A Data-Ownership Assuring Blockchain Wallet For Privacy-Protected Data Exchange

Subtitle Draft Version 0.8i

Daniel Hawthorne[1] Serafin L. Engel[1] Alex Norta[2]

[1] Pnyks Inc.
55E 3rd Ave San Mateo, CA 94401
USA

[2]Large-Scale Systems Group,
Tallinn University of Technology,
Akadeemia tee 15A, 12616 Tallinn,
Estonia

Contents

Disclaimer

Nothing herein constitutes an offer to sell, or the solicitation of an offer to buy, any tokens, nor shall there be any offer, solicitation or sale of Datawallet tokens in any jurisdiction in which such offer, solicitation or sale would be unlawful. You should carefully read and fully understand this whitepaper and any updates. Every potential token purchaser will be required to undergo an on-boarding process that includes identity verification and certain other documentation, which you should read carefully and understand fully because you will be legally bound. Please make sure to consult with appropriate advisors and others.

This white paper describes our current vision for the Datawallet platform. While we intend to attempt to realize this vision, please recognize that it is dependent on quite a number of factors and subject to quite a number of risks. It is entirely possible that the Datawallet platform will never be implemented or adopted, or that only a portion of our vision will be realized. We do not guarantee, represent or warrant any of the statements in this white paper, because they are based on our current beliefs, expectations and assumptions, about which there can be no assurance due to various anticipated and unanticipated events that may occur.

Please know that we plan to work hard in seeking to achieve the vision laid out in this white paper, but that you cannot rely on any of it coming true. Blockchain, cryptocurrencies and other aspects of our technology and these markets are in their infancy and will be subject to many challenges, competition and a changing environment. We will try to update our community as things grow and change, but undertake no obligation to do so.

Abstract

As people engage more deeply and often with Internet services like social media, search, and e-retail, they create ever more data. Well known social-media companies and a secretive data-brokerage industry monetize this personal data without explicitly consulting—or sharing the revenues with—the users who produced the content. Such is the dominant personal-data ecosystem. There is clear need for a transparent and fair data exchange based upon the expressive consent of the Internet users producing the data. This whitepaper specifies such a system built upon a user-controlled ”DataWallet” and a blockchain-based smart contract system to allow transparent and mutually beneficial exchange of data between consenting parties. We detail the requirements of such a system, its necessary structure, and the dynamic interactions of its components and stakeholders through an example focusing on a simple exchange of data for DXT (the data exchange token). We show how such an exchange is just one of many use cases for the proposed data exchange by describing how it can provide an augmented user experience for online video and music services as well as provide valuable insights to data creators about themselves.

Keywords: data exchange, blockchain, smart contract, decentralized, peer-to-peer

1 Introduction

As people spend ever more of their time engaging with online services, the volume and importance of the data created from these interactions concomitantly increases. Internet users are being cut out of this rapidly growing economic sector that has the power to impact their experience both online and off. There is an opportunity for data providers to reclaim the value they create online from the owners of the current data ecosystem.

A data-based economy emerges in which users produce an exponentialy increasing quantity of content online through social media, online searches, and the purchase of goods on e-commerce platforms. However, the current data-based economy, centered around the 'data-brokerage industry,' is deeply dysfunctional for both data creators and data consumers. The situation for data providers is captured well by a 2013 US senate court hearing[1] on the topic which concluded that customers of the data-brokerage industry have no control how their personal data and created content is used and monetized. Data providers know this—in a recent study[2], 81% of respondent state they do not feel secure using social media for sharing private information. The data-sharing economy is also broken for data consumers. This dysfunction can be understood in terms of the silo, quality and ethics problems. The *silo problem* is that the data is scattered across a field of walled gardens with most of the potential value only available if the data could be intelligently collated. This contributes to the *quality problem*. The available data is often the result of probabilistic matching models (with uninspiring results [1, 9, 12, 25]), outdated, or generally incongruent with true online engagement. Finally, the *ethics problem* is clear—data is acquired and monetized by corporations without informing the owning users. The advent of blockchain technology[3] poses a remedy to the current dysfunctional data-brokerage system. It enables transparent systems that can simultaneously give data creators back control over their data while providing data-informed businesses the highest quality, ethically sourced data.

The shortcomings of the current data brokerage system and the necessity for a new approach to willful data exchange will only become more clear given the current technological and social trends. First, data production is going to dramatically ramp up. A staggering 90% of all data has been created in merely the past two years and this market is predicted to grow $\tilde{2}7\%$ per year. This explosion of data will be further buttressed with the increasing adoption of Internet-of-Things (IoT) systems

[1] https://tinyurl.com/senate-hearing-2013

[2] http://www.adweek.com/digital/study-pew-public-perceptions-privacy/

[3] Briefly, the blockchain is a distributed database for independently verifying the chain-artifact ownership [24] in hash values that result from cryptographic digests [13, 19].

[28]. Second, the security and privacy situations for user data is broadly lacking[4]. Cloud-hosting is the default for IoT and social media storage despite providing a wider security surface area [15]. To assure the security and privacy of data, a suitable requirements-engineering framework is lacking [2] for system development.[5]

Given the deficiencies of the prevailing data brokerage system, it is surprising how few personal data-management tools are available to data creators. One promising option is to consider the development of specific blockchain-based web browsers[6] that strive to allow user control over self-generated content. However, this requires a major shift in user behavior with concomitant technological challenges. This paper proposes an alternative that disrupts the data-brokerage system, and not the data-creators Internet experience. We propose a blockchain-technology based DataWallet that restores trust and control for users while providing data-informed businesses the most complete data profiles possible. In this paper, we therefore systematically describe a system that enables this opt-in and mutually beneficial data-sharing ecosystem. We proceed by answering the following progressively refined sub-questions. What are the requirements of such a data-sharing ecosystem? What is the static architecture of the ecosystem that fulfills these requirements? And, finally, what are the dynamic interaction protocols of stakeholders and this architecture that enables the desired mutually beneficial data exchange economy?

The remainder of this paper is structured as follows. Section 2 introduces a running case and additional background literature. Section 3 defines the requirements for a blockchain-technology based datawallet system. Section 4 shows the system architecture of the datawallet that is derived from the requirements. Next, Section 5 explains the dynamic system engagement. Finally, Section 6 concludes this whitepaper and also discusses limitations, open issues and future work.

For concision, the description of the system is in the present tense, but the fully decentralized system described is the last milestone of the implementation roadmap (described in Appendix A). Additionally, for clarity we focus our description on the fully decentralized system (v3.0 in Appendix A.3). However, hosted versions of these components will remain available for interested community members (see v2.0 in Appendix A.3).

[4] These security issues extend to extremely sensitive information as the recent Equifax data breach illustrates [32]

[5] Specifically for the IoT domain, the security and privacy of forecast parabolically growing data production is not assured since existing security frameworks do not scale to large networks of heterogeneous devices. For example, the IoT-subclass of so-called wearable devices [3] without secure hardware and software stacks can not authenticate running software and are therefore unable to validate themselves.

[6] https://www.cryptocoinsnews.com/blockstack-joins-browser-wars-decentralized-tokenized-blockchain-web-browser/

2 Running Case and Background Literature

Figure 1 depicts the current state of user-content generation and monetization. Internet users pictured to the left, generating and consuming data through shopping, searching, and socializing on the web. While each service cultivates its 'walled garden,' some choose to engage in large-scale data exchanges among each other.

The right side of Figure 1 depicts the numerous centralized interests that consume personal user data. The corporate categories to the right may comprise profit-oriented organizations such as banks, advertising companies, tangible service-producers. Additionally, government-affiliated non-profit organizations also consume social-media data, e.g., the police and intelligence agencies.

Thus, while Figure 1 shows that individual users generate and communicate their created content data without monetary compensation, the corporate side must pay the social-media cloud members for content-data generated intelligence. The individual users currently lack an effective tool that allows them to monetize their own data at their discretion.

