FACULTY OF SCIENCES

Enforcing Data Protection in Solid: A Policy-Oriented Framework

Laurens Debackere
Student number: 01602685

Supervisors: Prof. dr. ir. Ruben Verborgh, Prof. dr. Pieter Colpaert
Counsellor: Dr. ir. Ruben Taelman

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de informatica

Academic year 2021-2022

GHENT UNIVERSITY

Samenvatting

Het Solid project heeft als doel om gebruikers opnieuw controle te geven over hun data, door applicaties en diensten los te koppelen van de opslag van gegevens die ze verwerken. Om deze visie te realiseren definieert men binnen dit project een specificatie voor persoonlijke datakluizen, gebaseerd op bestaande Web-standaarden en -principes zoals Linked Data, REST en OpenID Connect. Binnen het Solid Protocol 0.9 wordt gebruik gemaakt van Web Access Control om het voor een eindgebruiker mogelijk te maken om uit te drukken wie toegang heeft tot de gegevens in diens datakluis. Web Access Control heeft echter beperkingen op vlak van expressiviteit en interpreteerbaarheid, eigenschappen die enerzijds toelaten dat dit mechanisme efficiënt kan ingezet worden bij het bepalen van de autorisatie van een *client* maar het anderzijds moeilijk maken voor gebruikers om in de praktijk controle te houden over hun persoonlijke data.

Recente wetgeving op vlak van gegevensbescherming, zoals de Algemene Verordening Gegevensbescherming (AVG) die in mei 2018 van kracht werd in de Europese Unie, stelt echter specifieke vereisten aan toepassingen die persoonsgegevens verwerken op technisch, organisatorisch en juridisch vlak. Zo reguleert de AVG alle aspecten van gegevensverwerking en zorgt ze ervoor dat een eerlijk en transparant proces met betrekking tot het *data subject* moet worden gevolgd. Sindsdien werd gelijkaardige wetgeving rond gegevensbescherming ook in andere landen en regio's geïntroduceerd. Waar de AVG uitgaat van het principe van gegevensbescherming "by design and by default" worden we als eindgebruiker nog al te vaak geconfronteerd met ellenlange gebruiksvoorwaarden, toepassingen die op bedenkelijke wijze onze toestemming vragen en omslachtige processen om transparantie te verkrijgen omtrent hoe onze gegevens effectief worden verwerkt.

De principes van het Solid-project op vlak van controle over persoonsgegevens en transparante opslag en verwerking sluiten vrijwel naadloos aan bij de doelen van de wetgever bij het opstellen van de AVG. Echter voorziet de huidige Web Access Control specificatie niet de nodige granulariteit of contextuele informatie om aan de juridische vereisten te voldoen wanneer een toepassing of dienst geautoriseerd wordt om gegevens uit de datakluis op te halen. Recent werk van de Solid Data Interoperability panel, een werkgroep binnen de Solid gemeenschap, aangevuld met onderzoek rond de inzet van de semantische *policy languages* ODRL en SPECIAL in de context van Solid kan echter een oplossing bieden voor de beperkingen in de huidige standaard.

In deze thesis onderzoeken we dan ook de tekortkomingen van de bestaande Solid specificatie op vlak van autorisatie en authenticatie in de context van toepassingen die persoonsgegevens verwerken, waarbij we vertrekken vanuit de vereisten gesteld door de Algemene Verordening Gegevensbescherming. Op basis van de bevindingen uit deze evaluatie wordt een referentie-architectuur voorgesteld om de autorisatie-mechanismen van Solid in overeenstemming te brengen met de wettelijke criteria, waarbij we trachten om onze oplossing intuïtiever te maken voor zowel eindgebruikers als ontwikkelaars. We bespreken tot slot een implementatie van deze architectuur op basis van de Solid Application Interoperability ontwerpspecificatie en de Community Solid Server waarbij we aanknopingspunten voor verder onderzoek aanduiden op vlak van semantische modelering, efficiëntie en informatieveiligheid.

Summary

The Solid project aims to restore end-users' control over their personal data by decoupling applications and services from data storage. To realize this vision of data governance by the end-user, the Solid community is developing a specification for personal data vaults, also referred to as *Pods*, based on existing Web-standards and -principles like Linked Data, REST and OpenID Connect. In the context of the Solid Protocol 0.9, Web Access Control is used for enabling an end-user to express who can access the data in their Pod. Web Access Control does however have important limitations concerning expressivity and interpretability, properties that are simultaneously essential for efficiently evaluating the authorization of a given client and detrimental to an end-user's practical ability to control their personal data.

In contrast, recent legislation concerning data protection, like the General Data Protection Regulation (GDPR) which entered into force in the European Union in May 2018, defines very strict requirements for applications that process *personal data* at a technical, legal and organizational level. This way the GDPR regulates all aspects of *personal data processing* and ensures a transparent and fair process with respect to the *data subject* is strictly adhered to. Since the introduction of the GDPR similar legislation has been introduced in other countries and regions, such that its principles and requirements have been adopted far beyond the borders of the European Union. Whereas the GDPR assumes *data protection* principles to be applied "by design and by default" in *data processing applications*, we still find ourselves confronted with lengthy privacy policies, applications employing dark patterns to trick us into giving consent and cumbersome processes for obtaining the promised transparency as to how our *personal data* is being processed.

The principles of the Solid project with respect to control over our personal data as well as the transparent storage and processing it could enable fit seamlessly with the goals of the legislator while drafting the GDPR, namely enabling *data governance* and *data protection*. In contrast, the Web Access Control specification lacks the granularity and contextual awareness needed to enforce the regulatory requirements defined under legislation like the GDPR. Recent work by the Solid Application Interoperability panel, a panel in the Solid community, combined with research concerning the use of the semantic *data usage policy languages* ODRL and SPECIAL in the context of Solid could provide a solution to the limitations of the current standard.

In this thesis we will be identifying the shortcomings of the existing Solid specification concerning authorization and authentication in the context of applications which process personal data, where we will be using the GDPR as a reference for this evaluation. Based on our findings, a reference architecture is proposed in order to bring Solid's authorization mechanism in compliance with the relevant regulatory requirements, while improving end-user and developer experiences. Finally, we discuss a reference implementation of this architecture based on the Solid Application Interoperability draft specification and the Community Solid Server, which we also use to highlight further work concerning semantic modeling, efficiency and information security that is necessary for this architecture to become viable in practice.

Enforcing Data Protection in Solid: A Policy-Oriented Framework

Laurens Debackere

Abstract—**The Solid project aims to restore end-users' control over their data by decoupling services and applications from data storage. To realize data governance by the user, the Solid Protocol 0.9 relies on Web Access Control, which has limited expressivity and interpretability. In contrast, recent privacy and data protection regulations impose strict requirements on personal data processing applications and the scope of their operation. The Web Access Control mechanism lacks the granularity and contextual awareness needed to enforce these regulatory requirements. Therefore, we suggest a reference architecture for relating Solid's low-level technical access control rules with higher-level concepts such as the legal basis and purpose for data processing, the abstract types of information being processed, and the data sharing preferences of the data subject. Our architecture combines recent technical efforts by the Solid community panels with prior proposals made by researchers on the use of ODRL and SPECIAL policies as an extension to Solid's authorization mechanism. In implementing this architecture we were able to demonstrate its feasibility, even though performance optimizations will be required to enable its use in a practical setting. Nevertheless, our architecture shows promise in enabling data usage policies to resolve shortcomings of the access control mechanisms in Solid when implementing data processing based on consent.**

Index Terms—**Solid, Consent, Semantic Web, ODRL, Access Control**

I. Introduction

The Solid project[1] aims to realize Tim Berners-Lee's vision on decoupling personal data storage from the apps and services that use it, in order to return control and data governance to the user. Ultimately, Solid aims to re-establish a proper balance of power between service providers and their users [1], by providing the latter with the tools to make their own choices in data sharing and storage rather than having their data exist out of sight and out of control. To that end, the Solid community is developing a draft specification for decentralized personal data storage servers, also referred to as *Pods*.

At its core, the Solid Protocol version 0.9 [2] has three crucial building blocks that make up most of its footprint:

1) Solid implements parts of the **Linked Data Platform** W3C recommendation [3] to allow for read/write-access to the resources stored in a Pod with specific affordances for handling Linked Data.
2) Solid proposes **WebIDs** [4] and **Solid OIDC** [5] for *identification* and *authentication* purposes respectively. Through these standards, agents can be linked to a decentralized identifier expressing information on them like the agent's trusted identity providers. This allows for authentication between resource and authorization servers that have no prior trust relation.
3) **Web Access Control** [6] provides the critical controls over sharing of information stored in the Pod. Web Access Control is a cross-domain, decentralized solution for *authorizing* requests using *Access Control Lists* (ACLs) expressed as Linked Data. It identifies both *agents* and *resources* through the use of IRIs. Notably, ACLs can both be defined specifically for a given resource, or be inherited from a parent container.

While largely a refinement of existing legal frameworks like the 1995 Data Protection Directive and the 2005 ePrivacy Directive, the EU's General Data Protection Regulation[2] [7] set a major legislative milestone in the realm of data protection and privacy law when it entered into force in 2018.

It afforded *data subjects* with both transparency and greater control regarding the processing of their personal data by *data controllers* and took new and emerging technologies such as Big Data, AI, and the internet explicitly into account when it was first drafted. While far from perfect [8], it bestows a much greater deal of autonomy upon the *data subject* when making decisions regarding the processing of their personal data than has previously been the case.

One of the major shortcomings of the GDPR regulation boils down to the legal basis of consent and how it is typically realized on the Web [9], [10], [8]. In Article 6 of the GDPR, the six possible grounds for lawful data processing are laid out by the legislator, one of these being a *freely given consent* that can be withdrawn by the data subject at any time. The informedness of the data subject when giving their consent is emphasized greatly in the GDPR, meaning that a data subject should be able to accurately assess the consequences of the data processing to which they are consenting. In practice, the way consent is used by many services neither constitutes *consent* nor can it be considered *informed*. Most often the information required by Articles 13[3] and 14[4] of the GDPR is hidden away in lengthy privacy policies, which the data subject would have to read in their entirety to fully grasp the impact of their consent.

The prevalence of dark patterns on the internet [11], that are used to obtain the consent of a data subject, highlights a clear issue with respect to how this legal basis is being employed in practice. Today, the act of giving consent in an online setting is mostly a unilateral activity, where the data controller sets out

[1]https://solidproject.org

[2]http://data.europa.eu/eli/reg/2016/679/oj

[3]Article 13, "Information to be provided where personal data are collected from the data subject"

[4]Article 14, "Information to be provided where personal data have not been obtained from the data subject"

the conditions and the data subject has little impact on what information is being processed and how it is being handled. Solid's model for returning a user's control over their personal data might tip the scales in favor of the data subject when negotiating with a data controller in the context of consent. While in typical online service relationships, a data subject has little negotiating power and consent becomes a take-it-or-leave-it offer more often than not, Solid allows the *data subject* to have a clear overview of what data their Pod contains and granularly control with whom they share it. Therefore, it could bring crucial bilateral protections that consent depends upon in order to be used as intended by the legislator in data processing applications.

A. Motivation

While Solid has the potential to become a major driver for realizing the vision of informed consent as a legal basis for data processing as it was envisioned by legislators, several technical shortcomings still exist. As described earlier, Solid's current access control mechanisms, while suited for simple use cases, lack interpretability for average users and only capture very limited information on the identity of the parties involved, the data being exchanged, and the purpose and legal basis of this exchange. Furthermore, only limited analysis of how data sharing patterns and required legal safeguards can be implemented in Solid has happened so far [12].

The core idea of using *policies* in modeling and enforcing security and data privacy requirements for the Semantic Web has been the subject of prior work [13], [14]. Some extensions to the access control mechanisms in Solid based on the use of policy languages have already been proposed, such as the use of ODRL policies [15], [16] or the SPECIAL policy language[5] [17]. While these address some concerns with regard to interpretability and flexibility raised above, they also inherit or worsen some of the flaws of Solid's ACL mechanism. Issues include poor interpretability due to rule inheritance, increased runtime complexity of the authorizing process, and limited abstractions for identifying resources. Furthermore, the process by which a data controller requests the explicit consent and how this consent is then materialized in the new ACL policies needs to be considered in order to address the current shortcomings of Solid's ACL-based authorizations in a data processing context.

There is a distinction between the technical and end-user perspectives when using explicit consent as the basis for accessing resources in the data subject's Solid Pod. Whereas end-users need to understand *what data* they are sharing, *with whom*, for *what purpose* and *in which ways* these data are to be processed, a developer should not have to consider how their interactions map to these user-interpretable concepts. Rather we want developers to interact with the existing technical concepts from the Solid specification while having the Solid Pod or an intermediary validate whether these interactions are covered by a prior consent (or perhaps even some other legal basis). Therefore, we will define an architecture allowing for the decoupling of the legal and end-user interactions regarding consent from the technical interactions that were authorized by it.

[5]https://ai.wu.ac.at/policies/policylanguage/

Our contributions through this paper can be summarized as follows:

- defining the shortcomings of Solid's existing access control mechanism;
- presenting a framework reconciling end-user and legal requirements for data processing with Solid's existing access control model;
- implementing parts of this architecture in order to assess its practical feasibility and performance characteristics.

This paper continues as follows: Section II provides background information on the state of the art regarding Solid's authorization mechanism and briefly introduces concepts and standards that will be used throughout the rest of this paper. Section III details our proposed architecture, its interaction patterns and primary data structures. Section IV touches upon our implementation of the technical realm of this architecture and discusses its performance characteristics. Section V summarizes the reasoning behind this architecture, provides a brief interpretation of our results and discusses further work needed for the proposal to become viable in practice.

II. Background

In this section we will introduce the state of the art with respect to authorization and access control in Solid. Additionally we will present some of the technologies relevant to the architecture we are defining and evaluating in this paper.

A. Authorization in Solid

Solid's primary mechanism for authorizations is the Web Access Control (WAC) specification [6]. It employs the ACL ontology[6] to express *access modes* applicable to some resource for an agent, where both the agent and the resource are identified using IRIs. WAC supports four access modes in its rules, namely:

- **Read** Allowing for full or partial read operations on resources.
- **Write** Allowing for write operations on resources, i.e., create, update, or delete.
- **Append** Allowing for append operations on resources, i.e., to add information to the resource but not remove any.
- **Control** Allowing for read and write operations on the resource's *associated ACL resource*. This permits the grantee to delegate or revoke access to the resource.

Notably, these access modes are broad and do not map well to the more common CRUD[7] permission model [18]. Also, some of these access modes will align poorly with user expectations: e.g., what does it mean to have Append permissions over a container of resources?

The use of IRIs to both identify resources and agents might also contribute to poor user experience and lead to security

[6]http://www.w3.org/ns/auth/acl

[7]Acronym for Create, Read, Update, Delete.

breaches. For example, users might perceive an analogy between how they would typically manage a photo collection in a filesystem on a computer, and how pictures are stored in a folder in one's Pod. That way, an end-user could have some understanding of what kind of data are being shared, as they can easily open the files and look at their contents. However, the analogy falls short when it comes to *structured* data, which is commonly persisted as Linked Data in the Solid Pods. In this case, resource IRIs do not necessarily have meaning, and the organization of resources can be chosen arbitrarily by application developers. A similar concern is applicable to agent IRIs: How do I know my doctor's IRI is actually `https://nhs.gov.uk/id/123#me`? According to the UK Government's Department for Digital, Culture, Media & Sport's 2020 Cyber Security Breaches Survey [19] phishing attacks are one of the most common type of breaches experienced by UK businesses. Being just ordinary IRIs in the context of ACL rules, WebIDs suffer the same risk of being used in phishing attacks, where very similar looking WebIDs could be constructed that open the doors of your Pod to malicious actors. Detection mechanisms for phishing IRIs have been proposed, however these fall largely in the realm of heuristics.[20]

B. The Data Privacy Vocabulary

The Data Privacy Vocabulary[8] (DPV) [21] is a vocabulary that attempts to translate concepts and requirements related to the processing of personal information under data processing and privacy regulations, like GDPR, into classes and properties that can be used as Linked Data. It is structured to be extendable with concepts and requirements for specific jurisdictions, like the DPV-GDPR extension[9] that defines the GDPR-specific rights and legal bases concerning data processing.

C. Prior proposals for improving authorization in Solid

The Open Digital Rights Language (ODRL) [16] is a language for expressing policies that define permitted and prohibited actions over some entities. An ODRL profile and algorithm was proposed [15] as an extension of the existing ACL-mechanism used by Solid Pods to authorize requests. Furthermore, obligations and constraints can be imposed upon these actions. The proposed ODRL profile[10] enables the use of concepts from the Data Privacy Vocabulary in order to define policies that relate to data processing over some resources. The proposal also contextualizes the use of such policies for materializing complex data sharing preferences and legal bases for processing like informed consent. The authors [15] do highlight some significant challenges with their proposal, such as the efficiency of compliance checking with these ODRL policies, especially when used in a heterogeneous, decentralized architecture and combined with an inheritance mechanism, as well as the privacy risks associated with making these policies publicly accessible.

[8]https://w3id.org/dpv#

[9]http://www.w3id.org/dpv/dpv-gdpr#

[10]https://w3id.org/oac/

While evaluating different technical approaches that support the enforcement of legal rights given to data subjects under data protection regulation like GDPR, an assessment [17] was made of the affordances with respect to data governance provided by Solid and the SPECIAL project[11] in comparison to the current defacto standard of data subjects giving very broad consent to processing. In the evaluation of Solid in relation to data protection regulations it was found that Solid's current ACL-based solution for authorization falls short when trying to implement solutions adhering to the strict regulatory requirements set forth. Firstly, because of its poor user experience caused by issues like the lacking interpretability of access mode and resource identifiers for non-technical users, risk of phishing attacks due to the use of IRIs to identify agents, and the security concerns that arise from inherited ACL rules. Secondly, because ACLs fail to capture important concepts under data protection regulations that define what type of information is being shared, how that data will be processed and for what purpose, and which legal basis is used to warrant this processing. And lastly because the burden of modifying these ACL rules is currently delegated to application developers themselves, thus contradicting the original goals of returning control back to the end-user as developers have unlimited authority when modifying ACL rules and could resort to the dark patterns that have haunted modern-day implementations of consent on the web.

A layered, decentralized architecture for combining SPECIAL and Solid was also proposed and compared to these other approaches [17]. The concrete mechanics of the policy exchange and negotiation are left as future work by the authors, however their evaluation provides a good insight into the existing limitations of ACL based authorization when confronted with complex data processing applications.

D. Solid's Data Interoperability Panel

The Data Interoperability Panel within the Solid Community Panels[12] was started with the goal of standardizing the mechanics by which multiple applications can interoperate over the same data safely and effectively. In the process they try to increase user awareness and interpretability of the data stored in a Pod, by abstracting away complexities such as resource organization that are currently not governed by the Solid protocol, to finally enable multiple agents to safely and effectively interoperate over the same data. Most importantly, they aim to tackle these hurdles while preserving the fundamentals of the Solid protocol as it exists today.

In the context of the panel, two significant proposals have begun to take shape over the past year, namely the *Shape Trees* [22] and the Solid Application Interoperability [23]

[11]The SPECIAL project was a research project that aimed to deliver technologies to reconcile big data applications with the necessary regulatory compliance with respect to data processing. It delivered user interfaces for consent and processing transparency as well as ontologies for the logging by data processing applications and for modeling data usage policies of both data subjects and data controllers that are machine verifiable. (https://specialprivacy.ercim.eu)

[12]The Solid specification is drafted by different community panels, each focused on specific issues or domains that are relevant to Solid like authentication, authorization or data interoperability.

draft specifications. The former builds upon the existing specifications of RDF[13] and data shapes [24], [25], which respectively provide us with the foundations for interoperability through unambiguous identifiers (IRIs) and a structural schema against which individual RDF graphs can be validated. Where these existing specifications fall short however is in modeling complex resource hierarchies. Consider for example the organization of a collection of medical records that takes form in a Solid Pod where developers have relative freedom in both resource naming and the use of containers to gather their data. A Shape Tree defines structural constraints for a tree of resources in any ecosystem that has a notion of containers[14]. For each container, it allows shape constraints to be imposed on the contained resources. Shape Trees themselves can also contain other Shape Trees giving form to tree hierarchies (for example medical records as a whole may consist of medical images, prescriptions, bills, reports, etc.). The major strength of Shape Trees is that they can unambiguously define resource organization in a Pod and provide a higher-level abstraction that can be more easily understood by end-users. This way, Shape Trees guide applications and users by determining where data should be written to and where it can be read from. The modeling of related resource collections in this manner allows us to perform operations such as authorization, data migration and validation on this higher abstraction level as well. Especially in the context of authorization, defining rules at the level of Shape Trees rather than individual resources reduces complexity, the likelihood of errors and allows us to relate these higher-level conceptual resource aggregations to legal concepts such as Data Categories.

The Solid Application Interoperability (SAI) draft specification [23] leverages these proposed Shape Trees to standardize concrete mechanics by which applications and agents request access to information in a Solid Pod, the way by which they locate the concrete instances of the Shape Trees, and how they can interoperate over these. Up until now most of the specifics of these different operations were left open to individual application developers by the Solid specification, complicating interoperability over the same data. In the context of this paper, the standardizing of access requests is of specific importance, and will be used as a building block in our proposal. The SAI specification introduces the concept of an Authorization Agent as a service linked to an agent's WebID that manages the data under their control. It is tasked with processing access requests for the agent, managing previously granted permissions, and recording the concrete instances of Shape Trees through a collection of registries. While the specification is still under discussion by the panel, and some aspects of the mechanics of the authorization agent have not yet been fully defined or are deliberately being left open for implementation, we will be using many of the core concepts it sets forth in our proposal.

[13]The Resource Descriptor Format, core data model used in Semantic Web technologies to construct Linked Data resources.

[14]Solid builds upon the Linked Data Platform specification which governs the semantics of a container resource.

E. Linked Data Integrity & Authentication

The Data Integrity 1.0 draft community report [26] is a recent proposal by the W3C's Credentials Community Group, with the aim of providing authentication and data integrity capabilities to Linked Data resources through the use of mathematical proofs such as digital signature algorithms. It details a vocabulary for describing proof types, verification methods and algorithms. The origins of this work are to be found in the W3C's recommendation of the Verifiable Credentials Data Model [27], a data model that can be used to assert specific claims on a subject (such as a degree, driver's license, etc.) and which should be accompanied by a cryptographic proof that can assert their authenticity and integrity. These techniques will provide us with the necessary building blocks, in terms of authentication and accountability, we need to realize our proposed authorization architecture.

III. Architecture

As we have highlighted before, the Solid Protocol 0.9 [2] relies on the Web Access Control specification [6] as a mechanism for discretionary access control over the resources stored in a Solid Pod. While it offers adequate affordances for simple use cases related to authorization in social contexts, some of its capabilities and design choices may be problematic in more complex data processing applications that are governed by regulations like the GDPR. In contrast, our architecture splits out the implementation of consent as a legal basis for accessing personal data in the Solid Pod into two domains, shown in Fig. 1, where policies stored in the subject's Solid Pod form an interface between these different realms:

1) On the one hand, the *end-user domain* is governed by a so-called *Access Management Application* which is tasked with validating the data processing request coming from the responsible data controller against applicable legal requirements, end-user data sharing preferences and, if the processing request is approved, storing it as a Processing Grant in the data subject's Solid Pod.
2) On the other hand, the *technical domain* uses the *Authorization Agent*, as proposed by the Solid Application Interoperability specification, to handle concrete access requests made by applications and other agents in terms of Shape Trees, Data Shapes and ACL access modes. The interface between the two realms is formed by Processing Grants which are generated by the Access Management App and persisted in the agent's Solid Pod.

For authentication and identification of the different actors in the architecture we depend on the WebID [4] and Solid OIDC 0.1.0 [5] specifications that are defined within the Solid Protocol version 0.9 [2]. In the following paragraphs we will be expanding upon both the Access Management App, Authorization Agent and the proposed concepts of Processing Requests and Processing Grants used to bind these two services.

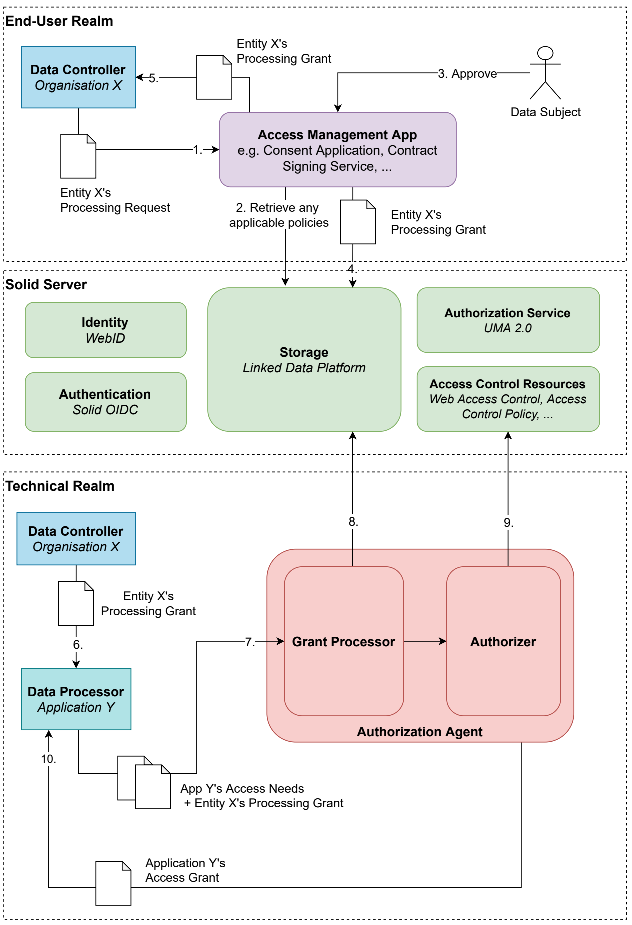


Fig. 1. Overview of our proposed architecture, linking an End-User realm governing Data Processing permissions with a Technical realm following the Application Interoperability specification

A. End-User Realm: Access Management App

The Access Management App is used by Data Controllers to obtain the necessary approval for the Data Processing they are requesting for some personal data categories and processing actions in fulfillment of a processing purpose that was allowed for through a specific legal basis. Once it has received a Data Processing Request, the Access Management App will first verify if the request is admissible and will attempt to match it against any explicit data sharing preferences the user might have in their Pod that can lead to an automatic granting of the request. If no preferences turn out to explicitly match the request, the data subject must be polled for their explicit consent. Once a Processing Request is granted, it is stored as a Processing Grant in the Solid Pod and delivered to the inbox [28] of the Data Controller.

B. End-User Realm: Processing Requests and Processing Grants

Whenever a Data Controller (Requesting Party) wants to obtain permissions for performing some data processing on the data subject, it will be constructing a Processing Request. This Request is constructed based upon a proposed ODRL

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/>.
@prefix dpv: <http://www.w3.org/ns/dpv#>.
@prefix cert: <https://www.w3.org/ns/auth/cert#>.
@prefix oac: <https://w3id.org/oac/>.

@prefix : <https://example.com/#>.

