information

*Article*

# Making Sense of Solid for Data Governance and GDPR

**Harshvardhan J. Pandit** [1,2]

[1] ADAPT SFI Research Centre, D02 FX65 Dublin, Ireland; harshvardhan.pandit@adaptcentre.ie
[2] School of Computing, Dublin City University, D09 PX21 Dublin, Ireland

**Abstract:** Solid is a new radical paradigm based on decentralising control of data from central organisations to individuals that seeks to empower individuals to have active control of who and how their data is being used. In order to realise this vision, the use-cases and implementations of Solid also require us to be consistent with the relevant privacy and data protection regulations such as the GDPR. However, to do so first requires a prior understanding of all actors, roles, and processes involved in a use-case, which then need to be aligned with GDPR's concepts to identify relevant obligations, and then investigate their compliance. To assist with this process, we describe Solid as a variation of 'cloud technology' and adapt the existing standardised terminologies and paradigms from ISO/IEC standards. We then investigate the applicability of GDPR's requirements to Solid-based implementations, along with an exploration of how existing issues arising from GDPR enforcement also apply to Solid. Finally, we outline the path forward through specific extensions to Solid's specifications that mitigate known issues and enable the realisation of its benefits.

**Keywords:** personal data stores; personal information management systems; security; privacy; data protection; ISO; semantic web



**Citation:** Pandit, H.J. Making Sense of Solid for Data Governance and GDPR. *Information* **2023**, *14*, 114. https://doi.org/10.3390/info14020114

Academic Editor: Georgios Kambourakis

Received: 18 November 2022
Revised: 2 February 2023
Accepted: 7 February 2023
Published: 12 February 2023



## 1. Introduction

Solid [1] is an ongoing effort to decentralise the control of data by moving its storage away from centralised systems and into *Pods* that are controlled by individuals [2]. The Solid specifications [3] define the implementation of Pods as an architecture containing identity management, access control, and communication. Users are provided with access control mechanisms to decide the storage, modification, and use of data in Pods by other users or applications (*apps*). By controlling access to their data within Pods, individuals also gain the ability to (re-)use it elsewhere for competing services or for other features—which is not possible through conventional methods where data is locked and controlled by service providers.

Solid was initiated in 2016 and is led by WWW-inventor Tim Berners-Lee [2]. It has gained interest due to its radical approach to move away from centralisationand lock-ins and lack of privacy. Solid represents a realisation of the data sovereignty philosophy where individuals 'control' their data and how it is used. Since this involves the use of personal data, the existing laws regarding data, privacy, and data protection, such as EU's General Data Protection Regulation (GDPR) [4] also apply to Solid. However, the radical deviation of Solid's implementations from conventional methods where data is collected and centrally retained by companies has resulted in uncertainty regarding how laws such as GDPR should be interpreted in light of the new use-cases, specifically regarding their sufficiency and potential for non-beneficial implications to centralised service providers [5–7].

To further this discussion, we ask the question: "*What assumptions from GDPR (and their interpretations) are still valid and applicable for a Solid user?*" In order to answer this, first, the use of Solid must be understood as defined by Solid's own specifications. This requires identifying what a Solid Pod is, how it is created, how it is obtained and used by individuals, and how apps interact with the data stored in it. This also requires understanding the

possible variations that may emerge as a result of applying existing practices for service provision by market actors, or as part of Solid's organic push towards re-implementing conventional models. Through these, we investigate who will implement and provide the various resources required by Solid and apps, how will users be involved in these processes, and who retains what degree of control. After establishing the basics of who does what in an implementation, the use-case must be interpreted through the investigative lens of GDPR, where the roles of controllers, processors, data subjects, and third parties must be identified and applied to the use-case. Then, the obligations associated with each entity must be assessed for applicability and fulfilment in order to determine compliance. Finally, for ensuring progress and benefits, we need to explore practical implications arising from Solid implementations by exploring how existing issues also apply to Solid-based use-cases, and identify specific paths for improvements and mitigations through developments involving Solid itself.

To enable these processes for Solid, we explore the primary questions as follows. To assist with understanding Solid, we summarise its specifications and relevant work in Section 2 with references for further information:

1. How to describe Solid Pods using existing 'cloud' terminologies (Section 3), and distinguishing implementation of functionalities? (Section 4)
2. What use-cases are possible from variations in resources and actors? (Section 5)
3. How does GDPR apply to an implementation of a Solid Pod? (Section 6)
4. What existing issues regarding privacy and data protection are also applicable to Solid Pods? What are their potential impacts? (Section 7)
5. What avenues are feasible for mitigating known issues through systemic extensions of Solid specifications? (Section 8)

The outcomes of this work are intended to benefit Solid stakeholders in understanding and applying GDPR (and other legal) concepts to their use-cases and thereby inform future technical and legal developments in the use of Solid. The intention of this article is to also highlight the risks involved in the use of Solid. There is no necessity or obligation for the Solid community to fix identified issues, especially where they are broader and universal in the context of web and applications. However, we hope that in highlighting them, Solid's vision of using decentralisation and machine-readable information to empower users in controlling their data can benefit from the identification of potential paths to mitigate known issues and innovate on better privacy mechanisms through developments within, using, and led by Solid itself.

## 2. Background Information and Relevant Work Regarding Solid

### *2.1. What Is Solid?*

Solid describes itself as a 'specification' for decentralised '*data stores*' called '*Pods*' that act as '*secure personal web servers for data*' with controls for accessing and using stored data. The Solid protocols [8] define the use of web-based technologies for creating and using *applications* that act on data within Pods. They also define functionalities related to identity [9], access control authorisations [10,11], handling of requests and notifications, governance of app requests and data access [12], and security considerations for implementations. The Solid project's website [1] showcases existing applications and development tools.

While Solid uses existing cloud technologies, such as servers, it differs by defining an additional layer or framework based on machine-readable policies to control usage in terms of identity, data storage, access control, and user management with the key differentiators being the promotion of decentralisation through user control over data. Thus, Solid provides specifications to re-imagine cloud technologies as a 'personal data store'.

In Solid, users and applications are represented as *Agents* whose identity is represented by a URL that also provides information (e.g., profile, metadata) and is used for authorisations. Apps request access using pre-defined methods, to which users can grant access and also revoke it later. Access to data within Solid specifications is determined based on: (1) data in question; (2) operation to be performed; and (3) entity requesting

access—where the entity can be a process, an app, or a user. These are expressed using *Access Request* which users make decisions that are stored as *Access Authorisations* [12]. Details of access are provided to requestors (e.g., apps) as *Access Grants* with specifics of what the access entails. The data within the scope of access is specified using *Data Grants*. The *Access Need* concept specifies information on the necessity of information (required or optional), scenario (personal or shared access), and description (human-intended text), with possible grouping into sets for collective reference and application [12]. We summarise this current model as the tuple: $\{data, operation, necessity, justification, agent\}$.

### 2.2. Known Implementations and Use-Cases

Solid as a specification has implementations [1] that are open and community-led, as well as closed-sourced commercial variants. These are utilised by *Pod Providers* to provision services with varying levels of freedoms (e.g., control over infrastructure) where either they or the users can choose providers for domains and servers, with a choice of locations (as jurisdictions). In addition, Solid can also be self-hosted, e.g., by manually installing it on a user-controlled server.

Notably, the Flemish government in Belgium has embarked on an ambitious project whereby all citizens will be provided a Solid Pod for storage and control over their government-issued documents which apps can request access to based on the user's consent only after establishing legal agreements with the government-established Data Utility Company outlining permitted purposes and processing [13,14]. An earlier article from involved researchers outlines further similar use-cases for citizens [15,16].

### 2.3. State of the Art Regarding Analysis, Applications, and Explorations of Solid

Researchers have explored the extension of Solid specifications to support policy management and its use in exercising more complex constraints over the use of consent and data in Pods [17–20], as well as using them to control the subsequent use of data beyond access [21]. Further extensions of Solid Pods have explored mechanisms through which possession of data (e.g., educational degree) can be verifiably demonstrated without sharing it [22], changing infrastructure to a local environment of a smartphone [23], and moving beyond documents to using 'knowledge-graphs' to achieve richer data utility [24].

Solid Pods have been demonstrated to implement GDPR's Right to Data Portability [25] and Right of Access [26]. Its legal considerations have been explored for the applicability of GDPR's obligations [27] to provide relevant questions to ask regarding Solid implementations. Similar explorations have also explored the purported value derived from decentralisation, its legality under GDPR, and the existence of issues [6,7], including the issue of a data subject being a controller [7]. There have also been security-focused investigations that explored the relevance of GDPR's obligations in implementations of Solid [28] that emphasise the need for further investigations of this topic. Similarly, the European Data Protection Supervisor (EDPS) has outlined Solid [29] amongst other 'Personal Information Management Systems' as a topic of interest regarding GDPR, with specific emphasis on risks, consent management, transparency and traceability, exercising of rights, data accuracy, data portability and interoperability, and security.

These works reflect a growing interest in understanding the use of Solid and its compatibility with GDPR. This article fills gaps in three important areas. First, the potential breadth and variety of arrangements that Solid could be used in necessitate a systematic method to describe use-cases before investigation. Second, exploring GDPR's relevance to implementations of Solid's specifications. Third, identifying which current risks and issues regarding GDPR compliance apply to Solid and identifying ways to mitigate them through further development of Solid's specifications.

## 3. Solid as 'Cloud Technology'

### *3.1. Motivation for Explicitly Defining Solid as a Cloud Technology*

ISO/IEC 17788:2014 Information technology—Cloud computing—Overview and vocabulary [30] defines *cloud computing* as a "Paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand" where resources can be servers, networks, software, applications, storage, etc. It defines *cloud services* as "One or more capabilities offered via cloud computing invoked using a defined interface". This definition suits implementations where a Solid Pod is a form of cloud service that enables applications and (cloud) services to utilise and/or store data. Since the Solid specifications do not place limits on how storage and computing are built and provided, we can apply the full extent of existing cloud infrastructure and provision methods to describe possible implementations.

By defining Solid as a Cloud technology, we benefit from identifying and applying relevant cloud terminologies, standards, guidelines, legal requirements, and obligations, and utilise existing domain expertise. For example, using ISO 35.210 Cloud Computing [31] standards for security, handling of sensitive data, interoperability, portability, policy management, and data governance; or ENISA's Cloud Computing Risk Assessment [32] as cybersecurity guidelines; or GDPR guidance on Controllers and Processors [33] that outlines requirements and responsibilities for use of cloud-based technologies through market providers.

Analysing use-cases first requires accurately representing the specifics in terms of actors, processes, and information flows. For Solid, we reuse the existing Cloud technology concepts adapted from ISO/IEC 22123-1:2021 Information technology—Cloud computing—Part 1: Vocabulary [34] to provide a framework through which implementations of Solid as cloud technologies can be documented for common understanding. For this exercise, we first checked whether each term had relevance to Solid and the topic of this paper, and then rephrased their definitions to specify relevant descriptions of information associated with the use of Solid Pods. The findings are summarised as a collection of entities in Figure 1.

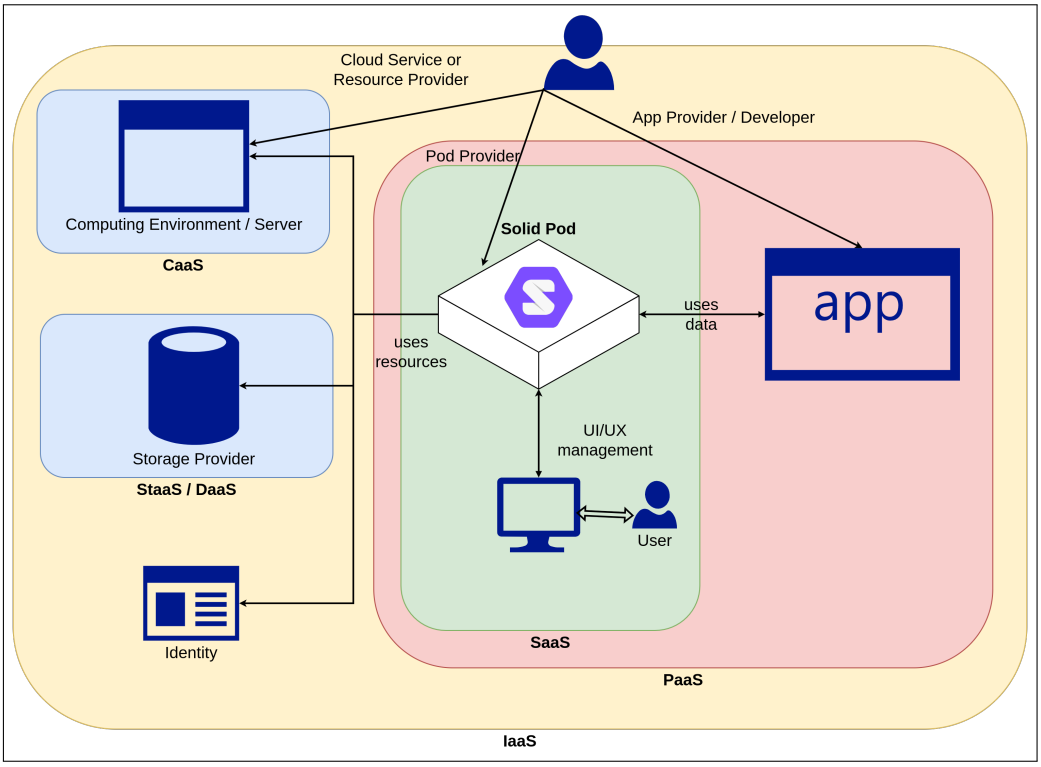


**Figure 1.** Simplified representation of applying Cloud paradigms to provisioning Solid Pods.

### *3.2. Actors*

Actors refer to entities that have specific roles within the Solid ecosystem. Currently, Solid specifications only refer to *Agents*, *Social Agents*, and *Applications*. We use the cloud-based terms from ISO/IEC terminology since they have standardised definitions and interpretations and also have relevance in legal compliance investigations.

- *Provider*—entity that makes available Pods or relevant resources (storage, computing, identity, application, domain, etc.), with prefixes indicating contextual concepts such as—*pod provider* for the entity that provides Pods, *app provider* for providing an app, *storage provider* for providing storage, and so on. Note that this concept only refers to the *provision*, which is separate from *development*.
- *Customer*—entity that has a 'business relationship' for the purpose of using Pods or its relevant resources. This includes purchasing, leasing, subscribing, or establishing any form of contract or agreement, with or without monetary transactions. Customers typically will have a direct relationship with a provider, which may be facilitated by one or more *broker* entities. For example, a *pod broker* is an entity that negotiates a relationship between a *customer* and a *pod provider*.
- *Developer*—entity with the responsibility for designing, developing, testing, and maintaining implementations of Solid Pods, applications, or relevant resources. Similar to *provider*, this concept can also be prefixed to indicate contextual roles, such as—*pod developer* for the entity that is responsible for the development of Pods (as platforms, software, or apps), *app developer* for apps, and so on. Providers and developers are important concepts to distinguish since they determine accountability and responsibility for resources, and may have obligations depending on the extent of their role in provision and development, respectively.
- *User*—natural person that uses Pods or relevant resources. Note that the ISO/IEC definition here also includes what Solid considers *Agents* as users, e.g., devices and applications. We make the distinction between *User* and *Agent* so as to distinguish between human and machine-based agencies—which is necessary to later analyse processes such as consent. We also distinguish between customer and user since they may not be the same—for example, when a Pod customer is a company that provisions Pods from a provider, customises it, and provides it to (end-)users. In this case, the company is a provider for the end-users but a customer for Pod providers.
- *Data Subject*—the individual that is the *subject* of the data within a Pod, and who may be different from the user. While ISO/IEC uses the terms PII Principal and Data Principal, we use data subject for its accuracy in this context as well as for consistency with GDPR investigations.

Solid specifications use the term *Owner* with the definition as: "An owner is a person or a social entity that is considered to have the rights and responsibilities of a data storage". However, this term is problematic in that it may get interpreted as referring to *data ownership* which is a specific legally relevant concept and may produce unintended applications in terms of copyright and intellectual property rights. It also creates issues through the use of *responsibilities* which are based on legal obligations and rights, and which do not necessarily fall upon the individuals. To avoid such implications and to restrict interpretations to well-established common practices, standards, and legal norms—we only use terms from ISO and GDPR.