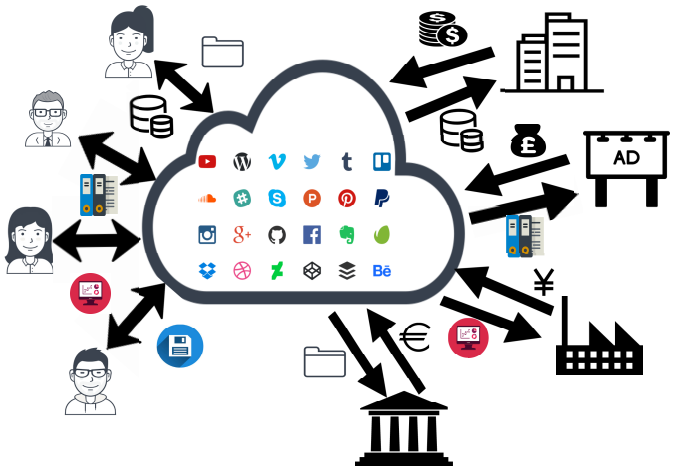

Figure 1: Current monetization situation of user-generated data content in social media.

That is why we have developed a personal data management platform—DataWallet, and a way to directly profit from personal data—dx-Insights, which is DataWallet's market and consumer insights data-application. We will use dx-Insights to provide grounded examples throughout the subsequent technical sections. This running

case is just one example of a personal-data-based product.[7] In Section 5 we highlight the expanse of possible personal-data-based products by describing two broad categories of products—data products for the data provider (e.g. for introspection, decision making, and self-improvement) and data products that *augment* existing services by customizing them with the user's data.

2.1 Background Literature

We now highlight the literature that motivates our approach and the terms necessary to understand the methodology employed in the subsequent sections. Our presentation of the DataWallet ecosystem follows the well-established model-driven design (MDD) methodology [5] to ensure quality system design, architecture, and implementation.[8] In line with this methodology we present a series of formalizations that provide an increasingly deeper treatment of the static and dynamic features of the proposed ecosystem. We employ (1) a goal-model to describe the requirements of the system, (2) an UML component diagram to outline the static architecture, and finally (3) a UML sequence diagram to illustrate the dynamic behavior of the system. We now describe these formalizations in turn.

First, goal models are part of the agent-oriented modeling (AOM) method [29] that we employ to formally specify the features of the blockchain-based DataWallet ecosystem. AOM goal models use the notation in Figure 2 to capture the functional requirements of a system in the form of functional goals, non-functional requirements, or 'quality goals', and agents (who may be human or artificial, i.e. software agents) with specified roles. We adopt an additional symbol, the right pointing arrow, for denoting multiple inheritance of higher-level functional goals to a lower-level functional goal, as seen in Figure 6.

Figure 2: Icons of AOM goal-model notation [29].

The center of an AOM goal model is a functional goal termed a 'value proposition'. This captures the overall systems goal and requires hierarchical decomposition for establishing manageable development complexity. Quality and emotional goals are attached to respective functional goals and depending on the position in

[7] In fact, dx-Insights is simply the first product available on an open data product exchange

[8] MDD entails a front loading of activities in a system-design process which results in a high degree of generated code and testing instances.

the hierarchy, all refining functional goals must also satisfy an assigned quality goal. The same holds for roles and their relationship to the functional goal hierarchy.

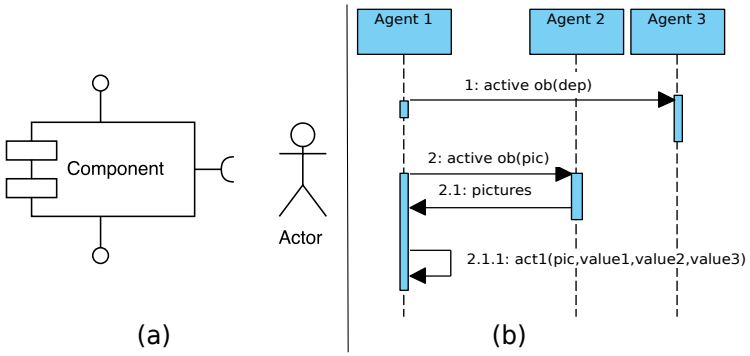

Figure 3: Notation of UML component diagrams in (a) and sequence diagrams in (b).

Second, we deduce a technology-agnostic UML component diagrams [6] from the AOM functional goals specified in the goal model to specify that static structure of the DataWallet system. Figure 3 (a) shows the UML notation elements where components are labeled rectangles. Components provide interfaces for intercommunication and are further refinable with sub-components. We also employ actors to indicate how components interact with the different stake-holders in accordance to the AOM goal-model actor-positions.

Finally, third, we use UML sequence diagrams [27] to specify the dynamic behavior of the DataWallet system. Figure 3 (b) shows an example with communicating entities on the top shown as labeled rectangles. Underneath each entity is a dashed line denoting timelines on which bars show when respective entities are active. Labeled directed arcs between those bars shows communication-message exchange along the timelines.

The aim of the described MDD approach is to capture the distributed-systems [30] nature of the DataWallet ecosystem. While Figure 1 reflects the current data ecosystem where the majority of data users produce is inside social media walled gardend, the advent of IoT, or cyberphysical systems (CPS) [26] will result in a new dimension in which users will increasingly create data. CPS are internet-integrated sets of smart objects, such as IoT-devices, that may be orchestrated by Clouds for complex processing tasks. As such, CPS integrate computational and physical capabilities that allow for interaction with humans through diverse means [4]. Such novel interaction vectors have the promise to expand the human's capabilities and

embodiment through computation, communication, and control to enable the next-generation avionics and vehicles, smart cities, Industry 4.0 production systems, e-healthcare, etc.

Finally, pertaining to blockchain technology, smart-contract technology is essential for the DataWallet system to achieve traceability and security. Essential for smart contracts is a decentralized validation of transactions for which so-called proof-of-work (PoW) [31] is the most often used. Smart contracts employ a public distributed ledger called blockchain that records transaction events without a trusted central authority. The current smart-contract standard is Ethereum [33], despite some widely acknowledged drawbacks. First, PoW transaction validation does not scale and thus, Ethereum is not feasible for most industry applications. Second, the Ethereum affiliated Solidity smart contract can not be formally verified [8] and was recently hacked[9] because of security flaws resulting in a loss of ca. $50 million[10]. More scalable is a smart-contract solution that uses proof-of-stake (PoS) [7] transaction validation and blockchain sharding [16]. For example, the smart-contract system Qtum [11] uses PoS already successfully in its application.

3 Goals of Data-Exchange Ecosystem

We define the goals and requirements of a data-sharing ecosystem built upon expressive consent using the goal model of Agent-Oriented Modeling (AOM). The AOM method is a socio-technical requirements-engineering approach used to model dynamic and complex systems composed of both human and software agents. To aid in comprehension of the requirements of the entire system, we decompose the model into coherent pieces centered around the two primary stakeholders, namely the data requesters and the providers. We begin by describing the joint high-level requirements of the data exchange in Section 3.1, and then turn to the requirements of the data providers in Section 3.2. Finally, we conclude by describing in Section 3.3 the requirements of the data requesters.[11]

[9] https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/

[10] https://bitcoinsmagazine.com/articles/ethereum-classic-hard-forks-diffuses-difficulty-bomb-1484350622/

[11] Note that when considering the requirements of the data requestor, the focus lies on those who desire a decentralized profile management solution which is the final step in the technology development outlined in Appendix A building from the centralized profile management solution currently developed by DataWallet.

3.1 High-Level Requirements

As seen in Figure 4, the principal value proposition of both primary stakeholders in the ecosystem, the data requesters and the data providers, is to engage in a consensual, transparent, and secure exchange around data. To seed this exchange, data requesters need a clear and easy method of creating a data request and incentivize its fulfillment. Similarly, data providers who either offer customizations to existing services or build novel services based upon data need a way of offering services in conjunction with requesting the necessary data and receiving compensation.

