techniques like differential privacy have shown promise in sharing noisy and/or anonymized data with limited value loss [13]. Techniques such as federated learning and multi-party-compute have been used to train models on distributed data sets [14][15]. In addition, data outsourcing techniques have been employed to separate the management of data from its storage [16]. All of these solutions are still yet to find practical applications for the most part and are hard for others to build on.

3.3.3 Web3 Solutions

The distributed nature of web3 technologies provide a natural way to explore data ownership and decentralize the control of data. Ceramic creates a way to create and link existing databases in a decentralized manner and manage their identity [17]. Ocean Protocol creates a data set market by allowing people to sell access to data sets and bring compute to data [18]. While these projects are inspired and create new ways to interact with data in a decentralized way, they don't address the problem of allowing individuals to own and control their data.

4 Data Ownership

In this section, we will define data ownership, explain why defining data as an asset is tricky, and establish the flow of value in the current data economy.

4.1 Ownership

The Snickerdoodle Protocol aims to shift the balance of power within the existing data economy by providing individuals greater control of their data. To do so, we must first understand the value within the existing data economy and what it means to control and own data.

Data Ownership: If an individual can exclusively control and manage the collection, storage, and usage of an attribute of their data corpus in a secure and private manner, then it can be said that they own that particular data attribute.

Ownership of data is a difficult concept to define. Unlike physical resources, data can be copied indefinitely, is generated constantly, and generally requires technical expertise and extensive cyber infrastructure for collection and value extraction. Because of these properties, the safety of data is crucial to data ownership and makes regulating the use of data inherently difficult.

To illustrate these properties, let us use the example of Alice: a customer shopping at a grocery store. By simply being at the store, Alice has generated data about which store she shopped at and when. When she checks out, she generates data about what products she has bought and what payment method she used. When she leaves, she generates data about how long she has been in the store. All of this data may have value, and several actors may be collecting it. The store may be collecting this data through surveillance or loyalty programs. Her phone may have software collecting her location data. Her credit card company may be tracking her spending habits. In the existing data economy, these entities performing the collection have total sovereignty of these data attributes. They may analyze this data for targeted advertising, conduct market research, or sell it to other parties. In addition, companies collecting this data may take all of these actions with varying standards of privacy, security, or anonymity for Alice. Alice is not likely to have knowledge about what data attributes were collected, who collected it, who has access to it, or how the data is used. She is also not compensated for the value that is extracted from this data, even though she is the entity from whom originated it.

In the above example, the concerns around collection and privacy are immediately apparent, but this example also brings into question what it means to own data. In Alice's case, her data is being collected by third parties who may sell or exchange it with other parties. Due to the infinite duplicability of digital data, any such exchange results in both parties possessing the data. In this case, who actually owns the data? Is it the party that collected it? Is it collectively owned by all parties that are currently storing it? Or is it owned by the party that originated it (Alice)? In the existing model, data is owned by the entities that store it, and may be legally attributable to the entity that collected it (CITE). According to definition 4.1, Alice would have control of her data if she knew what aspects about her were being observed, had control over

who was able to access the information, and the infrastructure used to collect, store, and access the data did so safely.

4.2 Data as an Asset

In an information economy, data is the most fundamental commodity there is. To understand the individualized data economy, we must highlight the difference between assets and commodities. While commodities are an asset class, they have unique properties. Unlike most other asset classes, commodities are traded at high volume. While it is possible to purchase small volumes of commodities, they are often illiquid due to the markets operating at high volumes. While fractional vehicles like ETFs do exist, these are fundamentally not equivalent, as they represent a symbolic debt obligation and not the resource itself. For example, it is possible to buy an individual gallon of oil but much harder to sell it as the markets do not trade oil by the gallon. Conversely, it is much easier to sell a thousand barrels of the same oil since the commodities markets trade at these quantities regularly. In much the same way, individual data is not valuable in today's data economy and is almost exclusively handled in the aggregate.

There are cases in which individualized markets have developed around commodities. For example, in the wake of the 2008 financial crisis, consumers rushed to purchase gold as trust in financial assets, currencies, and markets plummeted. Nations and banks often trade in gold in large quantities, and as a result, gold markets trade-in units ranging from the thousands to the millions of ounces. To facilitate the liquidity of consumer gold, several gold purchasing operations have sprung up to provide liquidity to the individual and aggregate gold for commodities markets. Because of the non-triviality of data ownership (see below), this infrastructure has yet to be created for data. This is what the Snickerdoodle Protocol aims to enable.

4.2.1 Utility of Data

Large scale data analysis has provided great utility in the twenty-first century and has revolutionized every industry from supply chain to medicine, biotech, sports, and self-driving cars. Most of these applications require substantial computational resources, such as using machine learning to generate human faces and stories, to less complex tasks like finding the quickest route home. This also leads to undesirable outcomes such as tracking individuals without their knowledge or consent.

The utility of data comes from the ability to analyze it and and produce *insights*, see 3.2.3. Insights represent actionable intelligence derived from a data set. As an example, Google uses an individual's search history to learn a preference so that a relevant advertisement may be served. A self-driving car analyzes data from its sensors to learn that the traffic light turned red and it should stop.

4.2.2 An individual's role in the data economy

While the data economy is said by the World Economic Forum to be valued around $3 trillion at the time of writing, the primary lack of incentive alignment in the modern data economy is that individual generating data do not have control over how their data is being used and is not compensated for their role in the data economy. Different governments recognize the need to give users control, such as the GDPR and CCPA. Another problem is that the lack of ownership and control of data leads to ways for governments to get around surveillance laws. Instead of getting a warrant to view individual data, governments can buy the data from data brokers. This creates an opaque way for governments to spy on individuals. For example, the U.S. government's ICE agency created an extensive database through publicly available information and data brokers that allow them to track everyone in the U.S. without a warrant.

4.2.3 Properties of Data

It's worth highlighting that when data is treated as an asset, the value of the data asset can depend on the properties of that data. The *utility* of the insights that data can provide will dictate how valuable the data is. *Privacy* of the data is vital in creating value out of data. If data is not private, then there no property of scarcity. An asset without scarcity is worthless. Worse, the lack of privacy incentives enables mass surveillance. Similarly, *security* is deeply tied to the value of data as an asset. Data that is not secure can easily be stolen or can be rendered unusable. Data should be *interoperable*, so it can be used across many platforms. If a user's data is encoded via a proprietary format or cannot be moved outside of a particular silo, there will be less demand for the data set. Mechanisms to fascilitate the *authenticity* of data are also

required. If the data can easily be faked or modified, then the insights derived from the data set become uncertain.

4.3 Flow of Data and Value in the Data Economy

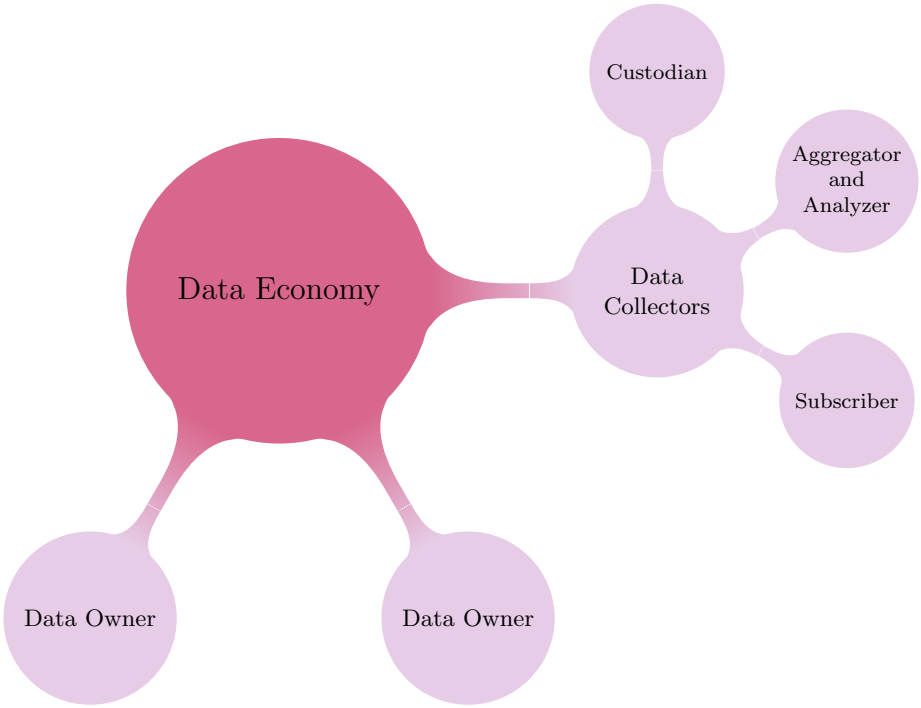

Figure 1: The actors in the data economy.