### *3.3. Functionalities*

*Functionality* refers to the capability or feature exhibited by a Solid Pod or its related resources such as storage, computing, identity, or applications. Functionalities can be categorised based on capabilities, interoperability, and portability of data and applications. *Pod capabilities* is the classification of capabilities that an implementation of a Pod supports in terms of letting customers or users manage them. For example, support for adding

additional storage, a computing or execution environment, or pre-configured applications. Capabilities can be described in relation to three broad concepts:

- *Application*—how users manage applications. This relates to how users discover applications, interact with requests for data use, 'install' apps, and perform configuration or other management and governance-related tasks.
- *Platform*—whether users can deploy, manage, and run processes. This relates to whether the relevant tasks necessary for a Pod—such as identity verification, authorisation, policy management, or anything that requires computing or execution can be managed by the users, or is provided via pre-configured environments, or can also be modified by users and/or applications for supporting other functionalities.
- *Infrastructure*—whether users can provision and control resources related to a Pod or an app, such as storage, computing, networks, etc.

*Pod service category* is the classification of services supported by a Pod based on available capabilities. This defines how Pods are implemented using physical and virtual resources, e.g., as Applications, Platforms, and Infrastructure, and how they are provided to customers or users. This section does not list the full extent of how cloud technologies and services can be provisioned in terms of 'anything as a service' (XaaS), but only considers the broad categories necessary to describe use-cases with concise and complete information.

- *Infrastructure as a Service (IaaS)*—Users can control infrastructure (e.g., storage space, computing servers) directly, with potential limitations to use specific providers or offered choices, e.g., operating systems, networking components (e.g., web server and firewalls), data stores (e.g., databases or triple-stores), and virtualisation capabilities. Solid Pods can be readily deployed as self-controlled servers in an IaaS environment (https://solidproject.org/self-hosting/css access on 1 November 2022).
- *Platform as a Service (PaaS)*—Users of a Pod are given a 'platform' through which they exercise their control over apps and resources without explicitly dealing with the underlying infrastructure. Platforms determine how users interact with their Pods, data, and applications, and are not currently defined by the Solid specifications. A platform can be a dedicated development over a Solid server instance or be an extension of existing platforms to support Solid as an additional protocol. Examples of platforms as both dedicated services (e.g., Inrupt Pod Spaces (https://start.inrupt.com/ access on 1 November 2022) and extensions (e.g., NextCloud [35] are available in Solid's documentation.
- *Software as a Service (SaaS)*—Users use Pods via controlled interfaces (e.g., web browsers and smartphone apps) and do not directly control resources. See 'Pod Providers' on Solid's website [1] for examples.
- *Compute as a Service (CaaS)*—Providers provide dedicated computing environments controlled by the user for (server or serverless) process execution, e.g., to process their own data or to enforce data localisation within controlled environments.
- *Storage as a Service (STaaS)*—This is a hypothetical extension where the Pod exposes different forms of storage as a service. For example, a SQL database, semantic-web triple-stores, binary or blob storages, or dedicated media storage services such as for photos and videos.
- *Data as a Service (DaaS)*—Another hypothetical extension as a service between data and apps that provides data-value and data-utility. For example, when companies and applications do not need to centrally collect, store, and manage data, but instead utilise the availability of data within a Solid Pod directly by using it on demand. These may include operations over (raw) data, invoke specific queries to obtain answers, be limited to only data collection or storage, or also involve storing ephemeral and persistent outputs from processes.

The ISO/IEC cloud standards define *Portability* as "ability to migrate an application from one cloud service to another". Applied to Solid, portability refers to the extent to which data and applications can be migrated (i.e., moved) to another Pod, for example as:

- *Data portability*—data can be migrated or moved outside of the Pod;
- *Application portability*—apps can be moved between Pods;
- *Pods portability*—Pods can be moved between providers;
- *Data Synctactic portability*—data is ported using well-defined data formats;
- *Data Semantic portability*—data is ported using defined semantics and data models;
- *Data Policy portability*—data is ported while complying with relevant policies;
- *Application Synctactic portability*—apps are ported by utilising well-defined formats;
- *Application Instruction portability*—app instructions (i.e., executable code) can be ported;
- *Application Metadata portability*—app metadata, such as profiles or established permissions and authorisations, can be ported to another Pod;
- *Application Behaviour portability*—apps are ported without changes in functionality;
- *Application Policy portability*—apps are ported while complying with relevant policies.

The ISO/IEC cloud standards define *Interoperability* as "ability of two or more systems or applications to exchange information and to mutually use the information that has been exchanged". Applied to Solid Pods, interoperability refers to how Pods and applications can exchange and mutually use data. In this, portability only refers to the ability to migrate or move data or apps outside of a Pod, while interoperability also refers to the usefulness of that data or app in the new Pod. For interoperability, the following interpretations for Solid are provided based on ISO/IEC-defined forms of interoperability:

- *Pods interoperability*—Data and apps in a Pod are interoperable with other Pods (same or different provider), and Pods support import/export features to achieve this.
- *Application interoperability*—Data is interoperable across different apps (same or different provider), and apps support import/export features to achieve this.
- *Data interoperability*—Data is interoperable with other data from the same or different provider. This means both data providers and/or consumers support the same (or a set of) data formats or schemas to achieve interoperability.
- *Synctactic interoperability*—Data and apps use interoperable formats that are understood by both providers and consumers (e.g., CSV, JSON) to achieve interoperability;
- *Semantic interoperability*—Data and apps use well-defined schemas, ontologies, or data models that are understood by both providers and consumers.
- *Behavioural interoperability*—Interoperability does not detriment functionality.
- *Policy interoperability*—Interoperability takes place while maintaining compliance with legal, organisational, Pod, service, or user policies.

### *3.4. Data Categories*

While Solid Pods are defined as a data storage service for individuals to store and control their (personal) data, they can also contain several other categories relevant to the functioning of Pods, applications, and services—specified by ISO/IEC terminology as:

- *Personal Data*—category for data associated with an individual (where exact definition depends on jurisdiction). Non-personal data is data that is not associated with an individual. Note that this concept is broader than PII which relates to *identifiability* of the data with an individual—for example, removal of identifiers may suffice to make the data non-PII but it would still be personal data, whereas its (complete) anonymisation results in non-personal data.
- *Customer Data* or *user data* to refer to data being under the (legal, contractual, or other forms of) control by customers and users, respectively. If customers, users, and data subjects are distinct—their respective data categories will also be distinct.
- *Derived Data*—category for data produced as a result of interactions with services and applications. This can be logs such as those associated with data access, usage, or processes. It also includes data associated with authorisations, e.g., produced as a result of managing permissions for an app.

- *Provider Data*—category for data under the control of providers—such as configurations for provisioned Pods, resources, or applications, or logs relevant to operational processes such as identities or used to calculate charges for use of resources.
- *Account Data*—information about accounts regarding Pods, resources, and apps.
- *Protected Data*—data needed to be protected by provider, user, or application.
- *Publicly Accessible Data*–here 'public access' only refers to access mode for data, and does not constitute permission to *use* or further *disseminate* it), and so publicly does not imply visibility or accessibility, but specifies access within contextual boundaries.

In these definitions, while *sensitive* information and data are not explicitly defined, they are covered through other ISO/IEC standards related to data security and governance, such as ISO/IEC 19944-1:2020 Cloud computing and distributed platforms—Data flow, data categories, and data use Part 1: Fundamentals [36] which refers to sensitive data categories associated with children, finance, health, and medicine—and provides guidance on how these are to be managed within cloud-based data flows. Through these, we understand the necessity to categorise data based on relevance to operations and actors, as well as sensitivity, and to use these categorisations within the relevant processes associated with security and oversight. Solid specifications currently do not explicitly define such categorisations, though they do support the declaration of categories for specific data within a Pod.

### *3.5. Contracts and Agreements*

The use of Solid Pods, related resources, applications, and other services is governed through agreements and contracts between providers and customers or users. Here, *agreement* is a document outlining an arrangement or understanding between entities, whereas a *contract* is a specific formal agreement between entities that is intended to be legally enforceable and is governed by relevant jurisdictional requirements and obligations regarding validity, enforcement, liabilities, and remedies. ISO/IEC terminology describes *service agreement* as an agreement between a provider and a customer regarding the provision of specific services. It also defines *service level agreements* (SLA) between providers, customers, and suppliers that identifies the services, how they are to be provided, their targets, commitment to specific objectives and their qualitative characteristics, and which can be part of another contract or agreement. The distinction between the two relates to the number of details and specifics in terms of what the service entails and what should be provided and/or expected regarding its use.

Applied for Solid, agreements can be associated with Pods, resources (e.g., storage, computing, identity), Apps, or Users. These can relate to specific functionalities (e.g., Pod storage space, computing in hours, resources for a specific app), and can be individually managed by customers or be part of a common contract governing terms of use for Pods and apps. In addition to these, the specific contracts and consenting agreements established by users are separate concepts that do not feature within ISO/IEC cloud-based terminologies, but are instead covered in separate standards associated with privacy—such as ISO/IEC 29184:2020 Information technology—Online privacy notices and consent [37].

The Solid specifications currently do not support or specify any form of agreement regarding the provisioning of Pods, resources, or Apps. However, Pod service agreements are mentioned externally—for example as part of the information on where to find a Pod provider. For interactions between Apps and users, the specifications only refer to 'authorisation' that is recorded and stored within the Pod.

## 4. Functionality Layers in a Solid Pod

In this section, we explore functionalities in terms of how data is stored, retrieved, and used through a Pod. In this, the relevance of cloud technologies is apparent in that each functionality can be implemented using a different and distinct set of technologies, and can be associated with a different cloud-based actor. These also relate to different capa-

bility types—such as where functionalities are implemented as infrastructures, platforms, or services.

The functionalities are represented as *layers* based on the systemic influence they have on each other where each layer depends on the implementation of the layers below it to define its own functionality. The layers are intended to assist in the conceptual description of Solid's implementations and provide a common basis for the co-ordination of efforts associated with each 'layer'. This is based on the OSI model consisting of seven layers describing the communication of information between two systems [38].

The layers also represent a 'separation of concerns', and act as a framework for indicating the scope of developments by indicating the affected layers. For example, a data storage solution has immediate relevance for the data retrieval layer as a dependency but is not as strongly coupled with the interface layer. This separation of functionalities as layers also assists later in the establishment of processes regarding accountability—such as notices and consent, security measures, and identifying the role of entities in determining how data is being collected and used based on which layers they control and what limitations are imposed. In addition, the layers enable accurate analysis of cases where Pods may be provisioned with specific limitations on some of these functionalities which may result in restrictions on the portability and interoperability of data and apps.

We identified the following layers (see Figure 2) by distinguishing between implementations of Solid Pods and applications in terms of interactions with data: *data storage* as the bottom layer given Solid's reliance on data storage as a central concept, followed by *data retrieval* to retrieve stored data, *computing* to (optionally) perform computations on it, *access control* to control retrieval of information, *communication* to share information between entities and resources—and finally *interface* for interactions and management processes by *actors*. Orthogonal to these are layers associated with *logging*, *policies*, *security*, and *identity* which have relevance to and are influenced from implementations of each layer.

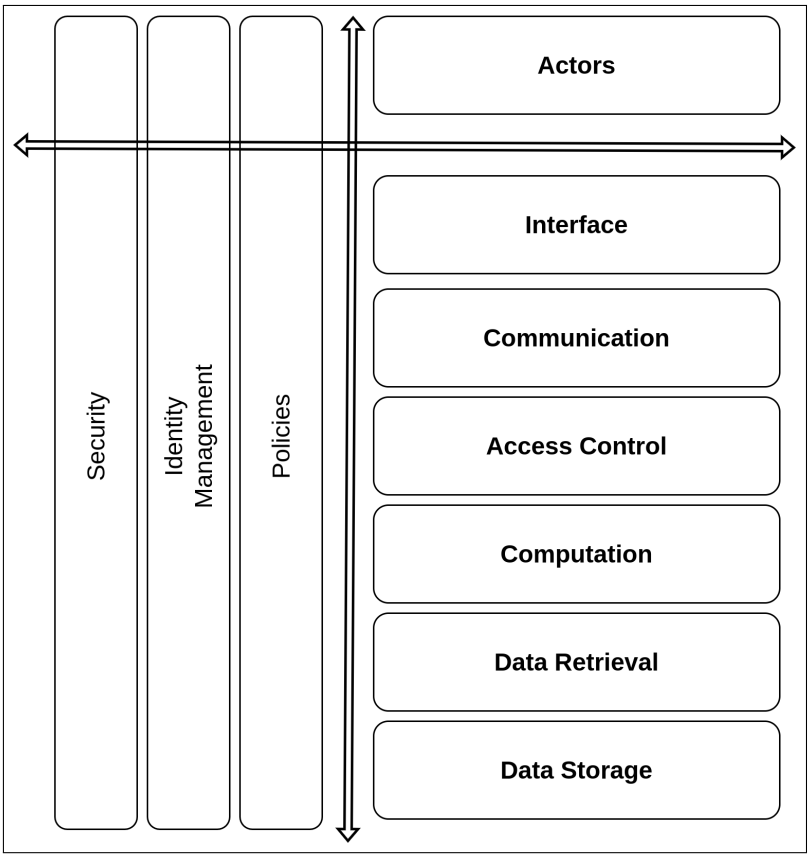


**Figure 2.** Representing Solid Pods as a collection of inter-related layers based on functionalities.

### *4.1. Data Storage Layer*

#### 4.1.1. Physical and Virtual Views

The Data Storage layer refers to the physical and virtual arrangement of data within a Pod or its aligned resources. Here, physical refers to the actual data storage mechanism, e.g., disks, bytes, or file systems, and virtual refers to non-physical or logical arrangements that provide alternative ways to represent and arrange data as an abstraction over underlying data mechanisms. An example of this is how a specific URL representing a path to a resource may or may not correspond to the actual location of that resource within a server. Another example is the encapsulation of information, such as a URL with a fragment identifier representing a specific element within the HTML document.

Defining virtual or logical views over physical resources enables data to be provided and stored using semantics or schemas—such as within databases. This enables the same physical data resource to be represented and used as different virtual resources, which opens possibilities for content negotiation and richer use of data within applications based on synctactic and semantic interoperability. For example, a collection of bytes on a disk can be interpreted as byte streams, documents, or semantic data (e.g., RDF triples), or they can be converted on-demand to requested formats and schemas.

The governance of access modes is based on how data can be stored, read, used, queried, and controlled—which affects how data and access are communicated. The current Solid specifications specify URLs for accessing data, which is similar to how documents are stored using file paths. Virtualisation can be used to expand what the paths point towards, e.g., through the webserver to map different paths to different data or retrieval methods. However, there are strong arguments to take advantage of virtualisation to promote greater and innovative use of data beyond fixed documents and formats [24].

#### 4.1.2. Cataloguing

The data storage layer also determines how data can be grouped together, such as in the form of folders, catalogues, registries, or collections. Such arrangements are crucial to create logical views that enable easier storage and access to related data within the same context. For example, apps such as those for contact books and photo albums rely on such contextual arrangements [24]. Limitations on the ability to have more than one grouping for the same data also place limitations on how it can be reused, as well as creates issues regarding privacy due to lack of separation capabilities [24]. Solid specifications currently specify *Data Instance* as a specific and unique instance of data that conforms to a *Shape Tree* (a specific schema), which is stored within a *Data Registry*. However, these do not resolve issues of limitations on data reuse since such shapes can be different for each user and app without a way to support interoperability between data providers and data consumers.

#### 4.1.3. Data Partitioning and Mirroring

Data partitioning is the separation of data, typically undertaken for considerations of efficiency, performance, control, or contextual needs. For example, data more likely to be requested from specific locations are stored closer to those locations to improve retrieval speeds. Another example is where sensitive data is stored in a separate physical or virtual location which can be supplemented with dedicated security measures. Data mirroring is the duplication of data—which can be performed separately or alongside partitioning for the same reasons of increased availability and efficiency, as well as to reduce costs arising from network egress. Both of these are common aims when using cloud delivery networks (CDNs), typically provided by a third party.