:medicalRecordsConsent a odrl:Policy, dpv:
     PersonalDataHandling;
    odrl:profile oac:;
    dpv:hasLegalBasis [
        a dpv:Consent;
    ];
    dpv:hasDataController <https://example.com/id/doctor#me
    >;
    odrl:permission [
        a odrl:Permission;
        odrl:assignee <https://example.com/id/doctor#me>;
        odrl:target dpv:HealthRecord, dpv:Prescription, dpv:
     HealthHistory;
        odrl:action dpv:Collect, dpv:Consult, dpv:Analyse,
     dpv:Alter;
        odrl:constraint [
            odrl:leftOperand oac:Purpose;
            odrl:operator odrl:isA;
            odrl:rightOperand :MedicalConsultation
        ]
    ].
```

Listing 1. Example Unsigned Processing Grant Request

profile [15] and concepts from the Data Privacy Vocabulary. An example request for medical records based upon explicit consent is shown in Listing 1. The request details handling of the personal data, in terms of legal basis (in the case of this paper we will only consider explicit consent), data controller, and specific permissions that will be needed in the context of the processing.

Each permission specifies what personal data categories it concerns as a target, what actions it needs to perform on this data, and constraints on the purpose or output of the processing. Other constraints could be envisioned as well, like technical measures used in the processing and associated risks, however these haven't been explored in the current proposal.

The *Data Processing Request* itself is presented to the Access Management App accompanied by a Data Integrity Proof that was generated by the Data Controller, this way the provenance and integrity of the request can be validated. Through this signing mechanism, the risk of spam or other malicious attacks with respect to the Access Management App and Processing Request procedures could be reduced, for example by assigning different trust levels to issuer services that can be used by Data Processors to sign their request based upon requirements like identity validation or regulatory compliance.

Finally, a *Processing Grant* is constructed from the Processing Request by first completing the legal basis, i.e., consent, with any other necessary attributes that were either gathered in interaction with the data subject or in an automated manner by the access management app. Thereafter any permissions that have not been witheld will be removed from the Grant, the RDF graph is supplemented with a Revocation Status attribute conforming to the W3C Revocation List 2020 specification [29] for revoking Data Integrity Proofs such that the Access Management App can revoke the Processing Grant at a later time, and a Data Integrity Proof will be created and signed by the Access Management App to indicate that the legal requirements for the data processing to be approved are fulfilled. The signed Processing Grant can be seen as an instruction for the authorization agent to provision certain types of information to a Data Controller and its designated processors.

C. Technical Realm: Authorization Agent

The Authorization Agent is largely based upon the proposed Solid Application Interoperability specification in terms of its semantics and API, which is still under discussion by the panel and thus might be subject to changes. This is enabled by the fact that mechanics of the authorization agent are largely left open to implementation, such that additional authorization checks can be executed between the *Access Needs* being presented to the authorization agent and the delivery of a so-called Access Grant that specifies the concrete data that has been elected for sharing with the application.

In fact, the only modification to the authorization agent interface that we are proposing in this paper is that a Processing Grant should accompany the Data Processor's access needs when access is being requested. This way the authorization agent can link the access request being made by the application or service, acting as a Data Processor for the Data Controller, to a valid legal basis for data processing. It then becomes the task of a Grant Processor module in the Authorization Agent to match the specified Processing Grant to the Processor's Access Needs in terms of Data Needs (Shape Trees) and Access Modes. The latter confronts us with the need for an unambiguous equivalence relation between the abstract definitions in the Processing Grant and their technical counterparts in the Access Needs.

Finally, once the *Grant Processor* has determined that the Data Processor's request actually matches our initial Processing Grant, it can proceed with an Authorizer that is tasked with modifying the atomic access control rules applicable to the instances of the Shape Trees that were specified in the service's Data Needs. Once this process has ended, an Access Grant is returned to the Data Processor and the necessary registrations are added to the Pod.

D. Auxiliary Rules & Policies

While the ODRL-based processing request and processing grant may suffice for defining the data processing that is being requested and approved on a business-level, it is insufficient for the authorization agent to relate these with the technical access needs specified by a Data Processor like an application. The semantic gap here is twofold, on the one hand we need to unambiguously define what data in the Pod falls under the approved processing and on the other hand we must know what actions on this data are permitted.

Firstly, the abstract data categories used to specify the personal data being shared under the approved data processing activities must be related to concrete technical data type information. As was elaborated upon in the background section, the combination of data shapes and shape trees as a mechanism for defining resource collections and their structure allows us to delimit conceptually related resources in the Pod like medical records, pictures, notes, etc. Through an additional set of rules that is configured by the data subject in their Solid Pod, a so-called Data Category Equivalence policy, we link the technical resource type information provided by Data Shapes and Shape Trees to Personal Data Categories as they are specified under DPV and used in the ODRL profile.

As higher level abstractions are used to define the actions that the processing allows for, we must also relate the Processing categories from the DPV with the Access Modes as they are used in both Solid's Access Control mechanism and the technical Access Needs specified by the Data Processor. These can be defined by the subject as Processing Access Needs, which are stored as an additional set of rules in the Pod.

Furthermore, while not elaborated upon in this paper, the proposed ODRL profile [15] was devised with the concept of data sharing preferences which allowed for the data subject to also express more complex data processing activities that could automatically be permitted to some requesting party based on purpose, data and processing categories. Such policies could also be persisted in the Solid Pod besides these previously noted equivalence relations and the concrete processing grants that flow from them.

IV. Implementation

In order to assess the performance of our architecture, an implementation[15] of the technical realm presented in section III-C was developed in TypeScript using the Components.JS framework [30]. Components.JS permitted us to develop a modular architecture, both with respect to the authorization logic being implemented as well as for the support of different capabilities defined in the Solid Application Interoperability specification. Furthermore, we rely on the Community Solid Server v4.0.1[16] to provide us with a modular Solid server instance that is compliant with the Solid protocol 0.9 [2].

The implementation itself takes advantage of the recently introduced concept of a UMA Authorization Service [31] in the Solid-OIDC 0.1.0 specification [5], which allows for the decoupling of authorization logic from the actual Solid server implementation. Nevertheless, a performance penalty is incurred because of the additional communication overhead involved. Through modular interfaces for respectively authentication and authorization capabilities, our Authorization Service can serve as a basis for research into different mechanisms for access control in Solid.

The modularity of this Authorization Service enabled us to implement authorization logic based on the Solid Application Interoperability draft specification. Whenever an Access Grant is provided by the Authorization Agent to a *data processor*, this information is recorded in the resource owner's Registry Set. The information contained in these registries is used by our authorization modules to subsequently authorize the incoming requests from a designated *data processor*.

[15]https://github.com/laurensdeb/interoperability

[16]https://github.com/CommunitySolidServer/CommunitySolidServer

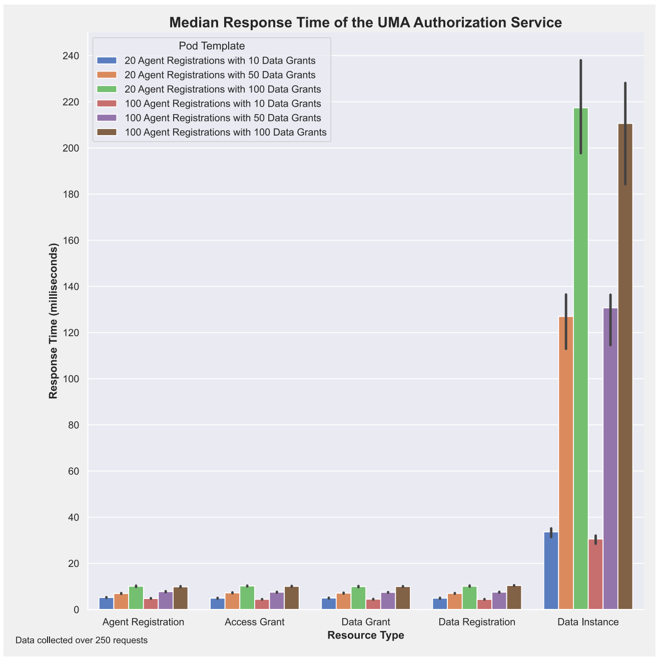


Fig. 2. Median response times by resource type for a token request to the UMA Authorization Service in different configurations of the Solid Pod. Data collected over 250 requests per configuration, 95% confidence intervals are marked in black. Validated using the 1.0.0 release of the implementation (Node v16.14.0 / Apple M1 Pro / 32GB RAM).

The implementation was benchmarked using a synthetically constructed Solid Pod template, of varying size in terms of Shape Tree instances and registered agents (*processors*). The benchmark itself consisted of 250 authorization requests for the various resource types that are defined under the Application Interoperability draft specification [23], where we measured the total time to authorize a request with the UMA Authorization Service. The results of our evaluation are shown in Fig. 2. While auxiliary data structures like Agent Registrations, Access Grants, Data Grants and Data Registrations only see a very limited impact in terms of median response times as we scale the dimensions of the Solid Pod, the Data Instances that store the actual information to which access is being authorized start showing much poorer performance. This leads us to conclude that the time complexity of authorizations for this resource type is heavily dependent on the amount of information to which access is being authorized, at least in the current implementation and under the current revision of the specification [23].

In addition to the implementation of a UMA Authorization Service, we also developed a modular Authorization Agent that can bridge the gap between the technical and end-user realms of our architecture. The Authorization Agent allows for the discovery of what resources an agent is allowed to access, and can process new requests based on Access Needs of the application or service in terms of Shape Trees [22]. Interfaces were also defined for integrating data usage policy languages, like ODRL or SPECIAL, into this component such that it can evaluate whether incoming access requests match a prior Processing Grant given through an Access Management Application.

V. Conclusion & Future Work

One of the major departures from previous proposals is that our proposed reference architecture aims to separate the problem of reconciling technical authorization with the legal requirements for data processing into two distinct domains. On the one hand, there is an end-user realm where the user is presented with requests for data processing in terms of processing actions happening on more abstract data categories, and where an end-user can determine explicit data sharing preferences. On the other hand, there is a technical realm where Solid's proposed Application Interoperability Specification governs how application developers can gain access to resources in an agent's Solid Pod, once a proper legal basis for processing has been established. The concept of Data Processing Grants that are verifiable through the W3C's Data Integrity Specification for Linked Data form the link between these two distinct domains, combined with policies that relate the meaning of Data Categories and Processing Actions to technical concepts that can be understood by the Solid Pod.

Whereas previous solutions aimed to integrate business concepts into Solid's existing authorization mechanisms, for example through expanding upon Access Control with purpose and policy concepts, our proposal takes advantage of the abstractions enabled by Application Interoperability specification and defines a layered architecture on top of it to introduce business and legal concepts.

Performance benchmarks of an implementation of the crucial authorization mechanism used by the technical realm of the architecture do show elevated response latencies as the dimensions of the Pod, in terms of data to which access is granted, increase. Nevertheless, our implementation was able to demonstrate the technical feasibility of authorization based on the abstractions offered by Shape Trees, as was proposed in the Solid Application Interoperability specification [23].

Some of the highlighted challenges in the ODRL proposal [15] have been addressed here, while others, like performance optimizations, will necessitate further consideration. By introducing the access management app and authorization agent as dynamic negotiators in the request flow we avoided the need for public policies to be advertised by the Solid Pod, which addresses important privacy concerns with a policy based solution (i.e., what if anyone can see that you have shared your medical records for the purpose of a past treatment with a psychiatrist). Additionally, by introducing the Authorization Agent as an intermediary for providing the technical access, we avoid the necessity of matching policies for each HTTP request on a resource in the Pod. Specific legal issues raised in the ODRL proposal [15] remain largely applicable to our proposal as well, i.e., the legal implications of user choice enabled by Solid's novel approach to data governance, the necessity of awareness of the applicable jurisdiction and its requirements for data processing activities and whether data sharing preferences, as were briefly touched upon, indeed constitute a form of consent.

While we have focused largely on the problem space of implementing explicit consent as a legal basis throughout this proposal, there could be room for enabling other legal bases to be enforced by the access management app as well. For example when processing is requested on the basis of a contractual obligation, the Access Management App could retrieve the contract from the subject's Pod and validate it against the identity of the requesting party for the data processing.

Acknowledgment

Parts of this work are based on the paper "A Policy-Oriented Architecture for Enforcing Consent in Solid" by Debackere et al.[32].

References

[1] R. Verborgh, "Re-decentralizing the Web, for good this time," in *Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*, O. Seneviratne and J. Hendler, Eds. ACM, 2022. [Online]. Available: https://ruben.verborgh.org/articles/redecentralizing-the-web/

[2] S. Capadisli, T. Berners-Lee, R. Verborgh, and K. Kjernsmo, "Solid protocol," Tech. Rep., Dec. 2021. [Online]. Available: https://solidproject.org/TR/2021/protocol-20211217

[3] S. Speicher, J. Arwe, and A. Malhotra, "Linked Data Platform 1.0," Tech. Rep., Feb. 2015. [Online]. Available: https://www.w3.org/TR/ldp/

[4] A. Sambra, H. Story, and T. Berners-Lee, "WebID 1.0," Tech. Rep., Mar. 2014. [Online]. Available: https://www.w3.org/2005/Incubator/webid/spec/identity/

[5] A. Coburn, elf Pavlik, and D. Zagidulin, "Solid-OIDC," Tech. Rep., Mar. 2022. [Online]. Available: https://solidproject.org/TR/2022/oidc-20220328

[6] S. Capadisli and T. Berners-Lee, "Web Access Control," Tech. Rep., Jul. 2021. [Online]. Available: https://solidproject.org/TR/wac

[7] "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)." *Official Journal of the European Union*, vol. L 119, 2016-04-27.

[8] "The GDPR: The Emperors New Clothes - On the Structural Shortcomings of Both the Old and the New Data Protection Law, author=Winfried Veil," *Consumer Law eJournal*, 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3305056

[9] M. Kretschmer, J. Pennekamp, and K. Wehrle, "Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web," *ACM Trans. Web*, vol. 15, no. 4, jul 2021. [Online]. Available: https://doi.org/10.1145/3466722

[10] G. G. Karcsony, "Managing personal data in a digital environment - did GDPRs concept of informed consent really give us control?" *International Conference on Computer Law, AI, Data Protection & the Biggest Tech Trens*, 2019. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3452573

[11] M. Nouwens, I. Liccardi, M. Veale, D. Karger, and L. Kagal, *Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence*. New York, NY, USA: Association for Computing Machinery, 2020, p. 113. [Online]. Available: https://doi.org/10.1145/3313831.3376321

[12] D. De Bot and T. Haegemans. (2021, Jan.) Data Sharing Patterns as a Tool to Tackle Legal Considerations about Data Reuse with Solid: Theory and Applications in Europe. [Online]. Available: https://lirias.kuleuven.be/retrieve/599839

[13] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *The Semantic Web - ISWC 2003*, D. Fensel, K. Sycara, and J. Mylopoulos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 402–418.

[14] P. A. Bonatti, S. Kirrane, I. M. Petrova, and L. Sauro, "Machine understandable policies and GDPR compliance checking," *CoRR*, vol. abs/2001.08930, 2020. [Online]. Available: https://link.springer.com/article/10.1007/s13218-020-00677-4

[15] B. Esteves, H. J. Pandit, and V. Rodrguez-Doncel, "ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2021, pp. 298–306.

[16] R. Iannella and S. Villata, "ODRL Information Model 2.2," Tech. Rep., Feb. 2018. [Online]. Available: https://www.w3.org/TR/odrl-model/

[17] G. Havur, M. Vander Sande, and S. Kirrane, "Greater Control and Transparency in Personal Data Processing," 01 2020, pp. 655–662.

[18] S. Villata, L. Costabello, N. Delaforge, and F. Gandon, "Social Semantic Web Access Control?" *Journal on Data Semantics*, vol. 2, 03 2012.

[19] UK Department for Digital, Culture, Media & Sport . (2020, Mar.) Cyber Security Breaches Survey 2020. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020

[20] M. Khonji, Y. Iraqi, and A. Jones, "Phishing Detection: A Literature Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013.

[21] H. J. Pandit, A. Polleres, B. Bos, R. Brennan, B. Bruegger, F. J. Ekaputra, J. D. Fernndez, R. G. Hamed, E. Kiesling, M. Lizar, and et al., "Creating a Vocabulary for Data Privacy: The First-Year Report of Data Privacy Vocabularies and Controls Community Group (DPVCG)," Oct 2019.

[22] J. Bingham and E. Prud'hommeaux, "Shape Trees Specification," Tech. Rep., Feb. 2022. [Online]. Available: https://shapetrees.org/TR/specification/

[23] J. Bingham, E. Prud'hommeaux, and elf Pavlik, "Solid Application Interoperability," Tech. Rep., May 2022. [Online]. Available: https://solid.github.io/data-interoperability-panel/specification/

[24] E. Prud'hommeau, I. Boneva, J. E. L. Gayo, and G. Kellogg, "Shape Expressions Language 2.1," Tech. Rep., Oct. 2019. [Online]. Available: http://shex.io/shex-semantics/index.html

[25] H. Knublauch and D. Kontokostas, "Shapes Constraint Language (SHACL)," Tech. Rep., Jul. 2017. [Online]. Available: https://www.w3.org/TR/shacl/

[26] M. Sporny and D. Longley, "Data Integrity 1.0," Tech. Rep., Jan. 2022. [Online]. Available: https://w3c-ccg.github.io/data-integrity-spec/

[27] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model v1.1," Tech. Rep., Nov. 2021. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[28] S. Capadisli and A. Guy, "Linked Data Notifications," Tech. Rep., May 2017. [Online]. Available: https://www.w3.org/TR/ldn/

[29] M. Sporny and D. Longley, "Revocation List 2020," Tech. Rep., Apr. 2021. [Online]. Available: https://w3c-ccg.github.io/vc-status-rl-2020/

[30] R. Taelman, J. Van Herwegen, M. Vander Sande, and R. Verborgh, "Components.js: Semantic Dependency Injection," *Semantic Web Journal*, 2022. [Online]. Available: https://linkedsoftwaredependencies.github.io/Article-System-Components/

[31] M. Machulak, J. Richer, and E. Maler, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," Tech. Rep., Jul. 2018. [Online]. Available: https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html

[32] L. Debackere, P. Colpaert, R. Taelman, and R. Verborgh, "A policy-oriented architecture for enforcing consent in Solid," in *Proceedings of the 2nd International Workshop on Consent Management in Online Services, Networks and Things*, Apr. 2022. [Online]. Available: https://www2022.thewebconf.org/PaperFiles/88.pdf

Afdwingen van Gegevensbescherming in Solid: Een Policy-gebaseerd Raamwerk

Laurens Debackere

Abstract—**Het Solid project tracht de controle van gebruikers over hun data te herstellen, door diensten en applicaties los te koppelen van hun data-opslag. Om dit beheer van persoonlijke data mogelijk te maken, gebruikt het Solid Protocol 0.9 de Web Access Control standaard, dewelke beperkt is qua expressiviteit en interpreteerbaarheid. Daarentegen stelt recente wetgeving rond privacy en gegevensbescherming strikte eisen aan toepassingen en diensten die persoonsgegevens verwerken en hun werkingsgebied. Binnen het Web Access Control mechanisme ontbreekt de vereiste granulariteit en contextuele informatie die nodig is om aan deze wettelijke vereisten te voldoen. Daarom presenteren we in dit werk een referentie-architectuur, die toelaat om Solid's elementaire autorisatie mechanisme te verbinden met bredere, juridische concepten zoals de wettelijke basis en het doel van de gegevensverwerking, de categorieën van persoonsgegevens die men wil verwerken, en de voorkeuren rond gegevensdeling van het data subject. Onze architectuur brengt recent technisch werk uit de Solid community panels samen met eerder onderzoek over het gebruik van de ODRL en SPECIAL *policy languages* als een extensie op het *access control* mechanisme van Solid. Aan de hand van een implementatie van deze architectuur konden we de technische haalbaarheid van het voorstel aantonen, hoewel optimalisaties op vlak van performantie een vereiste zullen zijn om ons raamwerk ook in de praktijk te gebruiken. De architectuur lijkt evenwel een beloftevolle poging te zijn om aan de hand van *data usage policies* tekortkomingen te verhelpen in de mechanismen voor toegangsbeheer binnen Solid, bij het implementeren van gegevensverwerking die onderhevig is aan juridische vereisten, zoals gesteld door de AVG.**

Index Terms—**Solid, Consent, Semantic Web, ODRL, Access Control**

I. Inleiding

Het Solid project[1] tracht de visie van Tim Berners-Lee te realiseren om de opslag van persoonlijke data los te koppelen van de applicaties en diensten die gebruik maken van deze gegevens, om op die manier gebruikers in staat te stellen om zelf hun data te beheren. Zo probeert Solid een meer evenwichtige machtsverhouding tussen dienstverleners en gebruikers te bekomen, door die laatsten de middelen te geven om hun eigen keuzes te maken omtrent gegevensopslag en -deling in plaats van hun data buiten hun bereik en controle te laten bewaren. Hiertoe ontwikkelt de Solid gemeenschap een ontwerp-specificatie voor decentrale persoonlijke datakluizen, ook wel *Pods* genoemd. In de kern bestaat dit Solid protocol versie 0.9 uit drie cruciale bouwstenen:

1) Solid implementeert delen van de **Linked Data Platform** W3C *recommendation* [1] om lees- en schrijftoegang mogelijk te maken voor gegevens die opgeslagen worden in de Pod, met specifieke functionaliteiten voor het verwerken van Linked Data.
2) Solid stelt het gebruik van **WebIDs** [2] en **Solid OIDC** [3] voor om *identificatie* en *authenticatie* mogelijk te maken. Door middel van deze standaarden kunnen partijen gelinkt worden aan een decentrale *identifier* die gegevens over hun uitdrukt, zoals de vertrouwde *identity providers*. Het laat toe om authenticatie te voorzien tussen *resource* en autorisatie servers die geen eerdere vertrouwensband hebben.
3) **Web Access Control** [4] maakt de cruciale controle over het delen van informatie in de Pod mogelijk. Web Access Control is een *cross-domain*, decentrale oplossing voor het autoriseren van verzoeken op basis van *Access Control Lists* (ACLs) die uitgedrukt worden als Linked Data. Het identificeert zowel gebruikers als data aan de hand van IRIs. In het bijzonder kunnen ACLs specifiek voor een bepaalde gegevensbron worden uitgedrukt of worden overgeërfd van een bovenliggende map.

Hoewel het grotendeels een verfijning betreft van bestaande juridische kaders zoals de Data Protection Directive uit 1995 en de ePrivacy Directive uit 2005, was de Algemene Verordening Gegevensbescherming[2] [5] een belangrijke juridische mijlpaal op vlak van wetgeving rond gegevensbescherming en privacy bij haar introductie in mei 2018[6].

De AVG voorziet voor een *data subject* meer transparantie en controle betreffende de verwerking van hun persoonlijke data door *data controllers*. Daarenboven heeft men bij het opstellen van deze verordening nieuwe en opkomende technologieën zoals *Big Data*, artificiële intelligentie, en het internet expliciet in beschouwing genomen. Hoewel de AVG ongetwijfeld tekortkomingen heeft [7], voorziet het in een veel grotere autonomie voor een *data subject* wanneer die beslissingen maakt over de verwerking van diens persoonsgegevens dan voorheen het geval was.

Eén van de voornaamste tekortkomingen met de AVG betreft de juridische basis van toestemming voor de verwerking van persoonsgegevens, en hoe deze typisch gebruikt wordt op het Web [8], [9], [7]. In artikel 6 van de AVG worden de zes mogelijke gronden voor rechtmatige verwerking van persoonsgegevens vastgelegd door de wetgever, vrijwillige toestemming die op elk moment ingetrokken kan worden door het *data subject* is één van deze gronden. Het informeren van het *data subject* wanneer die toestemming geeft wordt sterk benadrukt binnen de verordening, waarbij die accuraat moet kunnen inschatten wat de gevolgen zijn van de gegevensverwerking waarmee wordt ingestemd. Echter, in de praktijk implementeren diensten deze verwerkingsgrond op een manier

[1]https://solidproject.org

[2]http://data.europa.eu/eli/reg/2016/679/oj

die noch geïnformeerd is noch een eigenlijke toestemming inhoudt van het *data subject*. Vaak wordt immers door de AVG vereiste informatie verborgen in lange *privacy policies* die de gebruiker in zijn geheel zou moeten lezen om de impact van diens toestemming te kunnen inschatten [7].

De prevalentie van *dark patterns* op het Web [10], die gebruikt worden om de toestemming van een *data subject* te verkrijgen, wijst op een duidelijk probleem met betrekking tot hoe de verwerkingsgrond van geïnformeerde toestemming in de praktijk wordt gebruikt. Momenteel is het geven van een dergelijke toestemming in een online omgeving veelal een eenzijdige gebeurtenis, waarbij de data controller de voorwaarden uiteenzet en het *data subject* weinig impact heeft op hoe diens data wordt verwerkt en opgeslagen.

Het model wat Solid naar voren schuift om gebruikers opnieuw de controle te geven over hun persoonlijke data zou aanleiding kunnen geven tot een beter evenwicht in de machtsverhoudingen tussen *data subject* en *data controller*, wanneer die laatste vraagt om de toestemming van de gebruiker. Klassieke online dienstverlening biedt aan het *data subject* weinig mogelijkheden heeft om te onderhandelen waardoor toestemming eerder herleid wordt tot een alles-of-nietskwestie. Daarentegen maakt Solid het mogelijk voor een *data subject* om een duidelijk overzicht te krijgen van de data die hun Pod bevat en op granulaire wijze te kiezen welke data gedeeld wordt met wie. Hierdoor zou het cruciale, bilaterale beschermingen kunnen bieden die de verwerkingsgrond van geïnformeerde toestemming vereist om gebruikt te worden zoals bedoeld door de wetgever.

A. Motivatie

Hoewel Solid het potentieel heeft om een belangrijke drijfveer te vormen voor verwezenlijking van echte, geïnformeerde toestemming als rechtsgrondslag voor gegevensverwerking, zoals de wetgever voor ogen had, zijn er nog verschillende technische tekortkomingen die eerst verholpen moeten worden. Zoals eerder aangehaald zijn de huidige mechanismen voor autorisatie in Solid weliswaar geschikt voor eenvoudige gebruikssituaties, maar vertonen ze voor de gemiddelde gebruiker tekortkomingen op vlak van interpreteerbaarheid. Immers legt Web Access Control slechts in beperkte mate informatie vast over de identiteit van de betrokken partijen, de gegevens die worden uitgewisseld en het doel en de rechtsgrondslag van deze uitwisseling. Voorts is tot dusver slechts in beperkte mate geanalyseerd hoe patronen voor gegevensuitwisseling en de vereiste juridische waarborgen in Solid kunnen worden geïmplementeerd [11].

Het kernidee om *policies* te gebruiken bij het modelleren en afdwingen van veiligheids- en gegevensbeschermingsvereisten voor het Semantisch Web is al onderwerp geweest van eerder werk [12], [13]. Enkele uitbreidingen van de autorisatie mechanismen in Solid op basis van het gebruik van *policy languages* zijn reeds voorgesteld, zoals het gebruik van *ODRL-policies* [14], [15] of de SPECIAL *policy language*[3] [16]. Hoewel deze de hierboven genoemde bezorgdheden met betrekking tot interpreteerbaarheid en flexibiliteit deels wegnemen, erven of verergeren ze ook enkele van de gebreken van het ACL mechanisme van Solid. Deze problemen zijn onder andere slechte interpreteerbaarheid als gevolg van de overerving van toegangsregels, verhoogde uitvoeringstijd van het autorisatie proces, en beperkte abstracties voor het identificeren van data. Voorts moet het proces worden beschouwd waarbij een *controller* de geïnformeerde toestemming vraagt van een gebruiker en over de wijze waarop deze toestemming vervolgens in het nieuwe autorisatie-regels wordt gematerialiseerd, teneinde de huidige tekortkomingen van het autorisatie mechanisme in Solid te verhelpen bij de verwerking van persoonsgegevens.

[3]https://ai.wu.ac.at/policies/policylanguage/

Er is een belangrijk onderscheid tussen het technische en het eindgebruikersperspectief wanneer uitdrukkelijke toestemming wordt gebruikt als basis voor toegang tot gegevens in de Solid Pod van de betrokkene. Waar een eindgebruiker moet begrijpen *welke gegevens* zij delen, *met wie*, *met welk doel* en *op welke manier* deze gegevens zullen worden verwerkt, zou een ontwikkelaar niet moeten nadenken over hoe zijn interacties zich verhouden tot deze door de gebruiker begrijpbare concepten. We willen veeleer mogelijk maken dat ontwikkelaars interacties aangaan met de bestaande technische concepten van de Solid-specificatie terwijl de Solid Pod of een tussenpartij valideert of deze interacties onder een voorafgaande toestemming vallen (of misschien zelfs een andere rechtsgrond). Daarom definiëren we in dit paper een architectuur die het mogelijk maakt de juridische en eindgebruikersinteracties met betrekking tot toestemming los te koppelen van de technische interacties waarvoor toestemming is verleend.

Onze bijdragen in deze paper kunnen als volgt worden samengevat

- een overzicht bieden van de tekortkomingen van het bestaande *access control* mechanisme in Solid;
- het presenteren van een kader dat de eindgebruikers- en wettelijke vereisten voor gegevensverwerking verzoent met het bestaande autorisatie mechanisme van Solid;
- het implementeren van delen van deze architectuur om de praktische haalbaarheid in te schatten en de prestatiekenmerken te beoordelen.

Dit paper gaat verder als volgt: Paragraaf II geeft achtergrondinformatie over de stand van de techniek met betrekking tot het autorisatiemechanisme in Solid en introduceert kort de concepten en specificaties die in de rest van dit werk zullen worden gebruikt. In paragraaf III worden de door ons voorgestelde architectuur, de interactiepatronen en voornaamste gegevensstructuren beschreven. Vervolgens beschrijft paragraaf IV onze implementatie van het technische domein van deze architectuur en de prestatiekenmerken. Paragraaf V vat tot slot de redenering achter deze architectuur samen, geeft een korte interpretatie van onze resultaten en bespreekt het verdere werk dat nodig is om ons voorstel in de praktijk uitvoerbaar te maken.

II. Achtergrond

In dit hoofdstuk introduceren we de stand van de techniek met betrekking tot autorisatie en toegangscontrole in Solid. Daarnaast zullen we enkele van de technologieën presenteren die relevant zijn voor de architectuur die we in dit document definiëren en evalueren.

A. Autorisatie in Solid

Het primaire mechanisme van Solid voor autorisaties is de Web Access Control (WAC) specificatie [4]. Deze maakt gebruik van de ACL ontologie[4] om *access modes* uit te drukken die van toepassing zijn op een bestand of map voor een bepaalde *agent*, waarbij zowel de *agent* als de bron worden geïdentificeerd met behulp van IRI's. WAC ondersteunt vier toegangsmodi in deze toegangsregels, namelijk:

- **Read** Maakt volledige of gedeeltelijke leesbewerkingen op bronnen mogelijk.
- **Write** Maakt schrijfbewerkingen op bronnen mogelijk, i.e., creëren, bijwerken of verwijderen.
- **Append** Maakt het mogelijk om informatie toe te voegen aan bronnen, maar niet om gegevens te verwijderen.
- **Control** Staat lees- en schrijfbewerkingen toe op de *Access Control List* horende bij de bron. Dit laat de begunstigde toe om toegang tot de bron te delegeren of in te trekken.

Deze toegangsmodi zijn echter ruim gedefinieerd en sluiten niet goed aan bij het meer gebruikelijke CRUD-toestemmingsmodel[5] [17]. Ook zullen sommige van deze toegangsmodi niet steeds overeenkomen met de verwachtingen van de gebruiker: Bijvoorbeeld, wat betekent het om *Append* permissies te hebben op een map in de Pod?

Bovendien maakt WAC gebruik van een overervingsmechanisme om te bepalen welke *Access Control List* de effectieve ACL is voor een bestand of map in de Pod. Hoewel dit overervingsmechanisme redelijk eenvoudig te begrijpen is voor ontwikkelaars, kan het gedrag voor een onwetende eindgebruiker contra-intuïtief zijn of zelfs leiden tot onbedoelde onthulling van informatie. Als een gebruiker een applicatie bijvoorbeeld toegang verleent tot een map, verleent hij impliciet toegang tot alle gegevens en submappen die zich onder de map bevinden, inclusief nieuwe bronnen die zijn toegevoegd nadat de gebruiker toegang heeft verleend.

Het gebruik van IRI's om bronnen en gebruikers te identificeren kan evenzeer bijdragen aan een slechte gebruikerservaring en leiden tot privacy-inbreuken. Gebruikers zouden bijvoorbeeld een analogie kunnen zien tussen hoe zij gewoonlijk een fotoverzameling beheren in een bestandssysteem op een computer, en hoe foto's worden opgeslagen in een map op iemands Solid Pod. Daarbij kan een eindgebruiker enigszins begrijpen wat voor soort gegevens worden gedeeld, aangezien hij de bestanden gemakkelijk kan openen en de inhoud ervan kan bekijken. De analogie stopt echter wanneer het gaat om *gestructureerde* gegevens, die gewoonlijk als Linked Data in de Solid Pods worden opgeslagen. In dit geval hebben IRI's van bronnen niet noodzakelijkerwijs betekenis, en de organisatie van bronnen kan willekeurig worden gekozen door ontwikkelaars van toepassingen. Een soortgelijke zorg is van toepassing voor agent IRI's: Hoe weet ik dat de WebID van mijn dokter ook echt `https://nhs.gov.uk/id/123#me` is? Volgens onderzoek uit 2020 [18] van het Department for Digital, Culture, Media & Sport van de Britse regering, zijn phishing-aanvallen één van de meest voorkomende veiligheidsinbreuken die opgelopen worden door Britse bedrijven. WebID's zijn gewone IRI's in de context van ACL-regels, en lopen een gelijkaardig risico om te worden gebruikt in de context van phishingaanvallen, waarbij zeer gelijkaardig uitziende WebID's kunnen worden geconstrueerd die de deuren van de Pod openstellen voor kwaadwillende actoren. Detectiemechanismen voor phishing IRI's werden reeds eerder voorgesteld, maar deze vallen grotendeels in het domein van de heuristiek [19].

B. De Data Privacy Vocabulary

De Data Privacy Vocabulary[6] (DPV) [20] is een ontologie die concepten en vereisten met betrekking tot de verwerking van persoonsgegevens onder gegevensbeschermings- en privacywetgeving, zoals de AVG, tracht te vertalen naar klassen en eigenschappen die als Linked Data kunnen worden gebruikt. De ontologie is zo gestructureerd dat deze kan worden uitgebreid met concepten en definities voor specifieke jurisdicties, zoals de DPV-GDPR-uitbreiding[7] die de AVG-specifieke rechten en rechtsgrondslagen met betrekking tot gegevensverwerking definieert.

C. Eerdere voorstellen tot het verbeteren van autorisatie in Solid

De Open Digital Rights Language (ODRL) [15] is een taal voor het uitdrukken van *policies* die toegestane en verboden acties over bepaalde entiteiten definiëren. Een ODRL profiel en algoritme werd reeds voorgesteld [14] als een uitbreiding op het bestaande ACL-mechanisme dat door Solid Pods wordt gebruikt om verzoeken te autoriseren. Bovendien kunnen met deze techniek verplichtingen en beperkingen aan dergelijke acties worden opgelegd. Het voorgestelde ODRL-profiel[8] maakt het gebruik mogelijk van concepten uit de Data Privacy Vocabulary om *policies* te definiëren die betrekking hebben op gegevensverwerking over bepaalde bronnen. Het voorstel contextualiseert ook het gebruik van dergelijke *policies* voor het vastleggen van complexe voorkeuren voor het delen van gegevens en wettelijke grondslagen voor de verwerking zoals geïnformeerde toestemming. De auteurs [14] benadrukken enkele belangrijke uitdagingen met hun voorstel zoals de efficiëntie van de controle op de naleving van deze ODRL *policies*, vooral wanneer deze gebruikt worden in een heterogene, gedecentraliseerde architectuur en gecombineerd worden met een overervingsmechanisme, alsook de privacyrisico's die verbonden zijn aan het publiekelijk toegankelijk maken van deze *policies*.

Bij de evaluatie van verschillende technische benaderingen ter ondersteuning van de handhaving van gegevensbeschermingsrecht, zoals de AVG, werd een beoordeling [16] gemaakt

[4]http://www.w3.org/ns/auth/acl

[5]Acroniem voor Create, Read, Update, Delete.

[6]https://w3id.org/dpv#

[7]http://www.w3id.org/dpv/dpv-gdpr#

[8]https://w3id.org/oac/

van de mogelijkheden rond gegevensbeheer die worden geboden door Solid en het SPECIAL-project[9] in vergelijking met de huidige defacto-standaard waarbij betrokkenen zeer ruime toestemming geven voor verwerking. Bij de evaluatie van Solid met betrekking tot de vereisten voor gegevensbescherming is gebleken dat de huidige, op ACL gebaseerde oplossing van Solid voor autorisatie tekortschiet bij het implementeren van toepassingen die moeten voldoen aan de strenge wettelijke eisen die worden gesteld. Ten eerste, vanwege de slechte gebruikerservaring veroorzaakt door onder andere de gebrekkige interpreteerbaarheid van *access modes* en *resource identifiers* voor niet-technische gebruikers, het risico op phishing aanvallen vanwege het gebruik van IRI's om gebruikers te identificeren, en de veiligheidsproblemen die voortkomen uit overgeërfde ACL regels. Ten tweede, omdat ACL's belangrijke concepten in het kader van gegevensbeschermingsrecht niet vastleggen, zoals welk soort informatie wordt gedeeld, hoe die gegevens worden verwerkt en met welk doel, en welke rechtsgrondslag wordt gebruikt om deze verwerking te rechtvaardigen. En ten slotte omdat het wijzigen van deze ACL-regels momenteel wordt gedelegeerd naar de ontwikkelaars van toepassingen, wat conflicteert met de oorspronkelijke doelstellingen om controle terug te geven aan de eindgebruiker, aangezien ontwikkelaars onbeperkte bevoegdheid hebben bij het wijzigen van ACL-regels. Op die manier zouden ze wederom hun toevlucht kunnen nemen tot de donkere patronen die huidige implementaties van toestemming op het Web vertonen.

Een gelaagde, decentrale architectuur voor het combineren van SPECIAL en Solid werd eveneens voorgesteld en vergeleken met deze andere benaderingen [16]. De concrete mechanismen voor de uitwisseling van *policies* en onderhandeling voor toegang worden door de auteurs als toekomstig werk overgelaten. Hun evaluatie geeft evenwel een goed inzicht in de beperkingen van op ACL gebaseerde autorisatie wanneer deze geconfronteerd wordt met complexe gegevensverwerking.

D. Solid's Data Interoperability Panel

Het Data Interoperability Panel binnen de Solid Community Panels[10] werd opgericht met als doel het standaardiseren van de mechanismen waarmee verschillende applicaties veilig en effectief kunnen samenwerken over dezelfde data. Hierbij proberen ze het gebruikersbewustzijn en de interpreteerbaarheid omtrent in de Pod opgeslagen gegevens te vergroten, door complexiteiten zoals *resource*-organisatie weg te abstraheren, om uiteindelijk verschillende toepassingen in staat te stellen efficiënt en zinvol overheen dezelfde gegevens te laten samenwerken. Noemenswaardig is dat zij deze problemen trachten aan te pakken met behoud van de fundamenten van het Solid-protocol zoals het nu bestaat.

In de schoot van het panel hebben het afgelopen jaar twee belangrijke voorstellen vorm gekregen, namelijk de ontwerpspecificaties voor *Shape Trees* [21] en voor *Solid Application Interoperability* [22]. De eerste bouwt voort op de bestaande specificaties van RDF[11] en data shapes [23], [24], die respectievelijk de basis leggen voor interoperabiliteit door ondubbelzinnige identifiers (IRI's) en een structureel schema waaraan individuele RDF-grafen kunnen worden gevalideerd. Waar deze bestaande specificaties echter tekort schieten is in het modelleren van complexe *resource* hiërarchieën. Denk bijvoorbeeld aan de organisatie van een verzameling medische dossiers die vorm krijgt in een Solid Pod en waar ontwikkelaars relatieve vrijheid hebben in zowel het benoemen van bronnen als het gebruik van mappen om hun data te verzamelen. Een Shape Tree definieert structurele beperkingen voor een boomstructuur van bestanden in elk ecosysteem dat een notie van mappen heeft[12]. Voor elke map kunnen vormbeperkingen worden opgelegd aan de bevatte bronnen. Shape Trees zelf kunnen ook andere Shape Trees bevatten om zo vorm te geven aan complexere hiërarchieën (zo kunnen bijvoorbeeld medische dossiers als geheel bestaan uit medische beelden, voorschriften, rekeningen, verslagen, ...). De grote kracht van Shape Trees is dat ze de organisatie van bronnen in een Pod ondubbelzinnig kunnen definiëren en een abstractie op hoger niveau kunnen bieden, die door eindgebruikers gemakkelijker kan worden begrepen. Op deze manier gidsen Shape Trees toepassingen en gebruikers, door te bepalen waar gegevens naartoe moeten worden geschreven en waarvandaan ze kunnen worden gelezen. Door op deze manier verzamelingen van gegevens te modelleren kunnen we bewerkingen zoals autorisatie, datamigratie en validatie ook op dit hogere abstractieniveau uitvoeren. Vooral in de context van autorisatie vermindert het definiëren van regels op het niveau van Shape Trees, in plaats van individuele resources, de complexiteit en de kans op fouten, en stelt het ons in staat deze conceptuele *resource*-aggregaties op hoger niveau te relateren aan juridische concepten zoals categorieën van persoonsgegevens.

De Solid Application Interoperability (SAI) ontwerpspecificatie [22] maakt gebruik van deze voorgestelde Shape Trees om het concrete mechanisme te standaardiseren waarmee applicaties en gebruikers toegang vragen tot informatie in een Solid Pod, de manier waarop ze de concrete instanties van de Shape Trees lokaliseren, en hoe ze overheen deze data kunnen samenwerken. Tot nu toe werden de meeste details van deze verschillende operaties door de Solid specificatie overgelaten aan individuele ontwikkelaars, wat interoperabiliteit over deze gegevens bemoeilijkt. In de context van dit paper is het standaardiseren van toegangsverzoeken van bijzonder belang, deze functionaliteit zal dan ook als bouwsteen dienen voor onze referentie-architectuur. De SAI-specificatie introduceert

[9]Het SPECIAL-project was een onderzoeksproject dat tot doel had technologieën te leveren om big data-toepassingen in overeenstemming te brengen met de toepasselijke regelgeving rond gegevensverwerking. Het leverde gebruikersinterfaces op voor toestemming en transparantie omtrent de verwerking, en ontologieën voor de logging door gegevensverwerkingstoepassingen en voor het modelleren van *policies* inzake gegevensgebruik van zowel *data subjects* als *data controllers*, die machinaal verifieerbaar zijn. (https://specialprivacy.ercim.eu)

[10]De Solid specificatie wordt opgesteld door verschillende community panels, elk gericht op specifieke kwesties of domeinen die relevant zijn voor Solid zoals authenticatie, autorisatie of data-interoperabiliteit.

[11]Resource Descriptor Format, het centrale data model dat gebruikt wordt door *Semantic Web* technologieën om Linked Data bronnen op te stellen.

[12]Solid bouwt voort op de Linked Data Platform specificatie die de semantiek van dergelijke mappen of *containers* bepaalt.

het concept van een Authorization Agent als een dienst die gekoppeld is aan de WebID van een gebruiker en die diens gegevens beheert. De Authorization Agent heeft als taak verzoeken om toegang voor de gebruiker te verwerken, eerder verleende machtigingen te beheren en de concrete instanties van Shape Trees bij te houden via een verzameling *registries*. Hoewel de specificatie nog verder door het panel wordt verfijnd en sommige aspecten van de Authorization Agent nog niet geheel zijn gedefinieerd of bewust worden opengelaten voor implementatie, zullen we veel van de kernconcepten die erin worden beschreven in ons voorstel inzetten.

E. Linked Data Integrity & Authentication

Het Data Integrity 1.0 *draft community report* [25] is een recent voorstel van de Credentials Community Group van het W3C, dat tot doel heeft authenticatie en data-integriteit mogelijk te maken voor Linked Data-bronnen door middel van wiskundige bewijzen, zoals algoritmen voor digitale handtekeningen. Het omvat een ontologie voor de definitie van verschillende soorten bewijzen, verificatiemethoden en algoritmen. De oorsprong van dit werk ligt in de W3C *recommendation* van het Verifiable Credentials Data Model [26], een datamodel dat kan worden gebruikt om specifieke eigenschappen van een gebruiker uit te drukken (zoals een diploma, rijbewijs, ...) en dat vergezeld moet worden van een cryptografisch bewijs dat de authenticiteit en integriteit ervan kan bevestigen. Deze technieken zullen ons voorzien van de nodige bouwstenen, op vlak van authenticatie en data-integriteit, die we nodig hebben om de door ons voorgestelde referentie-architectuur te realiseren.

III. Architectuur

Zoals we al eerder hebben aangehaald, baseert het Solid Protocol 0.9 [27] zich op de Web Access Control specificatie [4] als mechanisme voor *discretionary access control* over de gegevens die zijn opgeslagen in de Solid Pod. Hoewel het voldoende mogelijkheden biedt voor eenvoudige gebruikssituaties met betrekking tot autorisatie in sociale contexten, kunnen sommige van de ontwerpkeuzes van WAC problematisch zijn in complexere gegevensverwerkingstoepassingen die onderhevig zijn aan regelgeving zoals de AVG. Onze architectuur splitst daarentegen de implementatie van toestemming als rechtsgrondslag voor verwerking van persoonsgegevens in de Pod op in twee domeinen, weergegeven in Fig. 1, waarbij *policies* die zijn opgeslagen in de Solid Pod van de betrokkene een interface vormen tussen deze verschillende domeinen:

1) Enerzijds wordt het domein van de gebruiker beheerd door een zogeheten *Access Management* Applicatie, die als taak heeft het verzoek om gegevensverwerking van de *Controller* te toetsen aan de toepasselijke wetgeving en de voorkeuren van de eindgebruiker inzake gegevensdeling, en die, indien het verzoek om gegevensverwerking wordt goedgekeurd, dit als een *Data Processing Grant* opslaat in de Solid Pod van de betrokkene.
2) Anderzijds gebruikt het *technische domein* de *Authorization Agent*, zoals voorgesteld door de Solid Application Interoperability specificatie, om de effectieve toegangsverzoeken af te handelen die door applicaties en gebruikers worden gemaakt in termen van Shape Trees, Data Shapes en ACL toegangsmodi. De interface tussen de twee domeinen wordt gevormd door de *Processing Grants* die worden gegenereerd door de *Access Management App* en bewaard in de Solid Pod van de gebruiker.

Voor de authenticatie en identificatie van de verschillende actoren in de architectuur zijn we afhankelijk van de WebID [2] en OIDC 0.1.0 [3] specificaties die werden gedefinieerd binnen het Solid Protocol versie 0.9 [27]. In de volgende paragrafen zullen we dieper ingaan op zowel de *Access Management* App, de *Authorization Agent* en de voorgestelde concepten van *Processing Requests* en *Processing Grants* die worden gebruikt om deze twee diensten aan elkaar te koppelen.

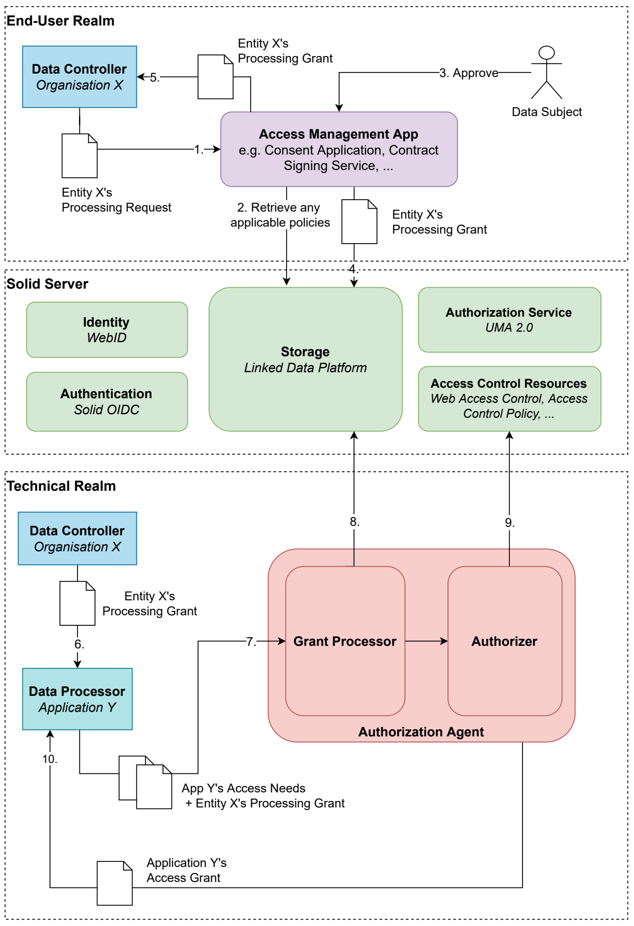


Fig. 1. Overzicht van onze referentie-architectuur, die een gebruikersdomein met de toelatingen voor gegevensverwerking koppelt aan een technisch domein volgens de Application Interoperability specificatie.

A. Gebruikersdomein: Access Management App

De Access Management App wordt door Data Controllers gebruikt om de noodzakelijke goedkeuring te krijgen voor de verwerking die zij verzoeken voor bepaalde categorieën persoonsgegevens en verwerkingshandelingen in kader van een verwerkingsdoel dat is toegestaan via een specifieke rechtsgrondslag. Zodra de app een verzoek om gegevensverwerking

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/>.
@prefix dpv: <http://www.w3.org/ns/dpv#>.
@prefix cert: <http://www.w3.org/ns/auth/cert#>.
@prefix oac: <https://w3id.org/oac/>.

@prefix : <https://example.com/#>.

:medicalRecordsConsent a odrl:Policy, dpv:
    PersonalDataHandling;
    odrl:profile oac:;
    dpv:hasLegalBasis [
        a dpv:Consent;
    ];
    dpv:hasDataController <https://example.com/id/doctor#me
    >;
    odrl:permission [
        a odrl:Permission;
        odrl:assignee <https://example.com/id/doctor#me>;
        odrl:target dpv:HealthRecord, dpv:Prescription, dpv:
     HealthHistory;
        odrl:action dpv:Collect, dpv:Consult, dpv:Analyse,
     dpv:Alter;
        odrl:constraint [
            odrl:leftOperand oac:Purpose;
            odrl:operator odrl:isA;
            odrl:rightOperand :MedicalConsultation
        ]
    ].
```