To enable this structured data request, data providers require an easy, secure, and transparent method of creating and managing a collated data profile. They then need a fair and transparent means to explore the available data requests, and engage in those they choose to. Similarly, they require a means to customize their existing services and engage in new services built upon their data profile. These are quality goals of the system, pictured in Figure 4 and structured according to [10, 14]. We now turn to describing each quality goal at length.

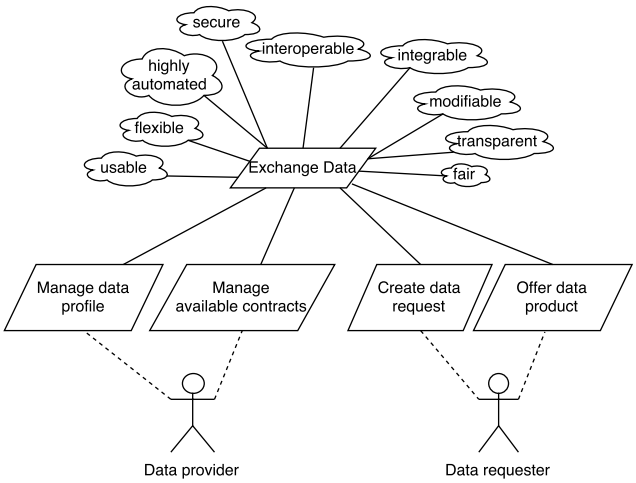

Figure 4: Root goal model for the data-wallet ecosystem.

Interoperable means that the core ecosystem must be able to interface and be composed with external components like data sources and applications. This is a challenging goal given the heterogeneity and dynamism of the external interfaces for existing data sources and applications. *Secure* indicates that the DataWallet ecosystem must resist unauthorized usage attempts and denial of service attacks while providing services to trusted data providers and requesters. *Flexible* data exchange indicates the ability of the system to integrate diverse and heterogeneous

data and data products with a uniform interface. *Usable* means that the DataWallet ecosystem must be clear and approachable for both data providers and data requesters. *Error avoidance* pertains to proactively anticipating and preventing commonly occurring collaboration errors. *Error handling* ensures system support for logging and gracefully handling software errors in the ecosystem. *Learnability* refers to minimizing the time it takes users to master the DataWallet ecosystem. *Transparent* data exchange pertains to ensuring both that the data provider and requester have a clear understanding of the data points, compensation, and services being exchanged, and that they are each away of each other's previous reputation. Finally, *Fairness* of the DataWallet ecosystem means that the stakeholders are treated without favoritism or discrimination and that all exchanges are clearly defined and explicitly accepted.

Additionally, there are quality goals that are not discernible during any particular interaction with the DataWallet ecosystem. *Integrable* indicates that all component must conform to a specified interface to facilitate the interaction of functionally contained components. *Modifiable* means that the system must adapts during its lifecycle to the application context, e.g, to accommodate new data-format standards.

3.2 Data-Provider Requirements

The hierarchy in Figure 5, represent the data-provider core requirements—managing the data profile and available contracts. Managing the data profile requires assurances that the data is secure and private which necessitates private key management and storage which is used to encrypt the data. The provider requires to be in control of which data is included in the profile and which fields are exposed to the data exchange at what point in time. The collation of the data profile requires the linking of the data sources, possibly though API access keys that need to be stored. This collated data profile should be updated according to the provider's data-directives.

The second refining hierarchy of Figure 5 illustrates the data providers' requirements for engaging and managing available data and product contracts. The core functionality desired is to find available contracts in accordance with the provider's data directives[12]. Providers require access to different kind of contracts—simple data requests like those posted by dx-Insights, novel data products, or personal-data-based augmented experience of existing products. Each of these three kinds of contracts may be targeted/restricted to a particular population of providers that needs to be automatically managed for the provider. Once accessed, the provider needs to be able to accept or reject contracts. To fulfill an accepted contract, providers

[12] depending on their optimal trade-off between privacy and efficiency they can choose to make some demographic fields available to the contract search to expedite the process

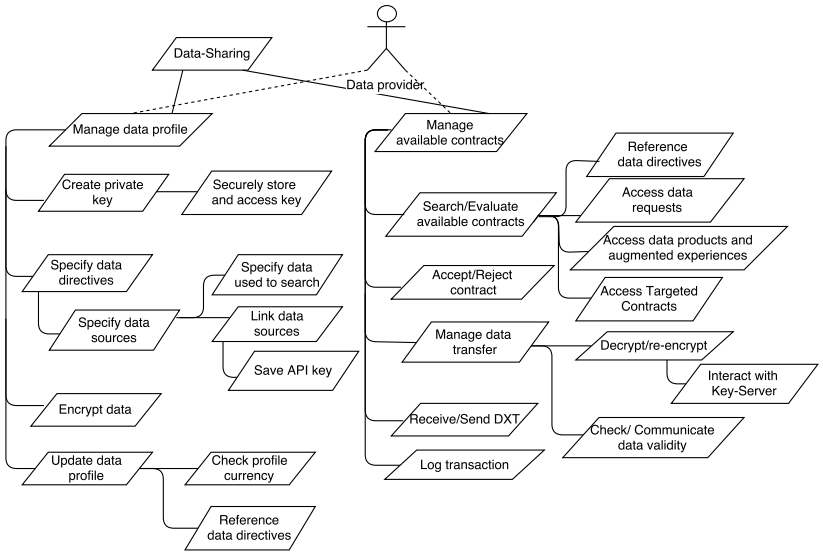

Figure 5: Data-provider profile management and contract engagement goals.

need to be able to send the agreed upon data points, which requires isolating the data points, decrypting them, check and communicate the data's validity to ensure fidelity, e.g., through a 'data hash' match, and ideally re-encrypting them with the requester's public key to ensure the sanctity of the transfer. Contract management also necessitates interacting with an escrow service to send and receive DXT after the exchange has completed. Finally, all transactions should be recorded in an easily-accessible and persistent log.

3.3 Data Requester Requirements

The goal model refinement for the data requester in Figure 6 displays their two primary requirements. First, they require being able to create a data request in exchange for compensation (DXT). Second, they desire to be able to offer a data product which requires specific data points, and therefore, much of the same functionality of a simple data request. We therefore employ the multiple inheritance symbol to highlight the overlapping goals, and use colorized goals to indicate where the requirements diverge (breaking multiple inheritance, see Section 2.1 for discussion).

As Figure 6 shows, both data requests and product offers need the ability to specify the required data points, and optionally to specify a particular population for which the contract is valid either through demographic parameters, or through an

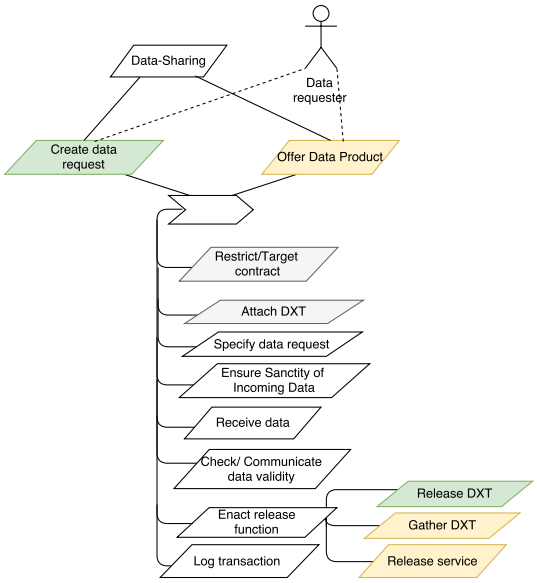

Figure 6: Data requester goals. Grey shaded goals are optional, and colored goals are owned by the correspondingly colored parent (breaking multiple inheritance).

access code/qr code. They then also need the ability to ensure the data they receive is not able to be interpreted along the transfer process, which is accomplished by sending a public key to the data provider to encrypt the data prior to transfer in a way that can only be decrypted with the requester's private key. The data they receive has to be able to be verified, e.g., through a hash match. Valid data must enact the specified release function. For the data request, it is the release of DXT allocated in escrow. For a data product, this requires the release of the promised service. Finally, all transactions must be recorded in an easily-accessible and persistent log.