For Solid, a Pod is considered a holistic storage managed under a single identifier (i.e., an IRI for the Pod). Internally, it can be split into separate instances, each of which can be a virtual resource attached to a Pod, or be separate Pods themselves. While the specifications themselves do not explore this topic in detail, we later reuse these concepts in terms of how apps and users manage data, their usefulness towards having additional security for

sensitive information, and using partitioning and mirroring to ensure (some) data is always stored or available within a jurisdiction.

### *4.2. Data Retrieval Layer*

#### 4.2.1. Retrieval Forms

The retrieval of data is based on the data storage layer in terms of physical arrangements—such as retrieving data as bytes, blobs, or streams; and virtual arrangements—such as retrieving using specific formats or schemas. In addition, the data storage layer also affects retrieval by defining how data can be accessed in terms of identifiers, paths, and arrangements. The implementation of the data storage layer, therefore, affects the retrieval of data in terms of enabling or disabling the abilities through which users and apps can access that data. For example, if data can only be retrieved as binary documents and using paths to specify targets—apps must either be developed to use this arrangement or require additional efforts to convert it into required arrangements.

Data retrieval can be implemented directly by the Pod (i.e., as operating software), or by intermediaries (e.g., apps, services, manually)—where the support for specific forms and requirements for retrieval determines what capabilities are possible and feasible for use of data. For example, if retrieval supports entire documents rather than individual records or partial data, apps such as contact books will request access to all possible data in order to retrieve names and contact numbers [24]. In such cases, where data storage may not support virtualisation by itself, retrieval can utilise intermediaries to collect and strip the data of unwanted elements to only return the requested information.

Solid specifications currently specify retrieval using IRI or URLs indicating the path of data, which is then handled by the Pod server to retrieve corresponding documents or *data instances*. Other forms of retrieval can also be added to this, for example as APIs over URLs to specify metadata such as requested formats, queries, or other pertinent information. Well-defined URLs can also be used to provide a common method to obtain data based on categories, formats, schemas, etc. For example, */data/contacts* as a common way to retrieve contacts irrespective of how they are actually stored within the Pod, and using parameters to specify formats or limit the number of records.

#### 4.2.2. Querying Data

Retrieval can also be based on queries—where specific criteria and constraints are expressed with which to identify and retrieve selective information. Queries can also provide a solution to the selective information problem described above, and also enable clearer access to information by specifying only relevant information to be retrieved. Further, queries can be used to create ad-hoc documents, or graphs, in requested formats—thereby also assisting in issues related to the interoperability of information. An additional utilisation of queries, as a broad concept and form of information retrieval, is to specify derivations (e.g., conversion of values) and inferences (e.g., derive statistical analytics) over data within the Pod. This allows apps to request information without accessing the data in question since the queries would only return the requested (generated) information.

However, compared to path-based document retrieval, queries are more expensive in terms of computation, more difficult to assess in terms of permission and privacy, and require more logic to be available for handling calls and results. Solid specifications currently do not specify any form of querying or mention support for its development, though approaches have been proposed to move Solid's storage and retrieval mechanisms from document-based to knowledge-graph [24].

### *4.3. Computation Layer*

#### 4.3.1. Pods as Controlled Execution Environments

Alongside queries, computational resources associated with Pods also provide the opportunity to create user-controlled execution environments which can be used to run processes *locally*. In this, since the data never leaves the Pod or its associated (user-controlled)

resources, the overall process can become more trustworthy than sending the data to an app. This is an especially powerful paradigm for several cases, such as: (i) where computations do not need large amounts of data and are infrequent; (ii) they involve sensitive data; (iii) the apps and actors cannot be trusted to keep data; (iv) the apps and actors cannot be trusted to delete data after use; and (v) the user wants to control the process themselves—e.g., to ensure there is no bias or inaccuracy in outputs.

The code or instructions required for computation can be provided by an app, or it can be retrieved by the user through methods provided by the app. For example, to calculate the statistical mean of the user's walking distance in the past week, the app can either submit a detailed query with the mean calculation or only indicate the method *mean* with parameters based on some common understanding of how it should be executed. Alternatively, the Pod itself can determine functions to execute based on the information requested. From the previous example, the app only requests 'mean walking distance' (e.g., using a URL, query, or API), and the Pod interprets the query, identifies and performs the computations involved, and returns results.

#### 4.3.2. Cost of Computations

Computations can be costly to execute based on their complexity and resource availability. Since Solid does not require Pods to have computational capacities beyond those required to perform basic operations related to data storage, retrieval, identity management, and access control, computational resources cannot be assumed to be present in all Pods. This means the support for computations would be added separately by users—for example, through provisioned servers or serverless environments, which may vary between Pods and providers. Moreover, if these are restricted to using specific vendors or platforms—can affect the interoperability of information and behaviours associated with the use of Solid Pods.

In addition to dedicated computations generally having some *cost* involved regarding processing resources or time, there may also be computations inherently involved in retrieval mechanisms—such as converting the stored data into requested forms. For example, there are computations involved in mapping request paths to underlying folder structures on a server, looking up indexes for which data paths are to be retrieved for a given API, or executing queries (of various complexities) to selectively retrieve information. These may result in additional *costs* if such features are not part of the Pod infrastructure and services, or are metered based on quotas. For example, several cloud providers charge the use of network communications (ingress/egress) with some initial amount provided for free. If users are not aware of this, even seemingly simple retrievals may end up costing money.

Cloud services also enable providing virtual models over existing data through the Data as a Service (DaaS) paradigm. One way to utilise this within the Solid ecosystem, is to provide DaaS features where apps can request (raw) data, information (e.g., in a schema), or answers (e.g., as queries—see retrieval in next subsection). In this case, the cost of retrieval can be paid by apps or shared with the users. This can also be used to incentivise data providers to submit data in well-defined and supported forms, or to offset their non-conformance by enabling other data consumers to convert the data in order to use it. Another avenue is to federate processes so that not all retrieval-related computations take place on Pods, and instead, some data is processed on the apps' servers or client-side (e.g., a web browser), such as through the use of triple pattern fragments (TPF) [39].

### *4.4. Access Control Layer*

Access Control refers to the method by which access to data within a Pod is controlled. It relies on the data retrieval and data storage layers to define how data is identified (e.g., specifying its path) and the method by which it is accessed (e.g., also by path, API, or query). Its scope is limited to operations over data within a Pod, i.e., read, write, and erase,

and excludes control over how that data is used, disseminated, or managed once it leaves the Pod (instead, this is specified by policies in a later section).

In the current Solid specifications, access control is implemented as a user accepting a request from an agent (e.g., app) for operations (read, write, erase) over data (as URLs). In this, the specifications do not clarify how such URLs paths are mapped to data categories, or express complex conditions as policies based on agent class, resource class, and origin information, constraints, and conditions of use—which are important as they are parts of service agreements and have legal implications.

For a Pod to implement different data storage and retrieval mechanisms, it would also need corresponding support from access control mechanisms to specify who can access data. Similarly, if retrieval methods can utilise queries and computations, these would also need support from access control mechanisms to control who can perform them and who they relate to data. For example, an app that is permitted to retrieve only statistical means using queries can exploit the lack of control over what data can be queried to perform inference attacks to retrieve data. Therefore, the correspondence between access control and data retrieval methods also becomes relevant to the implementation and investigation of security issues beyond simple interpretations of permissions and access to data.

### *4.5. Communication Layer*

Communication here refers to the method by which data and access are communicated (i.e., provided, requested, and retrieved) in the context of a Pod. Solid specifications utilise web protocols (HTTP) for communication of data and access control requests, where URLs are used to identify Pods, data within Pods, agents, and other resources. The HTTP protocols (GET, POST, PUT, DELETE, etc.) are used to express interactions with resources. These are then utilised by users and apps to interact with Pods and the data they contain. Once access authorisations have been established, subsequent communications as per the Solid specifications are required to provide a security token that proves the prior relationship and which can be verified as being valid.

If the data retrieval and access control mechanisms can utilise other forms of identifiers for referring to data, or support operations not part of the current access control methods supported by Solid, the communication layer will also need to be modified to support these. For example, the current URL-based method presumes corresponding access control permissions based on that specific path. If this is not the case, and paths are virtual views that are not the same underlying data storage or retrieval paths, then the communication layer would require to be aware of such changes so as to accept these requests.

Similarly, if the communication includes additional content such as policies or other metadata related to the use of data or apps, the corresponding aspects of how apps declare themselves or perform initial authorisation requests and subsequent data use requests also needs to incorporate these changes. Finally, other forms of communication methods such as well-defined or common established URLs, or APIs, can be established as wrappers around existing HTTP/URL-based methods to provide the necessary abstraction to hide implementation details of pods and data (e.g., exact path for data) from external agents.

### *4.6. Interface Layer*

The interface layer is where the users (i.e., humans) interact with a Pod and manage their data and access to it for apps. This includes features that use UI/UX in the form of panels, dashboards, or similar design elements. Currently, the Solid specification does not provide any such feature and leaves it to the Pod providers to implement one for their users. The Solid website lists applications [1] providing interfaces such as file managers, messaging clients, calendars, and inbox management for app requests and notifications.

An important consideration in the development of such interfaces is what functionality they depend on within Solid Pods. For example, file managers rely on data storage and retrieval methods to understand what should be presented to the users. Similarly, contacts and calendar apps rely on the ability to retrieve relevant data and metadata to populate

their respective interfaces. If the data storage and retrieval forms are fixed, then these apps can only utilise that as the basis to provide their functionalities—which can result in unintended excessive data exposure as well as difficulties in getting exactly or only the data required and in the correct forms [24].

Another consideration for interfaces is that if they require some computation and are expected to be executed on the client side (e.g., by the Pod or a web browser), there is no way to express or distinguish this from those that are a front-end to an external server. Users of interfaces thus may not be aware that they are interacting with an external agent or a locally executed process (which may or may not be controlled by an external agent).

Interfaces also involve the interactions that users undergo in relation to the use of their pod, data, and apps. For example, an app's request to use data may be presented externally to the pod as a notice on that app's website. This may involve information about why the app wants to access the data, what data it wants to access, and other pertinent information such as policies applicable. Such interfaces should also be considered when investigating Solid's use-cases since they are how users make decisions on whether to grant access. In addition, these interfaces are also important for legal investigations, such as the provision of privacy notices or the validity of consent.

Other forms of interfaces include notifications, notices, correspondences, updates, notifications, or other communications between entities (i.e., providers, developers, brokers associated with pods, resources, and apps), and users are also included within the scope. Solid's inbox functionality is relevant for such communications but is currently limited to only handling some interactions regarding other users and apps.

### *4.7. Actor Layer*

Actors are entities associated with specific roles in relation to the Pod, data, and apps. They use the interfaces and communication layers to send and receive data and requests. These are distinguished from Solid terms such as *Agents* to refer to *legally accountable entities* so as to understand who is responsible and accountable for implementations, decisions, and processes. For example, each of Pod, data, and apps can have distinct actors associated with its development (i.e., who created it or maintains it), provision (i.e., who provided it), users (i.e., who has access to it), and investigative roles such as auditors and governance agencies. Understanding who the actors are is a crucial step in investigations associated with data. The actor layer only considers those actors associated with data interactions.

We distinguish *agents* from *actors* and *entities* for the purposes of establishing accountability. This requires the identification or indication of actors with sufficient details so as to enable legal processes to apply. A good example of this is to specify the legal name of a company along with its address—which enables the identification of appropriate jurisdictional obligations and authorities. Solid currently does not mandate such disclosures, but instead leaves it up to each app to provide pertinent details via its identity profile document. It considers domain-based identity as a *strong* form of identification. It also does not explicitly support or require storing legal identity or other relevant information associated with actors within its *Agent Registries*, for example, so that the user can introspect actors and their identities at any time. Actors are also important to explicitly identify to form an agreement (e.g., contract, consent). Without explicit or implicit identification of actors, the establishment of agreements between the user and an app (for example) would not be valid or sufficient to trigger legal obligations and remedies. Note, this only refers to cases where information about actors is not provided.

Another important consideration of actors is related to data provenance. This refers to actors that provide specific data and may be required to be recorded as the source of that data. Solid's specifications support specifying who can write or edit data but do not record its provenance in a direct manner. Data sources can have an impact on the validity and authentication of information—such as for official documents and have legal rights associated with it—such as requesting rectification of inaccuracies.

### *4.8. Logging Layer (Orthogonal)*

Logging refers to generating and storing information about processes and interactions in relation to the Pod, data, or apps as a form of record-keeping. Logs can be distinguished as data logs (read, write, edit, erase), access control logs (recording creation and use of permissions), policies, identity management (e.g., registration, verification), and security (e.g., incident reports). In addition to these, logs may also be kept by pod, app, and data providers for contextual reasons such as to assist in resolving operational issues. We distinguish logs as being supplementary from intentional persistent information such as an index of access authorisations.

Logging is expressed as an orthogonal layer as it can apply individually or collectively to all other layers. For example, logs can be limited to data storage mechanisms, e.g., kept by the storage provider, or data retrieval methods, e.g., kept by a server handling requests, or computing layer, e.g., recording computations being executed, or access control layer, e.g., record the success/failure of requests.

Logs can be stored as data within a Pod and can constitute personal data based on their contents. They may be mandatory and beyond the control of the user, for example—as provider data to keep track of resource usage. Logs may be protected due to their sensitivity or importance in establishing accountability, for example—strong limitations on who can write and view access control logs in a Pod.

Solid currently supports an extremely limited form of logging, where decisions related to authorisations are recorded in a registry, and an *Access Receipt* is provided as a successful response after authorisation. Such receipts do not specify particulars about the scope or contents of access, how or where to store them, and are not necessary to be exchanged.

### *4.9. Policy Layer (Orthogonal)*

Policies is a broad term that refers to documented conditions, constraints, or agreements regarding data, operations on data, and actors with specific roles. In the context of Solid, policies can be associated with each layer, and are thus expressed as an orthogonal layer. We distinguish between the terms *policy* and *agreement*, where a policy is considered an agreement if it can be interpreted as a formal and binding document between actors. A policy that is not an agreement is used to refer to documented information regarding conditions, preferences, requirements, requests, offers, or other similar notions associated with Pods, resources, data, or apps.

Policies go beyond access control as expressed in Solid specification as they can support conditions related to data categories, actors, locations, jurisdictions, and technical and operational requirements, as well as richer contextual expressions such as limitations on duration and frequency, and also involve risks and rights-related concepts. These can be expressed as abstract or universally applicable information, or refer to specific jurisdictional interpretations, such as those explicitly mentioning GDPR.

Policies can also refer to contexts outside the Pod. This is a crucial distinction to put in the scope of the operations on data outside of a Pod. For example, if an app's policy only concerns what data it requires initially when asking for access, but not what it can do subsequently—this presents issues regarding privacy and security as the app may reuse or share that data for undeclared purposes.

Solid currently does not support policies beyond those associated with access control and grouping of agents into categories. It also does not require an app to declare what it intends to do with the data in the form of a policy or to record such policies within the Pod. While policies are referred to as additional links in an app's profile, there are no conditions for what information should be present in them, their completeness, or how these should be expressed in relation to the use of data within Pods. It is important to distinguish these comments as referring to how to retrieve and apply such policies and information to the use of Solid Pods with the understanding that these may be externally regulated, e.g., through legal obligations.

Policies can also be used for governing the *usage of data* and influencing computational executions. For example, data can be shared with *sticky policies* that dictate the conditions under which that data can be used as well as specifying obligations to be fulfilled in return [17]. Similarly, *usage policies* can be used to ensure that any processing operations on data are performed in conformance and compliance with specific requirements and contraints [21].

### *4.10. Security Layer (Orthogonal)*

Security is a broad topic that can be applied to every other layer. For example, data storage mechanisms can have security measures regarding data integrity, encryption, and usage limitations. Similarly, data access methods can implement security measures in terms of placing limitations on what data can be retrieved or incorporating access control and policy checking as a precursor to ensure valid access. Separately, the servers implementing Solid Pods or associated resources can utilise authorisation, DDoS protection, HTTPS, or other similar security measures as part of their infrastructure and software. Given the breadth of the topic and its universality to all aspects of Solid Pods, data, and apps—we consider security an orthogonal layer. For clarity, we distinguish between security and privacy as separate topics in the context of this article.