Listing 1. Voorbeeld van een niet-ondertekende Processing Grant Request

ontvangt, controleert het eerst of het verzoek ontvankelijk is en wordt de aanvraag vergeleken met eventuele expliciete voorkeuren voor gegevensuitwisseling die de gebruiker in zijn Pod heeft staan en die kunnen leiden tot een automatische goedkeuring van het verzoek. Als er geen voorkeuren blijken te zijn die exact overeenkomen met het verzoek, moet de betrokkene om zijn expliciete toestemming worden gevraagd. Zodra een *Processing Request* is ingewilligd, wordt het als een *Processing Grant* opgeslagen in de Solid Pod en afgeleverd bij de inbox [28] van de *Controller*.

B. Gebruikersdomein: Processing Requests en Processing Grants

Telkens wanneer een *Data Controller* (verzoekende partij) toestemming wil krijgen om gegevens over een *data subject* te verwerken, stelt die een *Processing Request* op. Dit verzoek wordt opgesteld op basis van een voorgesteld ODRL-profiel [14] en concepten uit de Data Privacy Vocabulary. Een voorbeeld van een verzoek om medische gegevens op basis van uitdrukkelijke toestemming is te zien in Listing 1. Het verzoek bevat details over de verwerking van de persoonsgegevens, in termen van rechtsgrondslag (in het geval van dit werk beschouwen we alleen uitdrukkelijke toestemming), de *data controller* en de specifieke toelatingen die in het kader van de verwerking nodig zullen zijn.

Elke permissie in dit verzoek specificeert welke categorieën persoonsgegevens verwerkt dienen te worden, welke handelingen met deze gegevens zullen worden gesteld, en welke beperkingen op het doel of de resultaten van de verwerking worden opgelegd. Er kunnen ook andere beperkingen worden bepaald, zoals de technische maatregelen die bij de verwerking worden gebruikt en de daarmee samenhangende risico's, maar deze werden in het huidige voorstel niet onderzocht.

De *Data Processing Request* wordt aan de *Access Management App* aangeboden samen met een *Data Integrity Proof* dat door de *Controller* is gegenereerd, zodat de herkomst en de integriteit van het verzoek kunnen worden gevalideerd. Door dit ondertekeningsmechanisme kan het risico op spam of andere kwaadaardige aanvallen met betrekking tot de Access Management App en de procedures voor de verwerking van *Processing Requests* worden beperkt, bijvoorbeeld door verschillende vertrouwensniveaus toe te wijzen aan de uitgevende diensten van de *Processing Requests* op basis van vereisten zoals identiteitsvalidatie of naleving van regelgeving.

Ten slotte wordt uit de *Processing Request* een *Processing Grant* geconstrueerd door eerst de rechtsgrondslag, d.w.z. toestemming, aan te vullen met vereiste attributen die ofwel in interactie met de gebruiker ofwel op geautomatiseerde wijze door de *Access Management App* zijn verzameld. Daarna worden alle toestemmingen die niet zijn weerhouden uit de *Processing Grant* verwijderd, wordt de RDF-graaf aangevuld met een Revocation Status attribuut conform de W3C Revocation List 2020 specificatie [29] voor het intrekken van *Data Integrity Proofs* zodat de *Access Management App* de *Processing Grant* op een later tijdstip kan intrekken, en wordt een *Data Integrity Proof* aangemaakt en ondertekend door de *Access Management App* om aan te geven dat aan de wettelijke eisen voor de goed te keuren gegevensverwerking is voldaan. De ondertekende *Processing Grant* kan worden gezien als een instructie voor de *Authorization Agent* om bepaalde soorten informatie te verstrekken aan een *Data Controller* en de door hem aangewezen verwerkers.

C. Technisch Domein: Authorization Agent

De *Authorization Agent* is wat betreft zijn semantiek en API grotendeels gebaseerd op de voorgestelde *Solid Application Interoperability*-specificatie, die nog door het panel wordt besproken en dus aan wijzigingen onderhevig kan zijn. Dit wordt mogelijk gemaakt door het feit dat de mechanismen van de *Authorization Agent* grotendeels open zijn gelaten voor implementatie, zodat aanvullende autorisatiecontroles kunnen worden uitgevoerd tussen het moment waarop de *Access Needs* aan de autorisatieagent worden gepresenteerd en de uitgifte van een zogeheten *Access Grant* die de concrete gegevens specificeert die kunnen gedeeld worden met de toepassing.

In feite is de enige wijziging van de interface van de *Authorization Agent* die wij in dit document voorstellen, dat de *access needs* van de *controller* vergezeld moeten gaan van een *Processing Grant*. Op die manier kan de *Authorization Agent* het verzoek om toegang dat wordt gemaakt door de toepassing of dienst, die optreedt als gegevensverwerker voor de *controller*, koppelen aan een geldige wettelijke basis voor gegevensverwerking. Het wordt dan de taak van een *Grant Processor* module in de *Authorization Agent* om de aangeboden *Processing Grant* af te toetsen aan de *Access Needs* van de Processor in termen van *Data Needs* (Shape Trees) en toegangsmodi. Dit laatste confronteert ons met de noodzaak van een ondubbelzinnige gelijkwaardigheidsrelatie tussen de abstracte definities in de *Processing Grant* en hun technische tegenhangers in de *Access Needs* van de toepassing of dienst.

Tenslotte, zodra de *Grant Processor* heeft vastgesteld dat het verzoek van de verwerker daadwerkelijk overeenkomt met onze initiële *Processing Grant*, kan deze verder gaan met een *Authorizer* die de taak heeft atomaire toegangsregels aan te passen die van toepassing zijn op de instanties van de Shape Trees die werden gespecificeerd in de *Access Needs* van de dienst. Zodra dit proces is afgerond, wordt een *Access Grant* teruggestuurd naar de Gegevensverwerker en worden de noodzakelijke registraties toegevoegd in de Pod.

D. Hulpregels & Policies

Het op ODRL gebaseerde verwerkingsverzoek en de bijhorende *Processing Grant* kunnen weliswaar volstaan voor het definiëren van de gegevensverwerking die op gebruikersniveau wordt gevraagd en goedgekeurd, maar ze zijn voor de *authorization agent* ontoereikend om deze te relateren aan de *access needs* die door een gegevensverwerker, bijvoorbeeeld een applicatie, worden gespecificeerd. De semantische lacune is hier tweeledig: enerzijds moet ondubbelzinnig worden gedefinieerd welke gegevens in de Pod onder de goedgekeurde verwerking vallen, en anderzijds moet bekend zijn welke acties op deze gegevens zijn toegestaan.

Ten eerste moeten de abstracte gegevenscategorieën die worden gebruikt om de persoonsgegevens te specificeren in kader van de verwerkingsactiviteiten, worden gerelateerd aan concrete, technische informatie over het datatype. Zoals eerder is aangehaald, stelt de combinatie van *Data Shapes* en *Shape Trees* als mechanisme voor het definiëren van gegevensverzamelingen en hun structuur ons in staat conceptueel gerelateerde bronnen in de Pod af te bakenen, zoals medische dossiers, foto's, aantekeningen, enz. Door middel van een aanvullende reeks regels die door de betrokkene in zijn Solid Pod worden geconfigureerd, een zogeheten *Data Category Equivalence Policy*, koppelen wij de door *Data Shapes* en *Shape Trees* verstrekte technische informatie over het datatype aan *Personal Data Categories* zoals die in het kader van de DPV zijn gespecificeerd en in het ODRL-profiel[14] worden gebruikt.

Aangezien abstracties van een hoger niveau worden gebruikt om de acties te definiëren die de verwerking mogelijk maken, moeten we ook de verwerkingscategorieën uit de DPV in verband brengen met relevante *Access Modes*, aangezien deze worden gebruikt in zowel het toegangscontrolemechanisme van Solid als de technische *Access Needs* die door de verwerker worden gespecificeerd. Deze kunnen door het subject worden gedefinieerd als *Processing Access Needs*, die een aanvullende reeks regels in de Pod vormen.

Voorts is het voorgestelde ODRL-profiel [14] ontworpen met het concept van voorkeuren voor gegevensuitwisseling in het achterhoofd, hoewel deze functionaliteit in dit paper niet in verder detail wordt beschouwd. Zo kan de betrokkene ook complexere verwerkingsactiviteiten vastleggen die automatisch aan een verzoekende partij worden toegestaan op basis van doel, gegevens en verwerkingscategorieën. Dergelijke *policies* worden ook in de Solid Pod bewaard, naast de eerder genoemde *Data Category Equivalence Policy* en de concrete *Processing Grants* die daaruit voortvloeien.

IV. Implementatie

Om de prestaties van onze architectuur te beoordelen, werd een implementatie[13] van het in paragraaf III-C gepresenteerde technische domein ontwikkeld in TypeScript met behulp van het Components.JS-framework [30]. Components.JS maakt een modulaire architectuur mogelijk, zowel met betrekking tot de autorisatielogica die wordt geïmplementeerd als voor de ondersteuning van verschillende functionaliteiten die zijn gedefinieerd in de Solid Application Interoperability-specificatie. Daarnaast maken we gebruik van de Community Solid Server v4.0.1[14] om ons te voorzien van een modulaire Solid server instantie die voldoet aan het Solid protocol 0.9 [27].

De implementatie zelf maakt gebruik van het recent geïntroduceerde concept van een UMA Authorization Service [31] in de Solid-OIDC 0.1.0 specificatie [3], waardoor de autorisatielogica kan worden losgekoppeld van de feitelijke Solid server implementatie. Desalniettemin wordt er een prestatieverlies geleden door de extra communicatieoverhead die hiermee gepaard gaat. Door modulaire interfaces voor respectievelijk authenticatie en autorisatie, kan onze Authorization Service dienen als basis voor onderzoek naar verschillende mechanismen voor toegangscontrole in Solid.

De modulariteit van deze Authorization Service stelde ons in staat autorisatielogica te implementeren op basis van de Solid Application Interoperability ontwerpspecificatie. Immers, telkens wanneer de *Authorization Agent* een *Access Grant* verleent aan een verwerker, wordt deze informatie opgenomen in de Registry Set van de eigenaar van de gegevens. De informatie in deze registers wordt door onze autorisatiemodules gebruikt om vervolgens de inkomende verzoeken van de verwerker te autoriseren.

De implementatie werd gebenchmarkt met behulp van synthetisch geconstrueerde Solid Pods, van wisselende grootte in termen van Shape Tree instanties en geregistreerde gebruikers (*gegevensverwerkers*). De benchmark zelf bestond uit 250 autorisatie verzoeken voor de verschillende datatypes die zijn gedefinieerd in de Application Interoperability ontwerpspecificatie, waarbij we de totale tijd hebben gemeten om een verzoek te autoriseren met de UMA Authorization Service. De resultaten van onze evaluatie zijn te zien in Fig. 2. Hoewel hulpdatastructuren zoals Agent Registrations, Access Grants, Data Grants en Data Registrations slechts een zeer beperkte impact zien op vlak van mediane reactietijden als we de dimensies van de Solid Pod schalen, beginnen de Data Instances die de eigenlijke informatie opslaan waartoe toegang wordt geautoriseerd, veel slechtere prestaties te vertonen. Hieruit kunnen we concluderen dat de tijdcomplexiteit van autorisaties voor dit resourcetype sterk afhankelijk is van de hoeveelheid informatie waartoe toegang wordt geautoriseerd, althans in de huidige implementatie en onder de huidige revisie van de specificatie [22].

Naast de implementatie van een UMA Authorization Service, hebben we ook een modulaire Authorization Agent ontwikkeld die de kloof kan overbruggen tussen de technische en

[13]https://github.com/laurensdeb/interoperability
[14]https://github.com/CommunitySolidServer/CommunitySolidServer

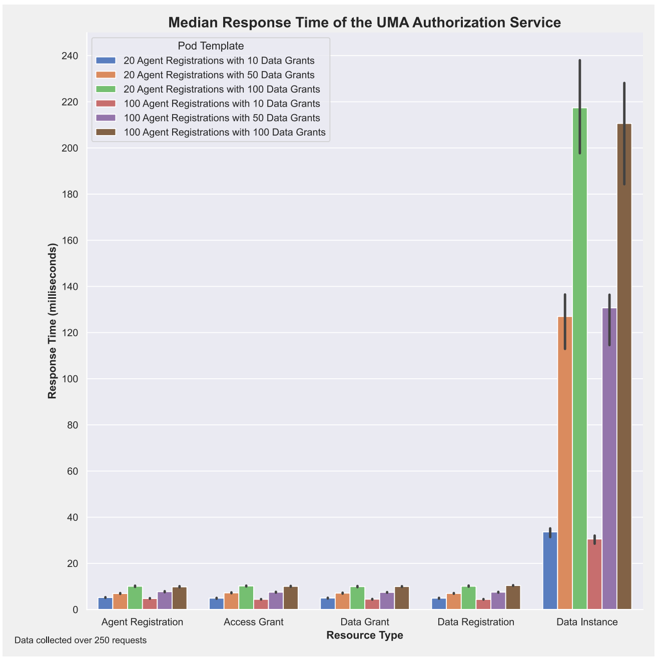


Fig. 2. Mediane responstijden per datatype voor een request naar de UMA Authorization Service bij verschillende configuraties van de Solid Pod. Data verzameld over 250 verzoeken per configuratie, 95% betrouwbaarheidsintervallen worden getoond in het zwart. Gevalideerd met de 1.0.0 release van de implementatie (Node v16.14.0 / Apple M1 Pro / 32GB RAM).

gebruikersdomeinen van onze architectuur. De Authorization Agent maakt het mogelijk om te ontdekken tot welke data een gebruiker toegang mag hebben, en kan nieuwe verzoeken verwerken op basis van de toegangsbehoeften van de toepassing of dienst in termen van Shape Trees. Er zijn ook interfaces voorzien voor de integratie van *data usage policy languages*, zoals ODRL of SPECIAL, in deze component, zodat die kan evalueren of inkomende *access needs* overeenkomen met een voorafgaande Processing Grant die is gegeven via een Access Management Applicatie.

V. Conclusies & Aanvullend Werk

Eén van de belangrijkste verschillen met eerdere voorstellen is dat de door ons voorgestelde referentiearchitectuur het probleem van het verenigen van technische autorisatie met de wettelijke vereisten voor gegevensverwerking in twee afzonderlijke domeinen opsplitst. Enerzijds is er een gebruikersdomein waar de gebruiker verzoeken om gegevensverwerking krijgt voorgelegd in termen van verwerkingsacties die plaatsvinden op abstractere gegevenscategorieën, en waar een eindgebruiker expliciete voorkeuren voor gegevensdeling kan bepalen. Anderzijds is er een technisch domein waar de Solid Application Interoperability ontwerpspecificatie regelt hoe ontwikkelaars van toepassingen toegang kunnen krijgen tot bronnen in de Solid Pod van een gebruiker, zodra een adequate rechtsgrondslag voor verwerking is vastgesteld. Het concept van Data Processing Grants die verifieerbaar zijn via de Data Integrity Specification van het W3C [25] vormen de schakel tussen deze twee domeinen, gecombineerd met *policies* die de betekenis van Data Categories en Processing Actions relateren aan technische concepten die door de Solid Pod kunnen worden begrepen.

Waar eerdere oplossingen beoogden juridische concepten te integreren in de bestaande autorisatiemechanismen van Solid, bijvoorbeeld door toegangsregels uit te breiden met een concept zoals verwerkingsdoel of verwerkingsgrond, maakt ons voorstel gebruik van de modulariteit die wordt voorzien door de nieuwe UMA Authorization Service en voorziet het in een gelaagde architectuur om *data usage policy languages* te introduceren.

Benchmarks van een implementatie van het cruciale autorisatiemechanisme dat door het technische domein van deze architectuur wordt gebruikt, laten evenwel verhoogde reactietijden zien naarmate de dimensies van de Pod, in termen van gegevens waartoe toegang wordt wordt verleend, toenemen. Desondanks was onze implementatie in staat om de technische haalbaarheid aan te tonen van autorisatie op basis van de abstracties die worden geboden door Shape Trees, zoals werd voorgesteld in de Solid Application Interoperability specificatie [22].

Sommige van de in het ODRL-voorstel [14] genoemde uitdagingen zijn hier aangepakt, terwijl andere, zoals optimalisatie van de prestaties, verdere overweging behoeven. Door de *access management app* en de *authorization agent* als dynamische onderhandelaars in de *request flow* te introduceren, hebben we vermeden dat *usage policies* openbaar door de Solid Pod moet worden geadverteerd, wat belangrijke bezorgdheden omtrent privacy wegneemt bij een oplossing op basis van dergelijke *policy languages* (bijvoorbeeld, wat als iedereen kan zien dat je je medische gegevens hebt gedeeld met het oog op een behandeling met een psychiater). Bovendien vermijden we, door de Authorization Agent te introduceren als tussenpartij voor het verschaffen van de technische toegang, de noodzaak om de *policies* te evalueren voor elk HTTP-verzoek op de data in de Pod. Specifieke juridische kwesties die in het ODRL-voorstel [14] aan de orde zijn gesteld, blijven ook grotendeels van toepassing op ons voorstel, d.w.z. de juridische implicaties van keuzes van de gebruiker die mogelijk worden gemaakt door de specifieke benadering rond gegevensbeheer in Solid, de noodzaak om rekening te houden met het toepasselijke rechtsgebied en de bijhorende vereisten voor gegevensverwerking en de vraag of voorkeuren voor het delen van gegevens, zoals kort werd aangestipt, inderdaad een vorm van toestemming inhouden.

Hoewel wij ons in dit voorstel grotendeels hebben geconcentreerd op het probleem van de implementatie van uitdrukkelijke toestemming als rechtsgrondslag, zou er ruimte kunnen zijn om ook andere rechtsgrondslagen door de *access management app* te laten afdwingen. Wanneer bijvoorbeeld om verwerking wordt verzocht op basis van een contractuele verplichting, zou de *access management app* het contract kunnen ophalen uit de Pod van de betrokkene en het kunnen valideren aan de hand van de identiteit van de partij die om de gegevensverwerking verzoekt.

Acknowledgment

Delen van dit werk zijn gebaseerd op ”A Policy-Oriented Architecture for Enforcing Consent in Solid” door De-

backere et al.[32].

References

[1] S. Speicher, J. Arwe, and A. Malhotra, "Linked Data Platform 1.0," Tech. Rep., Feb. 2015. [Online]. Available: https://www.w3.org/TR/ldp/

[2] A. Sambra, H. Story, and T. Berners-Lee, "WebID 1.0," Tech. Rep., Mar. 2014. [Online]. Available: https://www.w3.org/2005/Incubator/webid/spec/identity/

[3] A. Coburn, elf Pavlik, and D. Zagidulin, "Solid-OIDC," Tech. Rep., Mar. 2022. [Online]. Available: https://solidproject.org/TR/2022/oidc-20220328

[4] S. Capadisli and T. Berners-Lee, "Web Access Control," Tech. Rep., Jul. 2021. [Online]. Available: https://solidproject.org/TR/wac

[5] "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)." *Official Journal of the European Union*, vol. L 119, 2016-04-27.

[6] C. J. Hoofnagle, B. van der Sloot, and F. Zuiderveen Borgesius, "The European Union General Data Protection Regulation: What it is and what it means," *SSRN Electronic Journal*, 2018.

[7] "The GDPR: The Emperors New Clothes - On the Structural Shortcomings of Both the Old and the New Data Protection Law, author=Winfried Veil," *Consumer Law eJournal*, 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3305056

[8] M. Kretschmer, J. Pennekamp, and K. Wehrle, "Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web," *ACM Trans. Web*, vol. 15, no. 4, jul 2021. [Online]. Available: https://doi.org/10.1145/3466722

[9] G. G. Karcsony, "Managing personal data in a digital environment - did GDPRs concept of informed consent really give us control?" *International Conference on Computer Law, AI, Data Protection & the Biggest Tech Trens*, 2019. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3452573

[10] M. Nouwens, I. Liccardi, M. Veale, D. Karger, and L. Kagal, *Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence*. New York, NY, USA: Association for Computing Machinery, 2020, p. 113. [Online]. Available: https://doi.org/10.1145/3313831.3376321

[11] D. De Bot and T. Haegemans. (2021, Jan.) Data Sharing Patterns as a Tool to Tackle Legal Considerations about Data Reuse with Solid: Theory and Applications in Europe. [Online]. Available: https://lirias.kuleuven.be/retrieve/599839

[12] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *The Semantic Web - ISWC 2003*, D. Fensel, K. Sycara, and J. Mylopoulos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 402–418.

[13] P. A. Bonatti, S. Kirrane, I. M. Petrova, and L. Sauro, "Machine understandable policies and GDPR compliance checking," *CoRR*, vol. abs/2001.08930, 2020. [Online]. Available: https://link.springer.com/article/10.1007/s13218-020-00677-4

[14] B. Esteves, H. J. Pandit, and V. Rodrguez-Doncel, "ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2021, pp. 298–306.

[15] R. Iannella and S. Villata, "ODRL Information Model 2.2," Tech. Rep., Feb. 2018. [Online]. Available: https://www.w3.org/TR/odrl-model/

[16] G. Havur, M. Vander Sande, and S. Kirrane, "Greater Control and Transparency in Personal Data Processing," 01 2020, pp. 655–662.

[17] S. Villata, L. Costabello, N. Delaforge, and F. Gandon, "Social Semantic Web Access Control?" *Journal on Data Semantics*, vol. 2, 03 2012.

[18] UK Department for Digital, Culture, Media & Sport . (2020, Mar.) Cyber Security Breaches Survey 2020. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020

[19] M. Khonji, Y. Iraqi, and A. Jones, "Phishing Detection: A Literature Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013.

[20] H. J. Pandit, A. Polleres, B. Bos, R. Brennan, B. Bruegger, F. J. Ekaputra, J. D. Fernndez, R. G. Hamed, E. Kiesling, M. Lizar, and et al., "Creating a Vocabulary for Data Privacy: The First-Year Report of Data Privacy Vocabularies and Controls Community Group (DPVCG)," Oct 2019.

[21] J. Bingham and E. Prud'hommeaux, "Shape Trees Specification," Tech. Rep., Feb. 2022. [Online]. Available: https://shapetrees.org/TR/specification/

[22] J. Bingham, E. Prud'hommeaux, and elf Pavlik, "Solid Application Interoperability," Tech. Rep., May 2022. [Online]. Available: https://solid.github.io/data-interoperability-panel/specification/

[23] E. Prud'hommeau, I. Boneva, J. E. L. Gayo, and G. Kellogg, "Shape Expressions Language 2.1," Tech. Rep., Oct. 2019. [Online]. Available: http://shex.io/shex-semantics/index.html

[24] H. Knublauch and D. Kontokostas, "Shapes Constraint Language (SHACL)," Tech. Rep., Jul. 2017. [Online]. Available: https://www.w3.org/TR/shacl/

[25] M. Sporny and D. Longley, "Data Integrity 1.0," Tech. Rep., Jan. 2022. [Online]. Available: https://w3c-ccg.github.io/data-integrity-spec/

[26] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model v1.1," Tech. Rep., Nov. 2021. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[27] S. Capadisli, T. Berners-Lee, R. Verborgh, and K. Kjernsmo, "Solid protocol," Tech. Rep., Dec. 2021. [Online]. Available: https://solidproject.org/TR/2021/protocol-20211217

[28] S. Capadisli and A. Guy, "Linked Data Notifications," Tech. Rep., May 2017. [Online]. Available: https://www.w3.org/TR/ldn/

[29] M. Sporny and D. Longley, "Revocation List 2020," Tech. Rep., Apr. 2021. [Online]. Available: https://w3c-ccg.github.io/vc-status-rl-2020/

[30] R. Taelman, J. Van Herwegen, M. Vander Sande, and R. Verborgh, "Components.js: Semantic Dependency Injection," *Semantic Web Journal*, 2022. [Online]. Available: https://linkedsoftwaredependencies.github.io/Article-System-Components/

[31] M. Machulak, J. Richer, and E. Maler, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," Tech. Rep., Jul. 2018. [Online]. Available: https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html

[32] L. Debackere, P. Colpaert, R. Taelman, and R. Verborgh, "A policy-oriented architecture for enforcing consent in Solid," in *Proceedings of the 2nd International Workshop on Consent Management in Online Services, Networks and Things*, Apr. 2022. [Online]. Available: https://www2022.thewebconf.org/PaperFiles/88.pdf

Lay Summary

This thesis aims to address the challenge of implementing the relevant controls and safeguards for compliance with *data protection law* in applications and services that interact with personal data vaults following the Solid specification.

Solid

Personal data vaults provide a novel way of counteracting the asymmetries of power that exist in today's online applications and services. Such asymmetries are at the heart of the take-it-or-leave-it approach taken by many in asking for permission to access, process and store our personal data. By separating apps and services from the data they use, the Solid project aims to restore user choice, ensure data portability and improve transparency in data processing.

Data Protection Law: GDPR

When it came into force in May 2018, the General Data Protection Regulation harmonized rules and requirements concerning the processing of personal data of people in the European Union. At the heart of the Regulation is the concept of data protection, which aims to safeguard our personal data by requiring that it is used in a fair manner and following the principles of due process. Under the GDPR, the burden of proof for showing that these requirements are met falls on the party seeking to process the personal data rather than the individual whose information is being used.

Policy Languages & Data Usage

Policy languages enable operators of a distributed system to regulate the behavior of its components. An example of such a policy could be that a government requires its citizens, by law, to share income data with the tax authority. Semantic policy languages take advantage of the affordances of the Semantic Web to define machine-readable policies that are meaningful and enforceable at a global level. A specific subtype of policy languages enables the specification of data usage policies, which regulate the access, use and storage of data. If data usage policies only govern the initial access to data, they are also referred to as access control policies.

Problem statement

While the Solid specification already offers controls over the sharing of personal data, this is currently enabled through access control policies which offer limited affordances in ensuring compliance with data protection law. The strict requirements set by regulations like the GDPR thus require additional measures, outside of what is offered by the Solid specification, to be applied by organizations taking advantage of the data sharing enabled by Solid Pods. Such proprietary measures could be detrimental to the open ecosystem Solid tries to enable and reduce transparency for end-users.

Solution

In this thesis we propose an architecture for relating Solid's low-level access control mechanism with the higher-level requirements set by data protection law through the application of data usage policies. Additionally, we implement part of this architecture to evaluate its technical feasibility and performance. While our implementation shows adequate performance for small amounts of data stored in the Pod, further optimizations will be needed for the architecture to be practicable.

Acknowledgment

You may have heard of the expression "It takes a village to raise a child.", a proverb originating from a family of expressions that likely find their roots in African culture. At the start of this thesis I would like to add to that family of sayings, because it very much also takes a "village" to write a thesis. From the advise, feedback and interesting discussions I've had the pleasure of having with my supervisors, Ruben Verborgh, Ruben Taelman and Pieter Colpaert, over the last year to the collaboration and late-night brainstorming on the future of Solid and privacy on the Web within the Solid community panels, with people like Justin, Pavlik, Mathieu and Eric, and the feedback I've received from researchers across a wide range of fields, I can only conclude that this thesis was no 'one person job'.

Furthermore, I want to thank my friends and family for their support, proofreading sessions and motivation. And perhaps most importantly, for sometimes pulling me away from my keyboard.

And finally, I must not forget my colleagues with Digital Flanders and the Flemish Data Utility Company. Working with them over the last year on bringing Solid into a wide range of practical applications with the aim of improving life for every citizen in Flanders has simultaneously been one of the biggest challenges and most rewarding experiences so far.

Laurens Debackere,

Brugge, June 2022.

Permission of Use on Loan

Table of Contents

List of Figures

List of Tables

List of Listings

1

Preface

"You mean Laertes' son, from Ithaca. I saw him on an island. He was weeping in the palace of the nymph Calypso, who keeps him there by force. He has no way of getting back to his own land — he lacks companions."