The data economy is a complex system that collects data on individuals, shares that data with other actors, and runs analysis on that data. This flow is necessary to extract value from data. Individuals generating the data sets do not necessarily have a good way to take advantage of their data, and those interested in the data are not the entities that generate it. This dynamic is at the heart of the data economy. One group generates the data, and the other wants to gain insight from that data. The rest of the actors exist to provide infrastructure to support that dynamic.

4.3.1 Actors

We define the generator of the data as the *owner* and the actor who is interested in the data as the *subscriber*. Being the entity that creates the data, the owner should have ownership rights and thus control how their data is used. The subscriber is interested in gaining temporary access to that data and running an *algorithm* or *functionaltransformation* on a large data set to gain actionable intelligence. It's worth highlighting that, in our definition, the subscriber is only interested in the utility of the information embedded in the data, not the data itself.

Other actors in the economy provide the infrastructure. *Collectors* are the actors who monitor and collect data on owners. *Custodians* store that data. Because the utility of data increases when combined with other data, the *aggregator* aggregates data from different custodians and makes it easy to run analysis, or algorithms to generate insights.

Subscribers pay for the utility, and that payment flows back down through the economy. Also, note that the same entity can play multiple roles. E.g. Google is a collector, custodian, aggregator, and subscriber for ads on web searches. Subscribers pay Google to run an algorithm that analyzes people and gives ads to those they think are interested in the subscriber's product. Data owners are the people using Google search. They are getting back a free online search and pay Google by seeing ads and allowing Google to control the data.

A self-sovereign data economy requires the data owner must own their data and control how it is used. The owner must have granular control over the the collection, storage, aggregation, and algorithms that run on their data. Additionally, payment must be distributed equitably to the parties in this economy such that incentives amongst all actors are aligned.

4.3.2 Actors in the Protocol

We will simplify the actors for the initial version of the Protocol. The users of the Protocol are data owners who will collect, store, and manage their data from a *data wallet* (for more on data wallet, see 5.3.1). The data wallet will allow people to provide auditable consent to aggregation and allow certain algorithms or functional transformations to run on their personal data set by minting a non-transferable consent token (to learn more about the on-chain consent and-off chain aggregation architecture, see sections 5.2 and 5.3 respectively). Lastly, consenting data wallets will send the resulting anonymized insights to an aggregation provider, which will allow businesses to see the insights they have paid for (see 5.3.3.3).

General implementation details are given in section 5 with areas for future improvement given in section 7.

5 Implementation

5.1 Architecture Abstract

This section will discuss the implementation and design decisions of the Protocol. The Protocol has three primary components: a data wallet implementation, an on-chain data control plane, and aggregation service providers.

The data wallet is a software client that implements functionality which enables end-users to collect, index, and store their data as well as participate in the decentralized data network, see section 5.3.1. Organizations (consumers of data insights) will be able to query populations of data wallets through an on-chain control plane that adheres to a publish-subscribe (pub-sub) pattern, see section 5.2.

At the core of the control plane is an upgradable contract factory which produces independent instances of an EIP-721 compatible consent registry. Consent tokens claimed from these consent contracts are non-transferable but can be burned by the recipient. Claiming a consent token denotes a data wallet user's consent to participate in network queries in return for rewards (which may or may not be web3 digital assets).

Data wallets receive queries via Ethereum Virtual Machine (EVM) events emitted from consent registry contracts they have claimed tokens in. These events contain metadata encoding instructions (written in Synamint Query Language) to run computations on the data stored in the data wallet to produce insights. Once a data wallet has produced the requested insight, it performs a digital "handshake" with the aggregation service provider specified in the query metadata such that the data wallet owner receives a reward while the requesting organization receives the anonymized insight, see figure 4.

User consent and data flow is thus orchestrated in a distributed manner by the Protocol. It is also worth highlighting that while Snickerdoodle Labs will develop service infrastructure for the Protocol (such as producing a data wallet client and an associated SaaS product offering for enterprise participation in the Protocol). However, the protocol itself is permissionless and open so that anyone could implement a data wallet client or act as an aggregation service provider if the specifications of the Protocol is adhered to.

5.2 On-Chain Components

5.2.1 Consent Contract Factory

The on-chain components of the Protocol function as a decentralized, permissionless, and trustless data control plane. It specifically implements a publish-subscribe pattern in which organizations publish new instances of an EIP-721 compatible consent registry (see 5.2.2), and end-users subscribe to the registries by claiming a non-transferrable consent token. The publishing action is performed via the Protocol's upgradable consent contract factory, see figure 2. The factory contract will be the entrypoint to the Protocol for new insight consumers, since a consent registry is required to communicate with the network of data wallets.

The consent contract factory exists as a utility for insight consumers to create new consent registries. The factory is implemented with an upgradable beacon pattern, see figure 3 to enable gas-efficient deployments and to allow for seamless extensions of functionality via DAO proposals (see sections 5.2.5 and 6.1) .

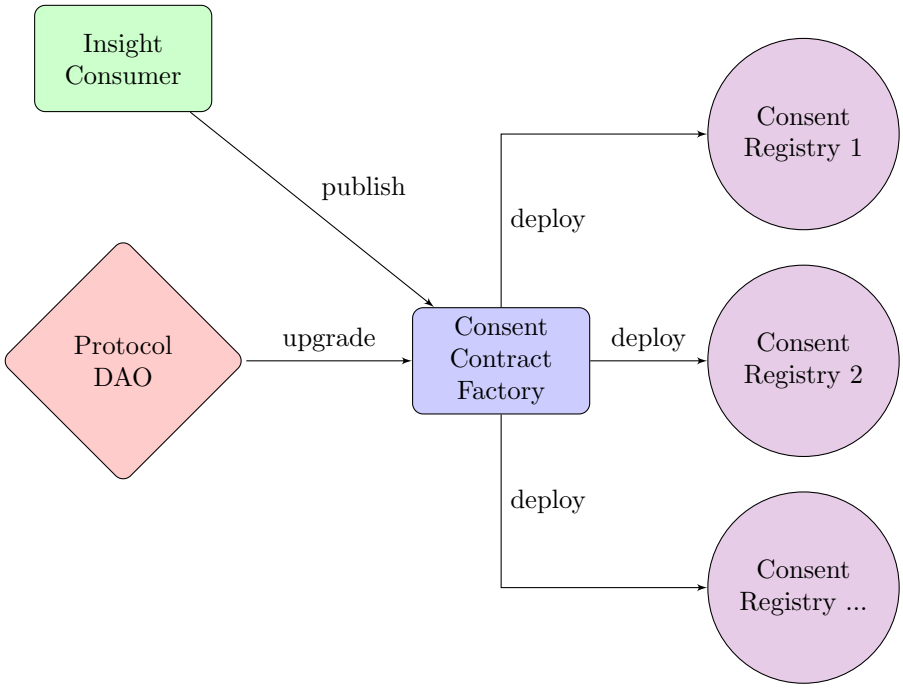

Figure 2: The core of the on-chain protocol implements an upgradeable, decentralized pub-sub network. Insight consumers publish consent contracts via the consent contract factory which is upgradable by the Protocol DAO. The EOA which publishes a consent contract receives the $DEFAULT\_ADMIN\_ROLE$ associated with that contract.

5.2.1.1 Factory Pattern The factory pattern defines a smart contract which is responsible for creating other contracts. The Protocol uses the factory pattern to simplify the deployment of new consent registries and to give new insight consumers a single point of entry into the data network.

5.2.1.2 Upgradable Beacon Pattern Consent registries are deployed as proxy contract instances that reference an upgradable beacon contract to obtain the correct address to delegate function calls, see figure 3. This upgrade pattern compliments the factory pattern by allowing for very gas-efficient deployments of new proxy instances. Proxy contracts only store storage variables and a pointer to their designated upgradable beacon contract. The upgradable beacon contract points to an implementation contract. The implementation contract contains all function implementations as well as the storage variable declarations that proxy contracts copy.

A Protocol upgrade to the consent registry functionality requires that a new implementation contract first be deployed to the blockchain. Then a DAO proposal must be initiated to point the upgradeable beacon to the new implementation address. All previously deployed and newly created proxy consent contracts inherit the functionality (and any new storage variables) defined in the new contract.

The upgradeability pattern is based on EIP-1967 and is implemented through OpenZeppelin libraries.

5.2.2 Consent Registries

Consent registry contracts are the primary on-chain mechanism by which insight consumers interact with with data wallet end-users. Consent registries allow organizations to create data pools by serving as an on-chain data structure that holds metadata regarding the conditions under which data is to be collected and used as well as a cryptographically verifiable list of externally owned accounts (EOAs) that have have given consent to participate in the data pool.

Consent registries expose an EIP-721 compatible interface. This is for developer integration convenience as it allows consent registries to be readable by most existing NFT (non-fungible token) indexing services (such as Snowtrace). Consent is denoted by ownership of a non-transferrable consent NFT which can be burned by the NFT owner at any time.