Solid specifications have dedicated sections to security and privacy considerations. For example, strong and weak identifiability of applications is defined to distinguish security in terms of an app's identifiability via existing domain certification methods [12]. Similarly, specific risks are acknowledged [1] with information on their relevance and mitigation.

### *4.11. Identity Management Layer (Orthogonal)*

*Identity Management* refers to how identities of actors and agents are managed through identifiers, credentials, authorisations, and authentication mechanisms. This is considered an orthogonal layer since identities may be associated with each layer. Though closely tied to security, policy, and access control layers, the management of identities on its own has enough significance to be considered separately. For example, the identity of a Pod is separate from the user's identity—as reflected through their URLs, and which may have their own requirements for being considered valid or trustworthy. Identity management can also be separated based on the involvement of entities and the resources they control. For example, a storage provider's identity could be a separate implementation with its own corresponding access control methods.

Solid currently specifies identity management in terms of users and apps through the use of WebID specification [9]. These may need to be additionally managed for the use of specific resources, for example, to translate an app's request to use some data into a data storage provider's identifier for accessing data.

## 5. Use-Cases Exploring Governance of Solid Infrastructure and Apps

The previous section established the terminology regarding actors, roles, functions, and processes associated with Solid. This section uses these terms to describe various possible use-cases for the implementation of Solid Pods and apps. The *use-cases* are a collection of different permutations and combinations for how Solid Pods can be implemented in terms of governance of resources and apps. They are not exhaustive, since there can always be new paradigms and methods that change how implementations are deployed and/or function. For relevance, see prior descriptions [13–16,25] and explorations of legal relevance [7,27,40]. Instead, the categorisation of use-cases focuses on the aims of this article, which are: (1) to identify data governance patterns involving Solid Pods; (2) express use-cases in terms of freedoms and limitations on abilities and actors; (3) to explore issues of trust and security that arise from specific governance patterns; (4) to explore variations in responsibilities and accountability; and (5) to guide how legal compliance investigations should approach the use of Solid.

The use-cases are categorised based on specific topics they concern, i.e., as infrastructure (Section 5.1), Apps (Section 5.2), and limitations or extensions to functionalities (Section 5.3). Each of these categories reflects a separation of concerns in terms of implementations and behaviours, and can be combined with each other to further create more complex use-cases. For example, the use-cases related to the infrastructure where Pods are managed by providers or users can be combined with other use-cases where apps can be installed from anywhere or only from an 'app store'. Each use-case is given a unique identifier (with prefix *UC-*) for convenience in referencing them within discussions.

### *5.1. Use-Cases Based on Infrastructure Management*

These use-cases are based on variations in the infrastructure used to implement and deploy Solid Pods. They concern the developers, providers, brokers, and consumers of Pods and associated resources (e.g., storage space, computing environments), and how these are provisioned to the (end-)users. These use-cases include limitations or restrictions placed on the use of (physical) resources (e.g., what storage providers are supported) but not those that concern how Solid as a software platform operates (e.g., API used to access storage or software-based limitations). The latter categorisation is explored in Section 5.3.

#### 5.1.1. UC-I1: Completely Self-Managed

The first use-case considers the user setting up everything themselves in terms of the necessary resources (server, storage disks, networking, domains). This could be a typical home setup where the user runs the server within their home or a commercial setup where an organisation deploys servers from their premises. It provides a great degree of freedom in terms of choice of what hardware and software (including Solid implementations) are installed, such as operating systems, firewalls, or any other applications and devices. At the same time, it also puts the onus of maintenance and responsibility of choosing technologies on the user as a (self-)provider.

#### 5.1.2. UC-I2: Solid with IaaS

Subsequent use-cases divulge from *UC-I1* by increasingly abstracting the management of resources. *UC-I2* uses the IaaS cloud paradigm where users provision infrastructure from a provider. Other than the management of provisioned resources, users retain the freedom and flexibility of choosing what software they install and use on their servers. Since the hardware is provisioned as part of IaaS, users have to abide by the availability and flexibility of chosen resources in terms of features, compatibility, and management options. Communications with resources are typically made through APIs defined by the resource providers. In some cases, an IaaS provider may only support or allow provisioning resources within its cloud service offerings or provide incentives such as lower costs for using the same provider. IaaS offerings typically also offer complete flexibility in choosing locations for each resource.

#### 5.1.3. UC-I3: Solid as PaaS

Further abstracting the management of resources, utilising the PaaS cloud paradigm for Solid enables users to request a Pod as a *platform* where they receive a pre-configured infrastructure where details and management of underlying resources such as servers, data storage disks, etc. are abstracted and hidden. Users may have the option to provision additional resources or change existing ones based on supported options. Providers may offer pre-configured choices of implementations based on Pod specifications, resource quotas (e.g., storage size), locations, and software (e.g., underlying operating system). In the case of PaaS, users have lesser freedoms since direct access to the hardware is restricted and all interactions happen through a dedicated virtual environment or APIs. However, users get more convenience since they have to manage fewer resources.

#### 5.1.4. UC-I4: Solid as SaaS

Applying the SaaS cloud paradigm for Solid means providing a Pod's functionalities in the form of software that the users can access and manage, e.g., through apps on their devices or through a web browser. In this case, all the underlying resources (e.g., servers, storage) would be hidden and managed by the provider. This represents the least amount of freedom since users would not have any choice in how their Pods are actually implemented, while also offering the most amount of convenience as the users obtain a complete system that is managed and maintained by the provider. However, it also represents increased responsibilities for the providers in managing the resources, deciding how they should function together, addressing security, as well as following any specific policy or legal obligation arising from such decisions.

The SaaS paradigm also enables capabilities such as implementing Solid specifications on top of existing products and services, such as that offered by NextCloud (https://nextcloud.com/blog/press_releases/pr20210414/ access on 1 November 2022), by providing Solid functionalities through reuse of the underlying native APIs. This approach can also be extended beyond SaaS to provide the resulting solution as PaaS or IaaS offerings. More importantly, if Solid specifications are constrained only to define how data is stored and accessed through URLs, along with some requirements on identities of users and apps, then the SaaS model enables any cloud-based storage provider (e.g., Dropbox, Google Drive, Microsoft OneDrive) to provide Solid-compatible solutions through their existing products and services that are popular and widely utilised.

#### 5.1.5. UC-I5: Solid with CaaS

CaaS in combination with Solid enables the use of other use-cases with additional resource management for providing computing functionalities, either in the form of servers or as serverless computing environments and APIs. This could be for executing processes associated with users (i.e., self-managed computing) or apps (i.e., controlled environment for third-party computing). The management of CaaS itself could be via IaaS, PaaS, or SaaS paradigms, which have different implications on roles and responsibilities associated with controlling the execution of processes. CaaS can offer valuable trust and security accommodations where data does not leave known boundaries or is always under the complete control of users. In such cases, CaaS can be an additional cloud-based service provisioned by providers or brokers that can be mutually used by users and apps.

#### 5.1.6. UC-I6: Solid with STaaS or DaaS

If data within Solid Pods contains value through additional operations or interpretations, rather than apps asking for the entirety of (raw) data and then processing it, Solid can use STaaS or DaaS paradigms where apps request specific information retrieval methods (e.g., SQL queries) or answers (e.g., via semantic reasoning) to be derived from data and provided in lieu of sharing data itself. This can be a more privacy-considerate model where apps never receive the actual data. The availability of information retrieval and question-answering mechanisms can be implemented as any of the other cloud paradigms (IaaS, PaaS, SaaS), with varying degrees of control possible on whether the users can control what data and features are exposed to apps.

The STaaS and DaaS functionalities also enable third parties to provide additional services on top of Pod implementations by acting as information brokers between the users and the apps. For example, a STaaS/DaaS service provider can provide an API to calculate fitness metrics by retrieving running logs from a Pod, operating on it, and returning the derived information to an app. Separately, the STaaS/DaaS service can also be executed locally (i.e., within the Pod) through CaaS capabilities.

### *5.2. Use-Cases Based on App Management*

These use-cases are based on different methods by which applications can be managed in relation to a Pod, where their processes are executed, and how they are governed in terms of trust and security mechanisms.

#### 5.2.1. UC-A1: Apps Are Unmanaged

This use-case reflects the scenarios where any app can be used by the user without any checks and balances in place regarding their origin, conformance, or any form of control. It reflects the current situation in Solid where there are no mechanisms in place that require an app to declare necessary information (e.g., app provider, policies, legal compliance information) and which are checked before allowing the app to access data. Note that this refers to additional requirements as distinct from conformance to the Solid specifications regarding identity and access control.

#### 5.2.2. UC-A2: Apps Follow Conformance Protocols

This use-case adds requirements for apps in terms of how they should express conformance towards specific protocols, specifications, standards, or legal obligations, which can be assessed or verified to ensure trust and accountability. A simple example of such conformance is the current practices regarding smartphone app stores that require applications to be packaged with specific metadata, which is checked before 'installing' it on a smartphone. In the case of Solid, since apps are not installed but rather 'registered', the same process can be performed as part of the registration. The criteria and conditions for conformance can be based on requirements derived from organisational policies, user or community guidelines, legal requirements, or other sources.

#### 5.2.3. UC-A3: Apps Are Managed through App Stores

App Stores or Marketplaces are mechanisms for the digital distribution of software and are a widely utilised service across devices and platforms. This also includes package and code repositories, distribution of pre-compiled binaries, code that is then compiled on the device, and virtualised applications that are provisioned via cloud services. App stores enable convenience for both providers and consumers by providing a common interface that can also provide features such as search, curation, recommendation, updates, versions, dependencies, configurations, security notifications, and installation management. App stores can be established as commercial enterprises (e.g., Apple, Google), or be community-maintained efforts (e.g., Linux package repositories), and can have dedicated tools and services associated with app installation, verification, and updates.

App stores may or may not have policies for applications to satisfy in order to be listed and provided to users. For example, Linux distributions such as Arch have separate repositories for packages that undergo some level of oversight by repository maintainers (i.e., official repositories) and those that do not (i.e., Arch User Repository In addition to these, app stores also require specific mechanisms to be in place to verify applications in terms of security and tampering. Apple's and Google's smartphone app stores also feature oversight bodies that audit applications and remove applications that do not conform to policies or are found to have violated terms.

Solid currently does not have an app store but does maintain a curated list of apps [1] NextCloud's implementation of Solid as an additional layer on top of its own services also provides some extent (currently unknown) for the usability of apps developed for NextCloud [35].

#### 5.2.4. UC-A4: Apps Are Vetted or Certified

Vetting or certification is the process by which an app is audited or assessed and provided a demonstrable and verifiable certificate or seal that it can produce as a trust mechanism. Certification criteria can be based on standards—such as the use of ISO certification agencies, established codes and guidelines, and legal compliance

requirements—such as GDPR's use of certifications and seals as a measure of demonstrating compliance conformity.

Certification processes are common in the provisioning of software, where developers sign their created applications, which are in turn verified by execution environments before permitting users to install or use them. Such mechanisms have also been integrated into app stores as part of security measures. In the case of Solid specifications, currently, apps only have to have a verifiable identity attached to the (web) domain address under which they operate. An extreme form of certification is the use of legally enforceable agreements on what the app is permitted or restricted to do, which is the mechanism used by the Data Utility Company in the Flanders use-case [13].

#### 5.2.5. UC-A6: Apps Are Installed 'Locally' in a Pod

Tangential to how apps are provisioned is the question of how apps interact with data in the Pod. Solid's specifications consider only cases where the app requests the use of data through URLs. However, following the CaaS paradigm, it is also possible for apps to be provided as entire or partial code bundles that can be installed within a locally controlled environment such as the Pod or associated server. This is akin to installing software on a device managed by the user. In such cases, the executions may all or partially take place within the local environment, with external communications for sending/receiving data and instructions as alternate mechanisms. The execution environment and control retained by apps are important considerations for investigating responsibilities.

#### 5.2.6. UC-A7: Apps Install 'Service' within a Pod

Similar to how apps can be local executions in respect to a Pod, apps can also install or provide 'services' which are installed or executed locally within a Pod. The term *service* here refers to the architectural concept where a process is executed in the 'background' such that its execution happens without an active front-end or interface for the user. Examples of such services include those used for sync, updates, information retrieval, and managing protocol handling (e.g., communications). A service can also be an isolated installation separate from apps. For example, users may choose to only install services that operate as part of their pods without the corresponding necessity for them to be expressed as 'apps'. Solid currently does not specify or allude to the use of such services, but their prevalence and use in other devices and platforms represent a possibility for them to be provided as part of implementations.

#### 5.2.7. UC-A: Apps Create a Locked Ecosystem

This use-case represents limitations or restrictions for only specific apps to be allowed to be used or installed for a given Pod. This is different from the use of an app store where any app that satisfies the criteria for inclusion can be used, and instead represents the case where a Pod provider specifies a restricted list of apps that can be used. Users have no choice but to use only these apps, and nothing else can be used without the Provider's support. Such measures can be enacted for purposes of ensuring rigid security, controlling data use, or locking users into the provider's ecosystem. Currently, Solid places no such restrictions.

### *5.3. Use-Cases Based on Extensions and Restrictions to Solid's Functionalities*

These use-cases represent additional extensions or restrictions that deviate from the current Solid specifications. These are mentioned as they impact how Solid Pods function, are provisioned, their portability and interoperability, and have important considerations on responsibilities of both providers and users, especially under GDPR.

#### 5.3.1. UC-L1: Limitations on Data and/or Apps

A Pod or app provider may place limits on data or apps whereby the user cannot exercise control in terms of editing or deleting it. For example, a Pod provider may

provision the pod with some pre-configured apps or data that the user cannot modify or remove but may have the option of adding and managing other additional data and apps. An extreme extension of this concept is the case where all data within the Pod is not within the user's control, and where the Pod is effectively a read-only storage solution. Less severe iterations are also possible, such as only allowing certain types of data to be stored in the Pod, or for the data to only be shared with certain apps.

#### 5.3.2. UC-L2: Shared Pods with Multiple Users

A Pod may have multiple users with or without separation of data amongst themselves. This is akin to multiple users using the same terminal or computer with separate accounts, who may be able to view, use, or control each other's dedicated data, and may have common locations for shared data. Such separations can reflect private groups such as families or organisational environments such as departments and teams. The sharing of a Pod has important considerations in terms of responsibilities, security, and the necessary software support required. Currently, Solid does support multiple users to access data within a Pod with varying degrees of control, but only supports a single user to be associated with the Pod in terms of its identity.

#### 5.3.3. UC-L3: Pod Where User Is Not a Data Subject

This use-case considers Solid Pods where the user is not the data subject, i.e., the data within the Pod is about an individual other than the user. This could be where the user is permitted to manage such data (e.g., as a parent or guardian or legal representative), or where Pods are used as dedicated storage areas for an individual's data and the users are part of the organisation for managing it. In such cases, the user deciding who accesses the data has implications for the data subject, which means the users also share responsibilities depending on the specifics of their role and relationship with the data subject.

#### 5.3.4. UC-L4: Virtualisation of Pods

A Solid Pod is presumed to be a single holistic resource. However, it is feasible for a Pod to be implemented as a virtualisation over several resources or even Pods that are hidden from external view and are used to separate and manage data and relevant concerns. The inverse of this is also feasible—where separate Solid Pods are in actuality the same underlying resource with virtual separation to represent data of different users. Such patterns require support from various implementation layers and may entail obligations for the entities deciding how such separations should be implemented and managed. For example, if Pods are implemented on a single storage infrastructure with separation managed virtually through software, then a valid security concern is whether accessing one user's data can accidentally retrieve another user's data.

## 6. Applying GDPR to Solid

The GDPR [4] is a relatively large and complex legislation with 99 Articles and 173 Recitals that define roles, compliance processes, and obligations along with supplementary guidelines and case law. For the rest of the document we abbreviate *Art.* for Article and *Rec.* for Recital following common conventions for indicating clauses. We utilise GDPR's Art.5 Principles as abstract and high-level goals through which other relevant articles and obligations are discovered. Our investigations and arguments have overlaps with similar prior analysis [7,27,40], which we advance through use of cloud-based interpretations and governance models for Solid's implementations.