Having specified the requirements of the data requester and provider in terms of AOM goal models, we now turn to the architecture that satisfies the requirements.

4 Data-Exchange Architecture

We outline the core architecture of a system that satisfies the requirements for both the data providers and the data requesters previously outlined. Figure 7 shows the overall component diagram of the DataWallet ecoystem. Note that there are three parts in the figures that group the components for the data provider, data-requester and data-exchange components respectively. Each part contains a crucial and central *Manager* component—the *Data-Profile Manager*, *Smart-Contract Manager*

and the *Request Manager*—whose multiple functionalities are also depicted.

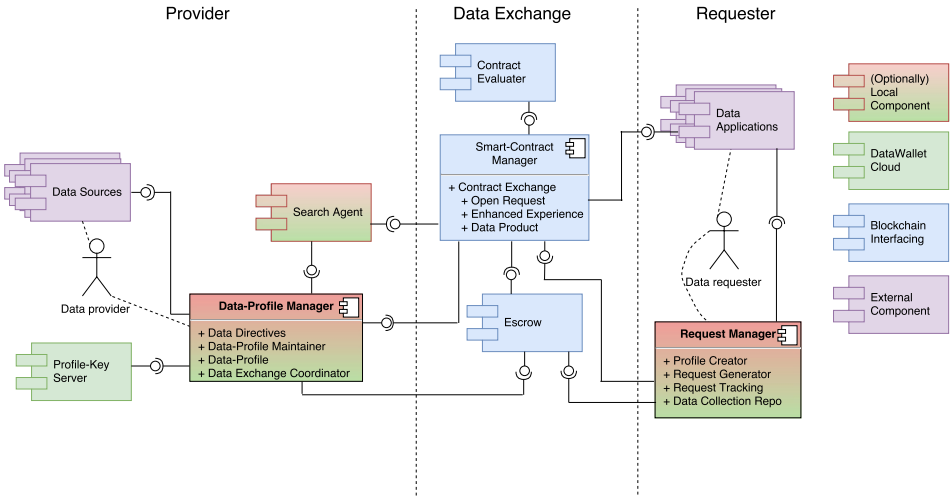

Figure 7: A UML component diagram of the data-wallet ecosystem.

We describe each part of the architecture in turn. Section 4.1 describes the components supporting the data requester. Section 4.2 denotes the corresponding components of the data provider. Finally, Section 4.3 details the blockchain-employing data-exchange components that mediate the data and service exchange between providers and requesters.

4.1 Data-Requester Components

The core component that the data requester directly engages with is the *Request Manager* and provides two interfaces—an API for programmatic requests from data apps and expert users and a user-friendly web-based system. The *Request Manager* allows for two major behaviors. First, it allows for the creation of a new data request, and second, it supports querying the status of existing requests and accessing the resulting data.

The *Request Manager* first guides new requesters through the creation of a profile and at least one public-private key pair to facilitate encrypted data transfer. The Request Manager then allows requesters to create data requests by populating a data-contract 'template' (*dx-contract template*), specifying the information in Table 1. An abstracted interface, a 'Data API', to collect data points that address common use cases, provides an alternative to manual contract specification. The Data API is built upon a pool of ready-made contracts developed and updated by the *DataWallet* and the community for common use cases.

To turn the template into a full-fledged data exchange contract and post the *dx-contract* that Table 1 depicts, the *Contract Exchange* requires the requester to allocate the specified DXT into the *Escrow* and affix the public encryption key to the contract. These data requests are placed in the *Open Request* section of the *Contract Exchange*. For example, the *dx-Insights App*, the first product on the *Data Application Exchange*, generates data requests for companies looking for customer and market insights based upon collated social media data from specific populations of providers, all of which are viewable on the open contract exchange.

		Contract Type	
		Data Request	**Data Product**
Field	**Mandatory**	Data points requested (E)	
		What data is used for	
		Compensation (DXT)	Product price (DXT)
		Number of Data Providers	Category of product (E)
		Rolling or Adoption Threshold	Product description (optional external describing links)
			Product-delivery mechanism
	Optional	Provider(s) demographic parameters (E)	
		Take-down date for request expiration	
		Data format, or pre-processing preferences (E)	
		Data routing port (if not the standard Requester data-manager)	
	Operationalizing Action	Allocate specified DXT into escrow (and additional percentage to support the contract-enactment ecosystem)	
		Attach a public encryption key	

Table 1: List of the template fields for a dx-contract. The first column specifies all the fields associated with a data request, while the second column indicates all fields associated with a data-product offering. **E** indicates an enumeration field, i.e., a field chosen from an pre-specified list of options.

In addition to the *Open Request* section of the *Contract Exchange*, there is the *Data Product* section, whose contracts are similarly created. As seen in Table 1, it shares some of the same fields as a data request, but requires the specification of additional fields such as the price of the product (in DXT) and a specification of how the *Data Application* is delivered upon execution of the contract. The application delivery mechanism can either be contract-internal logic that, for example, provides an access code and link to the data provider, or a description of off-chain delivery of access to the product. For example, *DataWallet* provides a personality analysis product that is based upon a provider's messages on social media. The contract on the *Data Product* section of the *Contract Exchange* indicates the required data points, the price for the product in DXT, and the delivery mechanism. The latter is a uniform resource locator (URL) transferred to the provider's *Profile Manager* that is connected to the hosted personality profile with an embedded, initial password,

which can be subsequently changed.

Through the *Request Manager*, requesters can also manage the contracts they have placed on the *Contract Exchange*. For 'rolling' contracts that allow providers to immediately execute the contract, requesters can see the current number of executed contracts and the resulting data (if the Request Manager's *Data Collection Repo* is used). For thresholded contracts, requesters can see whether it has executed (or set an alert when the threshold or an arbitrary number of providers have staked the contract).

4.2 Data-Provider Components

Data providers interface with the data-exchange via their *Data-Profile Manager* that has two major responsibilities. First, it constructs and maintains the provider's *Data Profile*, either through a cloud-based service, or on the provider's local system. Second, it enables the provider to use this profile and engage with contracts on the *Data Exchange Manager*.

The *Data-Profile Manager* constructs and maintains the provider's data profile in accordance to their *Data Directives*. The provider's *Data Directives* determine which data is retrievable from external *Data Sources* to customize the profile and search-agent behavior. For example, a provider shares likes and engagement times from a social media company, but not his private messages. Additionally, a data provider may specify that they want their respective data-profile updated at regular specified intervals, or on-demand. They could also customize their search-agent to have access to their age and geographic information to more quickly find viable contracts.

Given the data directives, the *Data-Profile Manager* coordinates with the *Requester* and the *Profile-Key Server* to construct and maintain the provider's data profile. First, the *Data-Profile Manager* guides providers through the generation of a profile encryption key which is then split with the 'remote' half of the key being stored on the *Profile-Key Server* to ensure no single point of failure.

The *Data-Profile Manager* enables providers to leverage their data on the *Contract Exchange* through its coordination of the provider profiles with the *Profile-Key Server*, *Smart-Contract Manager*, *Escrow*, and the provider's *Search Agent*. As described in Section 5.4, the *Search Agent* finds dx-contracts that are consistent with the requestor's data profile and directives. For example, the *Search Agent* returns compliant contracts posted by the dx-Insights App to a provider searching for contracts on the open exchange to acquire DXT for data. To execute the contract, the *Data-Profile Manager* must isolate the specified data points, decrypt them with the combined local and *Profile-Key Server* split-key, format the data as specified in the contract, re-encrypt the data with the requester's public key and send

the resulting data to the *Smart-Contract Manager* where the data is routed either to the *Request Manager's Data Collection Repo*, or an external *Data Application*, or server, as specified in the contract. The execution of this contract is described in detail below in Section 5.4.