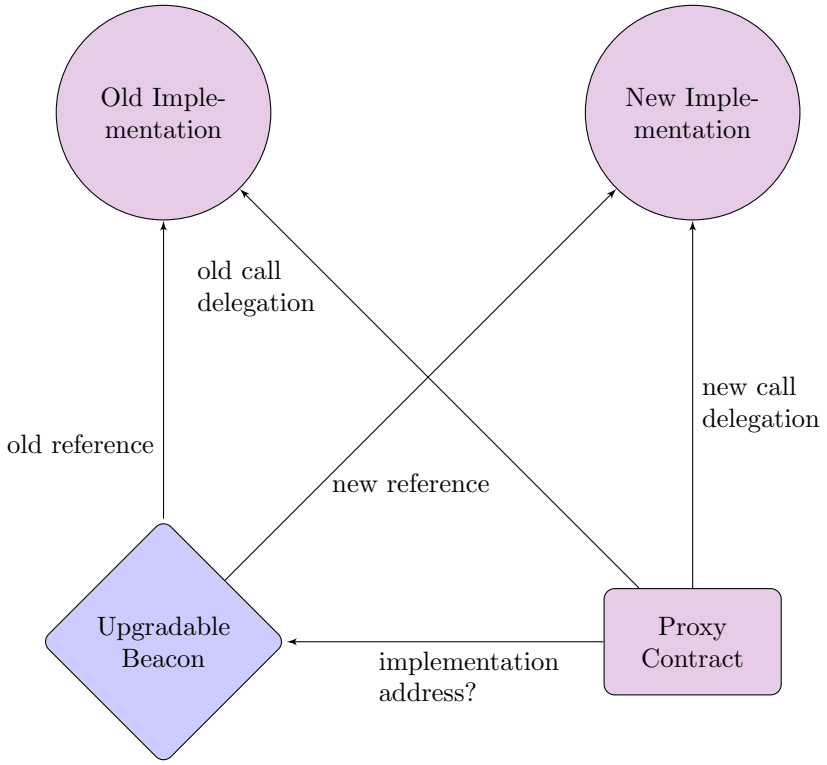

Figure 3: The upgradable beacon pattern was popularized by the Dharma protocol. It allows for many proxy contracts to be upgraded with a single transaction as well as for gas-efficient deployments of new proxy contracts. When a function call is directed at a proxy, the proxy retrieves the implementation address from the upgradeable beacon then delegates the function call to the contract at that address.

5.2.2.1 Request-for-Data Events After an organization has published a consent registry via the Consent Contract Factory (see 5.2.1), the organization can emit EVM events by calling a special function, *requestForData*, which takes a content identifier (CID) as its only input. This CID is used to retrieve the request specifications from a suitable content addressable network (like IPFS) that the data wallets will process (see section 5.3.1.1). Data wallets can detect past *requestForData* events by constructing EVM query filters and requesting all EVM logs that match those filters. Thus consent registries offer a tamper-resistant communication layer between organizations and participants in their data pools. See figure 4

Consent registries also specify a *queryHorizon* which inform data wallets of the oldest block number to search for these events. This variable is first initialized to the block number of the proxy contract deployment from the contract factory and can be updated by an EOA with the $DEFAULT\_ADMIN\_ROLE$ to a later blocknumber (but cannot be changed to an earlier block number). Setting a reasonable value for *queryHorizon* is important since many RPC providers only allow for query filters to search a limited history of the blockchain.

5.2.2.2 User Data Permissioning A *requestForData* event can indicate that it requires access to multiple attributes indexed by the user's data wallet client, such as country of origin, age, on-chain contracts they have interacted with using their linked EOA asset account. However, users can set granular permissions regarding their indexed data attributes.

Every consent token issued from a consent registry has an associated set of binary *agreementFlags*. There are 256 flags in total (the size of an EVM word) though not all flags will be assigned to specific data attributes at Protocol launch, leaving room for customization. Only the owner of a consent token can update the granular permissions denoted by the token's *agreementFlags*. The availability of granular consent data on-chain allows organizations to better understand what kinds of insights they will be able to obtain from their data pool before calling *requestForData*.

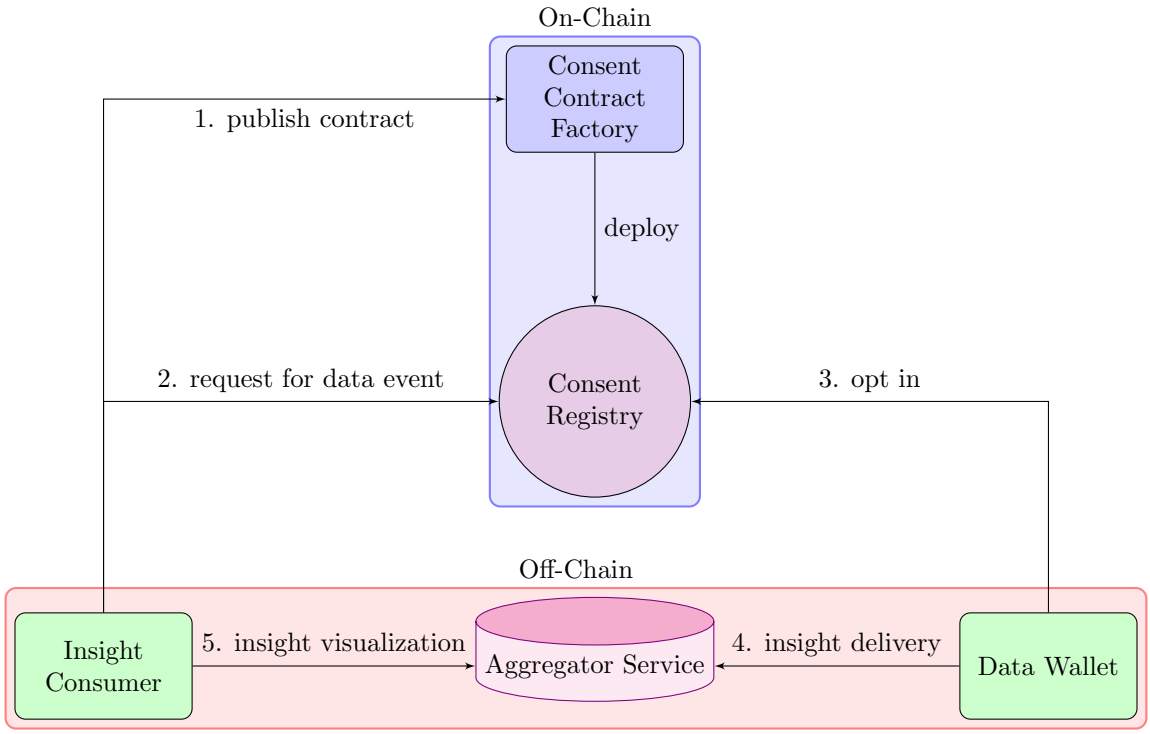

Figure 4: Insight consumers reach data wallet users by broadcasting events from a consent registry instance. Only data wallets that have claimed a non-transferrable consent token listen for events and deliver insights to the aggregation service URL listed in the associated request-for-data event.

5.2.2.3 Consent Invitations Consent registry metadata is used for the decentralized, permissionless, and trustless triggering of user flows that should be presented to the data wallet end user in a format appropriate for the data wallet client environment. In a web browser setting, the data wallet detects the current active URL, and via DNS over HTTPS (DoH) queries the TXT records associated with the apex domain. If the TXT record contains a reference to a Protocol consent contract address, the data wallet then fetches the URLs registered in the consent registry *domains* metadata storage variable and cross-references the domains listed from the contract to the current URL. If the data wallet detects that the current URL is included in the domains listed in the consent contract, the data wallet will inject a user flow into the browser DOM. The content of the user-flow is fetched from the URL specified by the consent registry *baseURI* parameter. The Web3 popup protocol can be extended to other environments including mobile browsing environments, VR experiences, gaming consoles, etc as indicated by figure 5. The user flows presented by these tamper-resistant popups serve as the on-boarding mechanism for end-user's to opt into a data pool.

5.2.2.4 Opt-In methods and Meta-Transactions The consent registries have two modes by which users can join a data pool: open-access and invite-only. Consent registries in which open-access is enabled (*openOptInDisabled* is $false$), any EOA is allowed to claim a consent token by calling the *optIn* method and paying the associated gas fees.

Registries where *openOptInDisabled* is *true* are invitation only and require a signature for an EOA with the $SIGNER\_ROLE$ in order to join the data pool associated with a consent registry. There are two available methods for invitation-only user opt-in: *restrictedOptIn* and *anonymousRestrictedOptIn*.

The former method requires that the recipient EOA be known in advance by the $SIGNER\_ROLE$ in order to construct the appropriate signature. The receiving EOA then becomes the only account that can call *restrictedOptIn* with that signature. If the recipient EOA address is not known ahead of time, the $SIGNER\_ROLE$ can construct a signature for use with the *anonymousRestrictedOptIn* method in which any EOA can submit the signature to that method in order to opt-in. Once a opt-in signature is used, it cannot be used a second time.