### *6.1. Lawfulness, Fairness, Transparency*

GDPR Art.5-1a stipulates that all processing must be "processed lawfully, fairly and in a transparent manner". The principle of *Lawfulness* refers to the necessity for any and all data processing to be justified through the use of one or more of the 6 lawful or legal basis defined by Art.6. These contain among others—consent (Art.6-1a), contract (Art.6-1b), legal

obligation (Art.6-1c), and legitimate interests (Art.6-1f). In addition to these, certain contexts have a separate legal basis that must be utilised, such as special categories of personal data requiring Art.9 legal basis, and cross-border data transfers requiring Art.45, Art.46, or Art.49. In addition to these, the principle of lawfulness also requires the processing to be lawful with other legal requirements outside of the GDPR.

The choice of which legal basis to use is dependent on the intended purposes, processing operations, and personal data categories, as well as their necessity and importance. While the GDPR does not specify any particular order or preference for one legal basis over another, an incorrect application can result in a compliance violation. The most common legal bases in use are related to the contract (e.g., service provision), legitimate interest (e.g., fraud prevention), and consent (e.g., opt-in marketing).

Another point of note is that under GDPR, each (singular or joint) Data Controller must have its own separate and independent legal basis, i.e., if there are two Data Controllers with separation of purposes and concerns, and both require consent, they must collect such consent separately. Such conditions for the *validity* of consent are noted in Art.7, Rec.32, Rec.33, and Rec.43, along with the necessity to maintain records in Art.7 and Rec.42.

In Solid, while there is a record for access granted to an app for specific data, it is not sufficient to meet GDPR's requirements for either valid consent or a valid record of consent. For example, Solid's mechanisms and records do not specify who the Data Controller is, the purposes for use of that data, the categories of data (especially for special categories), and if there are other legal bases also associated with that data.

Solid defines the concept *Access Needs* as "a specific explanation of why that data type is being requested" [12], which can be considered analogous to a purpose. However, the use of terms *access* and *needs* imply specific contextual connotations that may be interpreted ambiguously and in a manner not compatible with the use of *purpose* in data protection and privacy laws. For example, *access* can refer only to the initial access to data—which is only one type of possible operation and does not cover others in Art.4-2 such as subsequent uses and sharing. Similarly, *needs* implies some notion of necessity which may be misleading in cases where the data is *optional* or *not strictly necessary* to fulfil a purpose but is useful to enhance it.

In order to be compliant with the GDPR, it is vital for a Data Controller to declare their legal basis, and to indicate this clearly to the data subjects. In the case of Solid, there is no corresponding concept or method by which an app can indicate the legal basis for how it uses the data. Since the model for data access is based on asking permission from the user, this may be misconstrued as being only based on consent, but in actuality may be accompanied by other legal bases—such as contracts and legitimate interests. As a consequence, Solid also does not feature or support recording the legal basis as part of establishing the access control authorisation from an app request, nor to record subsequent accesses to the data being performed using specific legal bases.

An important point to consider in terms of behaviour is that organisations may share data with third-party organisations based on (assessed and validated) the third party's legitimate interests or as part of a legal obligation (e.g., Banks and Know-Your-Customer). In the case of Solid, because the data is stored on a Pod and managed by a user—the responsibility of handling such requests also falls on the users. This represents an unexplored area of GDPR, such as deciding what are the legal options for when Solid Pod users receive a request to access data but with the legal basis as a legitimate interest or legal obligation instead of consent or contract. The answer may be simpler in cases where users are also data subjects since this means they can exercise the right to object (Art.21) for legitimate interests while handling legal obligation requests can be challenging regarding establishing the validity of such requests.

Fairness refers to the necessity for processing to be in line with reasonable expectations of data subjects and which does not have unjustified adverse effects on them. This principle provides data subjects with some degree of protection from being exploited, manipulated, or otherwise having detrimental impacts arising from the processing of their personal data.

In order to assess the fairness, considerations include specific justifications presented for the collection and use of data, their comprehension by individuals and groups affected, whether there is discrimination and if there are detriments or obstacles for individuals to exercise their rights.

Transparency, also related to fairness, refers to the availability and comprehensibility of information regarding the processing of data (e.g., what, why, where, how, who)—which is typically associated with a *privacy notice* (or privacy policies). Such notices may be presented to individuals as part of their contract (e.g., terms and conditions), or in the consenting dialogue. GDPR necessitates the provision of such notices for transparency in cases where data is collected (Art.13) as well as not collected (Art.14) from the data subject.

Solid does not support nor require apps to use or provide information via notices when making requests, except for the ill-defined Access Needs descriptions. This means apps must use external mechanisms to provide such notices, such as through their websites or other interfaces, the details and specifics of which are not accessible nor recorded to the users. While an app's metadata can contain a link to the policy that contains this information, Solid does not mandate it to be present, resolvable, or assist in ensuring this information is available and accessible to users. In addition, the lack of acknowledgement of how further processing of data beyond initial access should be indicated or recorded as part of the data requests also compounds the detrimental impact. This results in difficulties in assessing the transparency of data processing operations.

### *6.2. Purpose Limitation*

GDPR's Art.5-1b requires data be "collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes". This necessitates all data requests to be associated with a purpose that not only covers the initial justification, but also any subsequent additional uses of that data. The resulting obligation requires Data Controllers to inform about specific purposes, typically through notices, and places constraints on what is considered a 'valid purpose' based on its clarity, comprehensibility, and specificity.

The association between purposes and legal bases requires the use of a specific legal basis to cover all possible purposes and uses of data, which is at odds with Solid's focus on only considering data operations within the context of a Pod. For example, if a service provider requests access to data for the purposes of providing a specific service with the lawful basis as a contract, it cannot subsequently collect and use that data (externally outside the Pod) to also perform other activities that are incompatible with the initial purpose. In such cases, the service provider is required to ask for consent (or choose another appropriate legal basis).

In order to assess whether purposes are valid and whether they are being correctly used as required by GDPR, a Data Controller is obliged to keep a Record of Processing Activities (ROPA, Art. 30). In addition, purposes also have to be made available to data subjects—typically through privacy notices. In the case of Solid, while there are no specific obligations to keep a record of such purposes within the Pod, the existing use of Access Needs and their groupings is similar to the use of purposes and purpose limitation principle, though inadequate to meet the GDPR's requirements.

### *6.3. Data Minimisation*

GDPR's Art.5-1c specifies the data minimisation principle, which requires data to be "adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed". Along with the lawfulness and purpose limitation principles, the data minimisation principle also requires data to be specifically and explicitly declared and associated with purposes. Through these, assessments of adequacy and relevancy are to be made based on whether the Data Controller is using the 'minimum amount' of data to fulfil a purpose or is engaging in the (unlawful) excessive collection of data that is not justified, nor required, or is based on future presumptions of usefulness. For Solid,

first there requires to be an explicit acknowledgement of what data categories are being processed in relation to which purposes (or as currently used—Access Needs). Only then it can be assessed whether the data being processed is adequate, relevant, and necessary for the indicated purpose. As outlined earlier, the data processing operations include those that occur within a Pod as well as external to it—which is not currently supported by Solid's specifications.

### *6.4. Accuracy*

The principle of accuracy in Art.5-1d requires Data Controllers to ensure data is "accurate and, where necessary, kept up to date". Through Solid Pods, in theory, data subjects having (modification) access to their own data also gain the ability to rectify any inaccuracies or deficiencies themselves. However, in practice this may be difficult in cases where data is obfuscated or not comprehensible by data subjects, or through lack of relevant tools, or because modifying such data may affect its integrity and validity for subsequent use (e.g., in official documents). In such cases, the ability for data subjects to exercise their right to rectify information (Art.16) can be facilitated by providing a communication mechanism that uses the Pod to store and share the rectified information with the Data Controller as well as any other parties (Art.19). Note that under GDPR, a Data Controller is obliged to accept a data subject's changes to their data, which effectively makes the data within a Solid Pod that is under control of the data subject an authoritative representation of their data.

### *6.5. Storage Limitation*

GDPR's Art.5-1e stipulates personal data be "kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed;". This reflects the conventional interpretation where data is collected and retained by the Data Controller, which needs to be re-interpreted for Solid's decentralised paradigm. First, the storage limitation principle emphasises the necessity to associate specific categories of data being processed with the purposes of processing. Second, it refers to the storage duration for data, which in the context of Solid can be interpreted as both duration of access to data in a Pod as well as the storage duration of data retained external to the Pod.

Currently, the Solid specifications do not support expressing duration associated with data access authorisations, which means apps have access to data in perpetuity until access is revoked. This also reflects a deviation from 'good practice' guidelines regarding consent [41] which suggests limiting the duration of consent and/or re-affirming it periodically. While Solid specifications provide a way to record the timestamp associated with authorisations, they do not have the necessary mechanisms in place to evaluate this periodically—as required for storage limitation and consent 'refresh' requirements. An important consideration for cases where data is not under the control of the user, such as that outlined in *UC-L1*, is that the storage limitation principle applies in full as per existing conventions to the entity that has control of this data.

In addition, while the use of *identification* in the storage limitation principle does allow such data to be retained after being made anonymous—it represents an exceedingly high-bar since under GDPR the criteria for anonymity cannot be satisfied by merely stripping identifiers [42]. In order to indicate this information, it is necessary to express both duration of storage and the processing operation that will be applied after this period, i.e., deletion or anonymisation. Where the storage limitation principle is instead implemented by limiting access control, the scope for continued use of data through anonymisation may not be possible without the Pod and/or users' permission and support. Of note, merely storing the anonymised data within a Pod that is or can be associated with a data subject has a strong possibility for the data to be considered *identifiable*, and thus become non-anonymised personal data again, even if the app does not use this identification information.

### *6.6. Integrity and Confidentiality*

GDPR's Art.5-1e states the principle of integrity and confidentiality that requires data to be "processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures". This means Data Controllers and Processors who process personal data (including collection and storage) should ensure appropriate technical (e.g., encryption, access control) and organisational (e.g., training, codes of conduct) measures are utilised based on specifics of processing operations, data categories involved, and other contextual information (Art.32, Rec.78, Rec.83). This includes performing appropriate risk assessment and implementing risk management and mitigation processes in a systematic manner integrated with the processing (Art.32, Rec.75, Rec.76. Rec.77). The use of 'security measures' is a shortened common form referring to both technical and organisational measures under GDPR based on their intention to protect and safeguard data and impacts to the data subject in the broadest sense.

While the implementation of measures related to security is typically the responsibility of a Data Controller, for Solid this changes based on who controls the implementation of Pods, resources, data management, and apps, even if it is the case that access to the storage is managed by the user. For example, if the user provisions a Pod using SaaS, they have no control over the underlying infrastructure, whereas if they provide it using IaaS, then they control the implementation of resources. The integrity and confidentiality principle is important given that it affects aspects such as backup, integrity controls, protection against attacks (e.g., hacking, DDoS), data breaches, maintaining logs, ensuring effectiveness for access control, etc. It is also important for considerations related to risk management and impact assessments (Art.32, Art.35, Rec.75, Rec.84, Rec.90–93).

One important consideration for Solid is that the GDPR requires Data Controllers to make informed assessments of a Processor's security measures before engaging them. Depending on who implements and controls what within a Solid-based use-case, the various responsibilities may need to be carefully considered and established to ensure appropriate safeguards are in place and ensure their legal relevance and compliance. While there are existing guidelines in place regarding implementations for resources (e.g., hardware, software), the new considerations relate to whether users will have the responsibility to assess such measures for Pod and resource providers based on existing common practices for provisioning cloud services—especially in IaaS paradigms. For example, Hetzner (https://www.hetzner.com/legal/terms-and-conditions/ accessed on 1 November 2022) a Cloud service provider outlines their role as a Processor, and that the customer is the responsible party regarding processing of personal data within provisioned services.

### *6.7. Accountability*

The last principle mentioned is accountability (Art.5-2), which states—"The controller shall be responsible for, and be able to demonstrate compliance" with the other principles. Depending on the role (Controller or Processor), the accountability principle requires planning and operational management along with documentation of specific obligations from GDPR. For example, implementing policies for 'data protection by design and default' (Art.25), appropriate contractual measures between Controllers and Processors (Art.29), ROPA records (Art.30), handling data breaches (Art.33, Art.34), Data Protection Impact Assessment (DPIA, Art.35), appointing a Data Protection Officer (Art.37–39), and utilising appropriate codes of conducts and certifications (Art.40–42). These obligations are required to be maintained and re-evaluated on an ongoing basis and are typically integrated into an organisation's management frameworks and policies. The records kept as part of these processes (e.g., ROPA) are utilised as the first avenues for inspections and audits into compliance practices.

The Solid specifications provide rudimentary measures for supporting such records through registries of access requests, but they do not satisfy the full extent of requirements

required by GDPR (e.g., ROPA Art.30). This is an important deficiency in cases where the user (or provider) has to maintain records associated with (other) resource providers and apps, either because they are the Controllers in such relationships or to audit and ensure accountability of other entities. Since the current state of specifications does not provide any avenue for even establishing who the other entities are (i.e., their legal identities and applicable jurisdictions), their legal compliance investigations face a steep setback. In the ideal case, the required information is recorded, or available through accessible links (e.g., the app's metadata). However, if this is not the case, and there is some mischief or malice being conducted, there is little potential for resolving this since authorities may have to conduct lengthy investigations to even establish the entities and their roles within a use-case. Pragmatism, therefore, necessitates such identities and relationships being established and recorded upfront to ensure appropriate levels of accountability are maintained.

### *6.8. Controllers and Processors*

GDPR Art.4 defines a controller as the entity that "alone or jointly with others, determines the purposes and means of the processing of personal data", and a processor as the entity that "processes personal data on behalf of the controller". Identifying who the controller(s) are in a given use-case is an essential task towards establishing who has the responsibility to fulfil GDPR's various obligations. For this, as per the definitions, the *determination* of a purpose, i.e., who decided what purposes should the data be processed for, and the *means* for carrying out that processing, i.e., how it should be implemented.

The nuance between a controller and a processor is an important one given the implications of additional obligations that come with being a controller. In most cases, even where a processor seems to be specifying the means for how processing takes place (e.g., use of a specific technology), it is essential to clarify that such operational decisions are permitted for a processor as long as they limit their processing to what has been agreed in the contract with the controller [33]. Therefore, merely determining the technological implementation of processing is not sufficient to become a controller if such implementations are backed by appropriate contractual terms, and where the processor is not the one that determines what data should be processed, its purposes, legal basis, etc.—which are to be decided by a controller.

Typically, under GDPR, the controller is the entity that decides what data should be collected, maintains it, and operates on it for some purposes. In Solid, it is tempting to assume that because the user has control over how their data is stored and the ability to manage its access for external entities (as apps), they automatically become full-fledged controllers under the GDPR [33]. However, in this, there are several important considerations and complications that require careful legal analysis and investigations to establish the exact responsibilities of the users as well as to avoid unnecessary burdens on individuals from a presumption of controllership.

Further, GDPR Art.2 and Rec.18 specify exemptions for processing undertaken by individuals as a "purely personal or household activity and thus with no connection to a professional or commercial activity". This means that a user maintaining their own data for personal reasons is exempt from GDPR's obligations, but only to the extent that such as activities are considered private. Existing case law [5–7] establishes that where such boundaries are crossed, e.g., by making data public or undertaking activities for commercial reasons (e.g., running a business), the exemption cannot be valid and the user is considered a controller for the processing of their own data.

In Solid, the determination of who is the controller, therefore, is based on what entity determines how data should be used and how it should be processed. Since this is highly dependent on how Solid is implemented in a use-case and that is effectively in control of resources and data, it is not possible to make blanket or universal statements regarding who is a 'Controller'. This is where the terminologies from Section 3 and the use-cases explored in Section 5 are helpful to initiate an inquiry into who the entities involved are, what

are their roles, and what influence or control they exert over which resources—towards determination of controllers within the use-case.