4.3 Data-Exchange Components

As Figure 7 depicts, the core component underpinning the *Data-Exchange* is the *Smart-Contract Manager* which coordinates the execution and blockchain interaction of dx-contracts. Note that much of the functionality described in this section as the province of the *Smart-Contract Manager* component is executed by the smart-contract itself (see Section 5.1 for the dynamics of the blockchain interaction throughout the lifecycle of a dx-contract). This requires the coordination of the blockchain components, the *Escrow* and *Contract-Evaluater* to ensure the compliant execution of dx-contracts, as well as the provider's *Data-Profile Manager*, *Search Agent*, the *Request Manager* and *Data Applications* components. The *Smart-Contract Manager* hosts the dx-contract exchange that internally is factored into two categories—the data request exchange and the data product exchange. Each have a public 'open' section and a 'targeted' section that is only viewable by providers with the appropriate rights (see Section 5.5 for details about this distinction).

The *Smart-Contract Manager* provides an interface to the data provider's *Search Agent* that allows access to compliant contracts, which conform to the provider's directives and coordinates with the *Search Agent* and the provider's *Data Profile* to verify that the latter is compliant with the requester's specifications (see Section 5.4 for the details of this process). The *Contract Evaluator* needs to verify various elementary aspects of smart contracts such as the soundness of control-flow, data-flow, resource allocation, exception, and compensation management [20, 21]. On the other hand, the *Smart-Contract Manager* also coordinates the enactment of contracts, e.g., checking to ensure the data transferred is compliant and then signaling the release of the DXT to the provider's data wallet. The *Smart-Contract Manager* follows a specific lifecycle [19] that comprises a specific setup [17], rollout and enactment [18] and rollback with termination phase [22].

5 Dynamics of the Data-Exchange Ecosystem

This section describes and formalizes the dynamic behavior of the decentralized DataWallet ecosystem with UML sequence diagrams [27] (in accordance with the notation of Figure 3(b)). The sequence diagrams show the exchange of data, value

stores (DXT), and data services between data providers and requesters with the components described in Figure 7. All exchanges described below ensure that 1) the data provider knows exactly what data is being exchanged, 2) the data is never available in clear text, 3) the data is only recoverable by the specific requester signified by the contract, and 4) the requester is getting data that conforms to their specifications. Ensuring these features require transparent and auditable software agents maintained by DataWallet, smart-contracts [23] that ensure transparent compliance of all parties, and the judicious use of asymmetric cryptography between data providers, data requesters, and DataWallet maintained software agents.

The remainder is structured as follows. Section 5.1 discusses the blockchain transactions that we consider for achieving event traceability. Section 5.2 describes the creation and management of data profile by data providers. Section 5.3 shows a data requester creating and posting data-request smart contract we term *dx-contract*. Section 5.4 details a simple data-transaction such as those conducted by dx-Insights. Slight modifications of the basic dynamic behavior required for simple data-transactions enables more sophisticated data and product exchanges. This broader functionality is shown in Section 5.5, which details how data products designed for the providers as end users can be exchanged, and finally, Section 5.6, which describes how a data provider can provide data to augment one of their existing services.

5.1 Blockchain Operations

Storing events on a blockchain is costly in terms of computing power and transaction fees. In the case of Ethereum, PoW (solving cryptographic riddles for transaction validation) proves to be a performance and scalability bottleneck. Thus, it is prudent to define the minimal set of transactions that allows for the required traceability in the DataWallet ecosystem.

Role				
rq	pr	cm	number	operation
x	x		1	DataWallet ID created (uid)
x			2	Data Sourced
		x	3	Smart-contract deployed
		x	4	DXT in escrow
	x		5	Data sent
x		x	6	Data received
		x	7	Data checked
x	x		8	DXT received
		x	9	Smart-contract completed

Table 2: Blockchain transactions for the DataWallet ecosystem.

Table 2 lists the operations with the *rq*, *pr*, and *cm* columns denoting the roles of *Data requester* and *Data provider*, *Smart-Contract Manager*, respectively. The next column gives an operation ID that we will refer to throughout the remainder

of this section to clarify how blockchain operations are used. The right-hand column briefly labels operations. Note that we keep the set of blockchain operations minimal to limit computational and transactional costs. The minimal set of operations is chosen to optimize the a utility landscape that spans cost, performance, DXT tracking, and latency calculations with respect to the current technology.[13]

We now provide a more thorough description of each operation indicated in Table 2. Operation 1 stores the unique id (uid) of newly created data provider and requester accounts. Operation 2 registers when a data source is integrated into a provider's profile. Both the time and source id are recorded. Operation 3 saves every contract placed upon the exchange. Operation 4 records the deposit of DXT in the *Escrow* associated with the contract. Operation 5, 6, and 7 indicates when data is sent by the provider, received by the requester, and checked against one-another to assure fidelity. Operation 8 stores the event when DXT have been received by the provider. Finally, Operation 9 records the successful enactment and termination of a smart-contract.

5.2 Create Data Profile

Data providers create a local data profile through a DataWallet supplied application that is optionally hosted locally with the data provider.[14] Figure 8 shows the message-exchange protocol for data-profile creation between the responsible agents.

Note that the message-labels of arcs in Figure 8 and all other sequence diagrams of Section 5 represent pseudo-code that we use to facilitate the protocol explanations. The creation of a profile is initiated by the data provider who interacts with their *Data-Profile Manager*. The *Manager* orders the creation of the profile based upon the users unique identification (uid) *cr_profile(uid)* and the creation of a private user-key *cr_pv_key(user)*. It then splits the key. It transmits half of the key to the *Profile-Key Server pv_key(uid)*, and stores the other half locally.

pNext, the *Data-Profile Manager* requests *rq_fill(dd)* that the provider fill out *update(dd)* their data directives to control/customize their data profile and search-agent behavior. The relevant directives are transmitted to the *Profile-Key Server* and the *Search Agent*. For example, the types/categories of requesters the provider wants to consider (either by exclusion via a blacklist, or by inclusion via a whitelist) is transferred to the *Search Agent*. Then the *Manager* requests access to the data

[13] as the technology improves the optimum point on the transparency/efficiency trade-off will change

[14] The details of this section follow providers who opt for a locally hosted data profile, but data providers who prefer to utilize DataWallet's cloud profile hosting can also engage with the exchange

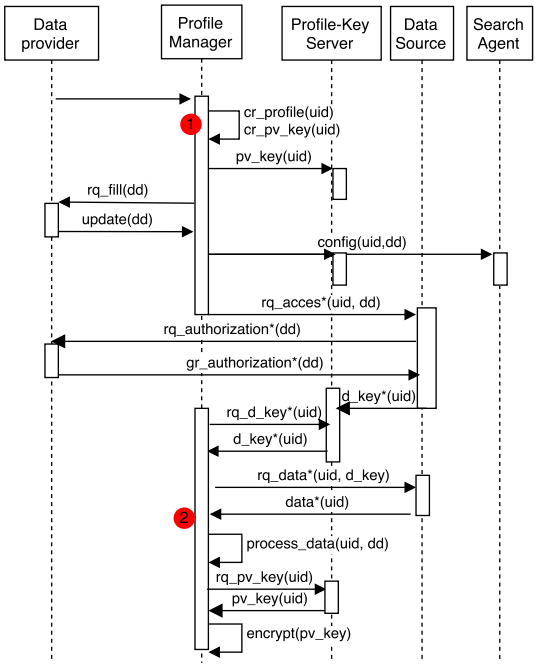

Figure 8: Data-profile creation in a UML sequence diagram.

points the provider specified from each external *Data Source rq_acces*(dd)*[15]. For most external *Data Sources*, this requires the provider to explicitly authorize the sharing of the specified data points *rq_authorization(dd)*. When the provider grants this authorization *gr_authorization(dd)*, the resulting data keys (in some cases API keys that provide programmatic access to the data) are transferred to the *Profile-Key Server*.