Support for both open and invitation-only opt-in flows makes the Protocol flexible to a variety of use-cases. However, end users are often hesitant to spent their own cryptocurrency in order to pay for transaction fees

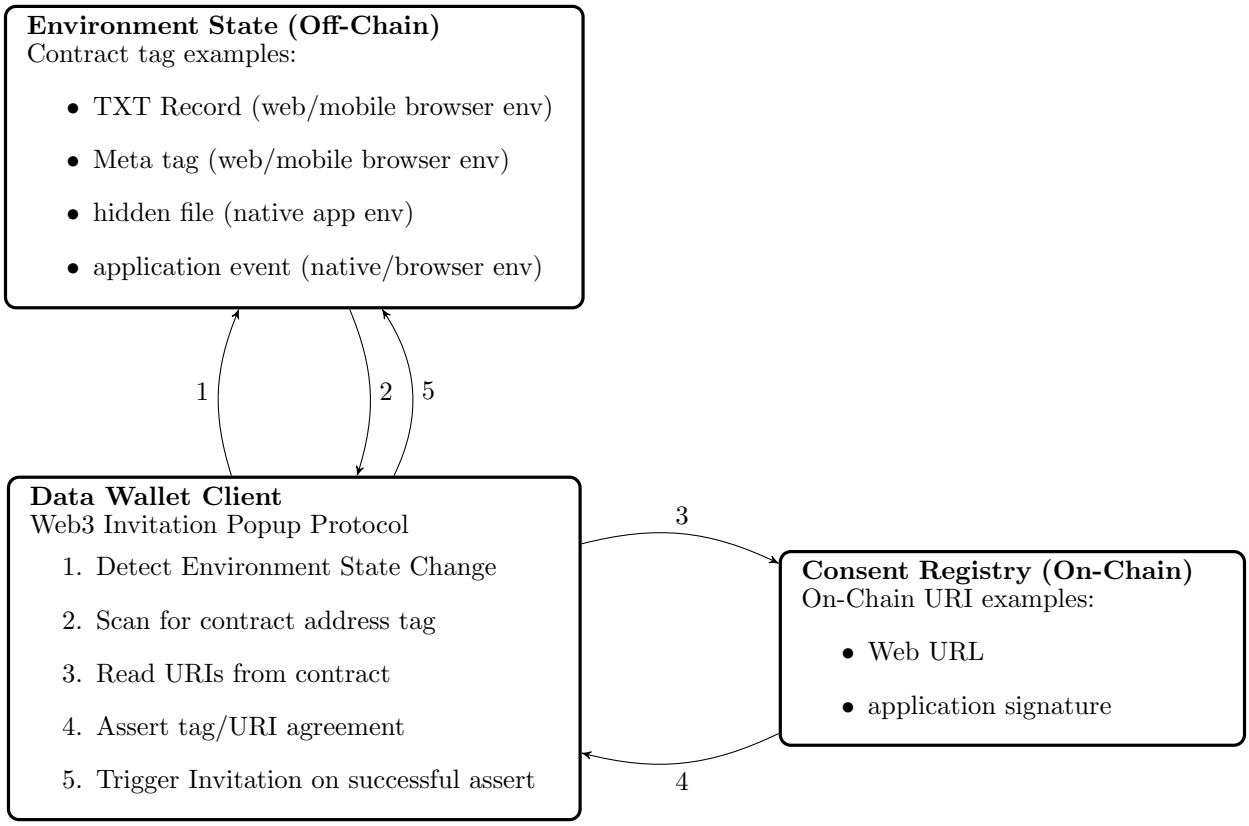

Figure 5: Actor model of the tamper-resistant web3 invitation popup protocol.

associated with a decentralized application or simply do not have the tokens necessary to do so. Additionally, requiring the user to spend their own assets to participate in the Protocol intoduces significant user friction and adoption hurdles. Consent registries implement EIP-2772 compatible metatransaction capabilities to circumvent this issue.

Metatransactions enable the delegation of a user's transaction gas fees to another party in a manner that ensures that the user's transaction cannot be maliciously altered. All opt-in methods implement support for EIP-2772 compatible metatransactions. This will offer the flexibility to have users pay for their own transaction gas fees or have the insight consumer pay.

5.2.3 Identity Crumbs

The Protocol introduces a special EIP-721 compatible registry, called the Crumbs Contract, to facilitate data wallet synchronization when an end user installs the client on a new device. When a user links a new EOA to their data wallet identity, the EOA encrypts their data wallet identity EOA (see section 5.3.1.2) and stores the encrypted data in the token URI of an entry in the Crumbs contract.

During a new data wallet client installation, the end user must simply link any EOA that has previously been linked to their data wallet identity in order for the data wallet to synchronize from their previously saved state that has been stored in the decentralized persistence layer of the data wallet network. The data wallet checks if the account being linked owns a token in the Crumbs contract; if so, it reads the encrypted content of the token URI, decrypts the information, and loads the public-private key pair into memory.

5.2.4 EIP-20 Token

The Protocol includes a fungible utility token adhering to the EIP-20 standard. This token will be used for paying various fees required to leverage the Protocol (like publishing a new consent registry) and will also be used for voting in the Protocol DAO (see Tokenomics in section 6). The associated voting mechanism that accompanies the possession of a token can be delegated to a different address without relinquishing ownership of the token

5.2.5 Decentralized Autonomous Organization

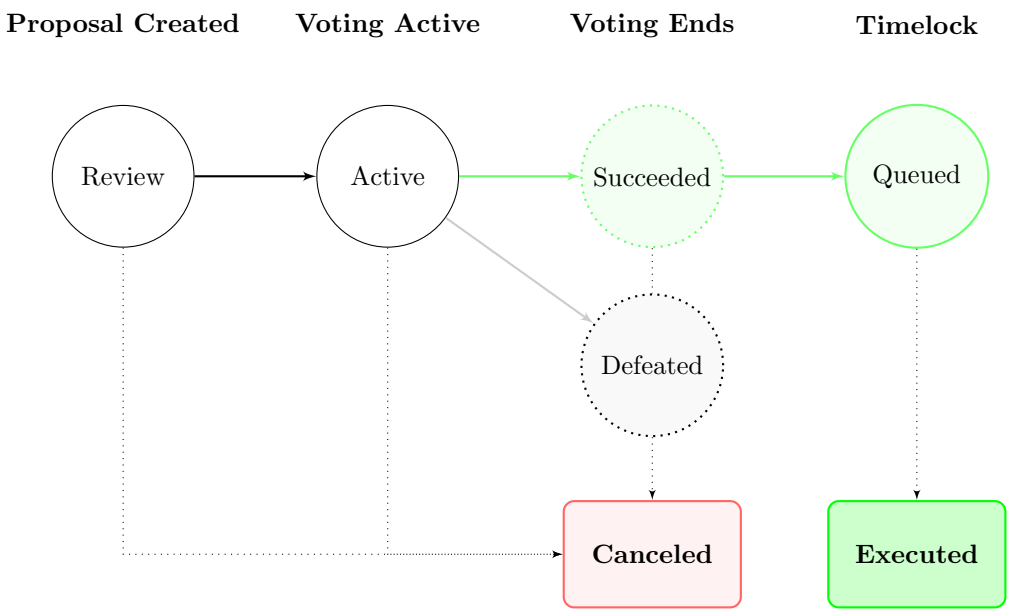

Figure 6: State machine diagram of the DAO proposal life-cycle.

The Protocol will include a decentralized autonomous organization (DAO) implementation as part of its on-chain components. The DAO will be responsible for proposing and executing upgrades to the Protocol. The particular pattern used by the Protocol DAO is based on Curve Finance's DAO and will be implemented with OpenZeppelin libraries.

Token holders (see section 5.2.4), are responsible for the creation and execution of DAO proposals. At mainnet launch it is anticipated that one token will render one vote, though this too could be modified via a DAO proposal. Token holders will have to reach a pre-specified quorum of voting power in order to successfully create a proposal in the DAO task queue.

Proposals will be subject to a delay of at least one block before voting begins as well as before a queued proposed can be executed in order to prevent flash loan attacks. Voters who initiate a proposal and subsequently relinquish their voting power by either selling their tokens or have their voting power revoked may have their proposal canceled if their remaining voting power is below the quorum threshold. The lifecycle of a DAO proposal is outlined in figure 6.