Where users control the implementation of pods and resources, they determine the means of storage (as a processing operation)—therefore, they should be considered controllers for the purposes of storing their information. Similarly, when users do not control the Pod infrastructure and do not have access to the underlying implementations, they should not be considered as controllers—but instead, the Pod and resource providers who retain control should be specified as controllers. In this, we presume that Pod users are the data subjects, because if they are not, then based on their relationship to the data subject (e.g., parents) they may end up being (joint-)controllers irrespective of the control over the underlying infrastructure.

The issue of Pod-based controllers is entirely separate from the use of data by apps, which will have their own separate considerations for determining controllers. Even if users (as data subjects) retain the ability to dictate which apps are permitted to access the data, and have control so as to revoke such access at any time—this cannot be sufficient for specifying them as controllers. This is because the determination of purpose would typically be performed by the apps, with the users merely acting to provide the required data. In cases where the users are involved in also specifying the purpose and/or how it should be implemented—such as with the CaaS paradigm in UC-Ix, the decision of who are controllers is based on the degree of control the users and apps have in deciding what should be performed as well as who/where it is implemented.

Where users (as data subjects) are deemed to be controllers, the exact implications of such a role are ill-defined due to the presumption of GDPR as well as its associated guidelines that the controller is an organisation. For example, it is entirely unfeasible for data subjects to be tasked with maintaining ROPA or records of their own consent, and it is outright nonsensical to imply that they should serve themselves with notices for transparency and accountability obligations. Therefore, where data subjects do become controllers, it should be taken as an indication of pointing out who is responsible for specific aspects of processing operations, such as the determination of storage or computation environments. This is no different than placing the responsibility of leaving a hotel room's door unlocked to the person using a hotel room.

While the arguments laid here regarding controllers are simplistic and lack any in-depth legal investigation, the intention is to prove sufficiently that investigations of determining controllers within Solid-based use-cases is a complex topic that warrants further research and investigations. In particular, to avoid excessive burdens on data subjects in managing their own data, and to offer a clear path forward by using the identified use-cases to determine who should be the controllers and processors. In this, we again emphasise the need and usefulness of a common terminology, which we provide in Section 3, and understanding the actual arrangement and utilisation of resources and determining which actors have what degrees of control, which we outline through use-cases in Section 5, as the necessary precursors to beginning an investigation of controller determination.

### *6.9. Exercising Rights*

GDPR Art.12–22 provide information on several rights that a controller has to make available to the data subject. These relate to transparency regarding processing actors and specifics (Art.13, Art.14), providing access to data (Art.15, Art.20), rectification (Art.16) and erasure (Art.17) of data, restriction (Art.18) and objection (Art.21) to processing, and to not be subjected to automated decision making (Art.22). Of these, the rights related to providing access to data and data portability can be readily implemented using Solid since the Pod already stores the data. Other rights that relate to rectification and erasure can also be readily implemented by modifying the appropriate data within a Pod.

Exercising the rights related to information provision requires such information to be recorded and accessible, which is currently not possible using Solid. For example, Solid specifications do not support indicating the source of data (e.g., user or third-party), the

identities of apps that collect this information, categories of data, who the data will be shared with, and several other key pieces of information required for the transparency obligation and typically covered by privacy notices. Similarly, exercising the rights related to restriction of processing, or objection to legitimate interests, or objecting to automated decision making also requires corresponding support from Solid implementations.

While it is always possible to provide information and exercise these rights outside of the Pod, e.g., on an app's website, these rights are closely tied to the access to data within a Pod. For example, when the user exercises their right to restrict processing based on the perceived unlawfulness of an app, this can be the equivalent of revoking any access to data even if there were other legitimate and lawful uses of that data by the same app. Without communicating such decisions (i.e., access revocation happened because a right has been exercised), the app (or more importantly the app's controller) may not be able to distinguish between the revocation of consent and exercising of rights related to objection or restriction.

### *6.10. Cross-Border Data Transfers*

GDPR operates under the EU regimes where a territorial scope is established for the free flow of data within the EU. Any data transfer outside of these jurisdictions (called 'third countries') requires a corresponding legal basis from Art.45 (adequacy decision), Art.46 (data transfer safeguards), or in their absence the use of explicit consent, contract, or other legal bases as per Art.49. To assist with the determination of whether these obligations apply, it is essential to identify and record the locations associated with data processing—including storage, computation, and communication.

In Solid, a Pod represents a storage resource stored at one or more locations, and which are accessed by apps. Depending on whether the Pod or app's resources involve locations outside the EU, and do not utilise any of the Art.45, Art.46, and Art.49 provisions, the resulting situation can be problematic with severe impacts on the ability of users to initiate an enforceable complaint (e.g., a non-EU based entity) and to exercise their rights (e.g., entities do not support EU rights). Such determinations require information about locations associated with the controllers and processors associated with the Pod, resources, and apps—which are currently not supported by the Solid specifications.

### *6.11. Handling Data Breaches*

GDPR Art.4-12 defines a data breach as "a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed". The EDPB's guidelines on data breaches outline breaches to have one or more of the following categories based on whether there is an unauthorised or accidental action on data regarding: (1) confidentiality— disclosure or access; (2) integrity—alteration; and (3) availability—loss of access or destruction. GDPR Art.33 and Art.34 specify obligations for controllers and processors to notify data breaches to the relevant controllers, authorities, or data subjects without undue delay, along with information about the extent of the breach in terms of affected data categories, consequences, and mitigation measures being undertaken.

In the case of Solid Pods, a breach can happen at the underlying storage resource, the software environment hosting the Pod (e.g., operating system, web server), the processes associated with implementing a Pod feature (e.g., database), or through access granted to users and apps. This can be for resources managed by users (e.g., IaaS, CaaS) or without the awareness of the users (e.g., Saas). In cases where the Pod is managed by a provider, with the user not having control over the implementations, the handling of data breaches is the responsibility of the resource providers. However, where the user manages their own infrastructure and software implementing a Pod, the responsibility of handling data breaches also rests with them.

It is important to note the distinction between GDPR's notion of responsibility and that of liability. Under GDPR, responsibility means the duty to carry out an activity, such

as notifying the authorities when a data breach takes place. By contrast, a liability is a typically pecuniary obligation that seeks to determine who is at fault. The handling of data breaches also includes ensuring appropriate technical and organisational measures are in place to avoid such breaches taking place, to minimise their consequences should they take place, and to have plans for addressing their impacts when they do take place.

For users implementing their own Pods (e.g., IaaS), this may mean additional duties for ensuring the appropriate security measures are in place, and carefully assessing all of their software's known vulnerabilities. While this may seem burdensome, this is no different than managing a personal server. Except that in the case of Solid, new attack surfaces emerge, such as cases where an app or malicious actor is provided access to all data on a Pod—through technical bugs, system vulnerabilities, or by convincing the user via social engineering and phishing attacks. Current Solid specifications have indicated some awareness of technical risks associated with breaches and security in general, but lack similar addressing of personal and social risks that lead to data breaches.

## 7. Existing Issues That Also Affect Solid

### *7.1. Transparency and Comprehension of Information*

One of the biggest issues for privacy is that individuals do not understand what is happening in terms of who the actors are and how they are using their data [43]. The *cliche* of lengthy 'privacy policies' and privacy notices using complex legal language has been well-studied and analysed, with several approaches proposed to mitigate these through the use of information extraction, summarising, and visualisation [44]. In addition, laws such as the GDPR are increasingly influential in determining the contents and provision methods of such information. This has been taken advantage of by developing methods that rely on legally required information being present within a document [44–46]. Communities have taken these approaches further by pooling and crowdsourcing information about privacy practices [47]. However, several issues persist—such as the information being (still) difficult to comprehend, understanding it in a contextually relevant manner [48], and their deviation from the actuality of data processing activities [48–50].

In the case of Solid, first, there is no consistent or clear method for how users should be provided with this information. Apps are supposed to have a profile that may contain a link to their policy, but there is no acknowledgement of what such policies should feature or how these relate to legal obligations. Combined with a lack of records on who is the actual legally-accountable entity accessing their data, users are completely at the mercy of potentially unknown entities with no transparency and accountability.

Where apps need to request access to data, it is unclear as to the necessity of providing information as a precursor to making the decision. This is a vital requirement where the basis for such data requests is consent—which Solid specifications also do not elaborate upon. The result is vague guidance on apps being required to ask *permission* to use data without any oversight on how that permission is sought or its validity in being considered *informed consent*. Even if such information was provided to the users via a website or other document in full compliance with the law, the Solid specifications do not acknowledge or provide support for users to be provided with this information in a manner that does not repeat the existing issues of information overload. Solid also does not record relevant information pertaining to such informed decisions, such as the location of where the user expressed their consent (e.g., a website), links to privacy notices, or receipts [51] that users and tools can utilise later.

### *7.2. Manipulative and Deceptive Practices in Consenting*

Further to issues regarding transparency and comprehension of information, the existing issues related to consenting being invalid, unfair, or outright deceptive also apply to Solid. For example, by keeping control of data with users, but control of the requesting mechanism with apps, the determination of what should be present in requests and choices offered to users is entirely controlled by external entities (e.g., app providers). This

effectively has the potential for existing widespread issues that take advantage of power imbalance with users to exploit them via various means—all stemming from the control of consenting mechanisms. Well-studied, widespread, and demonstrably illegal examples of such practices are:

- **Dark Patterns:** using UI/UX design to nudge, coerce, and manipulate users [49,50], e.g., clicking 'Accept All' because other options are hidden and require time to exercise or configure, or using pre-configured choices, or nagging when consent is refused.
- **Consent Walls:** withholding features and services until users give consent to excessive data use and sharing [50].
- **Hiding Impact and Extent:** using UI/UX design along with the user's lack of domain knowledge to hide the true extent and impact of consent—such as where accepting will result in sharing of data and profiling by thousands of companies [48,50], or that it involves sensitive or special data categories.
- **Assumption of Consent:** where consent is either entirely assumed (i.e., users are not even asked), or where the choices made by the users differ from what the controller records as consent (e.g., the controller may record 'Accept All' even when users have selected only some options) [48–50].

### *7.3. Withdrawing Consent*

Since Pods do not keep records of what the purposes, data, and context (e.g., duration, frequency) of consent is, or where users can find this information, users have no means to contact an application or exercise their rights—such as to withdraw their consent or to ask for restriction in processing. In this, it is critical to understand that Solid's use of *access control authorisation* cannot be treated as a fair equivalent of consenting and withdrawal on its own. This is because of the following reasons:

- *Access control* only governs access to data, therefore the permission it governs is restricted to access to that data. The Solid specifications do not mention any requirement to interpret revocation of access to also restrict further processing of data by an app.
- *Signalling consent withdrawal*—revocation of access can be for several reasons, for example, the user wants to stop providing access to that data, or they have identified a problem with an app, or some associated duration for the access has expired (e.g., access is valid for 3 months and must be confirmed periodically). By not distinguishing which of these decisions has resulted in the revocation of access, both users and apps are unaware of what has happened as well as what steps must be taken next. Therefore, where the user has decided to withdraw their consent, this should be a distinct action to enable the appropriate legal obligations to be triggered.
- *Communicating withdrawal to third parties*—users may give consent to more than just the app when sharing their data. For example, if the consent allows the app to access data, and to further share it with others, merely revocation will only be visible to the app. Since there was no record of what entities are involved in consent, the users cannot signal these entities themselves, nor can they ask the app to further communicate their withdrawal.
- *Granular withdrawal for some purposes*—GDPR requires consent to be granular in regards to purposes, i.e., separate purposes must have separate consent, such that consenting and withdrawal can be managed in isolation for each separate purpose. In the case of Solid, there is no indication of purpose when accessing data. While separate Access Needs can be managed using separate identifiers for security, these concepts do not cover purposes for how the data may be used externally to the pod. So, revocation of an access need may result in the necessary and corresponding withdrawal to other associated purposes the user agreed to when granting consent. Further, the specifications do not clarify the behaviour when two authorisations govern the same data and one is revoked.

### *7.4. Legal Basis: Ambiguity and Misuse*

GDPR's legal bases are strictly and rigidly interpreted in terms of their validity towards justifying processing. For example, it would be in violation of GDPR if a Solid app uses a contract or legitimate interest where the legal basis should have been consent. In addition, a Solid app's requests for consent should be separate from other matters and requests. This creates a potential for repeating existing problematic instances where controllers misuse legal basis (e.g., legitimate interest instead of consent [48–50,52]) and where users are not aware of other uses of data because of the notice for consent also containing information about other legal bases [50]. In Solid, since there is no concept of legal basis or consent, and all access requests are treated as a singular combined transaction, it is entirely possible for the consenting request to have specified that the data would be used for additional purposes through a separate legal basis, and which the users do not realise or misinterpret because of their understanding that Solid enables control of data *regardless* of legal basis.

### *7.5. Hidden Actors Exploitation*

Even where entities across both sides, e.g., users and app developers, are well-behaved, their use of other third-party vendors and services can be a source of unintended or negligent problems. For example, websites using a Consent Management Platform (CMP) to manage consent for their services have shown problematic uses where the CMPs decide what data is collected and use it for their own purpose [50,52]. While this is manifestly illegal under GDPR, as seen prolifically for misuses arising from real-time advertising and IAB's TCF framework [48], users and app developers may not be aware of such situations until the data has already been accessed and shared. In Solid, there are no checks and balances in terms of what an 'app' actually is in terms of legal entities, or a thorough analysis of the extent to which its design and mechanisms have issues that enable third parties to cause such mischief.

### *7.6. Risk Management—Data Sensitivity*

In a Solid Pod, there are no distinctions between data in terms of origin, sensitivity, legal obligations (e.g., mandatory information), or association with specific categories (e.g., to say data at a specific URL is health data). This is a problem because while neither users nor apps have this information, an app that uses data that is knowingly or unknowingly sensitive or special category may end up resulting in security issues or legal compliance violations. Such a lack of awareness regarding data also enables problematic actors to operate in a hidden manner, for example, hypothetically—if a fitness device was storing data in a Pod regarding movement logs, an app accessing this data for the purposes of showing statistics on how active the user is may find and exploit the location data present in logs in order to surveil and profile the user and to sell this information to third parties without the user's knowledge [53,54]—which are illegal under GDPR but not effectively enforced. The ambiguity on what access to data via a URL exactly means in terms of data categories further complicates the situation in the app's favour as they can claim the user is the responsible entity to ensure only needed data was transferred and the Solid specifications do not place any restrictions on access being based on specific data categories or requiring them to 'clean' or 'filter' data to ensure no additional information is provided. Combined with potential misuse from information obfuscation and dark patterns, the app may be able to further hide its activities and claim the user has consented to them [48].

Not knowing data categories involved and their sensitivity also has implications for the risk management and mitigation for all entities involved. If a Pod is provisioned using PaaS or SaaS, then the provider may be obliged to provide additional appropriate features if the use-case requires sensitive data (e.g., health data). Without such knowledge, providers only have to support the bare minimum security and risk features and may put the responsibilities and burdens on users to manage their own data-related risks.

### *7.7. Tracking and Profiling*

Tracking and profiling are severe problems for web-enabled platforms and devices as they provide a convenient method for information exchange. Techniques such as fingerprinting [55] are commonly used to identify individuals, and are associated with identifiers to profile them. Purposes of such activities can be justifiable, e.g., to combat fraud, or insidious, e.g., to sell profiled data to third parties. On the web, the use of cookies enables sharing of identifiers across providers, while on smartphones the device manufacturers themselves provide a unique identifier to enable such tracking and profiling.

In Solid, given that storage is offered without any oversight of what data is stored, who creates it, and why somebody uses it, it is possible that applications and actors develop new forms of fingerprinting and tracking techniques based on Solid's functionalities. For example, applications or the advertising libraries they use may develop a well-formed URL such as `/userID` to store and retrieve data associated with identifiers and profiles. This can be cleverly obfuscated so that it remains hidden from users. In addition, access to all kinds of data within a Pod can further enhance the profiling activities.

### *7.8. Lack of Effective Control via Limitations on Data Exploitation*

The mission of Solid is to give control of data back to individuals. However, if apps do not store data in *usable* forms, such as by utilising proprietary format, encrypting it, obfuscating it, or only storing partial data within the Pod—then the users have the data but no control to take advantage of it. While the GDPR tries to offset this through its right to data portability that requires data to be provided using machine-readable and commonly used formats, the actuality is that this right is not respected correctly [56,57]. Solid Pods provide a convenient method to retrieve and store data using GDPR [25], but without appropriate methods to ensure the data is actually useful and usable to the user and other apps, a Pod becomes limited to the user acting as a controller for the storage of information, while only some or apps with the ability to understand the data are able to benefit from it.