To then construct the provider's collated data profile, the *Data-Profile Manager* requests the data for each data source specified in the provider's *Data Directives*. This requires requesting each data source's key *rq_d_key*(uid)*, receiving it d_key*(uid). With the data key, the *Manager* then requests the data from the external *Data Source rq_data*(uid, d_key)*. After receiving all the data *data*(uid)* and collating it *process_data(uid, dd)*, the *Manager* encrypts the resulting profile *encrypt(pv_key)*, which requires requesting and receiving the remote half of the private key *rq_pv_key(uid)*/*pv_key(uid)*.

Finally, the completion of the profile is stored on the blockchain à la Operation 1

[15]Note that functions that are repeatable are indicated with an asterisks, e.g. *fn*(param)*. In this instance requesting access will be done for each data source specified in the provider's data directives

of Table 2. Specifically, the timestamp and uid of the profile is stored (as visualized with the red-circled 1 seen Figure 8

5.3 Create dx-Contract

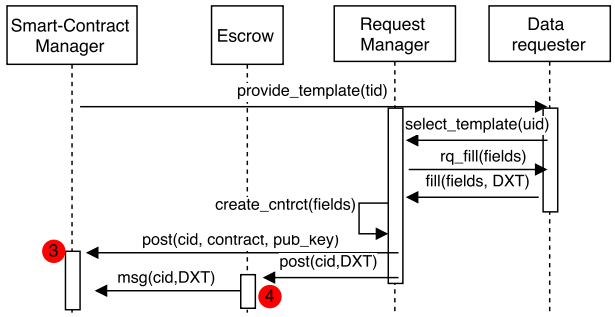

Figure 9: UML sequence diagram of the creation of data request, or dx-Contract.

Data requesters can post contracts on the open data exchange through the their *Request Manager*, which coordinates with the *Smart-Contract Manager* and *Escrow* as described in Section 4.1. As seen in Figure 9, in order to create a contract the requester first selects a template *select_template(uid)* from the pool available in the *Smart-Contract Manager*. The *Request Manager* ensures all required fields (see Table 1) are completed *rq_fill(fields)*. The *Request Manager* then creates the contract *create_cntrct(fields)*, and posts it to the *Smart-Contract Manager* along with the contract id (cid) and the provider's public key (pub_key) *post(cid, contract, pub_key)*. This contract posting is saved to the blockchain (Operation 3). The *Request Manager* also posts the necessary DXT into the *Escrow* account associated with the contract id *post(cid,DXT)*. This DXT allocation is recorded in the blockchain (Operation 4) and communicated to the contract in the *Smart-Contract Manager*.

5.4 Simple Data Exchange

Once a dx-contract (see Section 4.1) is placed on the *open-exchange* it is viewable by the data providers' *Search Agent*. In Figure 10, we describe the process by which a provider finds and executes a data request. While users likely prefer the agent to run in a batch mode, we examine the sequential fetching and evaluation of contracts for clarity. To initiate a search for a simple data request on the open exchange, a data provider directs their *Search Agent* to inspect the contracts posted on the *open-exchange rq_open_contracts(uid)*. In order for a request to be viable for

a data provider, the provider needs to possess all of the data points specified, and conform to the (optionally) specified demographic information. As seen in Figure 8, providers can, through their *data directives*, grant the *Search Agent* relevant information to ensure that all contracts returned satisfy both conditions. Granting the *Search Agent* the relevant data enables it to performing contract viability checks in the optimized contract management environment. Figure 10 therefore assumes that the *Search Agent* has all necessary information. Otherwise, additional communication and coordination are required between the *Search Agent*, the dx-contract, the provider's *Profile Manager*, and the *Profile Key Server* to decrypt the relevant data points and then ensure that they conform to the dx-contract's specifications.

An example of this process is dx-Insights posting a contract à la Section 4.1 and Table 1. For example, dx-Insights could be looking for the favorite television shows and musical artists of 25-33 year-olds who live in cities on the east coast. A data provider who was looking to exchange some data for DXT would be able to quickly find such contracts that are viable for them if they have given their *Search Agent* the relevant information. In this case that would be the list of fields available in their *data profile*, which include favorite tv shows and musicians, as well as the values of key demographic parameters like age and location.

Once a request has been determined to be viable, the data provider has the option to consent to the contract *consent(contracts)*. If the provider consents to the contract then their *Profile Manager* selects and decrypts the agreed upon data. This is accomplished by requesting the private profile key *rq_pv_key(uid)* from the *Profile-Key Server* and combining it with the locally stored half. Next, the data is hashed and sent to the contract *data-hash(cid)* as a reference for the fidelity of the data transfer. The *Profile Manager* then encrypts the data with the requester's public key *encrypt(req_pub_key)* (which is provided with the contract). At this point, if the contract requires, the provider stakes DXT into the *Escrow deposit+(dxt)*. Finally, the encrypted data is sent to the specified port of the hosted *Request Manager* or other compliant system.

Once the *Request Manager* receives the data it decrypts it with the appropriate private key *decrypt(req_pv_key)*.[16] It then sends a hash of the decrypted data to the *Contract Manager data-hash(cid)*. This triggers the *Contract Manager* to employ the *Contract Evaluator* to ensure that the sent and received data hashes match, and that all other conditions of the contract are met. If the contract is fulfilled *result(cid)* the *Contract Manager* instructs the *Escrow* to release the specified DXT *release(uid,cid)* to the *Data provider*. Finally, the transaction is appended to both the providers' and requesters' logs.

[16] The data-requester (and only the data requester) is able to decrypt the data they receive given the asymmetric cryptography scheme employed.

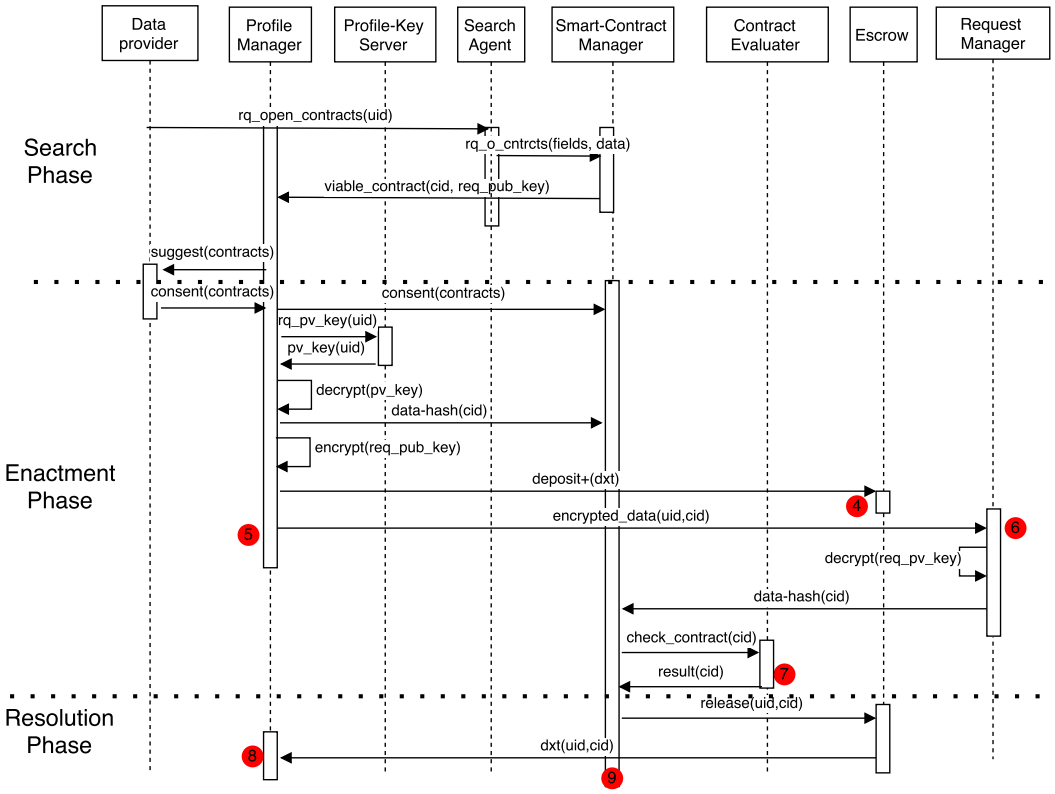

Figure 10: UML sequence diagram of the execution of a simple data request.