5.3 Off-Chain Components

5.3.1 Data Wallet

The data wallet is the primary client interface for end users to interact with the Protocol. It enables data ownership by facilitating user control and consent to the collection, storage, and usage of their data. The data wallet should provide the following functionality (as indicated by figure 7):

- Ingestion and indexing of user data from the data wallet client deployment environment
- Identity and verifiable credential generation/management
- Query/Reward discovery and management
- A query engine for individualized data mining, insight processing and delivery
- A consent management interface and granular access control
- Secure storage of the data

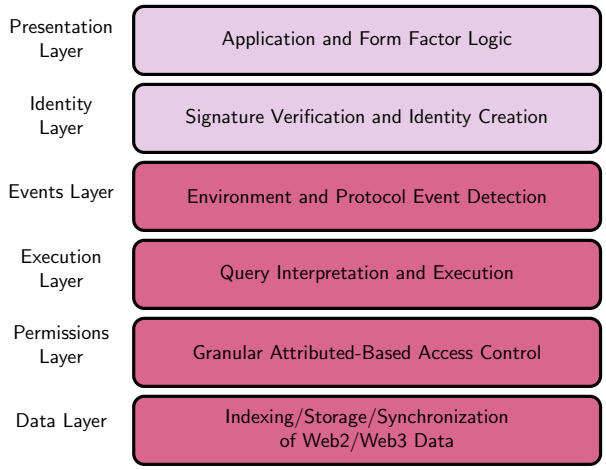

Figure 7: Logical structure of a data wallet client.

To the user, the data wallet operates in a conceptually similar way to a conventional cryptocurrency wallet, but with a wider scope. Instead of key and account management of a blockchain account, the primary purpose of a data wallet is to manage the storage, collection, and sharing of insights derived from user data. Data wallet functionality is form-factor agnostic (see section 7.1.5). However, browser extension and mobile applications are anticipated to be the primary channels for use.

5.3.1.1 Insight Acquisition and Control Flow The acquisition of insights from the data network begins at the consent contract factory (see section 5.2.1) where an organization must first publish a consent registry. Data wallets belonging to end users who have claimed a consent token via an invitation flow (section 5.2.2.3) detect *requestForData* events from the associated consent registry. The event will reveal a CID which resolves to query definition file (section 5.2.2.1). The query execution layer of the data wallet will parse the query definition, construct the associated abstract syntax tree (AST), and apply the logic to the data wallet persistence layer in a manner consistent with the conditions given by the user's on-chain permission settings (section 5.2.2.2). This control flow is outlined in figure 8.

5.3.1.2 User Identity Generation via Key Ratchets A data wallet should allow users to index transaction history and asset owership from multiple EOAs or smart wallets. However, if they were to use these addresses directly in the participation of the Protocol for consent token ownership it would readily allow for chain analysis of user behavior and compromise user data privacy. Therefor, a data wallet implementation should provide key ratchet utilities to allow for the deterministic generation of new EOAs (that cannot be linked back to the generating EOAs) which have the dedicated purpose of holding consent tokens.

A ratchet is a simple machine that only allows unidirectional state increments and prevents backward traversal of the state path. Likewise, a cryptographic key ratchet is an algorithm that allows for the deterministic generation of new public-private key pairs, using a prior key pair as inputs, in a manner that precludes the feasibility of determining what public-private key pairs were used in previous iterations. Cryptographic ratchet algorithms are widely used today in consumer-facing private messaging applications.

Algorithm 1: Key Ratchet Proto-algorithm

Require: EOA with message signing utility, Seed Message
Ensure: New EOA with no prior transaction history
  Seed Message Signature ← EOA.signMessage(Seed Message)
  new EOA ← pbkdf2sync(Seed Message Signature, EOA public address, 100000, 32, sha256)
  return new EOA

Procedure 1 outlines a simple technique, leveraging the Password-Based Key Derivation Function 2 (pbkdf2sync), to generate a new public-private key pair from the message signing utility exposed from most consumer crypto-wallets. This algorithm, used in conjunction with the Crumbs contract, described in section

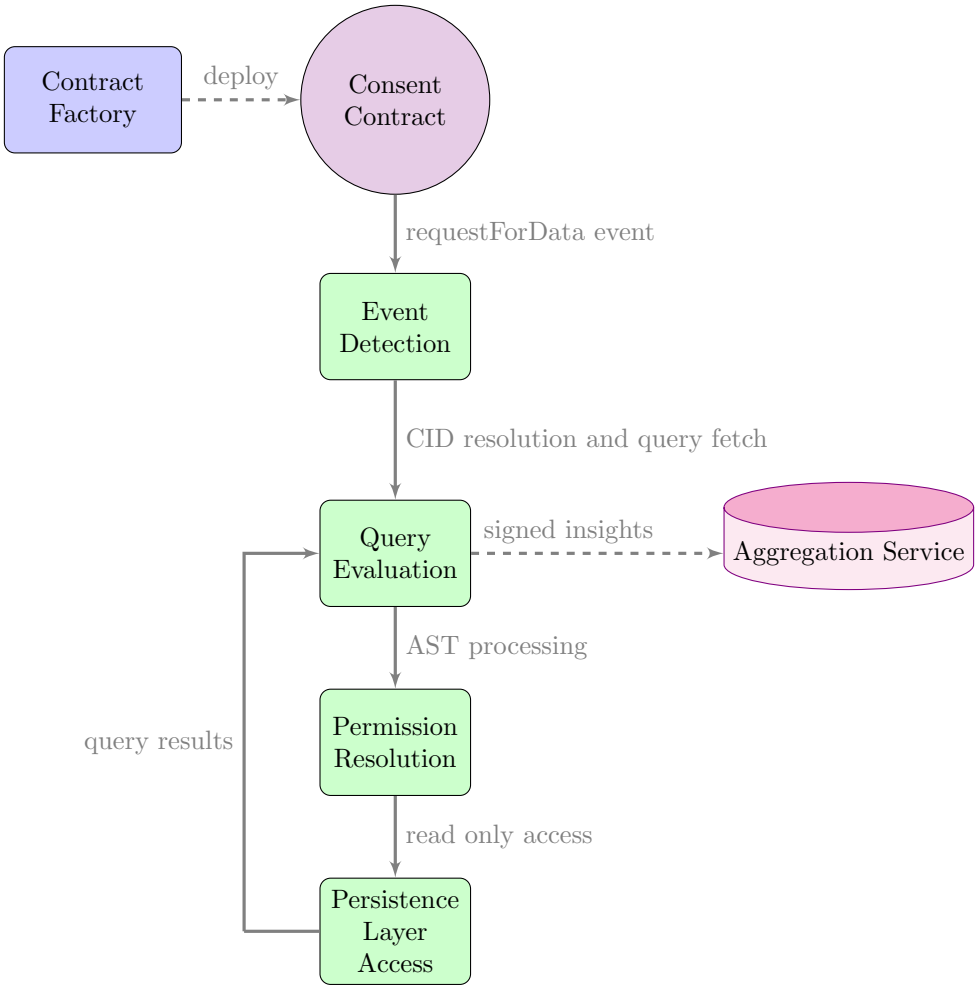

Figure 8: Control Flow from requesting to receiving an insight from the data wallet network.

5.2.3, can be used to enhance user data privacy by preventing the cross-referencing of linked EOAs on-chain while at the same time offering an improved user experience.

Specifically, by using the key ratchet algorithm to derive a dedicated in-memory key pair for consent, the data wallet form factor can sign metatransactions (see section 5.2.2.4) without multiple prompts from various wallet applications. Additionally, using derived EOAs for consent provides an additional layer of security for end user's, keeping their valuable asset-holding EOAs separate from those used for participating in various data pools.

5.3.1.3 Storage Secure storage of data is crucial to allowing end users to own their data. The Protocol does not specify a schema for local storage of user data; that is left to the party implementing a data wallet client. However, the storage layer of a data wallet client should allow for secure, tamper-resistant, and platform independent synchronization of user data across multiple client installations.

The initial data wallet implementation produced by Snickerdoodle Labs exposes a modular storage interface capable of integrating with various object storage provider technologies, such as Google Storage, Amazon S3, or decentralized options like the Ceramic Network.

It is also important to call out the wallet's storage of public-private key pairs. A data wallet should only store public keys and digital signatures associated with accounts linked to a user's data wallet, not private keys (other than the in-memory keys generated via key ratchet iterations used for holding consent tokens, see section 5.3.1.2). Instead, data wallet client implementations should delegate the management of asset-bearing keys to dedicated wallet applications like MetaMask, Coinbase Wallet, etc.

5.3.1.4 User Data Ingestion and Indexing

A data wallet client should implement automatic data collection utilities appropriate for the deployment environment of the client. For example, a browser extension client may collect metrics on sites visited and time spent on those pages. A mobile application client may collect geo-location information. Regardless of the attributes a data wallet client is collecting in a particular form-factor, attributes can be categorized via three important properties: explicit/implicit, first/third party, and authenticated/unauthenticated.

5.3.1.5 Explicit/Implicit

The data attributes collected by a data wallet client can be considered explicit or implicit attributes. Explicit data is data must be indexed and stored in its entirety by the data wallet client in order to produce insights from it. Explicit data offers no deterministic mechanism to regenerate the data set from scratch if it is lost. Implicit data does not require storage of every data element of the attribute in order to evaluate functions to produce insights and can be deterministically recovered if lost.

A simple example of an explicit data attribute would be a user's geo-location history. This data must be stored by the data wallet in its entirety to generate insights and if it is lost it cannot be easily reproduced without an dedicated backup. An example of implicit data is a user's on-chain transaction history. Transaction history can be recovered deterministically by simply linking asset accounts to the user's data wallet. Additionally, the entirety of the user's transaction history does not need to be available at all times, it can be fetched as needed when an insight requires accessing it.