A limited variant of this is where market actors create ecosystems based on the availability of data, such as through APIs and restrictions on how data is accessed, which may not empower the user at all. For example, consider smartphone applications that are entirely hosted within the user's *personal device* and also store their data on the device, but where the user can neither access that data nor enable other apps to use it. This is designed as a security measure, with the operating system determining limited forms of interoperability for data (e.g., contact book, messaging, camera). If applied to Solid, this would mean Pods that are entirely controlled by providers, with limitations on how apps can use the data implemented as part of the Pod (e.g., provider's APIs), and with users only having limited capability to store their data and accept its use by applications—similar to how smartphones operate today.

### *7.9. Legal Compliance and Enforcement*

The interpretation of GDPR roles and obligations is a complex affair that takes time to go through the legal processes since authorities require certainty in their investigations before announcing violations, and the subsequent decision-making by courts (and higher courts) takes a lot of time. This issue is combined with a lack of available resources for authorities (e.g., not being provided with required funds), and lack of domain expertise for increasing complexities in use-cases and technology means GDPR enforcement has identifiable and systematic problems [58]. This is not to say the law is ineffective or should not be followed—on the contrary, GDPR has resulted in benefits to individuals by raising the transparency of information and creating new rights that empower individuals [58,59].

For Solid, the existing issues with enforcement are made more severe because of deviation from established domain and GDPR terminology—which first requires efforts such as this article to establish how GDPR should be interpreted for Solid, and further because there are no systematic designs or implementations related to oversight, accountability, and

technical/organisational measures—which requires legal compliance to be investigated and applied *from basic principles*.

### *7.10. Burden on Users to Manage Privacy*

After all other issues, even where everything is valid in terms of laws and social norms, the resulting responsibilities for users to investigate each request and subsequent data use as well as to manage their own data and its security/privacy can end up becoming a burden. Such cases are common in the form of 'decision overload' [60] whereby users choose seemingly self-detrimental outcomes because of perceived non-contextuality and the lack of relevant controls. Solid's use of inconsistent terminologies, lack of user-side tools and services, and inability to have effective policies and agents that operate for and on behalf of the user increase the possibilities for users to perceive the management of Pods as a burden and create opportunities for other issues to be exploited. It is not necessary to solve this issue only through technological means, such as developing an automated policy-reasoning service that acts on the user's behalf. Instead, other forms of socio-technical approaches should also be considered that take advantage of the human-centricity of technologies. For example, crowd-sourced management of privacy concerns [47,61], the establishment of codes of conduct for Solid, and creating open app stores with vetting processes.

## 8. Path Forward towards Responsible Innovation

### *8.1. Establish Consistent Vocabulary*

Currently, the Solid specifications present new terms and concepts that do not align well with existing well-established domains and regulations. For example, Solid's use of 'owners' is inconsistent with cloud services where ownership and client/customers are well-defined concepts with legal relevance and interpretations. Similarly, Solid's use of 'Agent' is ambiguous and lacks relating Pods, resources, data, and apps to legal roles such as Controllers, Processors, and Data Subjects—which are necessary to understand and establish responsibilities and accountability within use-cases. Solid's 'policies' currently only contain rudimentary access control constraints, and lack the nuances and extent required for understanding and managing data practices based on legal (e.g., legal bases, purposes) and social norms (e.g., sensitivity, risks).

To resolve this disparity, we strongly recommend the establishment of a consistent vocabulary for expressing Solid's use-cases in terms of actors, roles, and processes—and to use these to define obligations, requirements, and other constraints on conformance. We provide an initiation of this through the application of existing standardised cloud terminologies to Solid in Section 5. The outcome of this should offer clarity regarding implementations in terms of which entities are involved, how to hold them accountable, and to take advantage of existing legal obligations and rights. This certainty will assist all stakeholders (individuals, companies, authorities) in establishing how Solid should be used as a legally-compatible paradigm and enable realisation of its intended benefits.

Specifically, we envision the following concepts as needing consistent vocabularies:

- **Resources:** Pods, storage (e.g., disks), computation, etc.
- **Entities:** Providers, consumers, users, etc. for Pods, resources, data, apps, identity.
- **Legal Roles:** Controllers, Processors, Data Subjects, Authorities
- **Agreements:** Consent, Contract; but also agreements associated with provisioning Pods, resources, data, and apps. This can also be extended beyond current conventions, such as by enabling users to have preferences and requirements, or using policies to establish agreements with apps and services on the use of data.
- **Notices:** for Pods, resources, data, apps, services, users, along with information on context such as specifics, ex-ante or ex-post, provider, and recipient.
- **Data:** establishing data categories for clarity of what data is actually being utilised, indicating sensitivity of data to understand risks and necessity of security, establishing when special categories of personal data might be involved to better understand impacts and legal obligations. In addition, separating data based on whether it

is related to users, use of apps (e.g., configurations), pod management (e.g., data registries, app authorisations).
- **Processes:** related to management and use of Logging, Policy Management, Identity Management, Data Management, Network Management, Data Storage, Compute, Data Query, App Management, Management.
- **Security:** related to what security measures are in place for Pods (e.g., firewalls), resources (e.g., access control), data (e.g., encryption), apps, and users.
- **Logs:** maintaining logs related to data (e.g., store, access, modify, source), apps (requests, authorisations), policies (agreements, preferences), identity (e.g., users' actions), and security.

In creating these concepts, existing vocabularies can be utilised or extended to avoid re-creating entirely new terms with unknown interpretations. Sources for these can be ISO standards such as those related to cloud service, or legal thesauri such as that established within the EU and based on GDPR, or community efforts such as Data Privacy Vocabulary (DPV https://w3id.org/dpv accessed on 1 November 2022) [18,62,63]. This step will enable common approaches to be discussed and developed by relevant communities and is a precursor to enable approaches mentioned in the following paragraphs that require machine-readable information for automation.

### *8.2. Clarity of Concepts and Processes for Legal Compliance*

Solid's deviation from established legal terminology creates difficulties in the interpretation and application of specific jurisdictional regulations, such as GDPR's requirements to establish purposes and controllers. To avoid such differences from creating obstacles, along with using established vocabularies, a Solid implementation must also be capable of offering clarity on how it relates to specific legal compliance concepts and obligations. For example, when a user accepts the use of an app from a provider, it should be able to understand the involved controllers and processors, as well as what rights are available and the information to exercise them. Similarly, it should be clear what legal basis and purposes the app is requesting to use the data for, what will happen once the data leaves the Pod (e.g., sharing with third parties), and if there are any sensitive categories that the user should be aware of.

In order for this information to be available and accessible, it should be mandatory for an app to provide this information, either with the request or through a linked resource, in machine-readable form. An app that does not satisfy this requirement should not be permitted to initiate a request, or it should be discriminated against as being untrustworthy. This information and other relevant records should be maintained as logs within the Solid Pod. For example, logs recording the creation of an authorisation decision along with specifics of entities involved, and its scope (e.g., purposes, data categories).

It should be clarified within the specification, that an app's request to use data and the corresponding authorisation is a form of consent, and therefore requires the necessary legally obligated criterion to be satisfied to be considered valid. Where this is not the case, the separate legal basis (e.g., contract) should be recorded and handled accordingly. For specifying that a Pod or an app requires GDPR compliance, the vocabulary should enable this information to be supplied in machine-readable form, such as part of the app's request, or a Pod's metadata (e.g., as `region:EU`). Where the legal basis is consent, the withdrawal of consent should be communicated to the app, either initiated by a Pod (e.g., sending a signal to the app's specific URL) or indicated the next time an app accesses data (e.g., reinterpreting HTTP status code *451 Unavailable for Legal Reasons*). In either case, it is vital for there to be no ambiguity in the indication that consent has been withdrawn, so that the appropriate GDPR obligations are triggered regarding halting the processing of data outside the Pod (if any) and communicating the withdrawal to other parties.

Since these represent significantly complex legal investigations with potential implications and responsibilities for data subjects to manage their own data, we suggest taking the use-cases from Section 5 as hypothetical scenarios to determine how Solid's management

of data and access control fits the existing norms of determination of processing used to define controllers, as well as to identify the lack of clarity or ambiguity that may be present owing to the novelty of technologies involved. Through this, we envision further work exploring specific legal implications of different models of governance (also from Section 5 such as common app stores or community-managed guidelines that could alleviate issues of trust and accountability associated with the use of Solid Pods.

### *8.3. Enable Use of Policies*

The current status quo is the situation where all decision-making power is concentrated with controllers and service providers because they are the ones that decide what data will be processed and for which purposes. The individuals on the receiving side of requests only have the options to either agree to given choices or lose out on whatever features are being provided. In addition, this means it is always the users who have to perform compatibility and risk assessments for a given request based on what is acceptable and necessary for them regarding privacy and security. It would be useful and empowering to both users if they had mechanisms that assist them in making such decisions by taking their pre-configured choices and matching them with an app's request to determine compatibility [17,18].

A hypothetical implementation of such assistive and decision-support systems can be created using two kinds of policies represented in Solid—(1) user's preferences (that MAY be satisfied); and (2) user's requirements (that MUST be satisfied). When an app request is initiated, the Pod would then check it for compatibility with requirements where no deviation is possible, and with preferences where some conditions may not be satisfied. The result of this (e.g., all requirements satisfied, some preferences not satisfied) can then be presented to the user (e.g., as part of the request, or in their dashboards) to help indicate whether the app satisfies their privacy requirements and to ease the burden of checking the app's data practices.

For such policies to be effective, it is also necessary for apps to provide corresponding information as part of the request. The current norm is that an app provides a vague summary of the information within a privacy notice [50], while the privacy policy page provides a large amount of information that is difficult to interpret for a specific request [45]. Both of these issues can be addressed by making it mandatory for an app to issue a request using machine-readable information and requiring that this information be limited to what the request is about. For example, rather than requiring the user to agree to all possible purposes that an app can implement across all of its services, the request must contain only the purposes and data required for specific services associated with the request. Such constraints can also encourage the use of *dynamic consent* or *just-in-time consent* that does not overburden users with excessive consenting at the start, but instead asks for consent in a partial and modular manner based on triggers such as the start of a feature or service use [64].

The role of policies goes beyond user assistance in decision-making as they are also helpful to manage access control authorisations in terms of generating and checking them for validity, and for logging data access in terms of entities and purposes. In the new model where policies are used to establish a user's preferences, requirements, and an app's requests, the access control authorisations are generated as a result of a successful agreement being reached between the user's and app's policies by their respective agents. The logs for access control, therefore, only need to indicate under which agreement the app is requesting data, or to be more explicit—for which purposes the data is being accessed is indicated by specifying the agreement under which that purpose was agreed upon (note: this is possible because GDPR requires separate consent and hence separate agreements for distinct purposes).

The usefulness of policies over access control authorisations is also evident from policies permitting more complex conditions to be expressed and checked, such as indicating requirements that any health data (categories) should only be used for medical research (purpose) by non-profit organisations (entities) within EU (location, jurisdiction). Policies

are also more transparent and accountable forms that can be preserved for the user to introspect at their leisure. For example, users can revisit their decision to share some data with a specific app based on the detection of corresponding changes in their preferences and requirements.

### *8.4. Programmatic and Machine-Readable Notices*

One of the issues we highlighted that is also applicable to Solid use-cases involves the misuse of notices in consenting and transparency obligations. This is possible because such notices are generated by controllers who have incentives to maximise the user's acceptance of choices regarding the collection and sharing of their data. To avoid such manipulations and exploitation from taking place, a radical solution is to shift the control of notices away from service providers and onto the user's who have to make decisions [65]. In such cases, the notices are generated on the user- or client-side (i.e., by the Pod), and use the information provided by the controller (i.e., in machine-readable form). While this has ample benefits for users in terms of customisation of content, interface, personalisation using preferences and requirements, and recommendations based on wisdom (e.g., community-voted guidelines), it is detrimental to companies who would want to retain control of notices so as to have the ability to tweak their notices for purposes of marketing (i.e., for legitimate purposes). In such cases, a compromise would be to provide granular degrees of control to controllers with the notice ultimately still being generated on the client-side [65].

### *8.5. User-Side Risk Management*

Another common norm that works against the interest of the users is that they are unable to assess the risks and impacts associated with a given request, either because of a lack of knowledge or because the necessary information is complex or hidden from them. With decentralisation, the use of machine-readable requests, and programmatic notices, it is also possible for Pods to provide risk assessment and management features for their users. For example, by detecting and highlighting that a given request involves the use of sensitive data, the notice will be generated with a corresponding indication of higher risk. Another example is where the user-agent assists with handling requests by performing risk assessments, and identifying problematic patterns such as the final agreement involving excessive data being collected, or data being shared with too many third parties, and suggesting corresponding changes to preferences and requirements in order to mitigate risks, such as by deselecting purposes that involve excessive third-parties. The patterns can be established based on common interoperable vocabularies being used in policies and using the existing state-of-the-art as well as crowd-funded lists as sources. In this manner, the users get the convenience of not requiring excessive investigations and decision-making with the assurance of their privacy not suffering as a consequence.

## 9. Concluding Remarks

In this article, we established how to interpret Solid first as a cloud-based technology, and then in terms of different functionalities. Through these, we provided a framework to assist in the representation of Solid-based use-cases and established how existing cloud services and their providers are associated with its implementations. This enables assessing the resulting behaviour of Solid Pods in terms of what resources are involved, who the entities associated with them are, who has control, who is accountable, and what freedoms and capabilities are exhibited by a Pod. We explored some implications of use-cases by considering the different arrangement of entities and resources and determined the resulting implications on control retained by the individual. This article thus establishes a framework for investigating whether a given use-case or implementation does indeed fulfil Solid's vision to empower individuals through data sovereignty, or continues current practices and its problems by only marginally changing the data storage mechanism.

We then investigated how GDPR applies to Solid by interpreting its principles in terms of Solid's concepts and implementations. Our analysis shows a clear and severe deficiency

between Solid's specifications and the concepts, compliance obligations, and enforcement as envisioned by GDPR. More critically, we found potential difficulties in investigations of GDPR compliance for Solid-based use-cases due to a lack of support in Solid specifications or its implementation in considering basic accountability and responsibility mechanisms afforded by legal processes. We also identified the necessity for further exploring the determination of controllers and processors based on the roles of entities within a Solid use-case regarding the determination of purposes and retention of control. This is separate from the issue of apps being associated with controllers—which we highlight as a problem due to the current lack of requirements for apps to declare who legally operates them. We have also highlighted the impacts arising from Solid in terms of exercising rights, cross-border data transfers, data breaches, and enforcement of GDPR principles.

Finally, we found that Solid is also prone to the myriad and severe problems present in conventional use-cases that have known GDPR compliance violations. Their severity was increased due to the lack of certain processes within Solid which makes it difficult if not impossible to apply known remedies. We discussed potential approaches that can resolve these problems, both using known and novel research and developments, with a specific focus on how Solid specifications can themselves be further developed or extended, and through the use of both technical and socio-technical innovations. To argue for their feasibility, we included relevant associations with existing efforts for each argument.

Thus, through this article, we have presented our findings regarding making sense of Solid in terms of how to first understand the use of Solid within a use-case, its implications on abilities and control by various actors, and then investigate its legality in terms of GDPR. Going beyond merely pointing out problems, we also discussed what should be developed to address specific problems. Our arguments point towards the necessity for first developing interoperable vocabularies as the basis for enabling all other automated approaches, and then utilise these as conformance criteria that are mandatory for Solid's resources to exhibit in implementations. Further, we identify specific tools and processes that can be developed to assist the user in their decision-making and to inspect their data use, while also suggesting the use of communities to mitigate burdens on individuals. Through these, we hope to have provided a path based on pragmatism and responsible development that can take Solid towards the realisation of its vision.