—Homer, *Odyssey, Book IV*

The unprecedented growth of online services and applications over the last two decades has fundamentally reshaped our society, determining the ways in which we communicate, do commerce, entertain ourselves, consume the news, ... This digital revolution has also cleared the way for a society where fundamental rights like privacy and data protection are under increased pressure, often leaving the governance of our data in the hands of the corporations that provide us with these essential tools of the modern era. In this thesis we research Tim Berners-Lee's Solid Project and how it could form the basis for a more equitable web where the balance of power between service providers and end-users is restored.

1.1 Surveillance capitalism

In the 2019 book "The Age of Surveillance Capitalism"[1] professor Shoshana Zuboff warns her readers of the way technology has radically changed the fabric of our society over the last few decades. Her book discusses the advent of a new economic model, called surveillance capitalism, that has fostered the explosive growth of platform economies like Google, Facebook, Microsoft, TikTok, WeChat, ... These services have all discovered new ways of extracting value from their customers, very different from the traditional product- or service-oriented approaches employed by the capitalists of the 20th century like Ford or Edison.

According to Zuboff, surveillance capitalism developed rather unassumingly in the background of major societal changes that followed World War II, like decolonization, the women's rights movement, the civil right's movement, the fall of the Berlin Wall, ... Over the course of this post World War II era, people have asserted a greater sense of self. Rather than the collectivism and sense of solidarity that drove the Allied forces to victory during the War, the relative peace of the last decades has led people to an exploration of themselves and their own identity, she argues. This trend is what some social scientists also call the second modernity[2].

Just a short two hundred years ago the first modernity originated at a time when industrialization allowed ordinary people to trade in their predetermined future as the "son of ..." or "daughter of ..." for a hard life in one of the new factories that the industrial revolution had given birth to. While they had some freedom to escape their descent, people were still expected to conform to their community's expectations and norms. This first modernity gave some headroom for self-determination, think for example of the many Europeans who endeavored on a quest for a better life in the Americas at the beginning of the 20th century, but nothing as radical as the changes that have occurred in our lifetimes. The people of the first modernity did lay the building blocks for our capitalist economic system, which was a critical enabler of these radical changes that were to follow, first as part of an underpaid workforce and later as a consumer when Henry Ford and his contemporaries discovered the power of mass consumption and transformed their workforce into a consumer base.

When Sir Tim Berners-Lee and Robert Cailliau invented the World Wide Web at CERN in 1989, they could never have imagined the profound impact their invention would have on our society. The individualism that had grown in our second modernity society over the preceding decades provoked a very real need for information. As our identities, beliefs and values were no longer predetermined by the environment we grew up in every individual had to discover and construct the open-ended reality that is their life. The Web was and is a critical enabler for the sharing of opinions, facts and thoughts that help us form this reality. Furthermore, it also broadened the horizon for so many, not restricting the dissemination of ideas to one single geographic community or neighborhood. But for all the good that the Web provided to these new generations of information-seekers eager to understand their world, it also formed the breeding ground for surveillance capitalism's meteoric rise.

Of particular note is that the original vision of the Web, as set forward by its inventors, was one of decentralization, where no single entity could control the dissemination of information. Think of how a hyperlink allows anyone to link to some article or website, without even requiring the permission of the author. This lack of centralized control enabled one of the central drivers of the Web's explosive growth: permissionless innovation.

It is this concept of permissionless innovation that enabled platforms like Google, Facebook and others to become the behemoths they are today, driven by an ever growing wave of people trying to determine their own identity in this era and connect with their kins. Social norms and expectations that were considered as if set in stone just a few decades ago, are now under constant discussion and scrutiny. It is up to each individual to navigate this society without certainties. This is the environment that gave birth to surveillance capitalism's economic driver. To these platforms it quickly became apparent that the information needs of a generation in search of their identity could be reappropriated for the profit of investors.

Surveillance capitalism, like any form of capitalism, involves the trade of a product or service. Surveillance capitalism's product is what Zuboff describes as *behavioral surplus*, it is the information that is extracted from our behavior as users on this platform not for improvement of the product but rather for the benefit of its investors. Such *behavioral surplus* is no longer the finished product these days, rather the aim is to predict and even influence future behavior of consumers. You may have heard the platitude "If it is free, then you are the product", but rather than the product we have become subjects whose personal data are being extracted, analyzed and sold for profit.

Crucially these platforms currently undermine the very foundational aspect of the web that enabled them to exist in the first place, they have largely traded in the decentralized roots of the Web for increased centralization and in the process hollowed out the very enabler of permissionless innovation. Other forms of capitalism also have some very negative consequences when allowed to go unchecked, think of the monopolistic behavior of the so-called Titans of Industry at the turn of the 20th century which gave birth to the first antitrust laws[3], how big e-commerce players use union-busting tactics in order to keep low-wage workers from improving their working conditions[4] or how Big Oil has knowingly contributed to global warming and polluted our environment in their search for Black Gold[5]. Traditionally, a combination of legislation and public outcry have aimed to balance the interests of our society as a whole with those of the shareholders of these big corporations. However, the unprecedented speed at which surveillance capitalism has become the dominant force behind the social media and other platforms we have all become so dependent on and the fact that it directly targets some of our most fundamental desires, like self-determinism, a feeling of belonging and our connections to the people we love have made it very difficult to clearly see what is happening here.

How can society fix this imbalance between the surveillance capitalism that drives the platform economies we have become so dependent on and the negative effects it has like misinformation and fake news, mental health problems, the filter bubble effect and privacy breaches? In 2017, confronted with both the tremendous positive effects of his invention as well as the negative impact of modern platforms and the economic systems that support them, Sir Tim Berners-Lee set forth three challenges for the Web[6]:

- Taking back the control over our personal data.
- Tackling the spread of misinformation on the web.
- Ensuring transparency for political advertising.

These challenges reflect the problems that we've highlighted in the preceding paragraphs and their supposed origins in the economic models of these large platforms which today function as islands of centralization in the decentralized world the Web once fostered. At the heart of the problem is the invisibility of these Platform's actions. As Zuboff describes it, we must be wary of not confusing the puppets (the platforms and services that we've become so dependent on) and their puppet master (the economics that feed of the behavioral surplus these platforms are so good at extracting from our actions). Over the last decade a lot of progress has been made in making this puppet master more visible, both at a legal, societal and technical level. Something which we will touch upon in the next sections of this introduction.

1.2 How do we define privacy and data protection?

Research into consumer awareness around privacy in Flanders[7], demonstrates an increased concern with users over how their data are being used by the applications and services they rely upon. Fifty-two percent of respondents are notably concerned with their privacy, in particular the lack of transparency into data processing and collection as well as the impact of social media are noted as areas of concern.

An important distinction to be made here is between privacy and data protection. Data protection ensures that the processing of personal data happens following the principles of due process and in a fair and transparent manner with respect to the data subject. The principle of data protection has shaped much of the contemporary European legal view regarding the protection of an individual's privacy, which is perhaps best demonstrated by its status as a fundamental right under the Charter of Fundamental Rights of the European Union and, perhaps more visibly, the General Data Protection Regulation (GDPR). While data protection in the EU delegates most of the legal responsibility to the party seeking to process personal data, the United States have followed a markedly different approach to privacy protection which puts a lot of the burden on the individual to make informed choices in a free market. While the US does have specific sectoral laws governing the protection of certain data, no all-encompassing protections exist akin to those afforded by the GDPR.

Even in the light of growing consumer awareness and strict regulation enforcing the principles of data protection at the EU-level, dark patterns are still employed by many applications and services [8], transparency into how information is being processed is often hidden away behind cumbersome processes and lengthy privacy policies, and take-it-or-leave-it approaches regarding consent are even now wide-spread[9]. Progress has been noted[8], however, mostly in the area of transparency through updated privacy policies and the use of cookie banners. Nevertheless, the mandated controls for end-users over their data are rarely provided and only a minority of web services are able to meet the high-standards set by GDPR[10].

1.3 Solid: Redefining a web of platforms into a web of data

The Solid project[1] aims to realize Tim Berners-Lee's vision on decoupling personal data storage from the apps and services that use it, in order to return control and data governance to the user. Ultimately, Solid aims to re-establish a proper balance of power between service providers and their users [11], by providing the latter with the tools to make their own choices in data sharing and storage rather than having their data exist out of sight and out of control. To that end, the Solid community is developing a draft specification for decentralized personal data storage servers, also referred to as *Pods*.

At its core, the Solid Protocol version 0.9 [12] has three crucial building blocks that make up most of its footprint:

1. Solid implements parts of the **Linked Data Platform** W3C recommendation [13] to allow for read/write-access to the resources stored in a Pod with specific affordances for handling Linked Data.

2. Solid proposes **WebIDs** [14] and **Solid OIDC** [15] for *identification* and *authentication* purposes respectively. Through these standards, agents can be linked to a decentralized identifier expressing information on them like the agent's trusted identity providers. This allows for authentication between resource and authorization servers that have no prior trust relation.

[1]https://solidproject.org

3. **Web Access Control** [16] provides the critical controls over sharing of information stored in the Pod. Web Access Control is a cross-domain, decentralized solution for *authorizing* requests using *Access Control Lists* (ACLs) expressed as Linked Data. It identifies both *agents* and *resources* through the use of IRIs. Notably, ACLs can both be defined specifically for a given resource, or be inherited from a parent container.

Recently, governments and businesses have started showing interest in Solid as an enabler for improved data sharing while respecting the requirements set by privacy and data protection law. In Flanders, the Data Utility Company is being launched to kick-start an ecosystem centered around data sharing enabled by technologies like Solid Pods. These evolutions emphasize the need for a thorough study of how Solid can meet or even exceed legal requirements. Research which encompasses not only a technical evaluation but also furthers our understanding of the user experience and user attitude with respect to this new model of data governance, the legal implications of the use of personal data vaults, and the economic model by which this data-sharing ecosystem can create value for the parties involved.

1.4 Motivation

Typical online service relationships show significant asymmetries of power between end-users and service providers[17], due to economical, technical and organizational choices. In contrast, Solid allows for a user to have a clear overview of what data their Pod contains and granularly control with whom they share access. Therefore, it could bring crucial bilateral protections that legal bases for data processing, like consent, depend upon in order to be used as intended by the legislator in data processing applications.

Moreover, as was highlighted in prior research[8], there is little effort currently being spent by data controllers in affording data portability to their end-users. A capability which is mandated by the GDPR, but obviously conflicts with business interests inspired by the surveillance capitalism model as presented by Zuboff[1]. Solid may be a critical enabler for such data portability given its use of Semantic Web technologies allows for interoperability by design. Also, the model of personal data vaults for storing data puts the controls over data sharing in the hands of the end-user instead of leaving these decisions up to the different service providers.

Nevertheless, the current Solid protocol 0.9[12] relies on Web Access Control to realize end-user control over data sharing, which has limited expressivity and interpretability. In contrast, recent privacy and data protection regulations impose strict requirements on personal data processing applications and the scope of their operation[18]. Crucially, the Web Access Control mechanism lacks the granularity and contextual awareness needed to enforce these regulatory requirements and shows critical shortcomings when used as a tool for realizing end-user data governance.

1.5 Research Question

In this thesis we investigate the implementation of consent-based data-processing in Solid, with the aim of validating in which ways it conforms to or falls short of realizing the legal requirements set forth by the GDPR regulation. In order to realize this goal we define a number of intermediate research objectives:

1. Identify the the relevant informational and technical requirements for consent-based data-processing under GDPR.
2. Provide an overview of prior research on the topic of reconciling Solid' authorization mechanism with the requirements for data processing under data protection law.
3. Evaluate the existing solutions for implementing *data processing* based on *consent* in Solid using Web Access Control.
4. Define a novel architecture for implementing consent in Solid, mindful of the requirements and shortcomings that were previously identified.
5. Implement and assess this novel architecture with respect to the identified legal requirements and shortcomings of prior proposals.

Through these research objectives we want to answer the following central research question: "Can we rely on data usage policies to implement consent-based data-processing in Solid which conforms to the requirements under the GDPR regulation and is applicable within the constraints of a practical application?". In order to evaluate this practical applicability requirement, we should validate that when given an authorization based on such a data usage policy an application can use these access rights in an interactive setting. This research question will in fact imply three separate sub-questions we need to answer:

1. Can data usage policies be integrated into the Solid protocol as it exists today?
2. Does our implementation conform to the informational and technical requirements of consent as a legal basis for data processing under GDPR?
3. Does our implementation have bounded performance guarantees that ensure it can be used in the constraints of an interactive, end-user application?

We will hypothesize that our architecture, which builds upon the work of the Solid Application Interoperability specification and prior research defining the use of ODRL policies as an authorization mechanism in Solid, in fact successfully integrates the model of data usage policies in Solid's existing protocol (1), can comply with the requirements set by the General Data Processing regulation (2), and provides bounded performance guarantees in authorization of a request from a specific client (3).

1.6 Outline

This thesis continues with chapters 2,3, and 4 that aim to provide the reader with information on the requirements for data processing under the General Data Protection Regulation as well as how Solid currently implements authorizations and what efforts have been made to improve this. Thereafter chapter 5 details our proposed reference architecture for the implementation and enforcement of the requirements set out by data protection regulations. In chapter 6 we present our implementation of this architecture using the Components.JS semantic dependency injection framework, followed by an evaluation in chapter 7. Finally, we present our conclusions reflecting on the fulfillment of the initial research objectives and define further work that will be necessary in this field.

2

Background: GDPR

"Speak softly and carry a big stick; you will go far"

—Theodore Roosevelt

While largely a refinement of existing legal frameworks like the 1995 Data Protection Directive and the 2005 ePrivacy Directive, the EU's General Data Protection Regulation[1] [19] set a major legislative milestone in the realm of data protection and privacy law when it entered into force in 2018. Its use of a "big stick" approach to compliance, through hefty fines and the introduction of new internal and external control mechanisms awakened both the legal and business community. It affords *data subjects* with both increased transparency and greater control regarding the processing of their personal data by *data controllers* and takes new and emerging technologies such as Big Data, AI, and the internet explicitly into account. While far from perfect [9], it bestows a much greater deal of autonomy upon the *data subject* when making decisions regarding the processing of their personal data than has previously been the case.

This chapter starts with an overview of the broader history and context of data protection and privacy regulations, information that is critical for understanding the objectives but also the limitations of such legislation. Thereafter we look at the specifics of the EU's General Data Protection Regulation, namely which parties are subject to it, what requirements it sets for data processing applications and how it relates to similar laws in other regions. Finally we aim to establish the specific informational requirements data controllers must retain, in fulfillment of research objective 1.

2.1 Legal views on privacy

Widely considered as the father of modern day data privacy law, in his 1967 book "Privacy and Freedom"[20] Alan F. Westin describes in detail the threats he sees our modern society pose to privacy and how we should tackle the challenges posed by new technologies like "closed-circuit TV cameras", "directional microphones", and the "computerization of vast amounts of personal data". For this last technological trend, Westin coins the term "data surveillance". The solutions proposed[21] in his work are fourfold; Firstly, a requirement of due process should be imposed on personal data processing. Secondly,

[1] http://data.europa.eu/eli/reg/2016/679/oj

the individual must be notified when their data are collected. Additionally, they must be able to examine the data being processed. Finally, the individual should be able to challenge its contents if inaccuracies are found. Most poignantly Westin notes that much progress could already be made with regards to limiting the encroachment on privacy he describes without necessarily modifying legislation but by introducing privacy controls like professional standards or even some "old-fashioned good manners"[21].

In her work "Privacy as Contextual Integrity", Helen Nissenbaum introduces privacy as "...one of the most enduring social issues associated with information technologies."[22]. Through the model of contextual integrity, which breaks down the issue of evaluating breaches of privacy into an assessment of the two social norms *appropriateness* and *distribution*, Nissenbaum aims to resolve the limitations of existing models for determining breaches in the context of so-called *public surveillance*. *Public surveillance* refers to the use of technology in novel ways which greatly expand our capacity to gather, process or analyze personal information such that existing law falls short in curtailing the privacy infringements these applications may enable. In contrast with some prior approaches towards defining privacy, Nissenbaum argues that there are no areas in life where "anything goes" with respect to our personal information. Nevertheless, in different contexts there will be distinct norms which apply to the flow of information. In the model of *contextual integrity*, these norms are split into two realms firstly norms of appropriateness and secondly norms of distribution. Both categories of norms should be respected for contextual integrity to be maintained. The norms of appropriateness define what details are appropriate to be revealed about a person in some context whereas norms of distribution regulate the transfer of personal information between parties.

The recently published W3C group draft note on "Privacy Principles"[23], aims to guide specification writers and policy makers by defining a reference framework for concepts related to privacy that can be applied at a global scale. The growing asymmetries of power that the Web has been confronted with over the last decades because of the ability to process, collect an analyze an ever increasing volume of data pose an important challenge with respect to privacy and data protection, according to the authors. And even in jurisdictions where data protection is legally established as the norm, dark patterns are prevalent in many services to evade the protections this legislation aims to give individuals[8]. Many of the ideas by Nissenbaum and Westin regarding privacy and the appropriateness of specific flows of information in certain contexts can be found in this draft document and should serve to inform future writers and implementers of web specifications.

An important concept which is also introduced in the W3C draft note[23] is that of *data governance*, a set of principles which enables actors to regulate flows of information. For a system to enable *data governance* for its individual users, it should allow these parties to define which flows of information can or cannot be produced, exchanged or processed about them by other actors. In fact, *data governance* goes beyond the idea of privacy and perhaps serves to enable what Nissenbaum names[22] "preferences" of the individual, when discussing a more conservative approach for handling the issue of public surveillance. As we will highlight in the next sections the choices of an individual with respect to flows of information are rarely absolute, for example how would a tax agency collect taxes if everyone simply chooses not to share their financial details. Thus, *data governance* might not always be the right model to strive for when implementing better *data protection* for the Web.

2.2 The EU's General Data Protection Regulation

The origins of the EU's General Data Protection Regulation can be found in the Charter of Fundamental Rights of the European Union[24], which aimed to consolidate the most important rights and freedoms applicable to EU citizens into a single, legally binding document. Articles 7, "Respect for private and family life", and 8, "Protection of personal data", define the freedoms that are implemented and enforced in secondary law like the GDPR. It is important to underline the origins of data protection as a fundamental right[2], firstly because these rights were defined long before the advent of today's platform economies and secondly because they give the individual the right to the processing of their personal data in a fair manner[25].

During a 2009 roundtable on Online Data Collection, Targeting and Profiling, then commissioner of Consumer Protection Meglena Kuneva described[26] personal data as "the new oil of the internet and the new currency of the digital world". This vision on personal data being analogous to a valuable commodity is not new, British mathematician Clive Humby was likely one of the first to note this parallelism back in 2006, at the time he substantiated this claim by noting: "It's valuable, but if unrefined it cannot really be used"[27]. The implications of this oil metaphor can be found throughout the GDPR, even though it has been noted[28, 29] that this analogy suffers from the lack of a clear definition of ownership. Similar to how society has struggled with the ownership of real-world valuable commodities. Especially when considering data protection as a human right, as it is defined within the European Union, this notion of ownership becomes all the more important.

At this point the fundamental difference between *privacy* and *data protection* should also be highlighted. From the perspective of the GDPR *data protection* is considered separate from the *right to privacy*. Notably, *data protection* should not only entail protecting the private life of the individual but also ensures that data are used in a fair manner and following the principles of due process. The United States on the other hand followed a fundamentally different approach to privacy law[18], where sectoral rules govern much of the *informational privacy* of individuals. This notion of *informational privacy* puts most of the burden on data subjects whom are tasked with performing a critical evaluation of privacy policies and then have to make an informed choice from the offerings in the free market. The burden in case of a breach then falls on the individual, whom should have made a more careful evaluation of the marketplace, at least according to this school of thought.

Importantly, many of the General Data Protection Regulation's rights and requirements were already present in prior law, such as the 1995 Data Protection Directive, which was plagued by poor compliance and inadequate enforcement because it had to be implemented in national law such that fines and remedies were left at the discretion of the member states[18]. By expanding the powers and responsibilities of Data Protection Authorities, increasing the fines that can be imposed when *data controllers* are found to be in breach with the GDPR and more broadly defining who can sue and the ways in which *data subjects* can seek remedy when they find that their rights have been violated, the GDPR aims to resolve many of the shortcomings of the Directive.

[2]Used interchangeably with the term "human right"

2.2.1 Terminology

Before defining the requirements imposed on personal data processing by the GDPR, we will first specify the terminology used to define the different actors involved, the regulated activities and the entities tasked with enforcement.

Firstly, we should delineate the concept of *personal data* which is described in Article 4 (Definitions) of the Regulation as "...any information relating to an identified or identifiable natural person...'[19]. The identifiability of the natural person is considered in a very broad sense, as it also includes indirect identifiability such that pseudonymous identifiers as well as information that is typically considered public knowledge may also fall under this broad definition of *personal data*.

The GDPR regulates the *processing of such personal data*, which is subsequently defined as "...any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means,..."[19] Analogous to the definition of *personal data*, this lays out a very wide scope for what can be considered *processing*. In practice almost everything one can do with *personal data* will fall under this definition of *processing*.

Three important actors are directly involved in the data processing activities as they are taking place. The *data subject* is the natural person to whom the *personal data* being processed can be related. Note that the nationality of this natural person is irrelevant to the GDPR, the fact that the person is in the EU suffices for the provisions of the GDPR to apply. *Data controllers* are the entities responsible for defining the purposes, identifying the legal basis, and implementing the necessary technical and organizational measures applicable to the data processing. They initiate the personal data processing and also assume liability in case of breaches and non-compliance to the Regulation. *Controllers* can delegate the task of processing personal data to *data processors*. In this case the due diligence of ensuring *data processors*, like cloud providers or advertising networks, are in compliance with the requirements of the GDPR, falls squarely on the *data controllers*.

Important in the enforcement of the Regulation are *Data Protection Authorities* which are established at a national level, and saw their role in supervision and compliance monitoring of *data controllers* considerably expanded and strengthened under the GDPR. These authorities are expected to be independent from political influence, can collaborate in broader investigative efforts and assist *data controllers* in achieving compliance. As a regulator the *Data Protection Authorities* are also responsible for processing the complaints of individuals, as well as notifications of breaches by *data controllers*. In large organizations the role of the *data protection officer* was introduced under GDPR. A DPO is responsible for the monitoring of compliance with the Regulation within an organization, furthermore they are the primary contact point for both the regulator and the *data subject*.

2.2.2 Scope

As we've noted before the GDPR interprets both *personal data* and *processing* activities in a very broad sense. In fact, only few exceptions exist that limit the applicability and scope of the Regulation. In the following section we'll identify the territorial scope, i.e. where the Regulation is applicable, and the material scope, i.e. to what activities and in which context the Regulation is applicable.

Territorial Scope

The GDPR evidently applies to organizations and entities that are established in Europe. However, the Regulation can also apply to organizations which have no real-world presence in the EU, if the controller or processor is "...offering services to data subjects in one or more Member States in the Union."[19]. This way the GDPR ensures that entities cannot just move data to a jurisdiction with more lenient regulations concerning *privacy* and *data protection*. Essentially, the GDPR is applicable as soon as any organization consciously processes the personal data of people in the EU, even if that organization is not based in the EU.[18]

Material Scope

By default the Regulation applies to *processing* of *personal data* in the very broadest sense, however two important exceptions have been introduced by the regulator. Firstly, it does not affect processing in the context of pure household or personal activities. An exception that is interpreted in a very narrow sense by regulators and courts, but exempts for example the interactions we have with friends and family on social media. Secondly, in the context of national security and prosecution of criminal offences data processing activities also fall outside the scope of the Regulation.

2.2.3 Requirements for Data Processing under GDPR

The GDPR defines four high-level requirements which make data processing proportional and lawful. Firstly, *data controllers* must adhere to the Fair Information Practices (FIPs). Secondly, there must be a proper legal basis for processing the *personal data*. Also, in the case of specific categories of sensitive data, the controller must have specific grounds for using this information. Lastly, in the case of transfer of data outside the EU there must be a lawful mechanism for this transfer to occur.

Fair Information Practices

The Fair Information Practices (*FIPs*) were introduced as a requirement by the original 1995 Directive with the goal of minimizing collection, processing and use of personal data. They consist of six principles that a data controller must take into consideration in the design, development and operation of *data processing* activities. In the next paragraphs we will briefly introduce these principles.

The principle of lawfulness, fairness and transparency is the first of these *FIPS*. It also relates to the central aim of *data protection*. Namely, that the processing of personal data occurs in a fair and transparent manner with relation to the data subject and respectful of applicable law, like GDPR. This stipulation is central to the requirement for controllers to give data subjects transparent information on their processing activities in language which they can understand.

Table 2.1: Overview of the Europeanized Fair Information Practices *processing activities* must adhere to under the GDPR.

Name	Description
1. Lawfulness, fairness and transparency	The data processing occurs in a fair manner, with a basis in law and transparent to the *subject*.
2. Purpose limitation	Data are only processed for a specific, concrete purpose which is known in advance to the *subject*.
3. Accuracy	The *personal data* being processed is proactively kept accurate and the *subject* can rectify inaccuracies.
4. Data minimization	The *personal data* processed must be relevant and limited to what is necessary in relation to the purposes.
5. Limited data storage	*Controllers* must not store data for longer than necessary and the criteria for deletion should be known beforehand to the *subject*.
6. Data Integrity and Confidentiality	The *controller* must have appropriate protections in place against loss, breach or unlawful processing of the *personal data*.

The second principle of purpose limitation, implies that data should only be stored and processed for a purpose that is known in advance to the *data subject*. It has been highlighted[18] that this purpose limitation is a challenge to modern Big Data and Machine Learning applications. Moreover we could also see it come into conflict with the goal of Solid to improve data re-use, as Pod providers would be confronted with this purpose limitation provision in their role of *Data Controller*[30]. In particular because the *FIPs* require this purpose be specific and concrete. Nevertheless, the GDPR does allow for a *controller* to determine that a new purpose is compatible with the original after an evaluation of the expectations of the *subject*, the context and nature of the data being processed, the consequences of the further processing for the *subject* and the link between the initial and new purpose.

With respect to accuracy, the *FIPs* mandate that *personal data* are correct and even rectified or erased where necessary. The accuracy principle takes into account the purposes for which the personal data are processed, such that the requirement is proportional to the *processing*. Finally, the *controller* must ensure adherence to the accuracy principle proactively and provide *subjects* with the possibility to correct data.

Data minimization determines that *personal data* must be "adequate, relevant and limited to what is necessary in relation to the purposes"[19]. Thus a *controller* must ensure that only the relevant *data* are being processed and collected for the purpose, and no additional *personal data*.

The fifth Fair Information Practice defines *limited data storage*, and requires that *controllers* not store personal data for longer than necessary and set a timeframe for deletion of the processed *personal data* beforehand.

Finally, the *data controller* must provide appropriate protections against loss or unlawful processing of the *personal data* of the data subject. This is also called the *data integrity and confidentiality* principle. Breaches against this and other practices must be reported to the Data Protection Officer, and, if the interests of the *data subject* are at risk, also to the supervisory Data Protection Authority.

Legal Basis

In previous sections we've already touched upon the difference between *data protection* and *privacy*, while both concepts aim to realize protection of the *private life*, the focus in *data protection* is to ensure fairness and due process with respect to use of the *personal data* of the subject. One of the central ways by which the GDPR realizes its aim of *data protection* is by mandating a proper legal basis for *processing*. The Regulation defines six lawful grounds for *personal data processing* in Article 6 "Lawfulness of processing", namely:

1. The *data processing* is based on the **informed, unambiguous and freely given consent** by the *subject* for specific purposes.

2. The *processing* is necessary because of a **contractual obligation** on the part of the *data subject*. For example, when applying for a loan with a financial institution, they can process your financial data to asses your creditworthiness.

3. *Processing* of *personal data* in order for the *controller* to comply with its **legal obligations**. A typical example of this ground is the data retention laws that frequently apply to communications providers.

4. *Data processing* necessary for protecting the **vital interests** of the data subject. The vital interests of the data subject could for example be in danger during a medical emergency.

5. The performance of a **task in the public interest** requires the *processing*.

6. The *processing* is performed based on the **legitimate interests** of the *controller*. This provision is often also called the *balancing provision* as the interests of the *controller* must be balanced with the interests and fundamental rights of the *subject*.

Only if at least one of these legal grounds applies can the *data processing* be considered legitimate under the Regulation. Our focus throughout this paper will be use-cases where the legal grounds 1 or 2 typically apply. Our motivation for this choice is that grounds 4 and 6 entail an element of subjective evaluation on the part of the *data controller*, which would be very difficult to model let alone enforce especially in a decentralized architecture as is defined by the Solid specification. With respect to grounds 5 and 3 an important challenge for enforcement of these grounds is the identification and modeling of obligations and permissions defined in legislation, the topic of Linked Legal Data[31] and the modeling of rules and requirements defined in data protection law[32] as well as reasoning over such resources[33] is a complex field of research which warrants a separate evaluation in relation to the problem space defined in this thesis.

Data Processing based on Consent or Contractual Obligation

In this subsection we will define the two legal grounds for data processing that capture the typical *data processing* activities which our reference architecture aims to support. Namely, the legal basis of consent or of a contractual obligation on the part of the *data subject*.

In particular with respect to the legal ground of consent specific requirements are imposed on the *controller* with respect to the information which should be disclosed to the *data subject*. When defining this legal basis in the Regulation, the regulator speaks of a "freely given, specific, informed and unambiguous indication of the data subject's wishes"[19]. This explicitly excludes[18] opt-out mechanisms, take-it-or-leave-it conditions when the *data subject* is presented with the consent request and vague wording or the use of fine-print to obtain consent. Also, specific attention was paid to consent given by minors where the local regulator can impose a stricter minimum age of consent. Lastly, the consent must be as easily revocable as it could be given by the *subject*. It is the burden of the *controller* to prove that they have met these requirements when obtaining consent. In practice however, the legal ground of consent is often misused [10, 34, 9] on the Web and it is even theorized[18] that it likely cannot be applied correctly in many common data processing applications.

A *controller* can also invoke a contractual necessity on the part of the *subject* as a ground for processing their *personal data*. The important criterion here is that the processing should be genuinely necessary in order to perform the contract. For example, prior to entering a loan a financial institution may perform a check to assess your creditworthiness, this is allowed under the legal basis of *contractual necessity*.

Requirements for Data Controllers

As our aim is to evaluate the compliance of authorization mechanisms with the requirements of *data protection regulation*, this last subsection will explicitly try to define the technical and informational requirements on the part of the *data controller*, based on [19, 18].

Firstly, it is expected that the *controller* keeps detailed records of its processing activities, as to prove that *personal data* was handled carefully and legitimately. This requirement puts the burden of proof with the *controller*, who has to show its compliance with legal requirements. Also, the controller must define a *data protection policy* which specifies why the data was gathered and for what purposes it is processed, how the accuracy of the data are maintained and whom can access the data.

The *controller* also must communicate with the *subject* in a transparent manner and using plain language. Furthermore, questions of the *subject* or supervisory authorities must be answered respecting strict timeframes set in the Regulation. Additionally, the *controller* must apply safe default settings and implement *data protection* by design, thus explicitly excluding dark patterns [8] that have haunted many modern-day implementations of consent on the web.

Finally, article 15 of the GDPR sets the minimum requirements for the information which a *data controller* should disclose to the *data subject* concerning the processing of their *personal data*. We will base ourselves on this article, and prior work regarding the modeling of the informational requirements under GDPR[35] to define a set of properties that should be captured by a *controller* to prove the lawfulness of processing.

At a high level an architecture for *data processing* under the Regulation should[36] capture the types of personal data being used by the service, the operations which are to be performed, the purpose of this processing, whom the recipients of the personal data are and where the data are to be stored[36]. Additionally, the legal grounds of this processing, i.e. the contract

Table 2.2: Overview of the required information disclosure by the *data controller* to a *data subject* in the context of *personal data processing* activities, under Article 13, 14 and 15 of the General Data Protection Regulation.

	Description	
1.	The **identity** of the Controller	Art.13-1a, Art.14-1a
2.	The **contact details** of the Controller	Art.13-1a, Art.14-1a
3.	The **identity** of the Controller's representative	Art.13-1a, Art.14-1a
4.	The **contact details** of the Controller's representative	Art.13-1a, Art.14-1a
5.	The **contact details** of the DPO	Art.13-1b, Art.14-1b
6.	The **purposes** of the processing.	Art.13-1c, Art.14-1c, Art.15-1a
7.	The **legal basis** of the processing.	Art.13-1c, Art.14-1c
8.	The **legitimate interests** of the Controller	Art.13-1d, Art.14-2b
9.	The **categories** of personal data concerned.	Art14-1d, Art.15-1b
10.	The **recipients or categories of recipient** to whom the personal data have been or will be disclosed.	Art.13-1e, Art.14-1e, Art.15-1c
11.	The **envisaged retention period** for the personal data will be stored	Art.13-2a, Art.14-2a, Art.15-1d
12.	The existence of the right to request from the controller **rectification** or **erasure** of personal data or **restrict or to object** to processing.	Art.13-2b, Art.14-2c, Art.15-1e
13.	The **right to lodge a complaint** with a supervisory authority.	Art.13-2d, Art.14-2e, Art.15-1f
14.	The **source** of the personal data.	Art.14-2f, Art.15-1g
15.	The **existence of automated decision-making**	Art.13-2f, Art.14-2g, Art.15-1h
16.	Where personal data are transferred to a **third country or to an international organization**, the data subject shall have the right to be informed of **the appropriate safeguards**.	Art.13-1f, Art.14-1f, Art.15-2
17.	The controller shall provide a **copy of the personal data undergoing processing**.	Art.15-3
18.	**Right to withdraw consent**	Art.13-2c, Art.14-2d
19.	Details on the statutory or contractual obligation	Art.13-2.e
20.	**Grounds** for not complying with informational right of the subject.	Art.13-4, Art.14-5

or consent the processing is based on needs to be recorded by the *controller* as the burden of proving the lawfulness of processing lies with them. These attributes align closely with the model defined in the SPECIAL Usage Policy Language[37]. Which uses a 5-tuple in its data model defining the information being processed, the purpose of this processing, the operation that will be performed, the storage location of the data and retention period, and what entities can access the outcome of the operation.

Table 2.2 highlights the required information disclosure to a *data subject* by the *controller* under Article 13[3], 14[4] and 15[5] of the GDPR. In Article 15, which describes the right of access, paragraphs 1 and 2 are of particular concern in the context of data sharing when using personal data vaults, given that a *subject* can consult their Pod at any time to receive a copy of their *personal data* such that requirements related to obtaining a copy of the personal data (as defined in paragraph 3) can be considered accomplished if the *controller or processors* do not maintain their own copies of the data.

[3]"Information to be provided where personal data are collected from the data subject"

[4]"Information to be provided where personal data have not been obtained from the data subject"

[5]Right of access by the data subject

Specifically in the context of *processing* based on consent, additional information should also be captured to ensure that the consent was informed, freely-given and unambiguous. Under the data model proposed by the GConsent[36] ontology, the entity providing the consent, status of the consent, context and expiry are persisted. Furthermore, when combined with the GDPRov[38] the data model is extended with information that can be used to assess the validity and qualitative requirements of the consent.

2.3 The Data Privacy Vocabulary

As we have highlighted before, the use Semantic Web technologies in the legal field is an active area of research, where scholars try to improve the discoverability of applicable law and prior rulings, describe the use of semantic reasoning techniques for evaluating compliance with legal requirements and develop ontologies to model real-world activities and processes in terms that relate them to the relevant legal provisions. One example of such a modeling language for defining *personal data processing* activities in relation to requirements set by *data protection* law is the Data Privacy Vocabulary.