5.3.1.6 First/Third Party

Data collected can come from different sources. Specifically, first-party data comes directly from the user, and third-party data comes from someone other than the user. For example, if the user directly inputs their name, their name would be considered first-party data; if the user imports their name from the DMV, that would be third-party data.

5.3.1.7 Authenticated/Unauthenticated

Data can be authenticated if its origin and validity can be verified through cryptographic means and unauthenticated if no such mechanism exists. For example, a wallet address can be authenticated via a signed message signature verification. Third-party data can be authenticated if it has a known credential authority (this is the fundamental operating principal of certificate authorities like Digicert).

5.3.1.8 Localized Processing

The data wallet is a local application that stores the owner's data securely and processes computations locally. By collecting and securely storing user data locally, the data wallet guarantees data ownership to the user by never sharing it. Because computations are running locally, the owner ensures that only analysis they've given consent to can run on their data. Insight consumers also benefit from this model as they can leverage data-driven insights without the risk of liability associated with the custody of user personal identifying information (PII). While the initial version of localized processing will be more limited, there is a myriad of ways we can modify this approach to add additional data safety and features (see section 7).

The wallet will learn what computations to run by listening to on-chain $requestForData$ events as depicted in figure 8. These queries are written in a simple language specified by the Protocol. This language, called Synamint Query Language, is discussed in section 5.3.2

5.3.2 Synamint Query Language (SyQL)

The Protocol specifies a simple query language, called Synamint Query Language (SyQL), which will allow insight consumers to broadcast conditional insight requests to consent registry cohorts in a transparent and interpretable fashion. SyQL is structured in JSON format containing information on the eligibility requirements, rewards, data to be collected, what processing to perform, and where to send processed data (as depicted in the insight delivery edge in figure 8). Queries written in SyQL should be stored on a content-addressable network, such as IPFS, to ensure tamper resistance.

A SyQL is written by specifying nested keywords with associated parameters that inform the data wallet client query processing engine what data attributes are being requested and what insights to return given conditional statement that are met by the user's data state. The keywords associated with SyQL and their intended usage and behavior are given in the following subsections (note that the SyQL specification is subject to change before mainnet launch of the Protocol):

5.3.2.1 version (required) The *version* keyword is reserved for specifying the version of the SyQL schema a query is based on. This keyword has no sub-keywords.

5.3.2.2 timestamp (required) The time when the SyQL query is created in ISO 8601 format, i.e., YYYY-MM-DDTHH:MM:SS. For an example, 20:20:39 on 13 of November 2021 is represented as 2021-11-13T20:20:39. This keyword has no sub-keywords.

5.3.2.3 expiry (required) The time when the SyQL query is expired in ISO 8601 format. Queries that are recieved after this time are considered stale and will not be executed or rewarded. There are no sub-keywords associated with this top-level keyword.

5.3.2.4 description (required) The *description* keyword is used for specifying text, markdown, or HTML intended to be displayed to the recipient of a query. There are no sub-keywords.

5.3.2.5 business (required) This keyword is reserved for indicating what entity is broadcasting a query. It has no sub-keywords.

5.3.2.6 queries (required) The *queries* keyword is used to indicate that a SyQL file is requesting access to the data wallet persistence layer. One or more instances must be specified with a queries block. These query instances can then be referenced by other top-level keywords. A query instances has the following sub-keywords:

- *name*: The *name* sub-keyword indicates which attribute must be accessed in the data wallet persistence layer. Supported attributes should include:
  - network: accesses the Web3 data associated with all accounts linked to a data wallet identity
  - age: access to the age of the data wallet user
  - location: access to the location data of the data wallet user
  - browsing_history: access to the browsing history of the data wallet user
  - gender: access to the gender field of the data wallet user
  - url_visited_count: access the number of times URLs are visited by the data wallet user
  - chain_transactions: accesses the transaction volume (in USD) and count by the data wallet user
  - balance: accesses the balance of the data wallet user on a per-chain basis
- *return*: The *return* sub-keyword specifies the object type that will be returned by a query. Supported types include:
  - boolean: true or false depending on the conditions applied to the attribute being accessed
  - integer: returns an integer object related to the referenced attribute
  - enum: returns an enum related to the referenced attributed. The enum keys are specified under enum_keys sub-keyword
  - object: returns an object to describe the referenced attributed. The object schema is specified in object_schema sub-keyword
  - array: returns an array to describe the referenced attributed. The array items are specified in array_items sub-keyword
  - string: returns a string to describe the attribute of interest. The string patter is described using string_pattern sub-keyword.
- *conditions*: Conditions are used in conjunction with the boolean return type. A conditions are used to specify the filter to apply to the attribute in order to determine if true or false should be returned. The following conditions are supported:
  - in: is the attribute in a set of objects

  - ge: is the attribute greater or equal than a given object
  - l: is the attribute less than an object
  - le: is the attribute less than or equal to an object
  - e: is the attribute equal to an object
  - g: is the attribute greater than an object
  - has: does the attribute include a set of objects

- *networkid*: This sub-keyword is used in conjunction with the balance attribute type. This sub-keyword allows for the specification of which layer 1 protocols a balance query should be run against. The following networkid are supported:
  - SOL: Solana network
  - 1: Ethereum Mainnet
  - 4: Ethereum Testnet (Rinkeby)
  - 42: Ethereum Testnet (Kovan)
  - 43114: Avalanche Mainnet
  - 43113: Avalanch Testnet (Fuji)
  - 137: Polygon Mainnet
  - 80001: Polygon Testnet (Mumbai)
  - *: all supported networks

- *chain*: This sub-keyword is used in conjunction with the network attribute type. This sub-keyword allows for the specification of which layer 1 protocols a network query should be run against. The following chains are supported:
  - ETH: Ethereum Mainnet
  - AVAX: Avalanche Mainnet

- *contract*: The *contract* sub-keyword is used in conjunction with the network sub-keyword. Specifying a contract indicates that the query is interrogating whether any accounts linked to a data wallet have made transactions meeting the following required characteristics:
  - address: address of the smart contract of interest
  - networkid: chain ID that the smart contract is deployed to
  - function: function ABI on the target smart contract
  - direction: was the user's account in the to or from field
  - token: is the contract an ERC20 or ERC721 standard
  - timestamp: did the account submit a matching transaction between start and end timestamp

- *enum_keys*: This sub-keyword is used in conjunction with the enum attribute type. Listing the keys that the attribute type supports.

- *object_schema*: This sub-keyword is used in conjunction with the object attribute type. Specifying the schema of the object including the properties, patternProperties (properties with regex formatted keys), and required properties of the object.

- *string_pattern*: This is used to describe the pattern of the string attribute, using Regular Expression (RegEx).

- *array_items*: This sub-keyword is used in conjunction with the array attribute type. Specifying the items of the array. The following array_items are supported:
  - boolean: an array of booleans
  - integer: an array of integers
  - object: an array of objects described with object_schema
  - array: an array of arrays
  - number: an array of numbers

5.3.2.7 returns (required) The *returns* keyword is used to specify one or more candidate return objects that may be delivered to an insight aggregator. A return object has the following sub-keywords:

- *name*: What is the type of return
  - callback: resolves immediately to a pre-specified message delivered to a callback url
  - query_response: resolves to the result of the specified query
- *message*: An explicit string message to be returned as a result. Used with the callback return type.
- *query*: A reference to a query specified in the queries block. Used in conjunction with the query_response return type.
- *url*: A complete URL specifying the location of the aggregation service associated with this SyQL file.

5.3.2.8 compensations (required) The *compensations* keyword is used to declare one or more possible digital assets associated with the SDQL file. Below are the following required characteristics of the compensations:

- *description*: A text, markdown, or html string for displaying to the user information about the digital asset.
- *callback*: A callback URL for claiming the digital asset.

5.3.2.9 logic (required) The *logic* keyword is used to specify arbitrary logic to components specified in the queries, returns, and compensations blocks.

- *returns*: A sub-keyword of logic used to specify an array of return expressions. A return expression can return objects declared in the returns block given that objects declared in queries have sufficient permissions to access the requisite attributes of the persistence layer.
- *compensations*: A sub-keyword of logic used to specify an array of compensation expressions. A compensation expression can return objects declared in the compensations block given that objects declared in queries have sufficient permissions to access the requisite attributes of the persistence layer.

5.3.3 Aggregation Services

Data ingestion and aggregation is the final stage of the insight aggregation process that allows subscribers to access the insights generated at the data wallet layer. This section will detail how the subscriber can see processed data from the user's data wallet. Any entity can assume the role of data ingestion service as long as they follow the protocol and are accepted by the DAO. Snickerdoodle Labs will offer the first data ingestion provider, which will be tied to a SaaS product.