The blockchain-transactions are indicated in Figure 10 with the numbered red dots that correspond to the transaction IDs of Table 2. Thus, when the provider stakes DXT (if required) this is stored à la 4. It is also recorded when the data is sent from the provider's *Profile Manager* (Operation 5), and when it is received by the *Request Manager* (Operation 6). Operation 7 record when the *Contract Evaluator* has validated the data exchange and other terms of the contract prior to contract enactment by the *Smart-Contract Manager*. Operation 8 indicates when the DXT is received by the provider. Finally, Operation 9 terminates the contract enactment.

The next two sections describe how minor alterations to the exchange dynamics outlined here can allow for a multitude of data and product exchanges.

5.5 Data-Product Exchange

Instead of requesting data in order to offer an external product such as dx-Insights or conducting research, some applications on the App Exchange are services for the data-providers themselves. The dynamics of such a *data product exchange* shares much with the previous spot data exchange of Figure 10, but optionally requires the data producer to exchange DXT in addition to their data, and must ensure the service delivery.

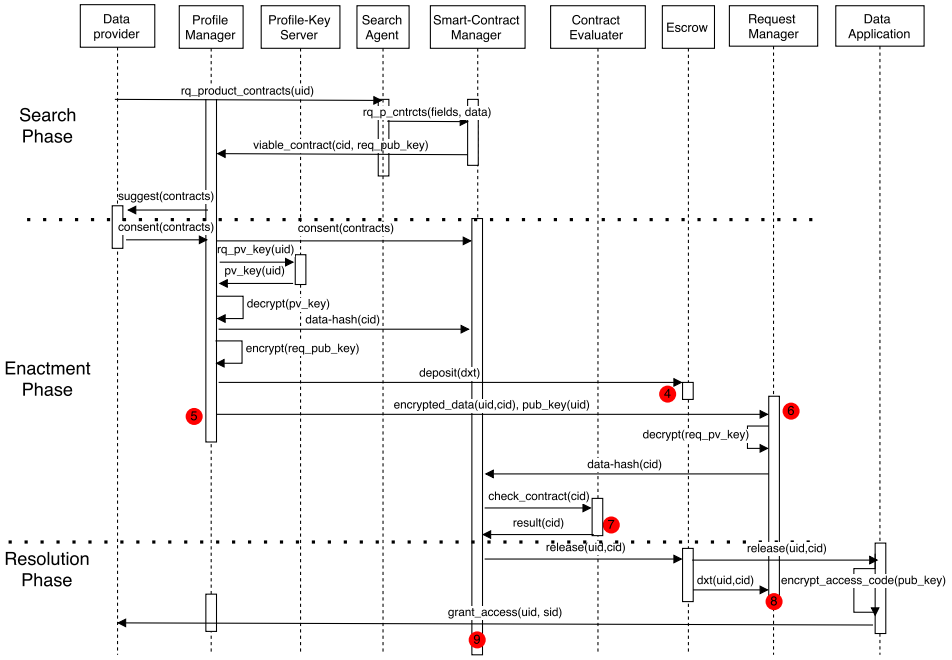

Figure 11: Reflective data-wallet experience in a UML sequence diagram.

Consider in the example of Figure 11 a developer who has designed an application that checks dark web sites to see if a user's emails and passwords have been compromised. They must specify a contract as described above and, as described in Section 4.1 and Table 1, additionally *must* set the category of product from an enumerated list, concisely describe the product in the contract, optionally referencing additional resources through an external link, describe the product delivery mechanism, state the amount of DXT required for the product and can optionally utilize a smart-contract for the delivery method. For example, a product access password check can be required for the token to be released. In the case of Figure 11, the developer uses the provider's public key to encrypt the password and transmit it to the

data provider and optionally requires a hash validation. Note, the requester must still stake a small amount of DXT to facilitate the contract management ecosystem.

After the completion of the data product contract, it is uploaded to the *data product exchange* where it is viewable to data-providers. Note that data products can also be targeted to specific providers using, e.g., qr codes, as described in the next section. For example, providers then searching in the data security category of applications see the application and (after the viability checks discussed above) choose to exchange for the product. The provider's data profile manager enacts the data transfer as previously described. However, there are additional optional steps of placing the specified DXT into escrow and providing a public key for access code encoding. Continuing with our example, the token would be placed in escrow until both the data and product access have been verified (via hash comparison on the smart contract).

5.6 Augmented Experience Exchange

By specifically targeting customers, developers and businesses can augment their users' experience with their products. For example, a music-streaming service can enhance their recommendation algorithm through personal user data. The dynamics of such an *augmented experience exchange* elaborates the previous *data service exchange*, but requires special targeted access rights to the contract.

An augmented experience contract *must* additionally specify the access rights. The simplest way to restrict access is to (reasonably) assume only people expressively invited have access to the contact's unique hash leveraging the *targeted exchange*'s 'security through anonymity'. If necessary, additional measures can be taken, e.g., requiring data providers to supply a passcode. This hash can be communicated to the data provider through qr code, or manual input into the provider's profile manager. Data provider's search agent can then access the targeted contracts they have been made aware of, and after checking the viability of the contract, give providers the option to consent and engage in the contract. The dynamics of this interaction parallel those of the *data service exchange* albeit with additional complexity external to the exchange. The data application developer must ensure that the augmented experience is built upon the provider's data and granted to their associated account.

6 Conclusion

This whitepaper presents the DataWallet ecosystem, a blockchain-technology based data-exchange application that allows data producers to reclaim the data they cre-

ate online from those that use it for their own profit. In giving data producers fine-grained control of their data, DataWallet solves several currently existing problems, not only for the data producers, but also for the data consumers. First, the *silo problem* is overcome by being able to deterministically combine datasets across the walled gardens of the internet. Second, collating these datasets in an intelligent way via smart contracts, resolves the *data quality problem*. Third, *the ethics problem* is resolved by ensuring that every data point was expressively consented to be shared by the people who produced it. Consequently, this whitepaper presented requirements that resolve these three issues for data consumers as well as the injustice of monetizing people's data behind their backs. The paper then derived the static architecture of the DataWallet-ecosystem from these requirements and subsequently described how its dynamic behavior enables disintermediated data-exchange using blockchain based smart-contracts.

The DataWallet requirements show that the ecosystem's main value proposition is to facilitate the secure and consensual exchange of data.[17] In Section 3, the value proposition was hierarchically refined with functional sub-goals that were assigned to the stakeholders of data provider and requester. The former controls the refining DataWallet functions of managing data profiles and managing available contracts. The latter controls the functions for creating data requests and offering data products.

In Section 4, we derived a static architecture for the DataWallet ecosystem from the goal models, which resulted in three coherent sets of distributed components. The data provider group of components are a data-profile manager that coordinates security through a profile-key server, the creation and maintenance of a data profile through external data sources, and a search agent to interface with the smart contract exchanges. The blockchain-technology driven data exchange is comprised of a smart-contract manager that coordinates with a DXT-housing escrow and a contract evaluator. Finally, the data requester group of components centers around a requester-manager component that interfaces with the escrow, the smart-contract manager, and the provider's data applications. In addition to these three groups of components the data-exchange contract template is an essential element of the static DataWallet ecosystem comprising mandatory and optional data fields, and specifications for operationalizing actions for data requests and products.

In Section 5, the running case was used to show several interaction protocols between the architecture components. It outlined the minimum set of eight blockchain operations that allow for immutable dynamic protocol-event traceability. Consequently, we showed four running cases of the dynamic protocol for creating data

[17] The quality goals of the system must be realized in the implementation with architecture and software styles and patterns, which are out of focus for this whitepaper.

profiles, performing a data spot-transaction for the provider consuming additional music, data-product exchange where the data provider checks if he has been hacked, and an augmented experience exchange where the data-wallet requester starts a media campaign. Note that all four protocols show at which points in time the blockchain stores transaction events by executing the eight detected operations.