The Data Privacy Vocabulary[6] (DPV) [39] is a vocabulary that attempts to translate concepts and requirements related to the processing of personal information under data processing and privacy regulations, like GDPR, into classes and properties that can be used as Linked Data. It is structured to be extendable with concepts and requirements for specific jurisdictions, like the DPV-GDPR extension[7] that defines the GDPR-specific rights and legal bases concerning data processing.

2.4 Conclusion

In this chapter we introduced the legal point of view on privacy and data protection, with a particular focus on the requirements set by the European Union's General Data Protection Regulation. While not a comprehensive overview of the legal realities involved in *data protection applications*, we highlighted the specific information which should be captured by *data controllers* and what considerations must be taken into account in the architecture and design of data protection applications, like the Fair Information Practices. These requirements should serve as the basis for evaluating existing solutions for authorization in Solid, and for defining our own architecture to implement compliance with *data protection* regulation in Solid.

[6] https://w3id.org/dpv#

[7] http://www.w3id.org/dpv/dpv-gdpr#

3

Related Work

The topic of data protection compliance using Semantic Web technologies has been the subject of prior work, in particular with respect to the modeling of legal requirements and processes involving personal data, as well as for expressing data sharing preferences. We start this chapter by discussing efforts regarding data usage policy languages based on Semantic Web technologies. Of particular importance in this area are ontologies like ODRL and SPECIAL, that aim to use these policy languages not only as a way to model permissions and prohibitions but also to enforce compliance. Thereafter we will focus on the work of Solid's Data Interoperability Panel with respect to improving developer experience regarding data discovery and authorization.

3.1 Semantic Policy Languages

Policy languages have been proposed[40, 41] as a flexible means of ensuring security and privacy on the Web, which due to its decentralized nature suffers from a lack of convenient security mechanisms for developers, content publishers and end-users. In general, machine-readable policy languages should allow for the regulation of behavior of components in a (distributed) system, without having to modify code or relying on manual interactions with each of the components that are governed by the policy. Through the use of Semantic Web technologies, interoperability issues between different policy languages can be resolved[40] and standardized, reliable semantics may be defined for evaluating these policies in the heterogeneous, decentralized landscape of the Web.

In this section we will first look at the field of data usage policies, and its subset of access control policies that will be the focus of subsequent chapters in the context of the Solid project. Thereafter we will highlight two examples of such semantic policy languages, ODRL and the SPECIAL policy language.

3.1.1 Access Control Policies vs. Data Usage Policies

A specific subset of policy languages enables us to define so-called data usage policies, in general these policy languages enable the restriction of how data are used in a distributed system[42]. Many current security systems cannot enforce

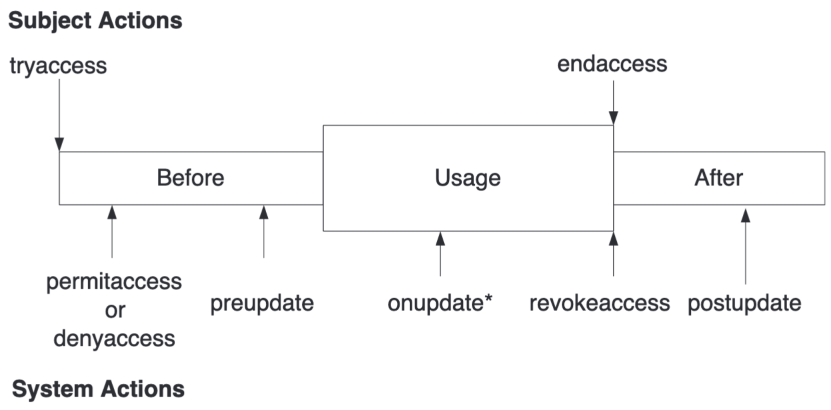


Figure 3.1: UCON model highlighting the continuity of enforcement throughout the lifecycle of data usage. Source: [44]

restrictions on the use of data after it was transferred to another system. Thus, once this exchange has occurred a sender must trust that the receiver will not misuse the information that was shared. Usage policy languages aim to resolve this shortcoming by broadening the scope of existing security mechanisms beyond just single systems and allowing end-users to express and enforce restrictions on the access, use and distribution of their information in a decentralized landscape, such as the Web.

With their UCON model, Park and Sandhu proposed[43] a pattern for usage control as a generalization of access control, such that it covers not only authorizations but also obligations and conditions. Importantly, UCON views enforcement as a continuous process. This contrasts sharply with how traditional access control mechanisms typically only make a one-time determination before resource access. Within the UCON model, three categories of enforcement mechanisms can be identified[42]; Proactive enforcement entails preventing policy violation before it occurs, through client-side or server-side techniques. Reactive enforcement is based on the assumption that it is hard to prevent the violation of a policy on a device that is not controlled by the enforcing party, such that it can be best detected after the violation has occurred. Lastly, trust-based enforcement mechanisms evaluate whether parties can be trusted to adhere to a data usage policy. In figure 3.1, the lifecycle of resource usage under the UCON model is shown.

Traditional access control policies, such as the ones we will describe in the next chapters in relation to Solid, are a specific, more limited subtype of data usage policies[43]. As we noted, they only govern resources before an exchange has occurred, in order to prevent unintended information disclosure or security breaches. Three main types of traditional access control are identified[45, 46], mandatory access control (MAC), discretionary access control (DAC) and more recently role-based access control (RBAC). Whereas MAC assumes access policies to be built into the design of the system and thus not modifiable, in DAC users with sufficient privileges can control what resources are shared with other users. MAC has therefore mostly been limited to military or government settings, whereas DAC has seen broader adoption. The newer RBAC type combines aspects of both MAC and DAC, by defining roles in the system and assigning access authorizations to these roles. Then, when a user is assigned to a role they are also granted all authorizations belonging to this role. On the other hand, with DAC the concept of a per-resource Access Control List is used[46] to express the authorized access of specific users for that resource, with a particular focus on protecting information within the constraints of a closed architecture. The distributed and open nature of the Web thus poses a significant challenge to access control policies as a security mechanism. Moreover, the academic model of the *access matrix*[47] that is typically used to assess these mechanisms has remained largely unchallenged and starts showing shortcomings when applied to an open-ended architecture where previously unknown users access resources and have to be authorized regardless of resource or user location. Nevertheless, these access control mechanisms are so prevalent because they allow for a straightforward determination[46] of the authorization of an agent and typically only require little awareness on the client-side of the authorization mechanism being used.

3.1.2 ODRL

The Open Digital Rights Language (ODRL) [48] is a rights expression language for defining policies that specify permissions, prohibitions and obligations over some entities. These deontic concepts allow for translating policies in natural language to a machine-readable format. Originally, it was introduced as a means to specify content licenses and regulate access to electronic assets, but through the ODRL profile mechanism it may be extended to support use cases in specific domains of application[49]. The ODRL core vocabulary identifies *policies* consisting of *rules* that govern *assets*. *Rules* determine that an *action* is permitted, prohibited or obligated in relation to the *asset*. Through *constraints* additional conditions under which a *rule* is applicable can be specified. The ODRL became a W3C recommendation in February, 2018.

An ODRL profile and algorithm has been proposed [50] as an extension of the existing ACL-mechanism used by Solid Pods to authorize requests. Through the semantics of ODRL, obligations and constraints can be imposed upon these authorized actions. The proposed ODRL profile[1] enables the use of concepts from the Data Privacy Vocabulary in order to define policies that relate to data processing over some resources. The proposal also contextualizes the use of such policies for materializing complex *data sharing preferences* and legal bases for processing like informed consent. The authors [50] do highlight some significant challenges with their proposal, such as the efficiency of compliance checking with these ODRL policies, especially when used in a heterogeneous, decentralized architecture and combined with an inheritance mechanism, the privacy risks associated with making these policies publicly accessible and the legal value of such *data sharing preferences* in the light of regulations like the GDPR.

Similarly, more general ODRL profiles on the topic of modeling legal requirements, such as those set by the GDPR, have been proposed. In [49], De Vos et al. define an ODRL profile to assess the compliance of business processes with regulatory requirements and define a translation of this ODRL profile into Answer Set Programming for enabling automated conformance evaluations. As an illustration of this profile, it is applied to key articles of the GDPR regarding the lawfulness of processing and transfer of personal data. The modified ODRL information model that was proposed to enable this compliance evaluation is shown in figure 3.2. Agarwal et al.[51] propose an analogous approach of using the ODRL to model legal obligations and contextualize this in a broader compliance assessment framework.

Some of the more general shortcomings of the ODRL that have been noted pertain to representing delegation, defining conflict resolution and resolving semantic ambiguities when defining duties of some party[52]. Nevertheless, the ODRL has

[1]https://w3id.org/oac/

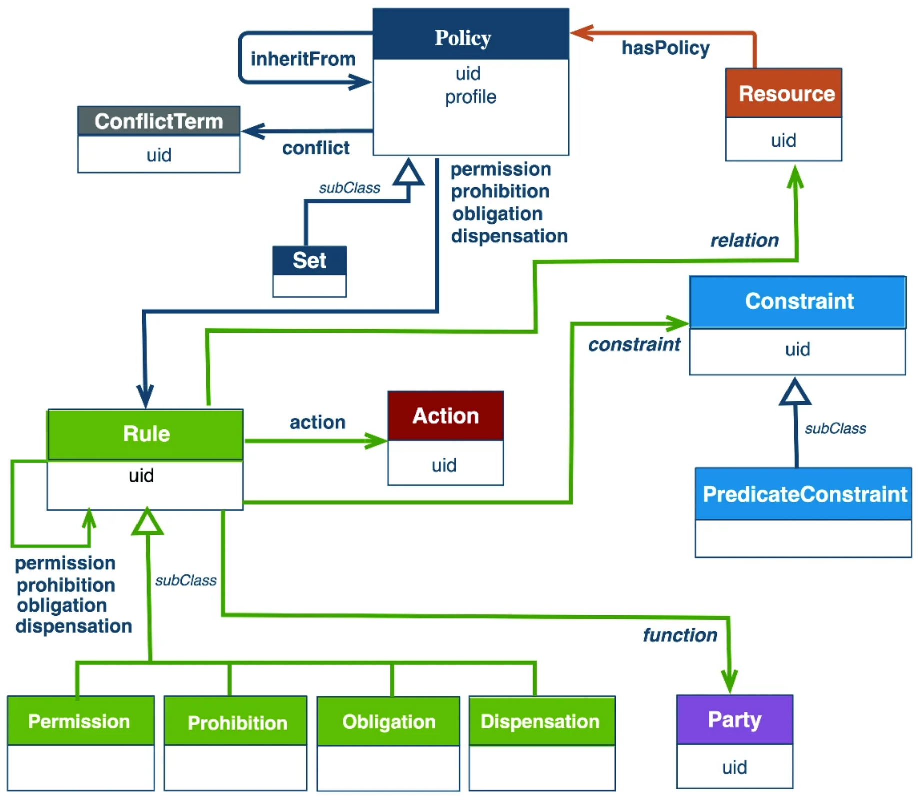


Figure 3.2: The ODRL information model

Source: [49]

been used in practice, for example in the RightsML Standard[53] which serves as a rights expression language in the domain of the media industry allowing for a media asset from a publisher to be annotated with instructions on the rights associated to the content.

3.1.3 SPECIAL

The SPECIAL project[2] was a research project that aimed to deliver technologies to reconcile big data applications with the necessary regulatory compliance with respect to data protection and privacy. It delivered user interfaces to be used by data subjects for consent and processing transparency. Moreover, ontologies were developed for the logging of data processing applications[3] and for modeling data usage policies[4] of both data subjects and data controllers that are machine verifiable.

Usage policies defined using the SPECIAL Usage Policy Language[54] consist of a set of five-element tuples, detailing the following information on the processing activity:

Data Information that is to be processed by the operation.

Purpose The purpose of the processing.

Processing A description of the processing itself.

Storage Where the result of the processing is to be stored and for how long.

Recipients The entities allowed to access the result of the operation.

The consent of a *data subject* for some processing is then materialized as the set of activities that are allowed by the data subject, in terms of these five-tuples. Similarly, a *data controller* specifies the processing activities they would like to perform in their own usage policy. If all operations in the *data controller's* usage policy are also contained in the *data subject*'s consent, the controller's policy is found to be in compliance with the subject's usage policy. Notably, the SPECIAL policy language is encoded in OWL2 to enable reasoning engines to efficiently perform these compliance checks in an automated manner[55].

While evaluating different technical approaches that support the enforcement of legal rights given to data subjects under data protection regulation like GDPR, an assessment [56] was made of the affordances with respect to data governance provided by Solid and the SPECIAL project in comparison to the current defacto standard of data subjects giving very broad consent to processing. In the evaluation of Solid in relation to data protection regulations it was found that Solid's current ACL-based solution for authorization falls short when trying to implement solutions adhering to the strict regulatory requirements set forth. Firstly, because of its poor user experience caused by issues like the lacking interpretability of access mode and resource identifiers for non-technical users, risk of phishing attacks due to the use of IRIs to identify agents, and

[2]https://specialprivacy.ercim.eu

[3]the SPECIAL Policy Log Vocabulary, https://ai.wu.ac.at/policies/policylog/

[4]the SPECIAL Usage Policy Language, https://ai.wu.ac.at/policies/policylanguage/

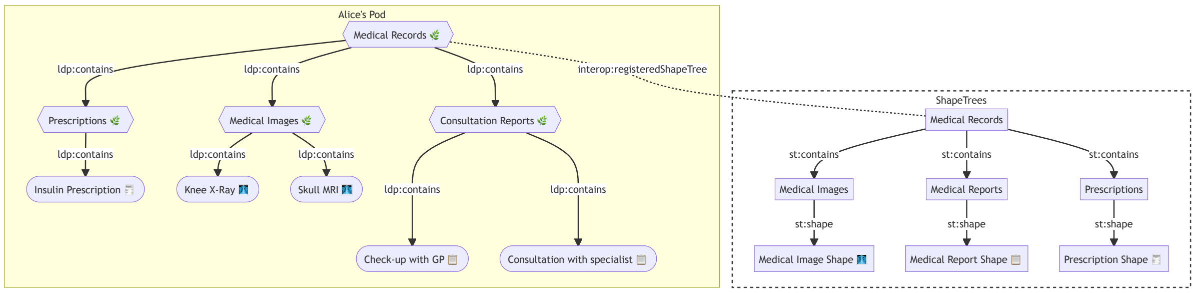


Figure 3.3: Overview of an example resource hierarchy for medical records using Shape Trees.

the security concerns that arise from inherited ACL rules. Secondly, because ACLs fail to capture important concepts under data protection regulations that define what type of information is being shared, how that data will be processed and for what purpose, and which legal basis is used to warrant this processing. And lastly, because the burden of modifying these ACL rules is currently delegated to application developers themselves, thus contradicting the original goals of returning control back to the end-user as developers have virtually unlimited authority when modifying ACL rules and could resort to the dark patterns that have haunted modern-day implementations of consent on the web.

A layered, decentralized architecture for combining SPECIAL and Solid was also proposed and compared to these other approaches [56]. The concrete mechanics of the policy exchange and negotiation are left as future work by the authors, however their evaluation provides a good insight into the existing limitations of ACL based authorization when confronted with complex data processing applications.

3.2 Solid's Data Interoperability Panel

The Solid Community's Data Interoperability Panel[5] was started with the goal of standardizing the mechanics by which different applications can interoperate over the same data safely and effectively. Furthermore, the panel's work tries to increase user awareness and interpretability of the data stored in a Pod, by abstracting away complexities such as resource organization that are currently not governed by the Solid protocol, in order to finally enable multiple agents to truly interoperate over the same data. Most importantly, the Data Interoperability Panel aims to tackle these hurdles while preserving the fundamentals of the Solid protocol as it exists today.

Two significant proposals have been taking shape in the panel over the past year, namely the *Shape Trees* [57] and the Solid Application Interoperability [58] draft specifications. The former builds upon the existing specifications of RDF[6] and

[5]The Solid specification is drafted by different community panels, each focused on specific issues or domains that are relevant to Solid like authentication, authorization or data interoperability.

[6]The Resource Descriptor Format, core data model used in Semantic Web technologies to construct Linked Data resources.

data shapes [59, 60], which respectively provide us with the foundations for interoperability through unambiguous identifiers (IRIs) and a structural schema against which individual RDF graphs can be validated. Where these existing specifications fall short, however, is in modeling complex resource hierarchies.

Consider for example the organization of a collection of medical records that takes form in a Solid Pod where developers have relative freedom in both resource naming and the use of containers to gather their data. A Shape Tree defines structural constraints for a tree of resources in any ecosystem that has a notion of containers[7]. For each container, it allows shape constraints to be imposed on the contained resources. Shape Trees themselves can also contain other Shape Trees giving form to tree hierarchies (for example medical records as a whole may consist of medical images, prescriptions, bills, reports, etc. as is shown in figure 3.3).

The major strength of Shape Trees is that they can unambiguously define resource organization in a Pod and provide a higher-level abstraction that can be more easily understood by end-users. This way, Shape Trees guide applications and users by determining where data should be written to and where it can be read from. The modeling of related resource collections in this manner allows us to perform operations such as authorization, data migration and validation on this higher abstraction level as well. Especially in the context of authorization, defining rules at the level of Shape Trees rather than individual resources reduces complexity, the likelihood of errors and allows us to relate these higher-level conceptual resource aggregations to legal concepts such as Data Categories.

The Solid Application Interoperability (SAI) draft specification [58] leverages these proposed Shape Trees to standardize concrete mechanics by which applications and agents request access to information in a Solid Pod, the way by which they locate the concrete instances of the Shape Trees, and how they can interoperate over this data. Up until now most of the specifics of these different operations were left open to individual application developers by the Solid specification, complicating interoperability over the same data. In the context of this research, the standardizing of access requests is of specific importance, and will be used as a building block in our proposal.

The SAI specification introduces the concept of an Authorization Agent as a service linked to an agent's WebID that manages the data under their control. It is tasked with processing access requests for the agent, managing previously granted permissions, and recording the concrete instances of Shape Trees through a collection of registries.

While the specification is still under discussion by the panel, and some aspects of the mechanics of the authorization agent have not yet been fully defined or are deliberately being left open for implementation, we will be using many of the core concepts it sets forth in our proposal.

[7]Solid builds upon the Linked Data Platform specification which governs the semantics of a container resource.

4

Authorization in Solid

To realize data governance by the user, the Solid Protocol 0.9 relies on Web Access Control as an authorization mechanism. Web Access Control is a cross-domain, decentralized solution for *authorizing* requests using *Access Control Lists* (ACLs) expressed as Linked Data. It is typically enabled by having the Solid Pod (*Resource Server*) make the authorization decision, based on identity details of the requesting party provided by an IDP[1]. This section starts with a technical introduction to Web Access Control through some practical examples, after which we define the technical capabilities of Web Access Control and highlight relevant limitations. In fulfillment of research objective 3 we will then look at the adherence of implementations of data processing using the Solid Protocol 0.9 to the legal requirements defined in chapter 2. Finally, the recent proposal of Access Control Policy as a new authorization framework and changes to the Solid-OIDC protocol in Solid are briefly discussed and contrasted with the existing protocol.

4.1 An introduction to Web Access Control

Solid's primary mechanism for discretionary access control is the Web Access Control (WAC) specification [16]. It employs the `acl` ontology[2] to express *access modes* applicable to some resource for an agent, where both the agent and the resource are identified using IRIs[3]. WAC supports four access modes in its rules, namely:

Read Allowing for full or partial read operations on resources.

Write Allowing for write operations on resources, i.e., create, update, or delete.

Append Allowing for append operations on resources, i.e., to add information to the resource but not remove any.

Control Allowing for read and write operations on the resource's *associated ACL resource*. This permits the grantee to delegate or revoke access to the resource.

[1] Identity Provider, in the context of Solid any entity implementing the Solid-OIDC specification.
[2] `<http://www.w3.org/ns/auth/acl#>`
[3] Internationalized Resource Identifier

The access subject[4] is typically referred to by their WebID. The WebID specification[14] defines how identification is realized in the Solid ecosystem, i.e. through a profile document that can be retrieved by dereferencing the WebID IRI. The WebID profile document itself contains identity claims on the identified subject, like their contact details through the `vcard`[5] ontology, interpersonal relations via `foaf`[6] or, perhaps more relevant to Solid, information on the trusted IDP of the user by using terms from the `solid`[7] namespace. Note that through the special agent IRIs `foaf:Agent` and `acl:AuthenticatedAgent` authorization can be granted to respectively any public, unauthenticated agent or any authenticated agent. Furthermore, it is possible to define groups of authorized agents as well as restrict agent access by the origin of their request.

The access object[8] can be either the IRI of a folder (*container*) or a file (*RDF-* or *non-RDF resource*) in the Pod. Furthermore, through the `acl:default` predicate it is possible for an authorization in the ACL resource to recursively apply to resources contained in the folder it references. This implies that WAC can have recursive inheritance with respect to container hierarchies, such that any member resource inherits the ACL of the closest container resource when recursing towards the root of the Pod.

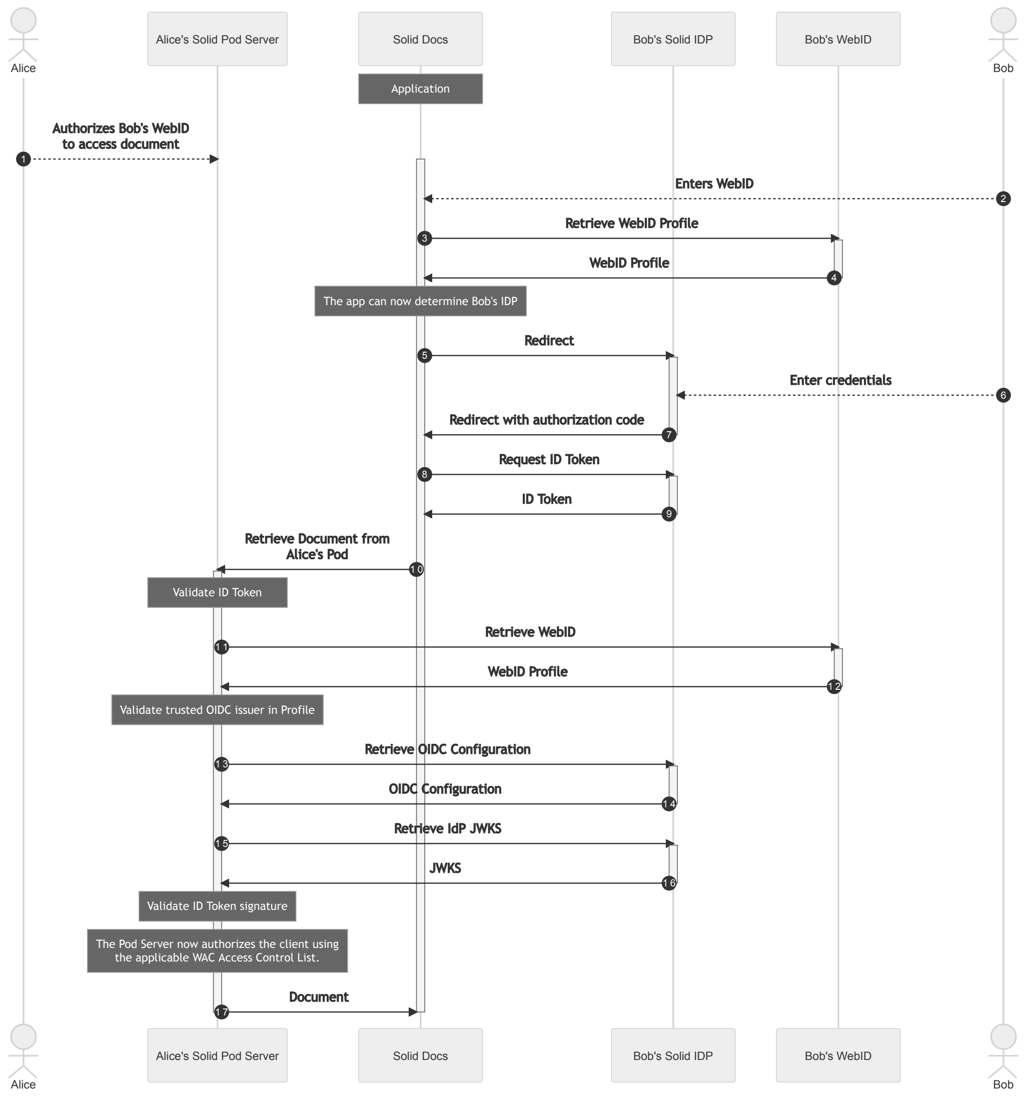
In figure 4.1 we use a sequence diagram to highlight how WAC currently facilitates the sharing of resources by having the Pod server evaluate both the authentication and authorization of the requesting party (*steps 10 to 15*). The WebID of actor Bob plays a crucial role here, as it will be used by actor Alice to identify Bob in the ACLs governing the resource being shared. At this point we must also note an important caveat with respect to *step 1*, where Alice grants access to Bob to access her document, as the Solid Protocol 0.9 does not normatively define the process by which Bob can suggest what access he needs for using an application to interact with the shared resources. Thus Alice, or the application she is using to interact with her Pod, has to assess Bob's access needs through an out-of-band process and then modify the WAC ACLs to reflect these requirements.

4.2 Technical Capabilities & Limitations

Web Access Control offers many of the capabilities in terms of data sharing controls that users of services like Google Docs or Dropbox will be familiar with. People can be marked as viewers, editors or administrators[9] of some resources in the Solid Pod, and these permissions are given to a party identified, not by their e-mail address, but by their WebID. The simplicity of this model, and analogy with this existing user experience are a core strength when using WAC in typical social interactions. In this setting we can also ignore the strict requirements of data protection law, as it likely does not apply in this *household* setting.

A first limitation of Web Access Control is that its access modes are broad and do not map well to the more common CRUD[10]

[4]Authorized agent
[5]`<http://www.w3.org/2006/vcard/ns#>`
[6]`<http://xmlns.com/foaf/0.1/>`
[7]`<http://www.w3.org/ns/solid/terms#>`
[8]Resource to which authorization is being managed
[9]i.e. able to delegate or revoke access modes for other parties, including themselves
[10]Acronym for Create, Read, Update, Delete.



Alice
Alice's Solid Pod Server
Solid Docs
Application
Bob's Solid IDP
Bob's WebID
Bob
1 Authorizes Bob's WebID to access document
2 Enters WebID
3 Retrieve WebID Profile
4 WebID Profile
The app can now determine Bob's IDP
5 Redirect
6 Enter credentials
7 Redirect with authorization code
8 Request ID Token
9 ID Token
10 Retrieve Document from Alice's Pod
Validate ID Token
11 Retrieve WebID
12 WebID Profile
Validate trusted OIDC issuer in Profile
13 Retrieve OIDC Configuration
14 OIDC Configuration
15 Retrieve IdP JWKS
16 JWKS
Validate ID Token signature
The Pod Server now authorizes the client using the applicable WAC Access Control List.
17 Document

Figure 4.1: Overview of how Web Access Control may currently facilitate the sharing of a document from the Pod of actor Alice with actor Bob. Note in particular how Alice's Pod Server is responsible for both authentication and authorization in steps 10-15.

permission model [61]. Also, some of these access modes will align poorly with user expectations: e.g., what does it mean to have Append permissions over a container of resources? There have been suggestions made by Solid community panels, to align the access modes with CRUD while retaining backwards compatibility with existing applications of the broader access modes. However, these discussions have not yet made their way into the specification.

Furthermore, we noted that WAC uses an inheritance mechanism to determine which ACL resource is the effective ACL governing some resource or container resource in the Pod. While this inheritance mechanism might be reasonably easy to understand for developers, to an unaware end-user, this behavior can be counter-intuitive or lead to unintended information disclosure. For example, when a user grants an app access to a container through `acl:default`, they implicitly grant access to all data transitively contained within, including any new resources that are added after the user granted access.

The use of IRIs to both identify resources and agents might also contribute to poor user experience and lead to security breaches. For example, users might perceive an analogy between how they would typically manage a photo collection in a filesystem on a computer, and how pictures are stored in a folder in one's Pod. That way, an end-user could have some understanding of what kind of data are being shared, as they can easily open the files and look at their contents. However, the analogy falls short when it comes to *structured* data, which is commonly persisted as Linked Data in the Solid Pods. In this case, resource IRIs do not necessarily have meaning, and the organization of resources can be chosen arbitrarily by application developers. A similar concern is applicable to agent IRIs: How do I know my doctor's IRI is actually `https://nhs.gov.uk/id/123#me`? According to the UK Government's Department for Digital, Culture, Media & Sport's 2020 Cyber Security Breaches Survey [62] phishing attacks are one of the most common type of breaches experienced by UK businesses. Being just ordinary IRIs in the context of ACL rules, WebIDs suffer the same risk of being used in phishing attacks, where very similar looking WebIDs could be constructed that open the doors of your Pod to malicious actors. Detection mechanisms for phishing IRIs have been proposed, however these fall largely in the realm of heuristics[63].

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.