5.3.3.1 Insight Ingestion The final step of the protocol is for data wallets to send insights from their queries to an endpoint. This endpoint is the ingestion provider and is specified in the SyQL query. Ingestion is an integral part of the Protocol as it allows insight consumers to visualize the insights they have paid for.

Once the ingestion provider receives the insight, it must store the insights events and manage access. They can also provide any additional services they want on top of this, such as providing customizable dashboards for advanced visualization. There is no way to mathematically guarantee any ingestion provider is behaving honestly and correctly. The lack of a strong guarantee means that actors in the Protocol have to put some degree of trust in ingestion providers. The Synamint Protocol reduces this problem by only allowing ingestion services that the DAO has approved. Any aggregation provider has to be trustworthy enough for the DAO to vote them in; if a provider is revealed to be a bad or incompetent actor, they can be voted out. For a deeper discussion of data safety concerns with ingestion providers, see section 5.3.3.2.

5.3.3.2 Data Safety

The initial version of the protocol requires trust in the Insight Platform to maintain data safety. As discussed above, there are no guarantees that the ingestion provider acts honestly. The ingestion provider can see, delete, and sell any insights they receive. Additionally, because the initial version of the wallet only implements simple anonymization techniques, ingestion providers could identify who shared the insight and reconstruct the raw data.

To reduce these privacy risks, the initial version of the protocol will only allow queries that don't reveal PII, and future versions will add anonymization techniques to insight computations (see section 7). To reduce the risk of malicious ingestion services, the Snickerdoodle DAO will maintain an allowlist of trusted actors. Additionally, Snickerdoodle Labs will be providing an ingestion service. The Snickerdoodle Ingisht Service is a SaaS product that will enforce data safety and has a massive financial and legal incentive to behave honestly.

5.3.3.3 SDL Insight Platform

The Snickerdoodle Insight Platform will be our SaaS product offering for interaction with the Snickerdoodle Protocol. The Insight Platform will provide an ingestion service and help manage interaction with the rest of the protocol. This includes support for creating and managing queries, consent contracts, rewards, and website integrations. The Insight Platform will also provide an analytics dashboard to help businesses see their insights.

6 Tokenomics

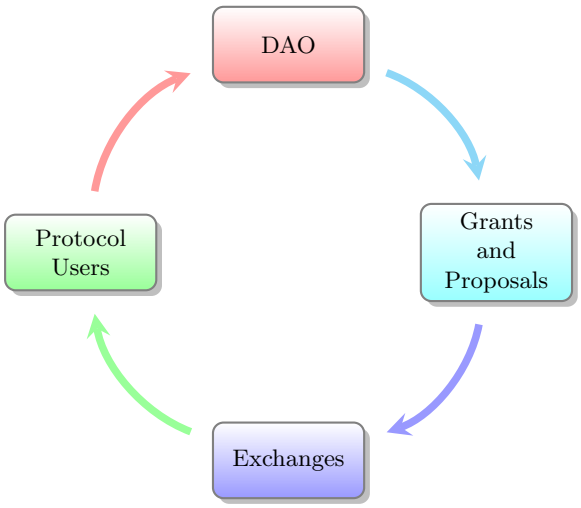

Figure 9: Lifecycle of the Protocol token.

The Protocol will issue a fungible token on mainnet launch, the technical aspects of which are given in section 5.2.4. The token will serve both network governance and protocol utility purposes outlined in the following subsections.

6.1 Governance

A fundamental use case the Protocol token will be network governance. At mainnet launch, it is anticipated that one token will be eligible to cast one vote in active DAO proposals. Token voting weight could be upgraded via future DAO proposals. For implementational details about the logical structure of the Protocol DAO, see section 5.2.5.

Protocol contract upgrades will be executable only via DAO proposals. The upgrade pattern used by the Protocol is the upgradeable beacon pattern, see section 5.2.1.2. DAO proposals would be capable up updating the implementation address of Protocol beacon contracts in order to augment or introduction new on-chain functionality.

Lastly, it is anticipated that the DAO will function as a decentralized treasury, accumulating tokens spent during the utilization of the Protocol. DAO proposals can be created to allocate these funds in a manner appropriate for the furthering of the Protocol as depicted in figure 9.

6.2 Utility Functions

It is anticipated that the Protocol token will have several utility use cases for initiating certain activities within the on-chain mechanics of the Protocol (see section 5.2 for details of on-chain components). It is desired by the Protocol designers that the token accrue value as the size of the data set represented by all data wallet users grows. The following mechanisms are likely to be implemented as on-chain logic in order to facilitate this goal:

6.2.0.1 Factory Fees The consent contract factory (section 5.2.1) will possibly require that token be spent in order to deploy a new consent registry. It is thought that requiring a token fee to publish a consent registry will disincentivize fragmentation of the data network, since it will be more expense to create multiple data pools than a single data pool. The token spent would be routed to the DAO treasury address. The fee amount would be set by DAO proposals.

6.2.0.2 Request-for-Data Fees In order to disincentivize frivolous *requestForData* events, it is anticipated that there will be a token fee required to emit these events. The fees could be a fixed cost or proportional to the number of consent tokens issued in the consent registry at the time of the function call. The token spent would be routed to the DAO treasury address. The fee amount and structure would be set by DAO proposals.

6.2.0.3 Stake for Ranking In the scenario where there are many consent registries published by many different organizations for varying purposes, it will be necessary to have a built-in search mechanism to enable data wallet users to find data pools that fit their interests. The Protocol may introduce a staking registry where consent registry publishers stake token in order to boost their indexed ranking in the list. This could be further granularized by introducing different verticals that stake could be allocated towards (some examples could be gaming, trading, consumer products, etc.). Data wallet implementations could leverage this staked ranking registry in order to present appropriate data pools to users based on the content of the data stored in their wallets.

Organizations could reclaim their staked token when a boosted ranking is no longer relevant for their consent registry, with some fraction of the token held back for the DAO treasury. Staking organizations that abuse the ranking system would be at risk of having their stake forfeited to the DAO treasury in its entirety and the ranking eliminated by a DAO proposal.

6.3 Token Distribution

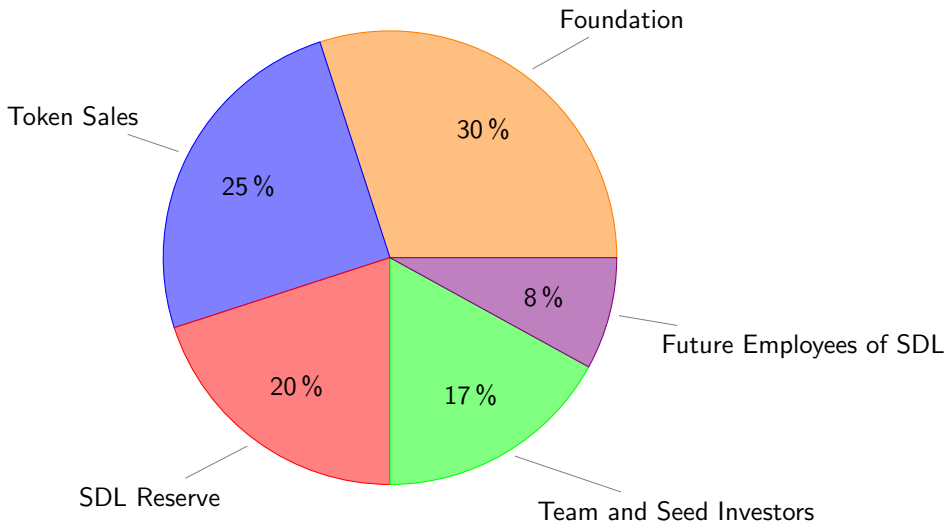

Figure 10: Protocol token distribution breakdown. The total supply is capped at 13.5 billion.

The Protocol token (section 5.2.4) is capped to a total of 13.5 billion. The allocation of the token supply is given in figure 10. The Protocol Foundation will be allocated the largest percentage of the supply at the time of token generation with Snickerdoodle Labs (SDL) receiving 20 percent of the supply.

7 Future Considerations

The First version of the Snickerdoodle Protocol won't be able to achieve everything we aim to achieve. This section will detail potential improvements we can make to the protocol and potential risks the protocol faces.

7.1 Potential Improvements

This section will deal with potential improvements we can make to the protocol. Many of the changes are related but each change can be made independently.

7.1.1 Data Sharing

Currently, the Protocol enables a basic version of data sharing, where only a simple set of predefined functions can be run on an individual's data and shared with one subscriber. An easy way to improve the protocol would be to enable more possible functions to run on this compute-to-data model. The currently allowed functions can only be run on a single individual's data. We could extend the protocol to run functions on multiple people's data before returning, thus enabling Multi-Party-Compute. Data is also only allowed to flow from data wallets to ingestion providers. Instead, we could create a back and forth between the data wallet and the subscriber. Allowing this back and forth would enable us to create a system to support federated learning. The subscriber would send over a pre-trained model, the wallet would update the model and send it back, and then the subscriber could combine the updates from every wallet and repeat the process. Additionally, we can make changes to improve the privacy of the Protocol. We could have the wallet use different anonymization techniques to prevent the wallet from sending PII to the insight service. Similarly, the wallet could expand its use of perturbation techniques and differential privacy. We can also add changes to how the protocol gets queries from the content addressing network to hide what queries subscribers are running. For example, we could create private query cohorts, so only a subset of the network knows what questions a business is interested in.