The resulting system will be developed in stages, as described in Appendix A.3. However, each stage of the development represents a viable product that addresses pressing issues for both data providers and data consumers, and culminates in a fully disintermediated, consent-based, data exchange. The resulting system has the potential to nurture the next generation of personal data products and AI-based customized experiences, finally allowing internet users to make their data work for *them*.

References

[1] C.C. Aggarwal and C.X. Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[2] I. Alqassem. Privacy and security requirements framework for the internet of things (iot). In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 739–741. ACM, 2014.

[3] O. Arias, J. Wurm, K. Hoang, and Y. Jin. Privacy and security in internet of things and wearable devices. *IEEE Transactions on Multi-Scale Computing Systems*, 1(2):99–109, 2015.

[4] R. Baheti and H. Gill. Cyber-physical systems. *The impact of control technology*, 12:161–166, 2011.

[5] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema. Developing applications using model-driven design environments. *Computer*, 39(2):33–40, 2006.

[6] D. Bell. Uml basics: The component diagram. *IBM Global Services*, 2004.

[7] I. Bentov, A. Gabizon, and A. Mizrahi. *Cryptocurrencies Without Proof of Work*, pages 142–157. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[8] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Zanella-Béguelin. Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM Workshop on Programming Languages and*

Analysis for Security, PLAS '16, pages 91–96, New York, NY, USA, 2016. ACM.

[9] D.M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[10] L. Chung, B.A. Nixon, . Yu, and J. Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[11] P. Dai, N. Mahi, J. Earls, and A. Norta. Smart-contract value-transfer protocols on a distributed mobile application platform. *URL: https://qtum. org/uploads/files/cf6d69348ca50dd985b60425ccf282f3. pdf*, 2017.

[12] N. Dalvi and D. Suciu. Management of probabilistic data: Foundations and challenges. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 1–12, New York, NY, USA, 2007. ACM.

[13] Zyskind G., NathanO., and Pentland A. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184, May 2015.

[14] G. Kotonya and I. Sommerville. *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[15] N. Kshetri. Privacy and security issues in cloud computing: The role of institutions and institutional evolution. *Telecommunications Policy*, 37(4):372–386, 2013.

[16] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 17–30, New York, NY, USA, 2016. ACM.

[17] A. Norta. *Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations*, pages 3–17. Springer International Publishing, Cham, 2015.

[18] A. Norta. *Establishing Distributed Governance Infrastructures for Enacting Cross-Organization Collaborations*, pages 24–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[19] A. Norta. *Designing a Smart-Contract Application Layer for Transacting Decentralized Autonomous Organizations*, pages 595–604. Springer Singapore, Singapore, 2017.

[20] A. Norta, P. Grefen, and N.C Narendra. A reference architecture for managing dynamic inter-organizational business processes. *Data & Knowledge Engineering*, 91(0):52 – 89, 2014.

[21] A. Norta, L. Ma, Y. Duan, A. Rull, M. Kõlvart, and K. Taveter. eContractual choreography-language properties towards cross-organizational business collaboration. *Journal of Internet Services and Applications*, 6(1):1–23, 2015.

[22] A. Norta, A. B. Othman, and K. Taveter. Conflict-resolution lifecycles for governed decentralized autonomous organization collaboration. In *Proceedings of the 2015 2Nd International Conference on Electronic Governance and Open Society: Challenges in Eurasia*, EGOSE '15, pages 244–257, New York, NY, USA, 2015. ACM.

[23] A. Norta, A. Vedeshin, H. Rand, S. Tobies, A. Rull, M. Poola, and T. Rull. Self-aware agent-supported contract management on blockchains for legal accountability. *URL: http://whitepaper. agrello. org/Agrello_Self-Aware_ Whitepaper. pdf*, 2017.

[24] B.S. Panikkar, S. Nair, P. Brody, and V. Pureswaran. ADEPT: An IoT Practitioner Perspective, 2014.

[25] S. Prince, P. Li, Y. Fu, U. Mohammed, and J. Elder. Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):144–157, Jan 2012.

[26] R. Ragunathan, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.

[27] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004.

[28] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164, 2015.

[29] L. Sterling and K. Taveter. *The art of agent-oriented modeling*. MIT Press, 2009.

[30] A. S. Tanenbaum and M. Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[31] M. Vukolić. *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*, pages 112–125. Springer International Publishing, Cham, 2016.

[32] R.M. Weber and B.D. Horn. Breaking bad security vulnerabilities. *Journal of Financial Service Professionals*, 71(1):50–54, 2017.

[33] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

Appendix

A Ecosystem Development Roadmap

The following describes the roadmap for development of the transparent, opt-in data marketplace. There are three major milestone versions of the marketplace ecosystem. Our current implementation—the v.10, the smart-contract based ecosystem v.20, and the decentralized stored v3.0.

It is important to note that the current implementation timeline is built upon a number of assumptions. There are both technological assumptions as well as community demand/business assumptions. The community demand assumptions, e.g., that there is a substantial demand for a fully decentralized solution à la Milestone 3, have informed our long-term development goals. However, these plans only reflect our research and experience in the current ecosystem. Subsequent research and community engagement may point to more desirable data storing, managing, and exchanging solutions. The technological assumptions include the emergence of scalable implementations of some currently unscalable solutions and open research questions. We have striven to factor in this uncertainty into our milestone progression, but the emergence of necessary technologies can impact the proposed timeline. Exogenous circumstances may therefore impact this roadmap, but significant deviations will be clearly communicated and explained to the community.

A.1 v1.0—Current Implementation

In the current implementation of the Data-Exchange, data providers manage their profiles through a mobile device. The data profiles are collated into a central database and all data sales are mediated by dx-Insights.

A.1.1 Data-Requester System

There is no open Data Product Marketplace. Companies interested in accessing providers' data do so through dx-Insights.

A.1.2 Data-Provider System

The Data-Profile Manager is the DataWallet mobile app, which is available for iOS and android. It communicates with a DataWallet-hosted Key-Server and a DataWallet-hosted centralized database. Potential data requests are posted directly in mobile application.

A.1.3 Data-Exchange System

All data exchanges originate from dx-Insights and posted in the mobile application.

A.2 v2.0—Smart-Contract Based

The Smart-contract based ecosystem embeds the promise of DataWallet—transparent, opt-in data exchanges—directly into the architecture. At this stage the fidelity of the system is not dependent upon a trusted party (DataWallet). The assurances of the system are structural and verifyable by interested members of the community. All PII is encrypted with a private key that only the data providers themselves posses. All data-transactions therefore necessitate the expressive consent of data providers. The Data Product Marketplace is open such that Data Providers are not limited to interacting with dx-Insights; Data Requesters can publish products to the marketplace. The Data Exchange is open—all data requests are executed by smart-contracts on the blockchain.

A.2.1 Data-Requester System

Data Product Marketplace is open. Requesters can create data requests through the online interface or programmatically with the API.

A.2.2 Data-Provider System

All data stored on DataWallet-hosted centralized database is encrypted with a private key that only the data providers themselves posses. Exchanges are searchable from the provider mobile app.

A.2.3 Data-Exchange System

The Data Exchange is open—all data requests are executed by smart-contracts on the blockchain coordinating the provider's data profile and the requester's Request-Manager.

A.3 v3.0—Fully Decentralized Data Profiles

The Ecosystem with fully decentralized data profiles is the ultimate realization of the self-sovereign wallet. Interested Providers can store their profiles on systems of their choosing and interact directly with the Data Exchange.

A.3.1 Data-Provider System

Data-Profiles can be created locally using open and auditable code. Data Profile structure and Search agents are similarly open sourced and customizable (as long as they provide the necessary interface to smart-contract exchange).