# Your doctor has Read, Write & Control Access to your Medical Records
<#records> a acl:Authorization;
          acl:agent <https://nhs.gov.uk/id/123#me>;
          acl:default <./MedicalRecords/>;
          acl:mode acl:Read, acl:Write, acl:Control.
```

Listing 4.1: Example ACL resource

Let us illustrate the mechanics of WAC using an example; a doctor is requesting access to the medical records of their patient. The use case is shown in detail through the sequence diagram in figure 4.2. If we were to realize this type of interaction pattern with Web Access Control, the patient would have to modify *(steps 2-3)* an ACL resource governing the medical records stored in their Pod, as shown in Listing 4.1. Through this ACL, the doctor (identified by their WebID `https://nhs.gov.uk/id/123#me`) obtains read and write permissions on the patient's medical records as well as control permissions such that information can be shared with trusted colleagues. Note that the choice of a container named "MedicalRecords" to retain your medical information is a completely arbitrary one, such that the interpretability of this ACL rule could be considerably worse if the developers of these medical record applications made arbitrarily different naming choices. Also, as noted before, the Solid protocol currently does not define how the doctor should request for their patient to grant these rights *(step 1)*. Having the interpretability and modification of an ACL rule depend fully on implementation choices of the developer is not a desired behavior for an authorization system, let alone one that aims to maximize end-user control.

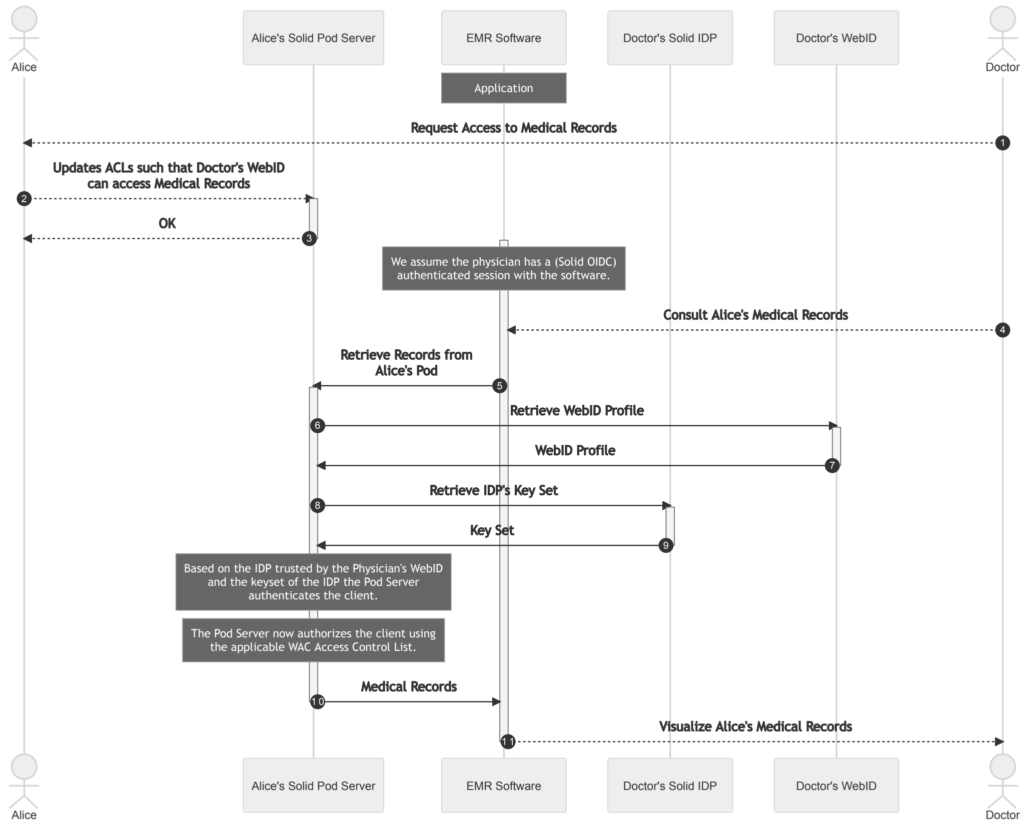

Alice
Alice's Solid Pod Server
EMR Software
Doctor's Solid IDP
Doctor's WebID
Doctor
Application
Request Access to Medical Records
Updates ACLs such that Doctor's WebID can access Medical Records
OK
We assume the physician has a (Solid OIDC) authenticated session with the software.
Consult Alice's Medical Records
Retrieve Records from Alice's Pod
Retrieve WebID Profile
WebID Profile
Retrieve IDP's Key Set
Key Set
Based on the IDP trusted by the Physician's WebID and the keyset of the IDP the Pod Server authenticates the client.
The Pod Server now authorizes the client using the applicable WAC Access Control List.
Medical Records
Visualize Alice's Medical Records

Figure 4.2: Illustration of the exchange of a patient's medical records through a Solid Pod, where the patient must modify ACL rules to authorize the physician's Electronic Medical Records software.

4.3 Adherence to Data Protection principles & Fair Information Practices

In this section we aim to relate informational and technical requirements of *data protection* law, and more specifically the GDPR, to the capabilities and limitations of the Solid protocol 0.9 with respect to authorization. Some of these requirements currently cannot be enforced or can only be captured through out-of-band mechanisms, in which case the *controller* must ensure compliance through additional, proprietary measures which potentially reduce transparency for the *subject* and may limit the interoperability that Solid aims to enable. We will start by focusing on the informational requirements as summarized in table 2.2 and how these are (partially) realized in the Web Access Control specification, thereafter we will focus on the "due process" requirements concerning *legal basis* and *purpose limitation* mandated under *data protection* law. This relates closely to the Fair Information Practices, which we will use to define areas of improvement for the Solid protocol when used in *data processing* applications.

4.3.1 Informational Requirements

The Web Access Control specification already captures information for the purposes of authorization, like the authorization subject (i.e., the WebID), the authorization object (i.e., the resources to which access is being granted) and the granted access modes. In table 4.1 we relate the information captured by WAC to the properties the GDPR requires data controllers to communicate to data subjects. A summary overview of informational requirements with respect to data subjects under the GDPR can be found in table 2.2, however we must note that a number of these fall outside the realm of what can reasonably be enforced by Solid as we will elaborate on in this section.

With respect to the recipients of the personal data, Web Access Control captures information on the authorized agent or group of agents via the `acl:agent` and `acl:agentClass` properties of the ACL. This property will reference to which agent or agents the authorization has been granted, using their WebID. However we must note that similar to an e-mail address, a WebID does not uniquely nor strongly identify a person or organization thus this identification mechanism may not comply with legal requirements unless it is supplemented with additional, out-of-band safeguards. Furthermore, additional recipients may be involved in further processing of information collected from the Pod such that the agents listed here may give only a partial view on the processing activity.

The resources to which access is being granted are also identified in the access control list. This attribute relates, at least partially, to the requirement of defining what categories of personal data are concerned. As we noted in the previous sections, resource identifiers are not necessarily meaningful thus supplemental information on what category of personal data these resources fall under will have to be provided. Type indexes have been proposed by members of the Solid community for the purpose of data discovery and resource organization, but could also prove useful for defining data category membership. Furthermore, it is possible that additional information outside of what is being shared directly through the Solid Pod is also being collected and processed by the *controller*, think for example of recommender systems which build automated profiles of their user's tastes and preferences.

When we consider the information that is not being captured by Web Access Control, knowing the *purpose* and *legal basis*

Table 4.1: Properties of a Web Access Control ACL and their relation to the informational requirements under Article 15 of the GDPR

Property	Description	Relation to GDPR's informational requirements
`acl:accessTo`	Identifies the resources to which access is being granted	The **categories** of personal data concerned (Art.15-1b)
`acl:default`	Identifies a container resource whose authorization can be inherited	
`acl:agent`	Identifies the agent to which permission is being given	The **recipients or categories of recipient** of the personal data (Art.15-1c)
`acl:agentClass`	Denotes a group of agents to which permission is being given	
`acl:mode`	References the access modes that have been granted	/

of the data processing may be the most important requirements towards realizing compliance with *data protection* law. Especially given the fact that this information is necessary for evaluating the *purpose limitation* and *lawfulness, fairness and transparency* criteria from the Fair Information Practices listed in table 2.1. Nevertheless, the determination of compliance to the specified purpose and legal basis is one which cannot be made on objective information alone, such that enforcement will likely have to involve human evaluation by a data protection officer, regulator or court.

The *limited data storage* requirement from the FIPs, determines that data should not be stored for longer than necessary. However, in the context of Solid Pods this determination is complicated by the fact that authorizations in Web Access Control do not define a period of validity. If such a concrete start and end time for an ACL rule were added, it could be objectively enforced by the Pod's authorization mechanism. The GDPR does, however, allow the period in which data storage and processing is warranted to be defined through other criteria as well, which may not be enforceable through automated means either.

Given the fact that we consider the data being shared for *data processing* activities is stored in the user's Solid Pod, some requirements of the GDPR will have already been (partially) fulfilled by design. First, the user can access the personal data they are sharing directly from their Pod, unless copies or derived work is being stored elsewhere by the data controller in which case additional transparency procedures will still be required. Second, given the fact that the most common Solid server implementations, like the Community Solid Server[11], explicitly give the owner `control` access over any ACL governing their Pod we consider the rectification, erasure and restriction of processing to be enabled by the Solid server, at least for the information it stores in the context of the processing. Furthermore, the source of the personal information will also be known to the end-user already. We must note however that these requirements also have implications for other processing and collection activities in the broader architecture of the service or application that processes the *personal data* and thus further legal, technical and organizational measures may be needed for attaining full compliance.

The right to lodge a compliant, existence of automated decision-making and transfer to third countries largely fall outside of the realm of what can be enforced by the Solid Pod. Therefore we will not consider these requirements as relevant to the architecture we will be presenting.

[11] https://github.com/CommunitySolidServer/CommunitySolidServer

4.3.2 Legal basis & Purpose of Processing

As we highlighted, the Web Access Control ontology does not capture the legal basis nor the purpose for data processing as mandated under the GDPR and similar data protection laws. From the perspective of a low-level authorization mechanism, this choice seams reasonable as we would be relying on the authorized party to self-report this information with no automated means of validating that the provided legal basis and purpose are in fact justified. Nevertheless, such information would be most relevant in the context of enforcement efforts by regulators and to provide transparency to the end-user.

Efforts to capture these details in Solid have been proposed, for example by combining the proposed ODRL profile for Solid's access control mechanism[50] with concepts from the Data Privacy Vocabulary[39]. Moreover, a recent survey[35] of policy languages and vocabularies for modeling information flows under the GDPR has concluded that when complemented with the ontology terms of the DPV[39] and GDPRtEXT[32] it is possible for ODRL to model 39 out of the 57 informational items that have been defined under the regulation. Therefore, it seems like a promising approach to supplement the information that is strictly needed for enforcement, like is currently modeled by WAC, with concepts from other ontologies like the DPV that can allow for greater transparency for both regulators and data subjects. Capturing this information would also help in improving compliance with the *lawfulness, fairness and transparency* and *purpose limitation* requirements defined by the Fair Information Practices under the GDPR.

4.3.3 Fair Information Practices

Besides the *lawfulness, fairness and transparency* and *purpose limitation* principles, it is important that we also consider other Fair Information Practices and how we could improve upon those in the design of Solid's authorization mechanism. In particular, because an authorization in the ACL does not define any period of validity or other temporal constraints it is not possible for Solid's authorization mechanism to enforce the requirement of *limited data storage*.

In relation to the *data integrity and confidentiality* principle, we note the challenges in terms of interpretability for ACL rules as well as the risks that WAC's inheritance mechanism may pose for unintended information disclosure. Additionally, Solid currently does not have standardized mechanisms for audit logging, which complicates the detection of a breach or occurrence of unlawful processing and thus fully relies on the adherence of the *data controller* to this requirement.

We must note however that some FIPs will already be adequately fulfilled, even in the current design of the Solid protocol, given that no additional copies of the data being processed are stored outside of the *subject's* Pod. For example, with respect to *accuracy*, if a *data subject* will be sharing data directly from their Pod this enables them to keep this data up-to-date and instantly rectify any inaccuracies.

4.4 The introduction of User-Managed Access in Solid-OIDC 0.1.0

In the 0.1.0 revision of the Solid-OIDC specification[15] the concepts defined under the User-Managed Access (UMA) 2.0 Grant for OAuth 2.0[64] authorization were introduced in Solid as a means of decoupling the authorization process from the Solid Pod (i.e., the Resource Server). These changes are still relatively new, and haven't been adopted in common implementations of Solid like the Community Solid Server[12]. Nevertheless, there seems to be significant potential in some of the affordances of the UMA specification, for example its *claims pushing* mechanism allows for requesting parties to provide details on their authentication and authorization, such as the customary OIDC ID Token, but which can now be supplemented with additional information like Verifiable Credentials. Furthermore, as we will show in the next chapters, the UMA Authorization Server can also serve as the basis for new authorization mechanisms to be implemented and evaluated, while not impacting the resource server.

In this section we will first look at how the typical request flow has been impacted by the introduction of User-Managed Access with respect to the Solid Server and a requesting party. Thereafter, we highlight the role of the authorization service under UMA 2.0 and its interaction with respect to other components like the Solid server and the Requesting Party. Finally, some important considerations with respect to the implementation and use of an authorization server are noted.

4.4.1 Introduction of UMA 2.0 in the request flow

In figure 4.3 we demonstrate the request flow that the Solid Docs application, also used in the example in figure 4.1, would have to perform under the revised Solid-OIDC specification because of the introduction of User-Managed Access 2.0. Highlighted in the gray boxes are additional requests introduced in the authentication/authorization-flow in Solid-OIDC 0.1.0.

The process of authenticating the end-user through Solid-OIDC has not changed, for this the specification still relies on the authorization code grant type of OpenID Connect 1.0 with PKCE[13]. However the request flow starts to differ significantly when the requesting party wishes to use the OpenID Connect ID Token to request some resource from a Solid Pod. At this point, the client will be presented with a 401 Unauthorized error *(steps 7-8)*.

The 401 error will refer to a UMA 2.0 Authorization Service (AS) that is trusted by the resource server (the Solid Pod). A Pod provider will choose their own AS, and could even change authorization services at some point in the future. In addition to a reference to the AS, the error will also return a so-called UMA ticket to the requesting party. This ticket value is used by the AS to identify the resources to which authorization of the requesting party should be determined.

In a subsequent discovery phase the token endpoint of the AS will be discovered by the requesting party such that the UMA ticket can be exchanged for an access token given the correct authentication and authorization claims *(steps 8-9)*. Thereafter, the UMA claims pushing process is performed with the Authorization Service's token endpoint *(steps 11-18)*. During this process, the requesting party presents its UMA ticket along with the previously obtained OpenID Connect ID Token and the

[12] https://github.com/CommunitySolidServer/CommunitySolidServer

[13] Proof Key for Code Exchange, extends the Authorization Code flow in order to mitigate the risk of interception of the authorization code.

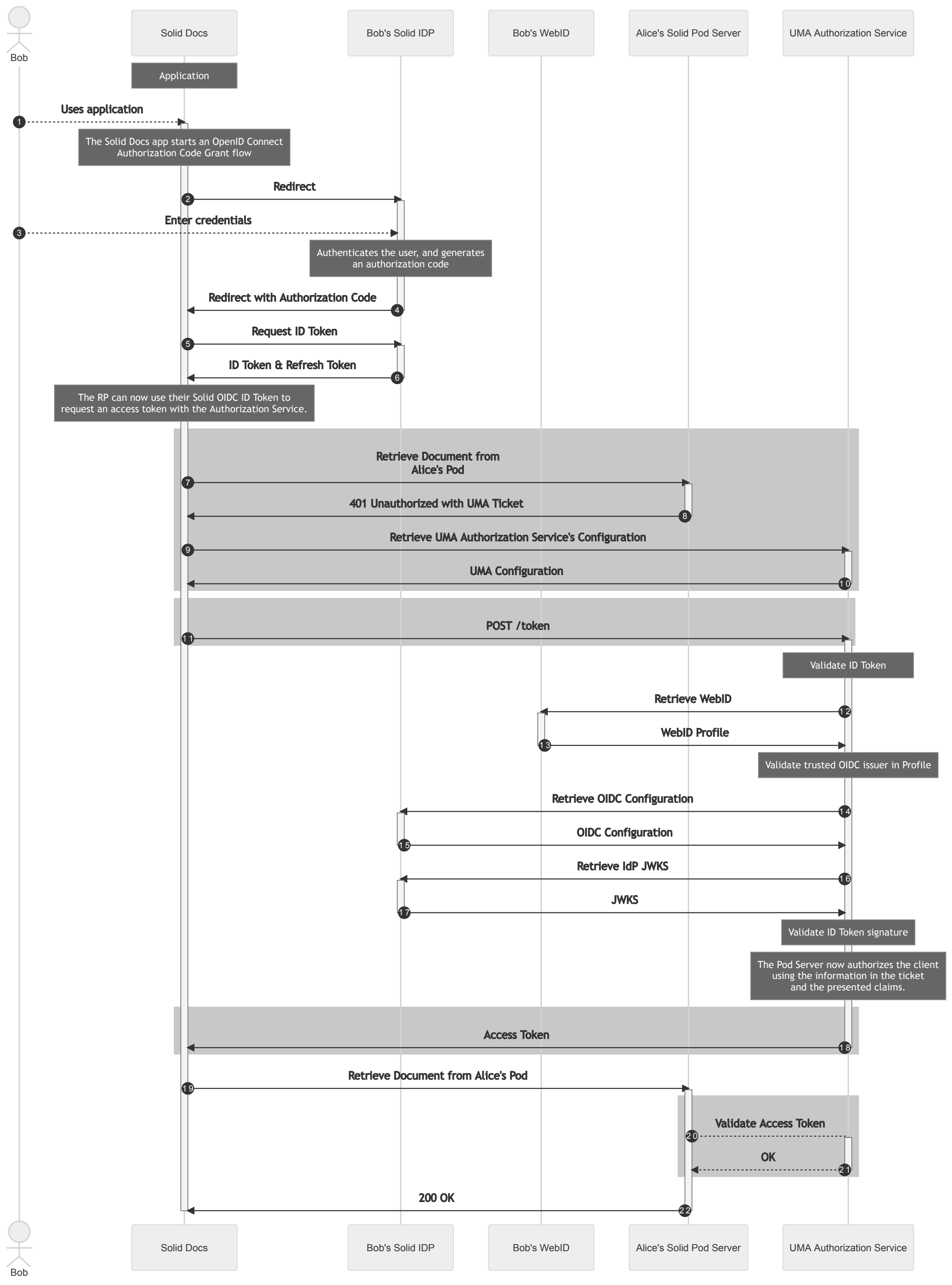

Bob
Solid Docs
Bob's Solid IDP
Bob's WebID
Alice's Solid Pod Server
UMA Authorization Service
Application
Uses application
The Solid Docs app starts an OpenID Connect Authorization Code Grant flow
Redirect
Enter credentials
Authenticates the user, and generates an authorization code
Redirect with Authorization Code
Request ID Token
ID Token & Refresh Token
The RP can now use their Solid OIDC ID Token to request an access token with the Authorization Service.
Retrieve Document from Alice's Pod
401 Unauthorized with UMA Ticket
Retrieve UMA Authorization Service's Configuration
UMA Configuration
POST /token
Validate ID Token
Retrieve WebID
WebID Profile
Validate trusted OIDC issuer in Profile
Retrieve OIDC Configuration
OIDC Configuration
Retrieve IdP JWKS
JWKS
Validate ID Token signature
The Pod Server now authorizes the client using the information in the ticket and the presented claims.
Access Token
Retrieve Document from Alice's Pod
Validate Access Token
OK
200 OK

Figure 4.3: Illustration of the authentication and authorization flow under the Solid-OIDC 0.1.0 specification using User-Managed Access 2.0 for Authorization. Sequences marked in gray highlight changes from the previous Solid-OIDC protocol.

AS will perform the same validations that were previously performed by the Solid Pod Server to determine the validity of the ID token for the WebID of end-user Bob *(steps 12-17)*.

Finally, the authorization service determines the authorization of the authenticated client with respect to the resources referred to in the UMA ticket. For this, the Authorization Service evaluates the applicable Web Access Control rules under the Solid Protocol 0.9. Nevertheless, in the future other authorization mechanisms like ACP or Verifiable Credential-based access control could also be enforced by the AS.

Once the claims pushing process has completed and the authorization service was able to determine the access modes of the requesting party, an access token is returned to the client that can be used to interact with the resource server *(step 18)*. The concrete validation mechanism for these access tokens, which is to be used by the resource server *(steps 20-21)*, is not normatively defined by the Solid-OIDC 0.1.0 specification but can be determined by the implementer of the resource server.

4.4.2 Role of the UMA Authorization Service

As we've noted a trust relationship will have to be established between the UMA Authorization Service and the Resource Servers (Solid Pods) in order for the access tokens generated by the former to be trusted by the latter. The Resource Server relies on the AS for authenticating the requesting party and evaluating the authorization it has for the requested resources. In the process, the resource server need not consider the evaluation of authorization rules, such as the ACLs under Web Access Control, in its design or implementation which can reduce its complexity. The establishment of this trust is not covered in a normative sense under the specification, rather it is left open to the implementation by developers and operators of Solid Pods. A similar reasoning is followed regarding how the resource server obtains the UMA ticket it should present upon a 401 Unauthorized error.

Crucially, under this new authorization flow the interface between the authorization service and the Solid Pod server is formed by the access modes that are contained in the access token. Thus, specification efforts will have to focus on defining these access modes independently of individual authorization mechanisms, like Web Access Control, such that the Authorization Service could support the enforcement of novel mechanisms as well. An important challenge to this are WAC's broadly defined access modes which today are tightly integrated into many implementations (cfr. section 4.2) and the fact that more granular, CRUD-inspired modes are still under discussion. Thus for now, the choice and meaning of access modes exchanged between resource server and authorization service is left open to the definition of implementers as well.

While introducing the authorization service reduces technical complexity on the side of the resource server, the AS will have to take over these responsibilities. Moreover, additional request latency is introduced due to the UMA flow incurring at least three additional round-trip times (RTT) in a cold-start situation where no cache of the authorization service's configuration exists with the requesting party, when compared to the prior Solid-OIDC flow. Depending on implementation choice on the part of the authorization service, the additional request latency for the requesting party could on average be as high as $2 \times RTT$ in the case where a new ticket is generated for each request to a new resource.

4.5 Access Control Policy: A proposal for improving Access Control in Solid

Access Control Policy (ACP)[65] is a recent proposal in the Solid Authorization community panel. It is intended as an alternative to Web Access Control, and addresses a number of the shortcomings we have highlighted about the latter. Consequently it has been proposed for inclusion in future revisions of the Solid Protocol. In this section we will briefly touch upon ACP and the ways in which it differs from Web Access Control, as well as link these improvements to the limitations of the current protocol as were described in section 4.3.

A first significant change in ACP is the fact that it explicitly decouples the location of resources from the storage of their respective access control lists. This decoupling is realized through the `acl` Link header relation. Notably, in ACP any resource has exactly one effective access control resource (ACR) such that it does not rely on a hierarchical mechanism as is defined by Web Access Control where a recursive traversal of parent containers is needed in order to discover the effective ACLs. We have previously identified this mechanism as a potential risk for unintended information disclosure, due to the fact that it is not explicit to the user.

ACP uses the concept of Policies to govern the access modes that are granted or denied for agents that conform to the requirements of a Matcher. The effective ACR then applies these policies to the resource it governs. Policies can be satisfied if all, any or none of its Matchers are satisfied. The fact that ACP allows the explicit prohibition of certain access modes to some agent is another significant difference from WAC, which does not allow for expressing such negative authorization rules.

A Matcher, as defined under the ACP specification, allows much greater flexibility in defining which agents an authorization applies to. Besides the identification of the concrete agents that are authorized like in WAC, ACP also explicitly defines properties such as the creator or owner of a resource, the type of a VC presented by the agent, or the OIDC OP to be used as criteria by matchers. Also, further matchers can be defined by implementers as a sub-property of `acp:attribute` such that this Matcher-class becomes an extension point of the ACP specification.

Nevertheless, ACP does not fully solve the limitations of WAC with respect to informational requirements under data protection law. For example aspects like purpose and legal basis are not captured in its design. Notably, resource and agent identification is still handled in the same manner as in WAC, with similar risks for phishing and unintended information disclosure. However, its extensible matcher concept may allow one to materialize a consent as a Verifiable Credential of a certain type for example, or define custom matchers for purpose, period of validity and legal basis.

In conclusion, ACP may offer a more flexible and expressive framework for authorizations in Solid while avoiding certain pitfalls of WAC like resource inheritance. However, it still does not fully align with specific informational requirements under *data protection* law and suffers from similar challenges as WAC with respect to resource and agent identification. Its proposed use of Verifiable Credential based authorization shows significant promise, but a practical implementation of this concept has not yet been demonstrated.

4.6 Conclusion

In this chapter we introduced Web Access Control, Solid's primary authorization mechanism under the 0.9 revision of the Solid Protocol. In an evaluation of its technical affordances we noted that its broad access modes, the inheritance mechanism it enables and the use of IRIs to identify both resources and agents could lead to unintended information disclosure or even security breaches when used thoughtlessly. Furthermore, the information recorded in authorization rules lacks some critical details for compliance with data protection law. For example purpose, legal basis, and time restrictions on the given access are not registered, while other information that is contained in the access control lists may be incomplete, like the recipients of the data being shared. The more recent introduction of User-Managed Access and proposal of Access Control Policy modify and extend the authorization process used by the Solid protocol significantly. While the latter's additional matchers allow for enforcement of new restrictions, such as a time limit on authorizations or the presenting of Verifiable Credentials by the requesting party, fundamentally these techniques remain low-level atomic building blocks and do not succeed in fulfilling all requirements that *data protection* law mandates. In the next chapter we will present a broader architecture with components that take care of these legal requirements at different levels of abstraction.

5

An Architecture for Enforcing Data Protection In Solid

This chapter of the thesis is also the basis of the peer-reviewed paper "A Policy-Oriented Architecture for Enforcing Consent in Solid"[66], which was accepted for publication at the Second International Workshop on Consent Management in Online Services, Networks and Things (CONSENT '22). Our aim here is to propose a reference architecture for relating Solid's low-level technical access control rules with higher-level concepts such as the legal basis and purpose for data processing, the abstract types of information being processed, and the data sharing preferences of the data subject. Our architecture combines recent technical efforts by the Solid community panels with prior proposals made by researchers on the use of the ODRL[50] policy language as an extension to Solid's authorization mechanism.

5.1 Background: Linked Data Integrity

Prior to introducing our architecture we must present the W3C's Data Integrity specification, as it is an important building block for ensuring data authenticity and integrity in the exchanges between the components of our framework. The Data Integrity 1.0 draft community report [67] is a recent proposal by the W3C's Credentials Community Group, with the aim of providing authentication and data integrity capabilities to Linked Data resources through the use of mathematical proofs such as digital signature algorithms. It details a vocabulary for describing proof types, verification methods and algorithms. The origins of this work are to be found in the W3C's recommendation of the Verifiable Credentials Data Model [68], a data model that can be used to assert specific claims on a subject (such as a degree, driver's license, etc.) and which should be accompanied by a cryptographic proof that can assert their authenticity and integrity. These techniques will provide us with the necessary security capabilities, in terms of authentication and accountability, which we need to realize our proposed authorization architecture.

5.2 Overview

As we have highlighted before, the Solid Protocol 0.9[12] relies on the Web Access Control specification[16] as a mechanism for discretionary access control over the resources stored in a Solid Pod. While it offers adequate affordances for simple use cases related to authorization in social contexts, some of its capabilities and design choices may be problematic in more complex data processing applications that are governed by regulations like the GDPR. In contrast, our architecture splits out the implementation of consent as a legal basis for accessing personal data in the Solid Pod into two domains, shown in figure 5.1, where policies stored in the subject's Solid Pod form an interface between these different realms:

1. On the one hand, the *end-user domain* is governed by a so-called *Access Management Application* which is tasked with validating the data processing request coming from the responsible data controller against applicable legal requirements, end-user data sharing preferences and, if the processing request is approved, storing it as a Processing Grant in the data subject's Solid Pod.
2. On the other hand, the *technical domain* uses the *Authorization Agent*, as proposed by the Solid Application Interoperability specification, to handle concrete access requests made by applications and other agents in terms of Shape Trees, Data Shapes and ACL access modes. The interface between the two realms is formed by Processing Grants which are generated by the Access Management App and persisted in the agent's Solid Pod.

For authentication and identification of the different actors in the architecture we depend on the WebID [14] and Solid OIDC 0.1.0 [15] specifications that are defined within the Solid Protocol version 0.9 [12]. In the following paragraphs we will be expanding upon both the Access Management App, Authorization Agent and the proposed concepts of Processing Requests and Processing Grants used to bind these two services.

5.3 End-User Realm

The end-user realm is responsible for enforcing compliance with data protection regulations in the context of consent-based data processing applications. Moreover it captures the additional informational items that are not captured by the lower-level Application Interoperability and access control specifications because they cannot be objectively evaluated in authorization. Such information will however be relevant in regulatory processes for data controllers and data subjects. Central to the end-user realm is the access management application, which will validate the incoming request for data processing by a data controller, present it to the end-user for their explicit consent and persist a Processing Grant once the consent is given. In this section we will further elaborate on the components of this end-user realm and the data model that is used.

5.3.1 Access Management App

The Access Management App enables Data Controllers to obtain the necessary approval for the Data Processing they are requesting for some personal data categories and processing actions in fulfillment of a processing purpose that was allowed for through a specific legal basis. Once it has received a Data Processing Request, the Access Management App will first verify if the request is admissible based on requirements set by the applicable data protection law. Subsequently, it will attempt

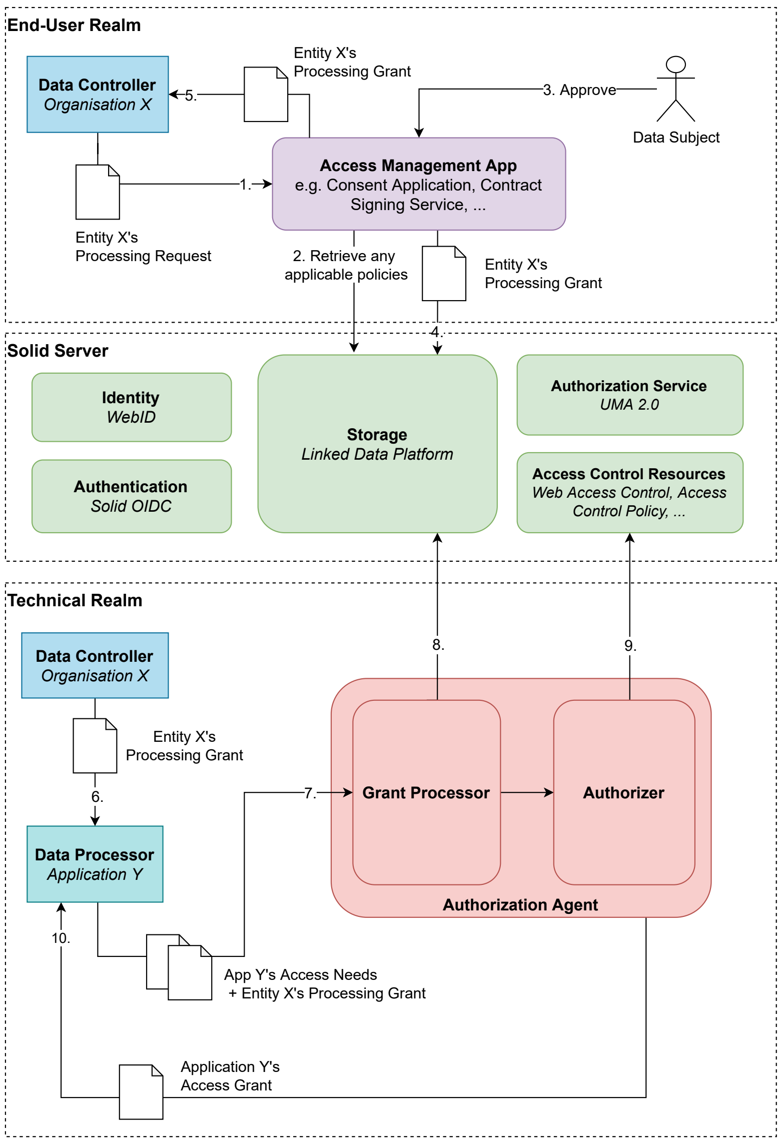


Figure 5.1: Overview of our proposed architecture, linking an End-User realm governing Data Processing permissions with a Technical realm following the Application Interoperability specification

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/>.
@prefix dpv: <http://www.w3.org/ns/dpv#>.
@prefix cert: <http://www.w3.org/ns/auth/cert#>.
@prefix oac: <https://w3id.org/oac/>.

@prefix : <https://example.com/#>.

:medicalRecordsConsent a odrl:Policy, dpv:PersonalDataHandling;
    odrl:profile oac:;
    dpv:hasLegalBasis [
        a dpv:Consent;
    ];
    dpv:hasDataController <https://example.com/id/doctor#me>;
    odrl:permission [
        a odrl:Permission;
        odrl:assignee <https://example.com/id/doctor#me>;
        odrl:target dpv:HealthRecord, dpv:Prescription, dpv:HealthHistory;
        odrl:action dpv:Collect, dpv:Consult, dpv:Analyse, dpv:Alter;
        odrl:constraint [
            odrl:leftOperand oac:Purpose;
            odrl:operator odrl:isA;
            odrl:rightOperand :MedicalConsultation
        ]
    ].
```

Listing 5.1: Example Unsigned Processing Grant Request

to match the request against any explicit data sharing preferences the user might have in their Pod that can lead to an automatic granting of the request. If no preferences turn out to match the request, the data subject must be polled for their explicit consent. Once a Processing Request is granted, it is stored as a Processing Grant in the Solid Pod, accompanied by a Linked Data Integrity proof, and delivered to the inbox [69] of the Data Controller.

5.3.2 Data Model: Processing Requests & Grants

Whenever a Data Controller (Requesting Party) wants to obtain permissions for performing some data processing on the data subject, it will be constructing a Processing Request. This Request is constructed based upon a proposed ODRL profile [50] and concepts from the Data Privacy Vocabulary[39]. An example request for medical records based upon explicit consent is shown in Listing 5.1. The request details handling of the personal data, in terms of legal basis (in the case of this paper we will only consider explicit consent), data controller, and specific permissions that will be needed in the context of the processing.

Each permission specifies what personal data categories it concerns as a target, what actions it needs to perform on this data, and constraints on the purpose or output of the processing. Other constraints could be envisioned as well, like technical measures used in the processing and associated risks, however these haven't been explored in the current proposal.

The *Data Processing Request* itself is presented to the Access Management App accompanied by a Data Integrity Proof that was generated by the Data Controller, this way the provenance and integrity of the request can be validated. Through this signing mechanism, the risk of spam or other malicious attacks with respect to the Access Management App and Processing Request procedures could be reduced, for example by assigning different trust levels to issuer services that can be used by Data Processors to sign their request based upon requirements like identity validation or prior regulatory compliance.

Finally, a *Processing Grant* is constructed from the Processing Request by first completing the legal basis, i.e., consent, with any other necessary attributes that were either gathered in interaction with the data subject or in an automated manner by the access management app. Thereafter any permissions that have not been witheld will be removed from the Grant, the RDF graph is supplemented with a Revocation Status attribute conforming to the W3C Revocation List 2020 specification [70] for revoking Data Integrity Proofs such that the Access Management App can revoke the Processing Grant at a later time, and a Data Integrity Proof will be created and signed by the Access Management App to indicate that the legal requirements for the data processing to be approved are fulfilled. The signed Processing Grant can be seen as an instruction for the authorization agent to provision certain types of information to a Data Controller and its designated processors.

5.4 Technical Realm

The technical realm allows for data controllers to trade in their data processing grants for authorizations to concrete resources, stored in the Solid Pod. Its functionality is enabled by the Solid Application Interoperability draft specification[58], which allows the Authorization Agent that is central to this realm to relate the processing grant to resources stored in the data subject's Pod.

5.4.1 Authorization Agent

The Authorization Agent is largely based upon the proposed Solid Application Interoperability specification in terms of its semantics and API, which is still under discussion by the panel and thus might be subject to changes. This is enabled by the fact that mechanics of the authorization agent are largely left open to implementation, such that additional authorization checks can be executed between the *Access Needs* being presented to the authorization agent and the delivery of a so-called Access Grant that specifies the concrete data that has been elected for sharing with the application.

In fact, the only modification to the authorization agent interface that we are proposing in this research is that a Processing Grant should accompany the Data Processor's access needs when access is being requested. This way the authorization agent can link the access request being made by the application or service, acting as a Data Processor for the Data Controller, to a valid legal basis for data processing. It then becomes the task of a Grant Processor module in the Authorization Agent to match the specified Processing Grant to the Processor's Access Needs in terms of Data Needs (Shape Trees) and Access Modes. The latter confronts us with the need for an unambiguous equivalence relation between the abstract definitions in the Processing Grant and their technical counterparts in the Access Needs.

Finally, once the *Grant Processor* has determined that the Data Processor's request actually matches our initial Processing Grant, it can proceed with an Authorizer that is tasked with modifying the atomic access control rules applicable to the instances of the Shape Trees that were specified in the service's Data Needs. Once this process has ended, an Access Grant is returned to the Data Processor and the necessary registrations are added to the Pod.

5.4.2 Data Model: Auxiliary Rules & Policies

While the ODRL-based processing request and processing grant may suffice for defining the data processing that is being requested and approved on a business-level, it is insufficient for the authorization agent to relate these with the technical access needs specified by a Data Processor like an application. The semantic gap here is twofold, on the one hand we need to unambiguously define what data in the Pod falls under the approved processing and on the other hand we must know what actions on this data are permitted.

Firstly, the abstract data categories used to specify the personal data being shared under the approved data processing activities must be related to concrete technical data type information. As was elaborated upon in the background section, the combination of data shapes and shape trees as a mechanism for defining resource collections and their structure allows us to delimit conceptually related resources in the Pod like medical records, pictures, notes, etc. Through an additional set of rules that is configured by the data subject in their Solid Pod, a so-called Data Category Equivalence policy, we link the technical resource type information provided by Data Shapes and Shape Trees to Personal Data Categories as they are specified under DPV and used in the ODRL profile.

As higher level abstractions are used to define the actions that the processing allows for, we must also relate the Processing categories from the DPV with the Access Modes as they are used in both Solid's Access Control mechanism and the technical Access Needs specified by the Data Processor. These can be defined by the subject as Processing Access Needs, which are stored as an additional set of rules in the Pod.

Furthermore, while not elaborated upon in this paper, the proposed ODRL profile [50] was devised with the concept of data sharing preferences which allowed for the data subject to also express more complex data processing activities that could automatically be permitted to some requesting party based on purpose, data and processing categories. Such policies could also be persisted in the Solid Pod besides these previously noted equivalence relations and the concrete processing grants that flow from them.