7.1.2 Atomic Reward Swaps

The initial version of the Protocol involves a trust-centric rewards-sharing mechanism. Ideally, there would not be any trust involved. The data owner only shares their insight once they know they will get a reward. The insight consumer will only send the rewards once they know they will receive a valid insight. One way to solve the transfer problem is with Zero-Knowledge atomic swaps. Only when both sides can guarantee that a valid trade will happen will insights and rewards be exchanged. Atomic swaps should be amenable to cross-chain interactions. For a discussion on ensuring that insights are valid, see section 7.1.4.

7.1.3 Cryptography

The ability to update the cryptography is built into the Protocol because it is essential to patch security holes. This same idea allows us to add new cryptography that can enable new ways for the system to provide data safety guarantees.

This wouldn't be a web3 white paper if we didn't mention how ZK-proofs could improve our system (lol this can be removed). ZK-proofs could allow data wallets to get consent tokens or rewards without others in the network knowing. Introducing new signing schemas will enable us to create proper data fiduciaries who can safely allow individuals to delegate their consent to others. Some interesting cryptography may enable data to be leased by only allowing decryption for a set amount of time. Additionally, we can use different cryptographic ideas to increase data tracking of data custodians (e.g., proof of storage and proof of deletion).

7.1.4 Data Outsourcing

Another avenue we want to look at for increasing the Protocol's usefulness is increasing the amount of data that the Protocol can collect. We can do this in several ways: enabling third-party verified data (e.g., DMV

signed driver's license), expanding the data wallet to allow other methods of storage (e.g., delegating my personal Dropbox to store data on my behalf), and creating better ways for individuals to combine their data preemptively to create data unions.

Lastly, we can improve a vast amount of functionality by increasing what can be managed on-chain by the DAO. In the future, the Protocol authors would like to see all aspects of the Protocol functionality governed by on-chain state. i.e. the allowable vocabulary of SyQL, data wallet storage schemas, data validation functions, etc.

7.1.5 Form Factor

The Protocol is designed to be form factor agnostic and cross-chain compatible. Implementation of a diverse array of data wallet client form factors will ensure that the Protocol is accessible to the widest audience.

7.2 Potential Risks

This section will detail risks with the protocol we are worried about and hope to address. These risks include worries about market adoption, security of the protocol, privacy, and equity of data as an asset.

7.2.1 Market Adoption

One of the most significant issues preventing the Protocol from improving the data economy is the adoption of the protocol; i.e., the Protocol can't help people own their data if no one is using the Protocol. This is especially difficult because we are trying to create demand and supply within a market. We have several ways we can increase the likelihood of adoption:

- Improving incentives, such as rewards from sharing data and even boosting rewards for early adopters
- Snickerdoodle Labs can partner with businesses to ensure the Protocol addresses their needs and increase the rewards available to people
- The tokenomics of the protocol can help increase early adoption

7.2.2 Security

Another risk addressing the Protocol is the security of the system. Individuals cannot own their own their data if it is not secure. Data wallets need to use strong encryption and access control techniques to protect collected data and should be upgradable to fix issues with security. For example, we can upgrade the encryption when systems need to become post-quantum secure. Another piece of attack vector that needs to be protected is when data and insights are shared. The transfer can be protected through standard means, such as TLS and HTTPS. It is harder to ensure strong security when data is sent to an aggregation service provider. People need to know their data is secure regardless of the aggregation provider. Reputation scores can help people choose which aggregation provider they want to send their data to, and the DAO vote on authorized aggregation services. There may be creative ways to add a token penalty if someone can prove an ingestion provider didn't hold their data securely. Another approach is reducing the harm if an ingestion service leaks data.

7.2.3 Privacy

Privacy issues are also a prominent issue facing the protocol. While the current economy does not lend much privacy to individuals, the Snickerdoodle Protocol aims to improve privacy significantly and shouldn't worsen privacy on web3. There are some potential privacy issues that the protocol faces. First, once data leaves a wallet and is sent to an ingestion provider, the user no longer has any direct control over the wallet. Ideally, the user should know, regardless of where the calculations on their data are being sent, that they aren't at risk of being identified, tracked, or any other risks of de-anonymization.

Second, the Protocol is built on a public blockchain. Metadata about the Protocol will be public for everyone to see. We don't want this metadata, such as what a user is consenting to or what rewards a wallet has received, to be able to lead to any negative consequences. Earlier in this section we've discussed several ways we could improve the protocol to prevent this.

Third, changes in the protocol through governance could lead to an erosion of privacy. We need to create checks to prevent this. Additionally, the protocol should be build and continued to built with forward-secrecy in mind. Specifically, changes in the protocol shouldn't reveal previously private data or messages.

A fourth issue is we currently don't know how sensitive a wallet address is in a post mass web3 adoption world. A wallet address that isn't linked to any other bit of information is not PII because it can't be linked back to anyone; however, a single link and suddenly everyone can know who that wallet address is linked to. If users in a web3 world use and link the same wallet with many different applications then for most people their wallet would be PII. This feels eerily similar to using email addresses in web2: they are used to login and verify an account and early on most people didn't mind giving them away willy-nilly but now if you do you know you'll get targeted with one form or another with spam.

7.2.4 Equity

We are turning data into an asset and like any asset issues of equity should be a major concern. From a governance perspective we need to make sure that supply doesn't consolidate and leave the protocol at risk. From an individual's perspective we need to make sure no group of people are unfairly and systematically given reduced value on their data, especially historically disenfranchised individuals. For example, not all data in the current economy is treated equal. Data from iPhones' are worth more because people who buy iPhones typically have more disposable income. The Protocol author's do not wish to create a system that naively perpetuates the current systems of wealth inequality.

8 Conclusions

This paper has discussed the current data economy, what is wrong with it, and how the Synamint Protocol aims to flip the power of the economy by allowing individuals to control their own data. The Protocol allows individuals to collect, store, share and provide verifiable consent to access their data through the use of a data wallet. The Protocol allows insight consumers who are getting value of analyzing data to continue to do so by spending Protocol tokens and without having to worry about the legal, technical, and ethical hurdles of storing and using people's data. The Synamint Protocol will be controlled by the Protocol DAO which will create a decentralized system of governance that allows the Protocol to improve over time and survive without being completely controlled by Snickerdoodle Labs. These changes can help further improve the Protocol's functionality and data safety guarantees.

References

[1] Kean Birch, DT Cochrane, and Callum Ward. Data as asset? the measurement, governance, and valuation of digital personal data by big tech. *Big Data & Society*, 8(1):20539517211017308, 2021.

[2] Minna Lammi and Mika Pantzar. The data economy: How technological change has altered the role of the citizen-consumer. *Technology in Society*, 59:101157, 2019.

[3] L.F. Garifova. Infonomics and the value of information in the digital economy. *Procedia Economics and Finance*, 23:738–743, 2015. 2nd GLOBAL CONFERENCE on BUSINESS, ECONOMICS, MANAGEMENT and TOURISM.

[4] OECD. Vectors of digital transformation. *OECD Digital Economy Papers n. 273*, 2019.

[5] Dan Ciuriak. Rethinking industrial policy for the data-driven economy. *Available at SSRN 3223072*, 2018.

[6] Trinh Viet Doan, Roland van Rijswijk-Deij, Oliver Hohlfeld, and Vaibhav Bajpai. An empirical view on consolidation of the web. *ACM Trans. Internet Technol.*, 22(3), feb 2022.

[7] Consolidation in the internet economy. 2019.

[8] Barbara Prainsack. Logged out: Ownership, exclusion and public value in the digital data and information commons. *Big Data & Society*, 6(1):2053951719829773, 2019.

[9] Brooke Auxier, Lee Rainie, Monica Anderson, Andrew Perrin, Madhu Kumar, and Erica Turner. Americans and Privacy: Concerned, Confused and Feeling Lack of Control Over Their Personal Information, November 2019.

[10] Facebook doesn't know what it does with your data, or where it goes: Leaked document. 2022.

[11] Carlos Santana and Laura Albareda. Blockchain and the emergence of decentralized autonomous organizations (daos): An integrative model and research agenda. *Technological Forecasting and Social Change*, 182:121806, 2022.

[12] Eugenia Politou, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. *State-of-the-Art Technological Developments*, pages 69–91. Springer International Publishing, Cham, 2022.

[13] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[14] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[15] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI global, 2005.

[16] Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. A data outsourcing architecture combining cryptography and access control. In *Proceedings of the 2007 ACM workshop on Computer security architecture*, pages 63–69, 2007.

[17] Ceramic protocol specification. 2022.

[18] Ocean protocol: Tools for the web3 data economy. 2022.