**Funding:** This work has been made possible through a Short Term Scientific Mission (STSM) grant from COST ACTION CA19134 Ditributed Knowledge Graphs (DKG)—funded by the Horizon 2020 Framework Programme of the European Union. The author (Harshvardhan J. Pandit) has received funding from the ADAPT SFI Centre for Digital Media Technology is funded by Science Foundation Ireland through the SFI Research Centres Programme and is co-funded under the European Regional Development Fund (ERDF) through Grant #13/RC/2106_P2.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The author expresses thanks to Ruben Verborgh, Esther De Loof, Pieter Colpaert, U.Gent, IMEC, SolidLab, Data Utility Company, and Beatriz Esteves for their support and assistance.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Solid Project. Available online: https://solidproject.org/ (accessed on 1 November 2022)
2. Mansour, E.; Sambra, A.V.; Hawke, S.; Zereba, M.; Capadisli, S.; Ghanem, A.; Aboulnaga, A.; Berners-Lee, T. A Demonstration of the Solid Platform for Social Web Applications. In Proceedings of the 25th International Conference Companion on World Wide Web—WWW '16 Companion, Montréal, QC, Canada, 11–15 May 2016; pp. 223–226. . [CrossRef]
3. Solid Technical Reports. Available online: https://solid.github.io/specification/ (accessed on 1 November 2022).
4. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation). *Off. J. Eur. Union* **2016**, *L119*, 1–88.
5. Edwards, L.; Finck, M.; Veale, M.; Zingales, N. Data Subjects as Data Controllers: A Fashion(able) Concept? *Internet Policy Rev.* **2019**. Available online: https://policyreview.info/articles/news/data-subjects-data-controllers-fashionable-concept/1400 (accessed on 1 November 2022).

6. Janssen, H.; Cobbe, J.; Singh, J. Personal Information Management Systems: A User-Centric Privacy Utopia? *Internet Policy Rev.* **2020**, *9*, 1–25. [CrossRef]
7. Janssen, H.; Cobbe, J.; Norval, C.; Singh, J. Decentralized Data Processing: Personal Data Stores and the GDPR. *Int. Data Priv. Law* **2021**, *10*, 356–384. [CrossRef]
8. Solid Protocol. Available online: https://solidproject.org/TR/protocol (accessed on 1 November 2022).
9. Solid WebID Profile. Available online: https://solid.github.io/webid-profile/ (accessed on 1 November 2022).
10. Web Access Control. Available online: https://solid.github.io/web-access-control-spec/ (accessed on 1 November 2022).
11. Access Control Policy (ACP). Available online: https://solidproject.org/TR/acp (accessed on 1 November 2022).
12. Solid Application Interoperability. Available online: https://solid.github.io/data-interoperability-panel/specification/ (accessed on 1 November 2022).
13. The Flemish Data Utility Company. Available online: https://www.vlaanderen.be/digitaal-vlaanderen/het-vlaams-datanutsbedrijf/the-flemish-data-utility-company (accessed on 1 November 2022).
14. Van Damme, S.; Mechant, P.; Vlassenroot, E.; Van Compernolle, M.; Buyle, R.; Bauwens, D. Towards a Research Agenda for Personal Data Spaces: Synthesis of a Community Driven Process. In *Proceedings of the Electronic Government*; Janssen, M., Csáki, C., Lindgren, I., Loukis, E., Melin, U., Viale Pereira, G., Rodríguez Bolívar, M.P., Tambouris, E., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 563–577.
15. Buyle, R.; Taelman, R.; Mostaert, K.; Joris, G.; Mannens, E.; Verborgh, R.; Berners-Lee, T. Streamlining Governmental Processes by Putting Citizens in Control of Their Personal Data. In Proceedings of the International Conference on Electronic Governance and Open Society: Challenges in Eurasia, St. Petersburg, Russia, 13–14 November 2019.
16. Verbrugge, S.; Vannieuwenborg, F.; Van der Wee, M.; Colle, D.; Taelman, R.; Verborgh, R. Towards a Personal Data Vault Society: An Interplay between Technological and Business Perspectives. In Proceedings of the 2021 60th FITCE Communication Days Congress for ICT Professionals: Industrial Data–Cloud, Low Latency and Privacy (FITCE), Vienna, Austria, 29–30 September 2021; pp. 1–6.
17. Havur, G.; Sande, M.; Kirrane, S. Greater Control and Transparency in Personal Data Processing. In Proceedings of the 6th International Conference on Information Systems Security and Privacy, Valletta, Malta, 25–27 February 2020; pp. 655–662.
18. Esteves, B.; Pandit, H.J.; Rodríguez-Doncel, V. ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), Vienna, Austria, 6–10 September 2021; pp. 298–306.
19. Debackere, L.; Colpaert, P.; Taelman, R.; Verborgh, R. A Policy-Oriented Architecture for Enforcing Consent in Solid. In Proceedings of the Companion Proceedings of the Web Conference 2022 (Virtual Event), Lyon, France, 25–29 April 2022; pp. 516–524.
20. Esteves, B.; Rodríguez-Doncel, V.; Pandit, H.J.; Mondada, N.; McBennett, P. Using the ODRL Profile for Access Control for Solid Pod Resource Governance. In Proceedings of the Semantic Web: ESWC 2022 Satellite Events, Crete, Greece, 29 May–2 June 2022; Groth, P., Rula, A., Schneider, J., Tiddi, I., Simperl, E., Alexopoulos, P., Hoekstra, R., Alam, M., Dimou, A., Tamper, M., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 16–20.
21. Akaichi, I. Semantic Technology Based Usage Control for Decentralized Systems. *arXiv* **2022**, arXiv:2206.04947.
22. Braun, C.H.J.; Käfer, T. Attribute-Based Access Control on Solid Pods Using Privacy-Friendly Credentials. In Proceedings of the Poster and Demo Track and Workshop Track of the 18th International Conference on Semantic Systems Co-Located with 18th International Conference on Semantic Systems (SEMANTiCS 2022), Vienna, Austria, 13–15 September 2022; pp. 1–5.
23. Jesús-Azabal, M.; Berrocal, J.; Laso, S.; Murillo, J.M.; Garcia-Alonso, J. SOLID and PeaaS: Your Phone as a Store for Personal Data. In Proceedings of the Current Trends in Web Engineering, Helsinki, Finland, 9–12 June 2020; Ko, I.Y., Murillo, J.M., Vuorimaa, P., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 5–10.
24. Dedecker, R.; Slabbinck, W.; Wright, J.; Hochstenbach, P.; Colpaert, P.; Verborgh, R. What's in a Pod? In Proceedings of the 6th Workshop on Storing, Querying and Benchmarking Knowledge Graphs, Hangzhou, China 23 October 2022; CEUR Workshop Proceedings; Volume 3279, pp. 81–96.
25. De Mulder, G.; De Meester, B.; Heyvaert, P.; Taelman, R.; Dimou, A.; Verborgh, R. PROV4ITDaTa: Transparent and Direct Transferof Personal Data to Personal Stores. In Proceedings of the Companion Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 695–697.
26. Esteves, B.; Rodriguez-Doncel, V.; Longares, R. Automating the Response to GDPR's Right of Access. In Proceedings of the 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022), Saarbrücken, Germany, 14–16 December 2022; pp. 170–175.
27. De Bot, D.; Haegemans, T. Data Sharing Patterns as a Tool to Tackle Legal Considerations about Data Reuse with Solid: Theory and Applications in Europe. Digita Research Reports. 2021. Available online: https://go.digita.ai/reuse-patterns (accessed on 1 November 2022).
28. Esposito, C.; Hartig, O.; Horne, R.; Sun, C. Assessing the Solid Protocol in Relation to Security & Privacy Obligations. *arXiv* **2022**, arXiv:cs/2210.08270.
29. TechDispatch #3/2020—Personal Information Management Systems | European Data Protection Supervisor. Available online: https://edps.europa.eu/data-protection/our-work/publications/techdispatch/techdispatch-32020-personal-information_en (accessed on 27 September 2022).

30. 14:00-17:00. ISO/IEC 17788:2014 Information Technology—Cloud Computing—Overview and Vocabulary. Available online: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/05/60544.html (accessed on 1 November 2022).
31. ISO Cloud Computing Standards. Available online: https://www.iso.org/ics/35.210/x/p/1/u/0/w/0/d/0 (accessed on 1 November 2022).
32. Cloud Computing Risk Assessment. Available online: https://www.enisa.europa.eu/publications/cloud-computing-risk-assessment (accessed on 1 November 2022).
33. Guidelines 07/2020 on the Concepts of Controller and Processor in the GDPR. European Data Protection Board (EDPB). 2020 Available online: https://edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-072020-concepts-controller-and-processor-gdpr_en (accessed on 1 November 2022).
34. 14:00-17:00. ISO/IEC 22123-1:2021 Information Technology—Cloud Computing Part 1: Vocabulary. Available online: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/08/03/80350.html (accessed on 15 October 2022).
35. EU Funds Creation of First Major European Solid Provider for Enterprises. Available online: https://nextcloud.com/blog/press_releases/pr20210414/ (accessed on 1 November 2022).
36. 14:00-17:00. ISO/IEC 19944-1:2020 Cloud Computing and Distributed Platforms—Data Flow, Data Categories and Data Use Part 1: Fundamentals. Available online: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/95/79573.html (accessed on 15 October 2022).
37. ISO/IEC. ISO/IEC 29184:2020 Information Technology—Online Privacy Notices and Consent. Available online: https://www.iso.org/standard/70331.html (accessed on 21 May 2022).
38. 14:00-17:00. ISO/IEC 7498-1:1994 Information Technology—Open Systems Interconnection — Basic Reference Model: The Basic Model. Available online: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/02/02/20269.html (accessed on 15 October 2022).
39. Verborgh, R.; Vander Sande, M.; Hartig, O.; Van Herwegen, J.; De Vocht, L.; De Meester, B.; Haesendonck, G.; Colpaert, P. Triple Pattern Fragments: A Low-Cost Knowledge Graph Interface for the Web. *J. Web Semant.* **2016**, *37–38*, 184–206. [CrossRef]
40. Janssen, H.; Cobbe, J.; Norval, C.; Singh, J. Personal Data Stores and the GDPR's Lawful Grounds for Processing Personal Data. *Zenodo* **2019**, 1–6. [CrossRef]
41. *Guidelines 05/2020 on Consent under Regulation 2016/679*; European Data Protection Board (EPDB). 2020. Available online: https://edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-052020-consent-under-regulation-2016679_en (accessed on 1 November 2022).
42. Finck, M.; Pallas, F. They Who Must Not Be Identified—Distinguishing Personal from Non-Personal Data under the GDPR. *Int. Data Priv. Law* **2020**, *10*, 11–36. [CrossRef]
43. Veale, M.; Borgesius, F.Z. Adtech and Real-Time Bidding under European Data Protection Law. *Ger. Law J.* **2022**, *23*, 226–256. [CrossRef]
44. Harkous, H.; Fawaz, K.; Lebret, R.; Schaub, F.; Shin, K.G.; Aberer, K. Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 531–548.
45. Kretschmer, M.; Pennekamp, J.; Wehrle, K. Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web. *ACM Trans. Web* **2021**, *15*, 1–42. [CrossRef]
46. Degeling, M.; Utz, C.; Lentzsch, C.; Hosseini, H.; Schaub, F.; Holz, T. We Value Your Privacy... Now Take Some Cookies: Measuring the GDPR's Impact on Web Privacy. In Proceedings of the 2019 Network and Distributed System Security Symposium, San Diego, CA, USA, 24–27 February 2019.
47. Terms of Service. Didn't Read. Available online: https://tosdr.org/ (accessed on 1 November 2022).
48. Veale, M.; Nouwens, M.; Santos, C. Impossible Asks: Can the Transparency and Consent Framework Ever Authorise Real-Time Bidding after the Belgian DPA Decision? *Technol. Regul.* **2022**, *2022*, 12–22. [CrossRef]
49. Toth, M.; Bielova, N.; Roca, V. On Dark Patterns and Manipulation of Website Publishers by CMPs. *Proc. Priv. Enhancing Technol.* **2022**, *2022*, 478–497. [CrossRef]
50. Santos, C.; Bielova, N.; Matte, C. Are Cookie Banners Indeed Compliant with the Law? Deciphering EU Legal Requirements on Consent and Technical Means to Verify Compliance of Cookie Banners. *Technol. Regul.* **2020**, *2020*, 91–135. [CrossRef]
51. Jesus, V.; Pandit, H.J. Consent Receipts for a Usable and Auditable Web of Personal Data. *IEEE Access* **2022**, *10*, 28545–28563. [CrossRef]
52. Matte, C.; Santos, C.; Bielova, N. Purposes in IAB Europe's TCF: Which Legal Basis and How Are They Used by Advertisers? In Proceedings of the Annual Privacy Forum (APF 2020), Lisbon, Portugal, 22–23 October 2020.
53. *Data Brokers: A Call for Transparency and Accountability*; Technical Report; Federal Trade Commission (FTC): Washington, DC, USA, 2014.
54. Urban, T.; Tatang, D.; Degeling, M.; Holz, T.; Pohlmann, N. Measuring the Impact of the GDPR on Data Sharing in Ad Networks. In Proceedings of the ASIA CCS, Taipei, Taiwan, 5–9 October 2020; pp. 222–235.
55. Laperdrix, P.; Bielova, N.; Baudry, B.; Avoine, G. Browser Fingerprinting: A Survey. *ACM Trans. Web* **2020**, *14*, 1–33. . [CrossRef]
56. Kröger, J.L.; Lindemann, J.; Herrmann, D. How Do App Vendors Respond to Subject Access Requests? A Longitudinal Privacy Study on iOS and Android Apps. In Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20), New York, NY, USA, 25–28 August 2020; pp. 1–10.

57. Urban, T.; Tatang, D.; Degeling, M.; Holz, T.; Pohlmann, N. A Study on Subject Data Access in Online Advertising after the GDPR. In Proceedings of the Data Privacy Management, Cryptocurrencies and Blockchain Technology, Luxembourg, 26 September 2019; Pérez-Solà, C., Navarro-Arribas, G., Biryukov, A., Garcia-Alfaro, J., Eds.; Springer: Berlin, Germany, 2019; pp. 61–79.
58. Four Years Under the GDPR: How to Fix Its Enforcement Access Now. 2022. Available online: https://www.accessnow.org/cms/assets/uploads/2022/07/GDPR-4-year-report-2022.pdf (accessed on 1 November 2022).
59. Schütz, P. *Data Protection Authorities under the EU General Data Protection Regulation*; A New Global Benchmark (Extended Version); Handbook of Regulatory Authorities; Edward Elgar Publishing: Cheltenham, UK, 2022.
60. Nissenbaum, H. A Contextual Approach to Privacy Online. *Daedalus* **2011**, *140*, 32–48. [CrossRef]
61. Wilson, S.; Schaub, F.; Ramanath, R.; Sadeh, N.; Liu, F.; Smith, N.A.; Liu, F. Crowdsourcing Annotations for Websites' Privacy Policies: Can It Really Work? In Proceedings of the 25th International Conference on World Wide Web, (WWW '16), Montreal, QC, Canada, 11–15 April 2016; pp. 133–143.
62. Pandit, H.J.; Polleres, A.; Bos, B.; Brennan, R.; Bruegger, B.; Ekaputra, F.J.; Fernández, J.D.; Hamed, R.G.; Lizar, M.; Schlehahn, E.; et al. Creating A Vocabulary for Data Privacy. In Proceedings of the 18th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE2019), Rhodes, Greece, 22–23 October 2019; pp. 714–730.
63. Kurteva, A.; Chhetri, T.R.; Pandit, H.J.; Fensel, A. Consent through the Lens of Semantics: State of the Art Survey and Best Practices. *Semant. Web* **2021**, 1–27. [CrossRef]
64. Tauginienė, L.; Hummer, P.; Albert, A.; Cigarini, A.; Vohland, K. Ethical Challenges and Dynamic Informed Consent. In *The Science of Citizen Science*; Vohland, K., Land-Zandstra, A., Ceccaroni, L., Lemmens, R., Perelló, J., Ponti, M., Samson, R., Wagenknecht, K., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 397–416.
65. Pandit, H.J. Proposals for Resolving Consenting Issues with Signals and User-side Dialogues. *arXiv* **2022**, arXiv:cs/2208.05786.