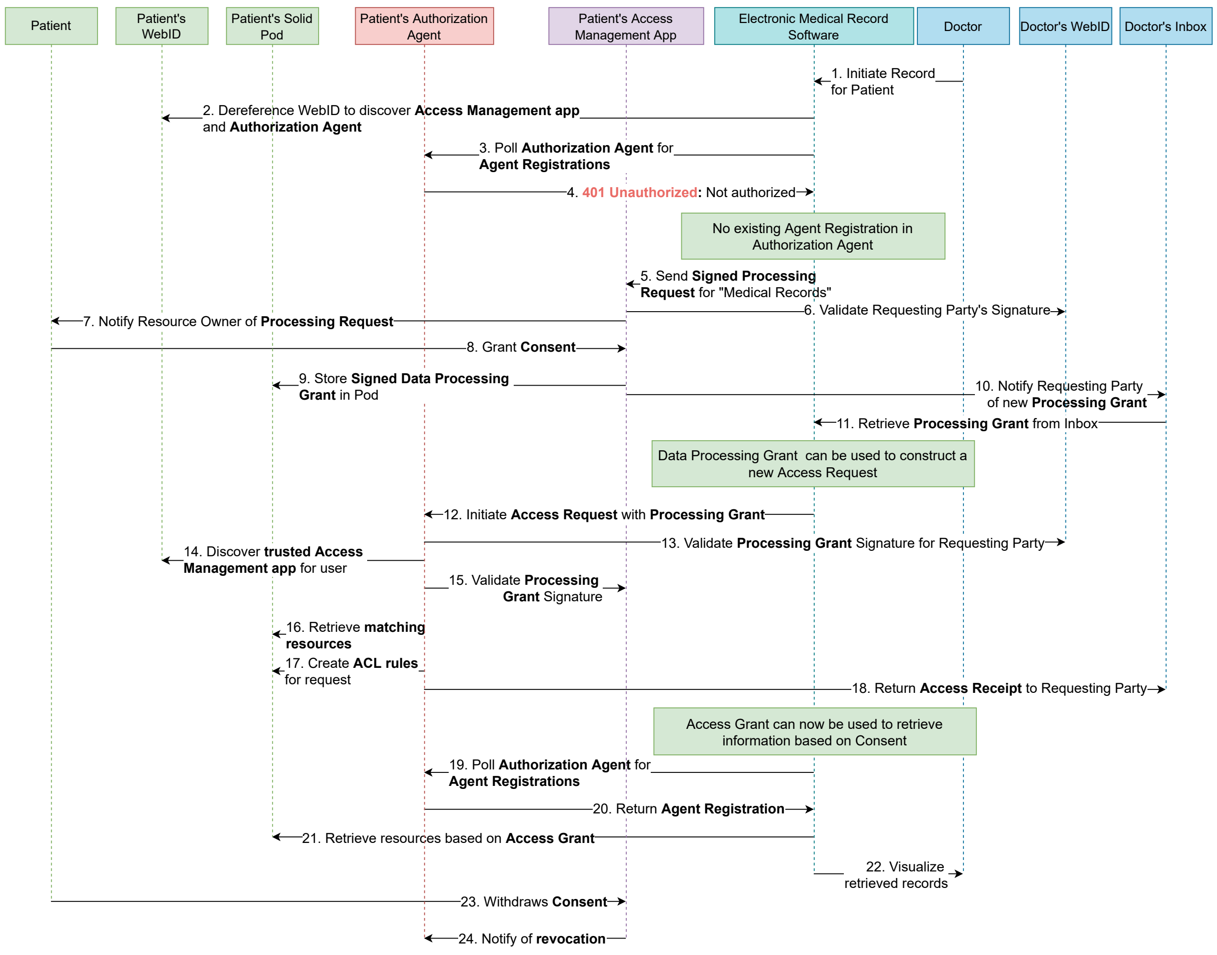

Patient
Patient's WebID
Patient's Solid Pod
Patient's Authorization Agent
Patient's Access Management App
Electronic Medical Record Software
Doctor
Doctor's WebID
Doctor's Inbox
1. Initiate Record for Patient
2. Dereference WebID to discover Access Management app and Authorization Agent
3. Poll Authorization Agent for Agent Registrations
4. 401 Unauthorized: Not authorized
No existing Agent Registration in Authorization Agent
5. Send Signed Processing Request for "Medical Records"
6. Validate Requesting Party's Signature
7. Notify Resource Owner of Processing Request
8. Grant Consent
9. Store Signed Data Processing Grant in Pod
10. Notify Requesting Party of new Processing Grant
11. Retrieve Processing Grant from Inbox
Data Processing Grant can be used to construct a new Access Request
12. Initiate Access Request with Processing Grant
13. Validate Processing Grant Signature for Requesting Party
14. Discover trusted Access Management app for user
15. Validate Processing Grant Signature
16. Retrieve matching resources
17. Create ACL rules for request
18. Return Access Receipt to Requesting Party
Access Grant can now be used to retrieve information based on Consent
19. Poll Authorization Agent for Agent Registrations
20. Return Agent Registration
21. Retrieve resources based on Access Grant
22. Visualize retrieved records
23. Withdraws Consent
24. Notify of revocation

Figure 5.2: Sequence diagram highlighting the exchanges for the motivating use case of a doctor requesting explicit consent for access to medical records of their patient.

5.5 Illustration with an Example Use Case

In this section we will be illustrating our proposed architecture through the motivating use case of a doctor looking to access the medical records of a patient stored in their Solid Pod based on an explicit, informed consent. We assume no previous consent or authorizations were given over the patient's Electronic Medical Records (EMR). Note that for clarity, interactions with the UMA Authorization Service are omitted. Figure 5.2 provides a complete sequence diagram, highlighting the relevant exchanges that are initiated by the physician and their electronic patient record application.

The exchange starts with a discovery phase *(steps 1–2)* where the application aims to determine which Access Management Application and Authorization Agent the patient has elected to use through their WebID. Once it has been determined that no previous registration exists for the EMR application with the Authorization Agent *(steps 3–4)*, a new Processing Request will be initialized and transferred to the patient's access management application *(step 5)*. After validation and explicit consent *(steps 6–8)*, a signed Processing Grant is created by the Access Management Application, stored in the patient's Pod and delivered to the physician's inbox *(step 9–10)*. Subsequently an Access Request for the patient's Authorization Agent can be constructed by the EMR application based on its Access Needs defined in terms of Shape Trees (in this case, shape trees relevant to the patient's medical records), and accompanied by the physician's Processing Grant *(step 12)*. After validation of the Processing Grant by the Authorization Agent *(steps 13–15)*, it is converted into ACL-rules for the instances of the Shape Trees mentioned in the Access Needs *(steps 16–17)*. An Access Receipt[1] is then returned to the physician's inbox *(step 18)*, finally allowing for the EMR application to visualize the patient's medical records *(step 19–22)*. If the patient subsequently chooses to withdraw their consent through the Access Management App *(step 23)*, the app will modify a Revocation List in order to revoke the Processing Grant that was initially provided and notify the Authorization Agent *(step 24)*.

[1]Notification referencing an Access Grant

6

Implementation

In fulfillment of research object 5, this chapter will detail an implementation of the technical realm of the reference architecture presented in chapter 5. The source code of this implementation is freely available on Github[1]. Firstly, the technology stack on which our development efforts were based will be summarized. Thereafter, the software architecture of the technical realm is analyzed and implementation choices are discussed. Finally, we touch upon the integration with the proposed end-user realm and how this could be realized in our implementation.

Additionally, we must clarify why the implementation of the end-user realm was left out-of-scope for the context of this thesis. The motivation for this choice is threefold; First, significant ontology work will be necessary in order to define a policy language that is suitable with respect to legal requirements and provides the necessary affordances for the proposed mapping to definitions of the technical realm. Second, the end-user interactions with the access management application should be considered from a user-experience point-of-view in order to assess whether user expectations are met in terms of data governance and transparency. Last, a legal assessment of the access management application and the specific capabilities it realizes in the context of data protection regulation should be made. Each of these aspects relates to a distinct field of research, and should warrant a much more thorough evaluation than we can provide in this thesis.

By implementing the technical realm of the architecture we will also be able to evaluate its impact on performance for clients requesting information based on access grants that have been defined by the authorization agent. As interactions with other components of the architecture like the access management app should occur much less frequently, typically only when the purpose or requested data categories change, assessing the performance of the technical realm will already give us adequate insights into the characteristics and practicability of our proposal.

6.1 Context & Technology Stack

Our implementation for the technical realm of the reference architecture is enabled in large part by the Components.JS semantic dependency injection framework[71]. Not only because it lies at the heart of Community Solid Server and its modular architecture but also since additional components that were developed in the context of this research have followed sim-

[1]https://github.com/laurensdeb/interoperability

ilar design principles. As a starting point of this work, we based ourselves on an existing MIT-licensed implementation of a User-Managed Access 2.0 Authorization Service[2], using Components.JS dependency injection, which was expanded upon with authorization logic following the Solid Application Interoperability draft specification[58][3]. Furthermore, an integration was developed between the Community Solid Server and its authentication and authorization mechanism and this UMA Authorization Service[4] as well as several auxiliary services and utilities to enable the practical use of the Solid Application Interoperability draft specification.

6.1.1 Components.JS

Dependency injection (DI) is a common pattern in programming, allowing for the loose coupling of objects by defining their relations not through concrete implementations but only via minimal interfaces. Importantly, the technique decouples the specific wiring of various components from the main application logic through configuration. A DI framework is then tasked with retrieving, instantiating and injecting the required dependencies into the objects.

Typical dependency injection frameworks have relied on local semantics, such that they only work within the context of a single application and for a specific DI framework. Components.JS[71] was developed with the goal of providing a globally interoperable and addressable DI framework for JavaScript and TypeScript using semantic web technologies. It offers useful affordances for research, like unambiguous configuration definitions for experiments and modularity across projects through its global semantics.

HandlersJS[5] is a set of MIT-licensed Components.JS modules, developed by Digita[6]. HandlersJS offers generic capabilities like HTTP request handling, logging, storage and error handling that can be used through dependency injection. It avoids developers having to implement these generic functionalities themselves, and thus can be used as a building block in more complex software architectures based on Components.JS DI.

6.1.2 Community Solid Server

The Community Solid Server[7] (CSS) is an open-source implementation of the Solid protocol 0.9 co-developed by Inrupt, inc. and Ghent University for research and development purposes. It has a configurable and extensible architecture, enabling researchers to quickly reconfigure the CSS for their experiments or extend it with modules enabling new functionality. To realize this level of openness and modularity, the CSS relies on the Components.JS framework.

In the context of our work we will be using the latest stable release of the CSS at the time of writing, v4.0.1. This version introduced support for seeding pods through configuration, a useful affordance during testing, and improved error handling

[2]https://github.com/laurensdeb/interoperability/tree/main/packages/uma
[3]https://github.com/laurensdeb/interoperability/tree/main/packages/aa
[4]https://github.com/laurensdeb/interoperability/tree/main/packages/css
[5]https://github.com/digita-ai/handlersjs
[6]https://www.digita.ai/
[7]https://github.com/CommunitySolidServer/CommunitySolidServer

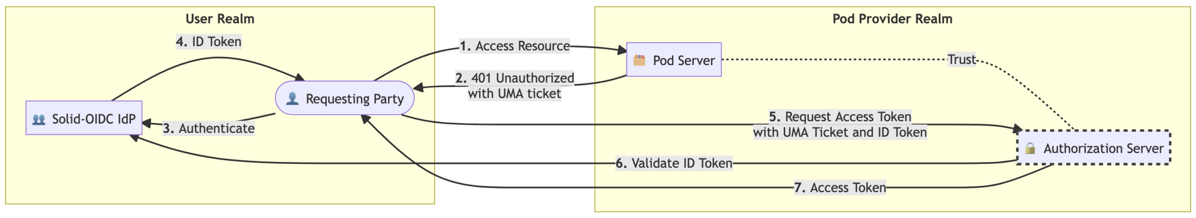


Figure 6.1: Interactions between components with respect to the UMA 2.0 Authorization Service in the Solid-OIDC 0.1.0 specification.

by allowing error instances to contain metadata that can be used in response handling.

6.1.3 User-Managed Access 2.0

The User-Managed Access (UMA) 2.0 grant type for OAuth 2.0 was recently introduced in the Solid authentication protocol, Solid-OIDC 0.1.0[15], with the aim of decoupling the authentication of agents in the Solid ecosystem from the authorization logic. The UMA specification[64] defines the use of an Authorization Service (AS) as a separate component managing authorization to resources provided by a resource server.

Figure 6.1 provides an overview of how the UMA authorization service integrates in the typical request flow using Solid-OIDC authentication. Inherent to the newly introduced UMA AS is the fact that it is trusted by the Pod Server through some out-of-band process, not governed by the specification. Therefore we consider this AS to fall in the realm of control of the Pod provider. Upon executing a request to the Pod server a 401 error indicates to the requesting party the need to perform a UMA token request. The responsibility then falls on the UMA AS to execute authorization logic using authentication details of the requesting party and information concerning the resource to which access is being requested.

We must note at this point that important aspects to the UMA-flow like the communication between AS and Pod Server as well as the scope of the access tokens the AS is delivering are not normatively defined under the Solid OIDC 0.1.0 specification. A consequence of this choice is that requesting parties may be confronted with the overhead of the UMA token request more than once in their interactions with a single Pod server. Moreover, the lack of interface definition between the AS and Pod server in the specification leads to a tight coupling between these components, unless if such an interface is introduced at another level in the software architecture.

6.2 Technical Realm: Authorization Service

The UMA Authorization Service (AS), which is used in our implementation, is based on a prior effort[8] of the author to develop an open-source, modular AS in TypeScript that can support different authentication and authorization mechanisms through Components.JS semantic dependency injection. This implementation was still lacking integration with the Community Solid Server, an obvious prerequisite in order for it to be enabled in an architecture such as the one we described in chapter 5. Additionally, the UMA authorization service had to be extended with authorization logic which can interpret the Solid Application Interoperability draft specification[58].

6.2.1 UMA-support for the Community Solid Server

In order for the Community Solid Server (CSS) to become compliant with the User-Managed Access 2.0 profile of OAuth 2.0, as introduced in the Solid-OIDC 0.1.0 specification, modifications in four main areas were required. Firstly, an interface must be defined for interactions with the UMA Authorization Service (AS), which can be used by other modules and allows for a more flexible integration between the CSS and different implementations of the AS. Second, authentication modules in the CSS should allow for the use of access tokens provided by the AS in requests. Third, the authorization framework must interpret the access modes that are present in the AS's access token and use these in subsequent request handling. Last, the processing of errors in the CSS has to be modified such that the `WWW-Authenticate` which is returned upon an authentication or authorization failure includes information for the client to initiate the UMA token request.

UMA Client

In figure 6.2, the different classes relating to the UMA client interface and its implementation are shown. As was discussed before, the Solid-OIDC specification does not normatively define the interface between the AS and Pod Server. Through the use of a `UmaClient` interface, we attempt to introduce this interface at a software architecture level. The `UmaClient` allows other modules, dependent on interactions with the AS, to:

- Request a UMA permission ticket with the AS that can be returned in the `WWW-Authenticate` header upon a `401 Unauthorized` error. Such a ticket can be used by the client in the UMA claims pushing process to exchange a Solid-OIDC ID token for an Access Token.
- Verify an Access Token with the UMA AS and determine what access modes where granted to which resource based on the token.
- Retrieve the configuration of the UMA Authorization Service.

[8] https://github.com/laurensdeb/interoperability/tree/main/packages/uma

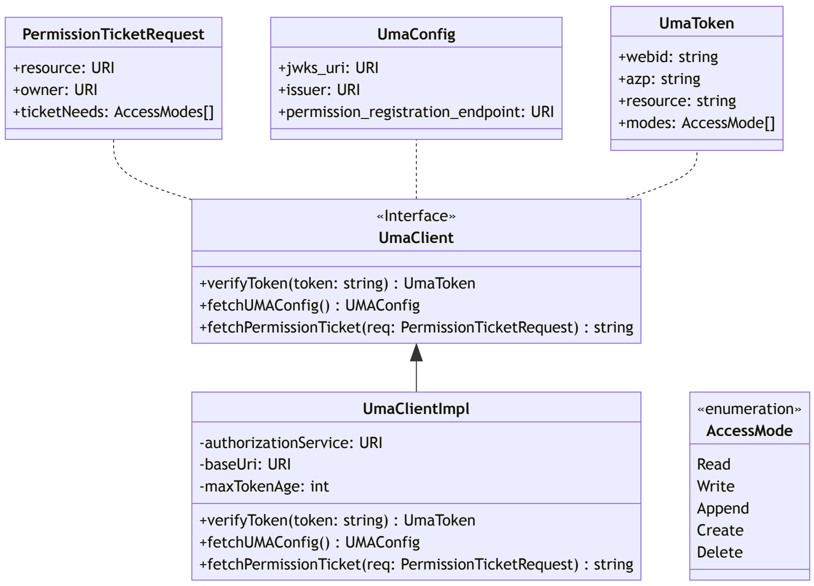

PermissionTicketRequest
+resource: URI
+owner: URI
+ticketNeeds: AccessModes[]
UmaConfig
+jwks_uri: URI
+issuer: URI
+permission_registration_endpoint: URI
UmaToken
+webid: string
+azp: string
+resource: string
+modes: AccessMode[]
«Interface»
UmaClient
+verifyToken(token: string) : UmaToken
+fetchUMAConfig() : UMAConfig
+fetchPermissionTicket(req: PermissionTicketRequest) : string
UmaClientImpl
-authorizationService: URI
-baseUri: URI
-maxTokenAge: int
+verifyToken(token: string) : UmaToken
+fetchUMAConfig() : UMAConfig
+fetchPermissionTicket(req: PermissionTicketRequest) : string
«enumeration»
AccessMode
Read
Write
Append
Create
Delete

Figure 6.2: Classes & interfaces for the UMA client that was developed for the Community Solid Server. The `UmaClient` was introduced to provide an interface to the Authorization Service for other modules.

Importantly, the `UmaClient` interface does not make any assumptions as to the serialization of access tokens. In the case of the UMA AS we will be using, the access tokens are JWTs. However, an opaque token could just as well be validated through token introspection in the `verifyToken` implementation.

Concerning the information exchanged between AS and Pod server, we opted for access modes and resource URIs to be the primary building blocks of our interface. This allows for a clear separation of concerns, where authorization logic in the AS determines which access modes are granted to what client and the Pod server is tasked with determining which modes are required for a specific request. Notably, the current Web Access Control specification does not follow this distinction yet such that it contains both the definition of what access modes are needed for a request as well as the semantics of how these modes are granted.

A concrete implementation of the `UmaClient` interface was made for the UMA AS we will be discussing in the next section. This implementation, `UmaClientImpl`, requires limited prior configuration outside of the base URI of the Pod server, the URI of the authorization service and the maximum age of trusted access tokens. Trust between the AS and the Pod server is established out-of-band, through configuration of the former.

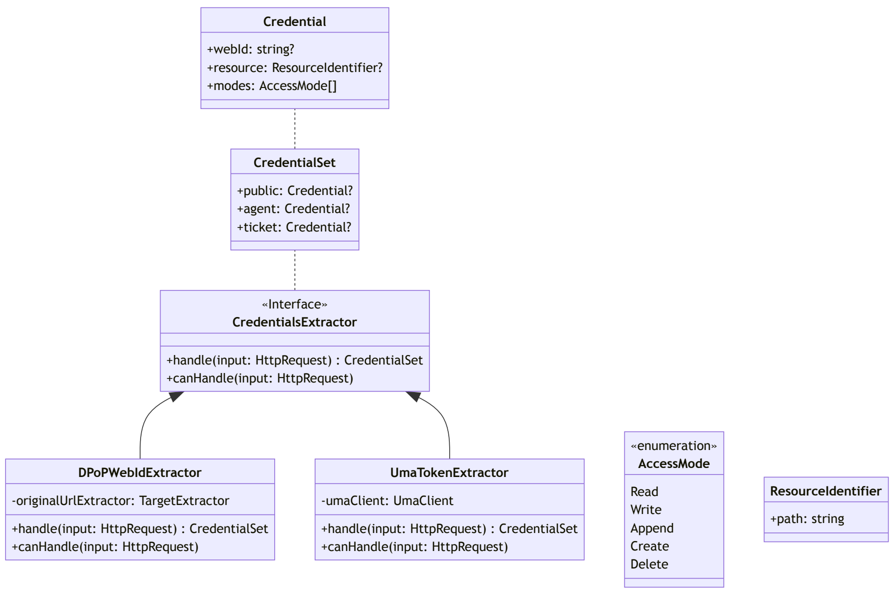

Credential
+webId: string?
+resource: ResourceIdentifier?
+modes: AccessMode[]
CredentialSet
+public: Credential?
+agent: Credential?
+ticket: Credential?
«Interface»
CredentialsExtractor
+handle(input: HttpRequest) : CredentialSet
+canHandle(input: HttpRequest)
DPoPWebIdExtractor
-originalUrlExtractor: TargetExtractor
+handle(input: HttpRequest) : CredentialSet
+canHandle(input: HttpRequest)
UmaTokenExtractor
-umaClient: UmaClient
+handle(input: HttpRequest) : CredentialSet
+canHandle(input: HttpRequest)
«enumeration»
AccessMode
Read
Write
Append
Create
Delete
ResourceIdentifier
+path: string

Figure 6.3: Classes & interfaces pertaining to authentication in the Community Solid Server. The `UmaTokenExtractor` was introduced to authenticate requests made using a UMA Access Token.

Authentication

As the v4.0.1 release of the Community Solid Server does not conform to the 0.1.0 revision of the Solid-OIDC specification[15], the Pod server still performs authentication of clients through Solid-OIDC (DPoP-bound[9]) ID Tokens. The information derived from this client authentication is then used by the Pod server in the subsequent authorization process based on Web Access Control.

In an effort to support the access tokens generated by the Authorization Service as a means of authentication for a request, a new implementation of the `CredentialsExtractor` interface was defined. A `CredentialsExtractor` processes an incoming HTTP request in order to extract the credentials that were presented in this request for authentication purposes. These extracted credentials are returned as a `CredentialSet` that can be used by the authorization logic thereafter. Given the specific nature of the UMA access tokens as both a means of authentication and authorization, the `CredentialSet` type was amended to include the `ticket` type. Additionally, the associated `Credential` type now also includes fields for a resource identifier and access modes that may be present in a token presented to the Pod server. Figure 6.3 gives an overview of the `CredentialsExtractor` interface, and its implementations.

The `UmaTokenExtractor` was implemented to extract, validate and process UMA access tokens that are presented with a request. It relies on a `UmaClient` for its operation, such that the concrete validation logic for the access tokens is abstracted away. Moreover, the `UmaTokenExtractor` may be used in addition to the existing `DPoPWebIdExtractor`, that is able to authenticate requests via Solid-OIDC ID tokens, such that backwards compatibility can be offered to clients of the Pod server.

[9] DPoP: Demonstrating Proof-of-Possession, https://datatracker.ietf.org/doc/html/draft-ietf-oauth-dpop

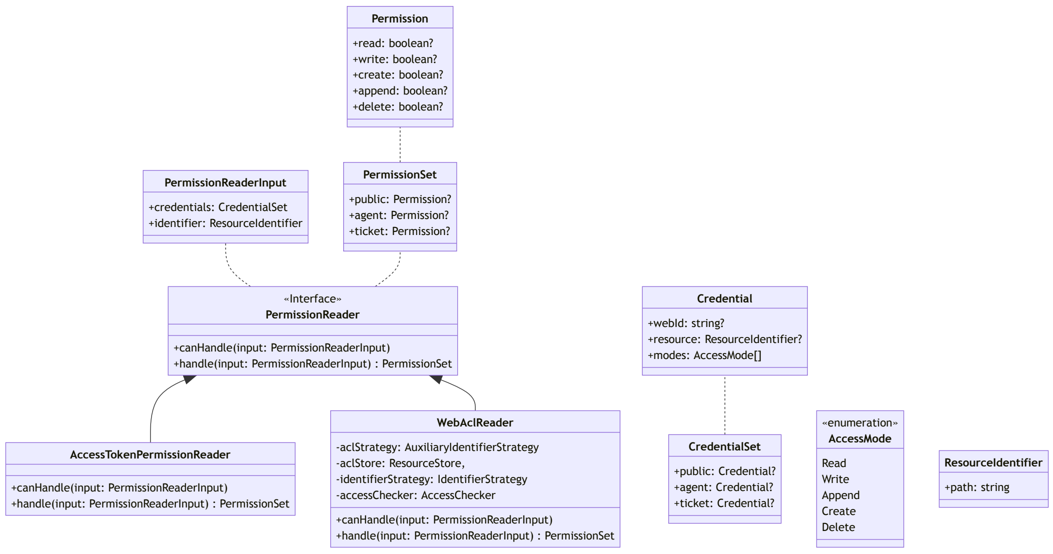

Permission
+read: boolean?
+write: boolean?
+create: boolean?
+append: boolean?
+delete: boolean?
PermissionReaderInput
+credentials: CredentialSet
+identifier: ResourceIdentifier
PermissionSet
+public: Permission?
+agent: Permission?
+ticket: Permission?
«Interface»
PermissionReader
+canHandle(input: PermissionReaderInput)
+handle(input: PermissionReaderInput) : PermissionSet
Credential
+webId: string?
+resource: ResourceIdentifier?
+modes: AccessMode[]
AccessTokenPermissionReader
+canHandle(input: PermissionReaderInput)
+handle(input: PermissionReaderInput) : PermissionSet
WebAclReader
-aclStrategy: AuxiliaryIdentifierStrategy
-aclStore: ResourceStore,
-identifierStrategy: IdentifierStrategy
-accessChecker: AccessChecker
+canHandle(input: PermissionReaderInput)
+handle(input: PermissionReaderInput) : PermissionSet
CredentialSet
+public: Credential?
+agent: Credential?
+ticket: Credential?
«enumeration»
AccessMode
Read
Write
Append
Create
Delete
ResourceIdentifier
+path: string

Figure 6.4: Classes & interfaces pertaining to authorization in the Community Solid Server. The `AccessTokenPermissionReader` was introduced to process access modes originating from the UMA AS.

Authorization

Only limited modifications pertaining to authorization were necessary in order to enable compatibility between the CSS and our UMA Authorization Service. Especially given the fact that UMA offloads the evaluation of authorization logic to the AS, our main implementation of the `PermissionReader` interface, that the CSS relies upon for determining a client's authorization based on their `CredentialSet`, does little more than check whether the access token in fact does pertain to the resource that is being requested. If so, it will apply the access modes in the token to the resulting `PermissionSet`.

Once again we should remark that the `AccessTokenPermissionReader` which realizes this functionality, is compatible with the existing `WebAclReader`. This way local Web Access Control rules can still be applied to clients that are authenticating through Solid-OIDC ID tokens.

Error Handling

Lastly, modifications to error handling in the CSS were necessitated by the UMA specification[64]. In particular, when a resource server is presented with a request that is not properly authenticated, and thus a `401 Unauthorized` error is thrown, it must return to the requesting party a UMA ticket as well as the base URI of the AS in the `WWW-Authenticate` header of the response. This way, the requesting party can initiate a request with the AS for an access token by presenting the UMA ticket alongside authentication and authorization claims in the UMA claims-pushing process.

In order for this `WWW-Authenticate` header to be added to the response, metadata must be available with the

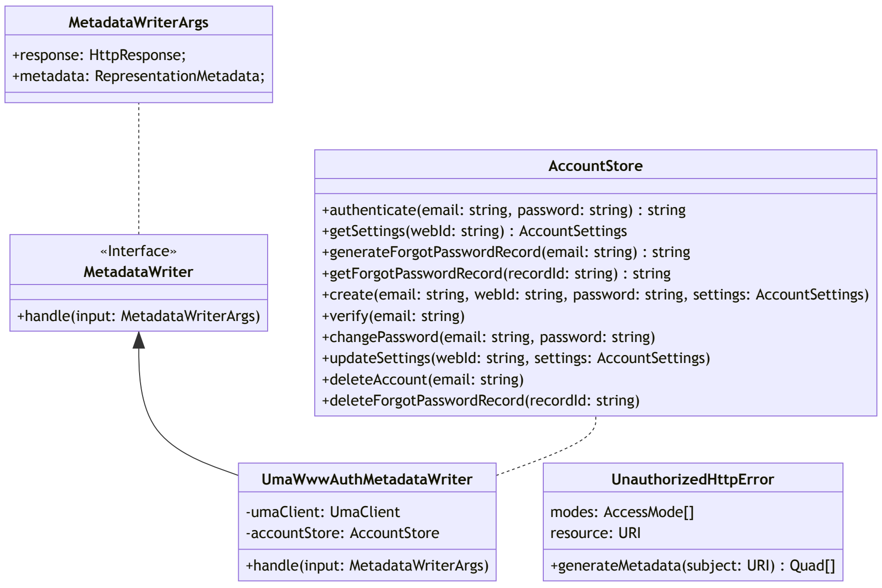

MetadataWriterArgs
+response: HttpResponse;
+metadata: RepresentationMetadata;
«Interface»
MetadataWriter
+handle(input: MetadataWriterArgs)
AccountStore
+authenticate(email: string, password: string) : string
+getSettings(webId: string) : AccountSettings
+generateForgotPasswordRecord(email: string) : string
+getForgotPasswordRecord(recordId: string) : string
+create(email: string, webId: string, password: string, settings: AccountSettings)
+verify(email: string)
+changePassword(email: string, password: string)
+updateSettings(webId: string, settings: AccountSettings)
+deleteAccount(email: string)
+deleteForgotPasswordRecord(recordId: string)
UmaWwwAuthMetadataWriter
-umaClient: UmaClient
-accountStore: AccountStore
+handle(input: MetadataWriterArgs)
UnauthorizedHttpError
modes: AccessMode[]
resource: URI
+generateMetadata(subject: URI) : Quad[]

Figure 6.5: Classes & interfaces modified pertaining to error handling in the Community Solid Server. The `UmaWwwAuthMetadataWriter` was introduced to add a UMA-specification compliant `WWW-Authenticate` header to the response based on metadata in the UnauthorizedHttpError.

`UnauthorizedHttpError` that defines the resource URI and the minimally required access modes for the request. The `UmaClient` then provides the `UmaWwwAuthMetadataWriter` with the ability to request a permission ticket with the AS that offers information regarding the request to be authorized.

Importantly, in order to obtain a permission ticket the permission registration flow of the federated UMA 2.0 specification[72] is followed. This is not normatively part of the Solid-OIDC 0.1.0 specification. Nevertheless, it affords a loose coupling between the AS and Pod server in the process of obtaining the permission ticket.

In order for this permission registration to be performed, identification of the resource owner through their WebID must be presented to the AS as well. This explains the injection of an `AccountStore` instance in the `MetadataWriter`, which permits us to query the registered Pods in the Pod server for their owner. An alternative would have been to rely on the OAuth 2.0 Resource Set Registration specification[10], to register ownership upon resource creation instead of with the permission registration flow. However, this would have entailed much more intricate modifications to the Community Solid Server and statefulness on the side of the UMA Authorization Service.

6.2.2 A UMA Authorization Service using Components.JS

In the context of another project of the author, an MIT-licensed implementation of a UMA Authorization Service was developed in TypeScript using Components.JS dependency injection and based on generic modules for HTTP request and error handling

[10] https://docs.kantarainitiative.org/uma/rec-oauth-resource-reg-v1_0_1.html

from HandlersJS. The source code of this implementation is available via GitHub[11]. An overview of its architecture and most important capabilities will be given in this section.

Overview

At the core of HTTP request handling using the HandlersJS framework is the `HttpHandler` abstract class, which uses the `Observable` pattern from RxJS[12] to enforce asynchronicity following the reactive programming pattern. The API surface of the Authorization Service is formed by four implementations of this abstract class, each of which is responsible for a different capability defined by either the UMA 2.0 Grant specification[64] or the Federated Authorization for UMA 2.0 specification[72]. An overview in the form of a class diagram is given in figure 6.7.

The `TokenRequestHandler` implements the token request endpoint as defined in section 3.2 of RFC6749, which defines the OAuth 2.0 Authorization Framework[73]. The `TokenRequestHandler` parses the request conformant to the generic definition of the token interface, and then relies on `GrantTypeProcessor` instances to handle the specific grant types, like UMA 2.0.

The `PermissionRegistrationHandler` class realizes looser coupling between the Authorization Service and the Resource Service (in our case, the Solid Pod server) by delegating the generation of UMA tickets to the AS. For this purpose section 4.1 of the Federated Authorization for UMA 2.0 specification[72] defines the request of a UMA permission ticket through the permission registration flow. The `PermissionRegistrationHandler` implements the interface of this specification, with the minor addition of a reference to the owner's WebID in the registration request such that the AS can determine whom the owner of the resource is. Such information is not part of the main UMA specification, as there it is assumed the ownership of a resource was established out-of-band.

Lastly, the `UmaConfigRequestHandler` and `JwksRequestHandler` are responsible for respectively discovery of the authorization service and establishing trust with its access tokens. The former provides clients with discovery metadata on the AS, conformant to the OAuth 2.0 server metadata RFC[74]. Trust between the AS and Pod Server is established by using asymmetric cryptography, such that a JSON Web Key Set[75] advertising the public keys of the AS must be available for a client. We will further detail the trust model between authorization service and Pod server in the next subsections.

[11] https://github.com/laurensdeb/interoperability/tree/main/packages/uma

[12] https://rxjs.dev/

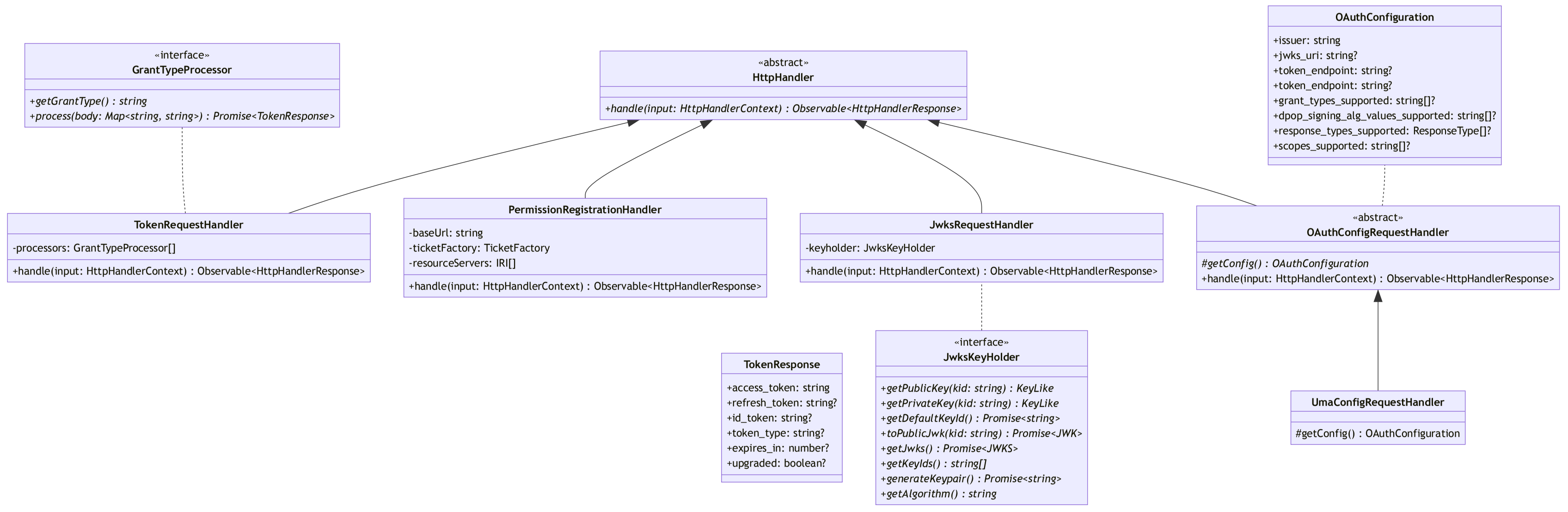

«interface»
GrantTypeProcessor
+getGrantType() : string
+process(body: Map<string, string>) : Promise<TokenResponse>
«abstract»
HttpHandler
+handle(input: HttpHandlerContext) : Observable<HttpHandlerResponse>
OAuthConfiguration
+issuer: string
+jwks_uri: string?
+token_endpoint: string?
+token_endpoint: string?
+grant_types_supported: string[]?
+dpop_signing_alg_values_supported: string[]?
+response_types_supported: ResponseType[]?
+scopes_supported: string[]?
TokenRequestHandler
-processors: GrantTypeProcessor[]
+handle(input: HttpHandlerContext) : Observable<HttpHandlerResponse>
PermissionRegistrationHandler
-baseUrl: string
-ticketFactory: TicketFactory
-resourceServers: IRI[]
+handle(input: HttpHandlerContext) : Observable<HttpHandlerResponse>
JwksRequestHandler
-keyholder: JwksKeyHolder
+handle(input: HttpHandlerContext) : Observable<HttpHandlerResponse>
«abstract»
OAuthConfigRequestHandler
#getConfig() : OAuthConfiguration
+handle(input: HttpHandlerContext) : Observable<HttpHandlerResponse>
TokenResponse
+access_token: string
+refresh_token: string?
+id_token: string?
+token_type: string?
+expires_in: number?
+upgraded: boolean?
«interface»
JwksKeyHolder
+getPublicKey(kid: string) : KeyLike
+getPrivateKey(kid: string) : KeyLike
+getDefaultKeyId() : Promise<string>
+toPublicJwk(kid: string) : Promise<JWK>
+getJwks() : Promise<JWKS>
+getKeyIds() : string[]
+generateKeypair() : Promise<string>
+getAlgorithm() : string
UmaConfigRequestHandler
#getConfig() : OAuthConfiguration

Figure 6.6: Overview of the HTTP Handlers in the UMA Authorization Service, which implement the User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization[64] and the Federated Authorization for UMA 2.0[72] specifications.

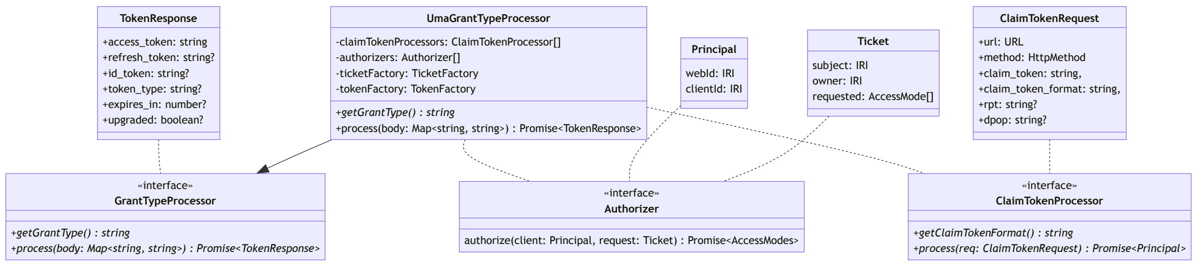


Figure 6.7: Overview of the modules involved in processing the UMA token request.

Token Request & Claims Pushing

Whenever an incoming token request is received by the `TokenRequestHandler`, it will perform a validation of the request in conformance with the broader OAuth 2.0 Authorization Framework's requirements. Thereafter it delegates further processing to a dedicated `GrantTypeProcessor`, specific to the grant type of the token request. In our case the `UmaGrantTypeProcessor` will handle the token request according to the UMA 2.0 Grant[64] specification.

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Auma-ticket
&ticket=eyJhbG...tSw1y-r_yJeQp9Y9l_Q
&claim_token=eyj0...YWxpY2UvcmVnaXN0cmllcy9hZ2Vu
&claim_token_format=http%3A%2F%2Fopenid.net%2Fspecs%2Fopenid-connect-core-1_0.html%23
    IDToken
```

Listing 6.1: Example UMA Token Request

Listing 6.1 shows a UMA token request, which can be identified by its grant type (`urn:ietf:params:oauth:grant-type:uma-ticket`). Additionally, the request includes the UMA permission ticket which identifies the request to be authorized. In the request we present here, the practice of UMA *claims pushing* is being demonstrated. *Claims pushing* entails that the requesting party includes a `claim_token` asserting identity or authorization claims in the token request. This mechanism is used by the Solid-OIDC 0.1.0 specification for the requesting party to present their Solid-OIDC ID token to the authorization service.

The `ClaimTokenProcessors` which are injected into the `UmaGrantTypeProcessor` are responsible for validating the identity and authorization claims presented to the AS during claims pushing. In order to support Solid-OIDC ID tokens as a means of authentication with this endpoint, the `DpopClaimTokenProcessor` class was implemented. Additionally, other `ClaimTokenProcessors` could be envisioned, for example to validate Verifiable Credentials presented by a client (a proposed means of authorization in the ACP specification, as was highlighted in a previous chapter).

Once the identity of the requesting party has been asserted through one of the `ClaimTokenProcessors` and information on the request to be authorized was extracted from the UMA ticket, the `Authorizer` modules execute their authorization logic based on the authenticated `Principal` and the information in the parsed `Ticket`. The `Principal` is identified both through their WebID as well as the Client ID of the application they are using. The `Ticket` includes information on the subject resource of the authorization request, the owner of this resource and the necessary access modes. The result of the authorization logic will be a set of Access Modes that are available for the requesting party in this context.

The resulting access modes from the authorization framework will be serialized as an access token by a `TokenFactory`. This access token identifies the resource, the access modes that have been granted, and the authorized party. As the token is linked to a concrete resource, it is currently not possible to reuse the access tokens implying that a UMA token request will be required for every new resource access in our authorization framework. This choice is motivated by the fact that the Solid-OIDC 0.1.0 specification does not normatively require these access tokens to be DPoP-bound. Consequently, leaking an UMA access token would allow for its use by parties that cannot provide a DPoP-proof and thus undermine Solid's security model. Restricting access tokens to a per-resource basis mitigates this risk somewhat. Additionally, allowing for broader use of a single UMA access token may blur the line between the responsibilities of the AS and Pod server (or require much more information to be captured by the access token), whereas a stricter separation of concerns was the primary motivation for the introduction of UMA.

Trust model between Authorization Service and Pod Server

Both the access tokens and UMA permission tickets that are exchanged between the client and the authorization service are serialized as JSON Web Tokens[76]. This format offers data integrity and authenticity capabilities when verifying the signature in these tokens with a JSON Web Key (JWK)[75].

Alternatively, the token introspection flow of the OAuth 2.0 framework[73] could also have been used to validate access tokens exchanged between the client and Pod server when authenticating a request. In this case, these tokens can even be considered as opaque access tokens by the resource server thereby delegating verification to the authorization service and reducing the impact of e.g. a future format change to the access token. Obviously such an active endpoint for token introspection does introduce an additional HTTP request each time a client accesses a resource using an access token. In contrast, the JSON Web Key Set of an authorization service can even be cached and validation is performed locally on the side of the Pod server, which can provide a minor performance improvement.

6.2.3 Authorizing requests using the Solid Application Interoperability specification

Through the `Authorizer` interface that has been described before, authorization logic based on the Solid Application Interoperability (SAI)[58] specification was implemented. As we highlighted in chapter 3, the SAI specification allows for access control rules to be defined at higher abstraction levels, in terms of Data Shapes and Shape Trees. Our proposed reference architecture relies on this higher abstraction level to relate business and legal concepts from the end-user realm to the low-level resource-oriented model for authorization which is employed by the Solid Pod server.

The `InteropAuthorizer` class bridges the `Authorizer`-interface defined in the UMA authorization service module to the data model and concepts of the SAI specification. This class was designed with modularity in mind, as the SAI specification is still evolving with respect to authorization (in particular concerning the delegation of access) and thus it should be possible to quickly introduce new authorization rules or modify existing ones.

Interoperability Authorization Strategies

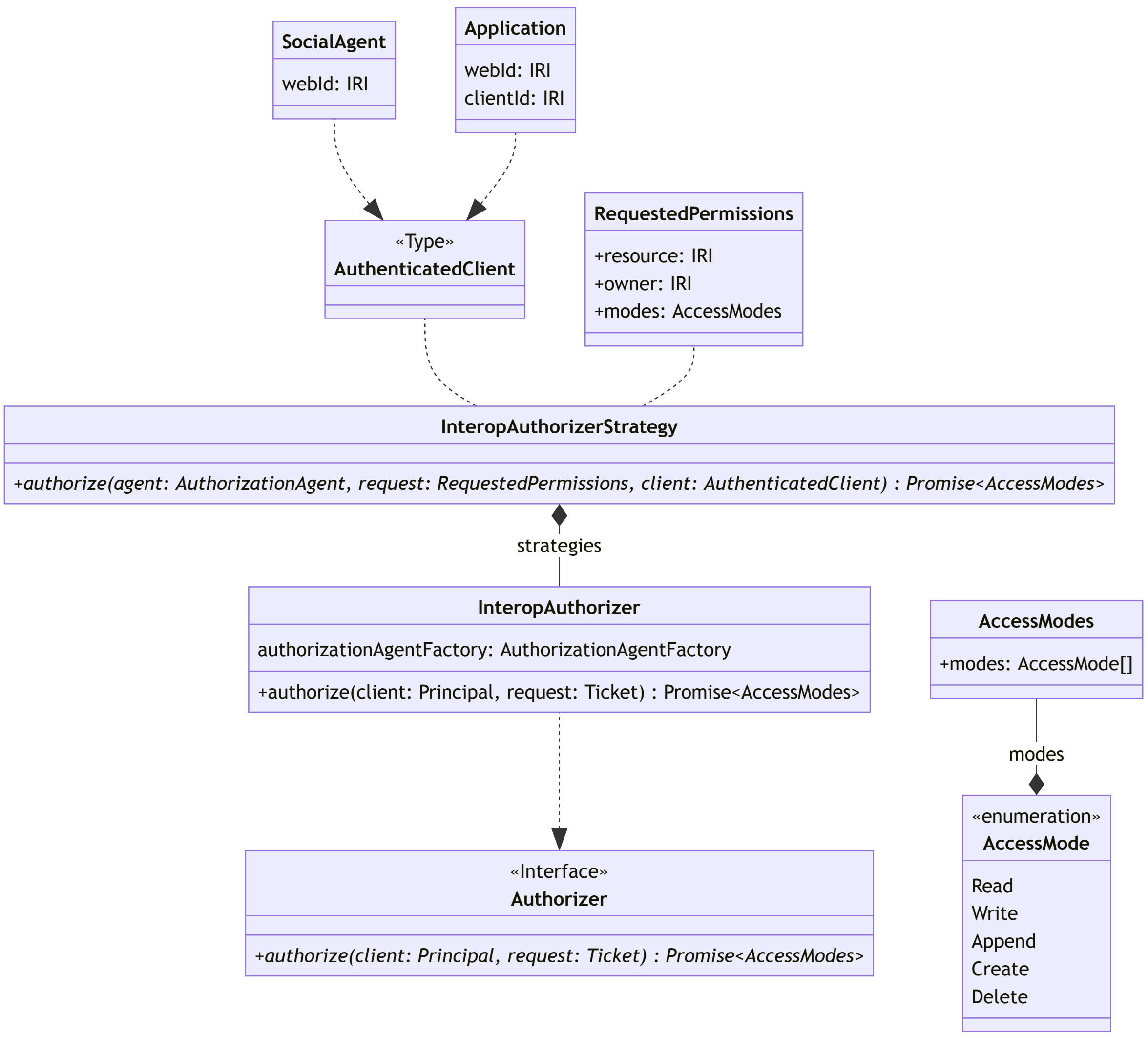
The SAI specification defines an agent's registry set as the source of truth for authorization of applications and other social agents with respect to the data of the owner. The registry set can refer to data stored across various different Pods, as long as an entry in the *data registry* exists. The *agent registry* and *authorization registry* will capture the authorization that was given to some agent in terms of references to concrete data registrations and access modes. Figure 6.9 gives an overview of how this registry set is referenced from the WebID of the agent and how a single agent can have multiple *data registries* pertaining to data in different Pods.

Data registrations relate resources stored in the Solid Pod to Shape Trees[57], which thus enable the enforcement of structural constraints on the data being stored in the Solid Pod. The resources within this registration are also referred to as data instances. Whenever an agent requests access to the resources in the Pod of a user, they will specify their *access needs* in terms of these Shape Trees. Consequently, the requesting party does not need to know in advance where the concrete data are located in the Solid Pod before requesting access (as would have previously been required under Web Access Control, where ACLs are defined in terms of resource IRIs). When the Pod owner decides to grant the access needs of an agent, a new *agent registration* along with *access* and *data grants* will be made available to the agent. Additionally, an *access* and *data authorization* are stored internally in the *authorization registry* as a source of truth for the authorization agent.

The `InteropAuthorizer` we implemented uses these *access* and *data grants* for determining the authorization of an agent. Moreover, the `Authorizer` not only governs access to the concrete data instances but also considers the discovery path that is defined by the SAI specification for an agent to determine which resources it can access, through the agent registration and its related grants.

The specific authorization rules derived from the SAI specification are realized by implementing the `InteropAuthorizer-Strategy` interface. Such a strategy is given request details like the resource, resource owner and necessary access modes as well as the identity of the authenticated client making the request. Under the SAI specification, the authenticated client is either a Social Agent, identified by a WebID, or an Application, identified by a Client ID. Through the `AuthorizationAgent` interface[13] that is provided to the strategy for consulting the registry set, relevant information to the authorization process can be retrieved as defined by the SAI specification. Finally, the *strategy* returns a set of access modes that were deemed applicable for the given agent and resource.

[13] Not to be confused with the actual Authorization Agent which we will define in the next section.



SocialAgent
webId: IRI
Application
webId: IRI
clientId: IRI
«Type»
AuthenticatedClient
RequestedPermissions
+resource: IRI
+owner: IRI
+modes: AccessModes
InteropAuthorizerStrategy
+authorize(agent: AuthorizationAgent, request: RequestedPermissions, client: AuthenticatedClient) : Promise<AccessModes>
strategies
InteropAuthorizer
authorizationAgentFactory: AuthorizationAgentFactory
+authorize(client: Principal, request: Ticket) : Promise<AccessModes>
AccessModes
+modes: AccessMode[]
modes
«enumeration»
AccessMode
Read
Write
Append
Create
Delete
«Interface»
Authorizer
+authorize(client: Principal, request: Ticket) : Promise<AccessModes>

Figure 6.8: Architecture of the Solid Application Interoperability `Authorizer`-module.

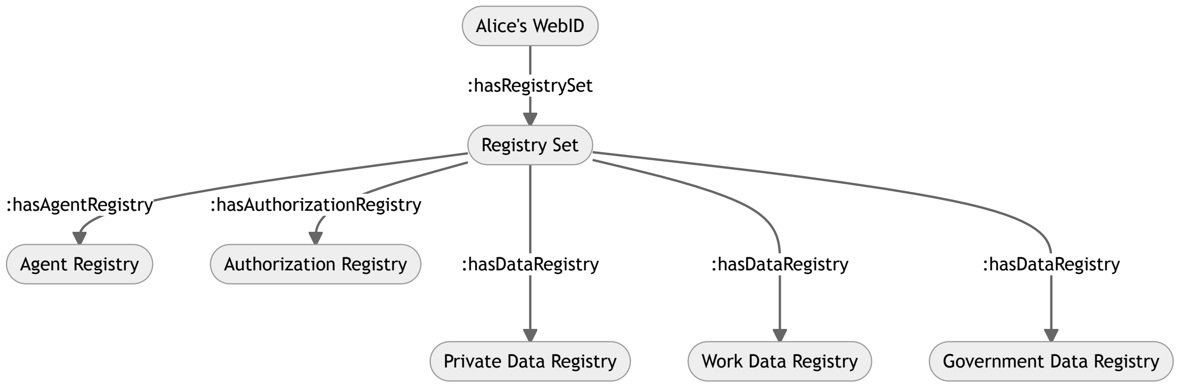


Figure 6.9: Overview of the registries used by the Solid Application Interoperability specification

Optimization

Three main optimizations were introduced in the authorization logic to reduce the time needed for evaluating the different *strategies* when processing a UMA token request. Firstly, because of the loose coupling between the AS and the Pod Server a large number of network requests may be necessitated by the evaluation of a *strategy* that relies on the information in the Pod owner's *registry set*. In order to mitigate this, the strategies themselves are evaluated in an asynchronous manner by taking advantage of the concurrency enabled through JavaScript's `Promise` construct. Second, short circuiting was introduced in the evaluation of authorization logic. Thus, as soon as any of the authorization strategies matches the resource, this will terminate the authorization process. For this optimization we do make the assumption that a resource is only part of a single Shape Tree at any given time, for a given client.[14] Lastly, we use an LRU cache for the `AuthorizationAgent` instances, combined with memoization[15], to speed up repeated requests from the same client. Nevertheless, the underlying *registry set* may change such that this caching mechanism must have a very limited expiration time, as to prevent a client from accessing resources after their authorization was revoked.

6.3 Technical Realm: Authorization Agent

Finally, we will describe our implementation of the Authorization Agent, which facilitates the interactions between agents in the Solid Application Interoperability specification[58] by performing the crucial tasks of data discovery and access negotiation. Under the SAI specification, each WebID references their *authorization agent* through the `interop:hasAuthorizationAgent` predicate. By performing an authenticated request on this authorization agent, an agent can discover their

[14]We must remark that this assumption is correct in the context of the "physical" Shape Trees we rely on in our implementation, which are based on resource containment.

[15]A form of caching, where the resulting function value for the given arguments is stored.

agent registration and subsequently determine what data they are allowed to access. In this section we will briefly highlight how this discovery process was realized in our modules.

Previously, we noted that the SAI specification is still under discussion, such that there currently is no normative interface definition yet for how an agent should "ask" for a registration with the Authorization Agent. Therefore, we have not completed an API endpoint for this functionality in our implementation. Rather we will present the interface definitions that should facilitate such a registration process and link the technical realm, which we've presented here, to the end-user realm through the concept of access control and data usage *policies*.

For the development of this authorization agent, we again opted for a software architecture based on Components.JS dependency injection. Additionally, we chose for the Authorization Agent itself to have a proper identity in the Solid ecosystem, by defining its own WebID that is used in authentication and authorization with the resource server storing the *registry set*. Other implementation of the *authorization agent* interface exist[16], however these implicitly assume the identity of the resource owner by keeping alive their Solid-OIDC ID token. In giving the resource server its own identity, we avoid the complexity associated with this identity management.

In the next sections we will provide an overview of the two core features of the authorization agent, negotiating authorization and enabling data discovery, and how these were realized in our implementation.

6.3.1 Data & Authorization Discovery

Early on in the design of the SAI specification, The Solid Data Interoperability panel adopted the principle that data discovery and authorization are inextricably linked. The *authorization agent* facilitates this process of data discovery by allowing an agent to retrieve their *registration* in the registry set of the resource owner. After performing a `HEAD` request on the *authorization agent* of the resource owner, an agent registration is returned as a `Link` header that can be used to continue the data discovery. This step in the discovery process ensures the resource owner does not disclose the existence of sensitive information in their data registry.

We noted before that the *authorization agent* assumes its own identity for the purpose of accessing the registry set, this is currently realized by giving the *agent*'s WebID read permissions via Web Access Control on the registry set. For convenience, this process was integrated into our Community Solid Server UMA modules, such that the Pod template being used automatically enables these ACL rules. This fact also highlights the convenience in combining traditional ACL-based authorization with newer authorization mechanisms that can be integrated into the UMA AS.

6.3.2 Agent Registration

In the case no agent registration exists yet, we did not have any normative specification as to how an access negotiation step would have to be implemented. Therefore we decided to limit our implementation to providing interfaces that can be used

[16] https://github.com/janeirodigital/sai-impl-service

when a definition of the required interactions makes its way into the SAI specification. Notably, we paid particular attention to how these interfaces would bridge the end-user and technical realms defined by the reference architecture we presented in chapter 5.

To provide a link between the technical realm of our architecture and the end-user realm which we presented before, the policy language used to express the data processing granted by the access management app will have to be understood by the Authorization Agent. We enable this behavior by having the `AgentRegistrationService`, that is responsible for processing an incoming request by an authenticated agent in terms of `AccessNeeds` pertaining to generic Shape Trees, rely on an `AccessPolicy` interface which converts the *access needs* of the agent to an *access authorization*, following the semantics of the Policy language. This *access authorization* can then be stored by the authorization agent and converted into an *access grant* and set of *data grants*.

Figure 6.10 shows these interfaces and how they relate to each other. Thanks to Components.JS dependency injection, an implementer of a usage policy language could integrate their work into the `AgentAuthorizationService` simply by implementing the `AccessPolicy` interface and wiring their instance into the framework via dependency injection.

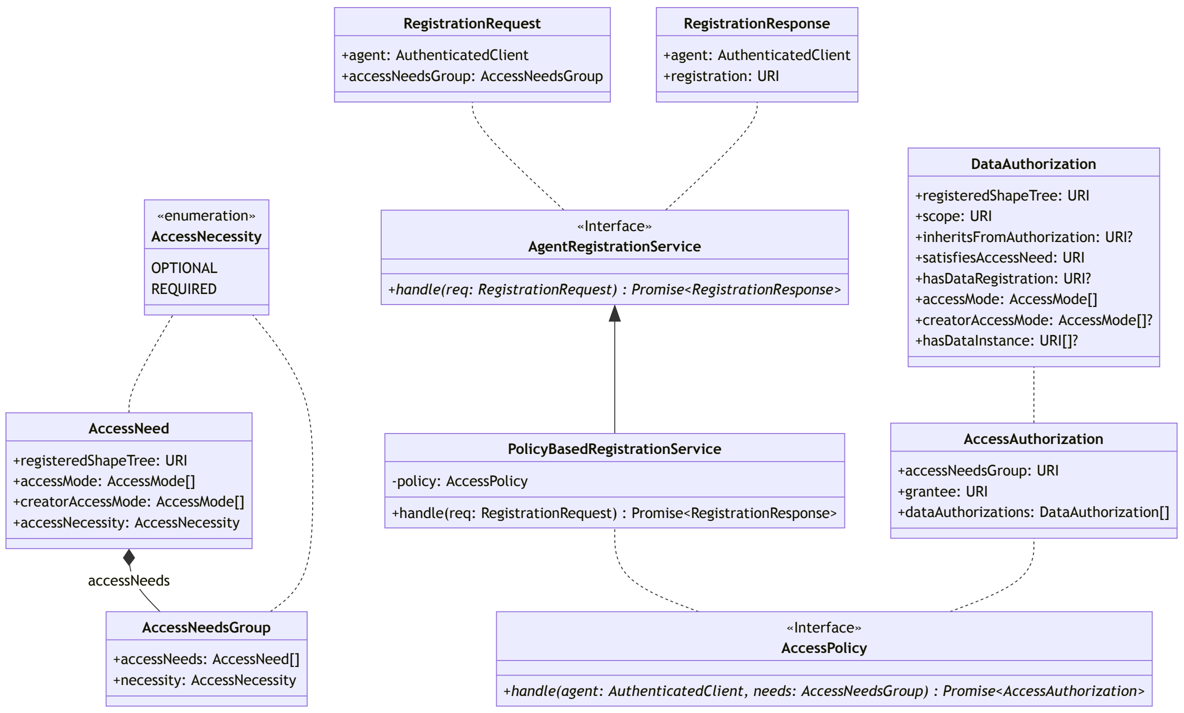

RegistrationRequest
+agent: AuthenticatedClient
+accessNeedsGroup: AccessNeedsGroup
RegistrationResponse
+agent: AuthenticatedClient
+registration: URI
DataAuthorization
+registeredShapeTree: URI
+scope: URI
+inheritsFromAuthorization: URI?
+satisfiesAccessNeed: URI
+hasDataRegistration: URI?
+accessMode: AccessMode[]
+creatorAccessMode: AccessMode[]?
+hasDataInstance: URI[]?
«enumeration»
AccessNecessity
OPTIONAL
REQUIRED
«Interface»
AgentRegistrationService
+handle(req: RegistrationRequest) : Promise<RegistrationResponse>
AccessNeed
+registeredShapeTree: URI
+accessMode: AccessMode[]
+creatorAccessMode: AccessMode[]
+accessNecessity: AccessNecessity
PolicyBasedRegistrationService
-policy: AccessPolicy
+handle(req: RegistrationRequest) : Promise<RegistrationResponse>
AccessAuthorization
+accessNeedsGroup: URI
+grantee: URI
+dataAuthorizations: DataAuthorization[]
accessNeeds
AccessNeedsGroup
+accessNeeds: AccessNeed[]
+necessity: AccessNecessity
«Interface»
AccessPolicy
+handle(agent: AuthenticatedClient, needs: AccessNeedsGroup) : Promise<AccessAuthorization>

Figure 6.10: Interfaces of the Authorization Agent relating to the creation of a new authorization.

7 Discussion

Our central aim at the start of this thesis was to assess how consent-based data processing applications could be implemented in Solid, in conformance with the requirements of the General Data Processing Regulation, and while improving upon existing approaches using the Web Access Control mechanism for authorization. In this chapter we will discuss our results; Firstly by reflecting upon the existing affordances offered by the Solid Protocol 0.9 and the results of prior work. Thereafter, we will discuss our proposed reference architecture in the light of the requirements under the GDPR and the improvements it enables for end-users or *data subjects*, *data controllers* and application developers. Lastly, our implementation of the technical realm of this architecture is discussed and a limited evaluation of its performance characteristics is made.

7.1 Assessment of Web Access Control & Prior Research

In chapter 4 we highlighted technical capabilities and limitations of the Web Access Control specification, as well as the informational requirements of the GDPR that can be fulfilled through its design. As was shown in table 4.1, the Web Access Control specification only partially captures some informational requirements, like categories of personal data concerned by the processing and recipients of this data, while others are not modelled. The fact this information is not captured also implies that the technical measures that can be enforced by the Solid protocol 0.9 are limited, for example *purpose limitation* and *limited data storage* cannot be ensured because no purpose or time restrictions are modeled with the authorization rules.

While it is possible for current applications interacting with *personal data* in Solid Pods to conform to the informational and technical requirements of the GDPR, this will require additional, out-of-band mechanisms on the side of the *data controller* to ensure compliance and capture the relevant information for transparency to regulators and end-users. Moreover, as these custom processes for ensuring compliance fall outside of the realm of the Solid specification their design and implementation may be proprietary to the *controller* or their software vendor. These implementations of the GDPR's requirements could also exhibit similar shortcomings and dark patterns as exist today in more traditional applications and services[8], where significant effort on the part of the *data subject* is needed to obtain transparency into how personal data are processed.

Also, with respect to the legal basis of consent, the process by which this consent is given to a *data controller*, and thus authorization rules are modified to make resources in the Pod accessible, is currently not governed by the Solid specification.

This implies that the user needs to trust the *data controller* to properly modify authorizations in their Pod when a consent is given, as well as revoke the relevant authorizations upon an opt-out. Additionally, the management of this consent will be a vendor- or application-specific affair instead of being integrated into the Solid specification and ecosystem.

The Access Control Policy specification[65] does show promise in expanding the information that can be modeled and rules which can be enforced through the access control mechanism. Its use of Verifiable Credentials could, for example, enable the modeling and enforcement of time, legal basis and purpose constraints for an authorization. Nevertheless, it also exhibits similar shortcomings in terms of interpretability of authorization rules (due to their resource-centric nature and broad access mode definitions), and does not touch upon the authorization process for applications.

Also, when more explicitly considering the role of the Pod provider as a *data controller*[30] in a direct exchange of personal data stored the Solid Pod based on consent, we remark a shared responsibility between the *data controller* of the data processing application and this Pod provider to ensure compliance with the GDPR when exchanging information. As Pod providers are also subject to the GDPR as *data controllers*, they could be subject to hefty fines in case information is exchanged without proper legal basis. If the compliance of data processing applications is not verifiable and enforceable through the Solid specification or a similar open standard, custom and potentially even proprietary mechanisms will have to be devised by Pod providers in order to capture relevant informational requirements and ensure adherence to technical principles. Such an evolution could be detrimental to the open nature of the Solid ecosystem, and its aim of enabling permissionless innovation and end-user governance over personal data.

Some of the problem areas which we have identified for WAC and ACP in this work have also been the noted in prior research regarding access control frameworks for Linked Data[77, 56]. For example, challenges like access negotiation mindful of user privacy, understandability of access control rules, and effectiveness of the administration and enforcement mechanisms have been pointed out[77]. These areas fundamentally impact the security, data governance and user-experience affordances provided by access control frameworks, and must be properly addressed when aiming to deliver real-world applications using these mechanisms to enforce authorization.

Prior work, like the proposed ODRL profile for access control in Solid[50] shows promise, by fulfilling additional informational requirements (through its use of the Data Privacy Vocabulary)[35] and allowing for improved end-user interpretability of the rules governing use of their data. Moreover, the proposed policy language uses the concept of data categories as an abstraction on top of concrete resources to which access is being authorized and proposes a procedure for how access requests can be made by applications. Be that as it may, the authors do highlight some issues with their proposal, for example in the efficiency of evaluating their proposed ODLR profile, the complexity of inheritance mechanism for ACLs which their architecture also retains, the privacy risks attached to discovery of data categories through the public ODRL policies, the legal meaning of such policies and interoperability with Pods that do not support this profile.

7.2 Architecture

One of the major departures from previous proposals is that our proposed reference architecture aims to separate the problem of reconciling technical authorization with the legal requirements for data processing into two distinct domains. On the one hand, there is an end-user realm where the user is presented with requests for data processing in terms of processing actions happening on more abstract data categories, and where an end-user can determine explicit data sharing preferences. On the other hand, there is a technical realm where Solid's proposed Application Interoperability Specification governs how application developers can gain access to resources in an agent's Solid Pod, once a proper legal basis for processing has been established. The concept of Data Processing Grants that are verifiable through the W3C's Data Integrity Specification for Linked Data form the link between these two distinct domains, combined with policies that relate the meaning of Data Categories and Processing Actions to technical concepts that can be used by the Solid Pod.

Whereas previous solutions aimed to integrate business concepts into Solid's existing authorization mechanisms, for example through expanding upon Access Control with purpose and policy concepts, our proposal takes advantage of the modularity enabled by the novel UMA Authorization Service and uses a layered architecture on top of it to introduce business concepts. While this does not solve the core limitations of the existing ACL mechanisms as were mentioned in previous chapters, like inheritance concerns and over-permissioning, it prevents us from importing these same issues into the higher level concepts of Access Grants and Data Processing Grants. Furthermore our architecture aims to reduce assumptions made on the supported features by the Solid server by externalizing the concepts of an Access Management App and an Authorization Agent such that these can be separate services a user chooses to add to their WebID and link to their Solid Pods. Additionally, an explicit dependency on the authorization logic supported by the UMA Authorization Service can be avoided in this framework, as long as a sufficient mapping from the business domain to the authorization mechanism supported by the AS exists for the Authorization Agent to perform. Especially in the light of the recent proposal of the Access Control Policy (ACP) language [65] as an alternative to Web Access Control, which addresses a number of the shortcomings we have highlighted about the latter, this decoupling is a desirable property.

Some of the highlighted challenges in the ODRL proposal [50] have been addressed here, while others will necessitate further consideration. With respect to efficiency, by introducing the Authorization Agent as an intermediary for providing the technical access, we avoid the necessity of matching policies for each HTTP request on a resource in the Pod and convert this into a negotiating step that theoretically should only occur if the access needs or the processing grant of the application have been modified. Moreover, in our proposal, policies do not show the hierarchical inheritance that is common for ACL resources thus compliance checking does not need to take this into account. Still, it might be the case that complex data sharing preferences used by the access management app or elaborate processing grants could lead to an unacceptable time complexity when used in practice. By introducing the access management app as a dynamic negotiator in the flow we avoided the need for public policies to be advertised by the Solid Pod, which addresses important privacy concerns with a policy based solution (i.e., what if anyone can see that you have shared your medical records for the purpose of a past treatment with a psychiatrist). Specific legal issues raised in the ODRL proposal [50] remain largely applicable to our proposal as well, i.e., the legal implications of user choice enabled by Solid's novel approach to data governance, the necessity of awareness of the applicable jurisdiction and its requirements for data processing activities and whether data sharing preferences, as were briefly touched upon, indeed constitute a form of consent.

While we have focused largely on the problem space of implementing explicit consent as a legal basis throughout this proposal, there could be room for enabling other legal bases to be enforced by the access management app as well. For example when processing is requested on the basis of a contractual obligation, the Access Management App could retrieve the contract from the subject's Pod and validate it against the identity of the requesting party for the Data Processing.

7.3 Implementation

In the implementation of the technical realm of the reference architecture we presented, a couple of novel contributions can be identified. Firstly, a UMA Authorization Service as well as modules for integrating this novel component into the authorization logic of the Community Solid Server v4.0.1 were developed. Second, authorization logic based on the Solid Application Interoperability (SAI) draft specification[58] was implemented and wired into the UMA Authorization Service through Components.JS semantic dependency injection. Lastly, we realized an Authorization Agent, conformant to the current SAI draft specification[58], which is capable of bridging the technical and end-user realm of our architecture.

In this section the capabilities of the presented architecture with respect to research into and development of authorization mechanisms will be highlighted. Thereafter, we will provide our results with respect to an evaluation of the performance constraints of the SAI-based authorization logic when integrated into the UMA authorization service. Finally, we will discuss the challenges that still exist and what optimizations we envision could improve performance further.

7.3.1 Affordances for researchers and developers

Through the introduction of UMA 2.0 into the latest editor's draft of the Solid-OIDC specification[15], an explicit distinction is made between the roles of the Solid Pod, as a resource server, and the authorization service, which is tasked with authenticating and authorizing requests. This fundamental *separation of concerns* also opens up new directions for research into different authentication and authorization mechanisms to be used in the broader Solid ecosystem. While previously the `PermissionReader` interface in the Community Solid Server already offered some flexibility as to the implementation of new authorization logic, it was necessarily still constrained by its integration with the authentication mechanisms supported by the Solid server.

In contrast, the `Authorizer` pattern in our UMA Authorization Service allows for developers to implement new access control mechanisms in a modular manner. When complemented with `ClaimTokenProcessor` implementations authentication of the UMA token requests can be done using existing methods like Solid-OIDC, but also through novel mechanisms like Verifiable Credentials.

With respect to research into policy languages, and more specifically the area of data usage policies which we briefly described in chapter 3, our implementation of authorization logic based on the Solid Application Interoperability draft specification[58] may enable practical applications of prior research, like the ODRL profile for Solid[50]. The authors note that in order to relate concrete resources to a *personal data category* as defined under the GDPR, they rely on this category to be self-reported in the associated ACL resource. The use of Shape Trees in authorization, as is the case in our implementation, could resolve this issue by using the structural constraints of the Shape Tree to relate resources to more abstract concepts like *data categories*.

7.3.2 Real-world performance evaluation

In order to evaluate the performance of authorization based on the Solid Application Interoperability draft specification, we set up an experiment where access authorizations for a single agent are simulated over the course of 250 requests. The data to which access is being authorized were synthetically generated based on the SAI specification's data model and a hypothetical Shape Tree for a project management application. Furthermore, the evaluation was done for varying dimensions of the registry set in the Solid Pod. In particular the number of Agent Registrations, Data Grants and Data Registrations were used as parameters for the generation of this synthetic dataset.

After generating six configurations of the synthetic Solid Pods, with either 20 or 100 agent registrations and 10, 50 or 100 data grants, we set out to evaluate the authorization performance. In order to assess performance we measured the complete response time of the UMA Authorization Service's token endpoint when presented with a ticket for a resource and proper authentication of the client. Data grants and data instances were randomly sampled when simulating requests in order to challenge caching behavior of the implementation.

The median response times per resource type and for each of the configurations are shown in figure 7.1, with detailed results summarized in Appendix 2. In this overview we notice there is a measurable impact when increasing the number of data grants, and thus also the number of data registrations, especially with respect to the authorizing access to data instances. This behavior can be explained by the fact that while access grants and data grants are considered immutable resources under the SAI specification, the same does not apply to data registrations. Thus an additional network request is always required when authorizing access to data instances in order to assess that the instance is still contained in a registration for which a data grant exists. In the case of the datasets with 100 data registrations, median response time already exceeds 200 milliseconds and even peaks above 1 second. On the other hand, an increase in the number of agent registrations does not appear to have a significant effect on the median performance numbers we've obtained. Across the different resource types, we notice the most significant increase in median response time with respect to the data instances, again likely due to the fact that authorizations for this resource type necessarily introduce an outgoing request to each data registration until one is found referencing the data instance.

Additionally, we also assessed the impact of a varying number of instances per data registration for a fixed number of data grants and agent registrations. For this assessment, we fixed the number of agent registrations and data grants at respectively 20 and 50 and varied the number of data instances between 5 and 100. In figure 7.2 the median response times are visualized, where we notice that there is no significant impact with an increasing number of data instances. Only in the authorization of the data instance resource type itself do we notice a slight difference in median response times. However,

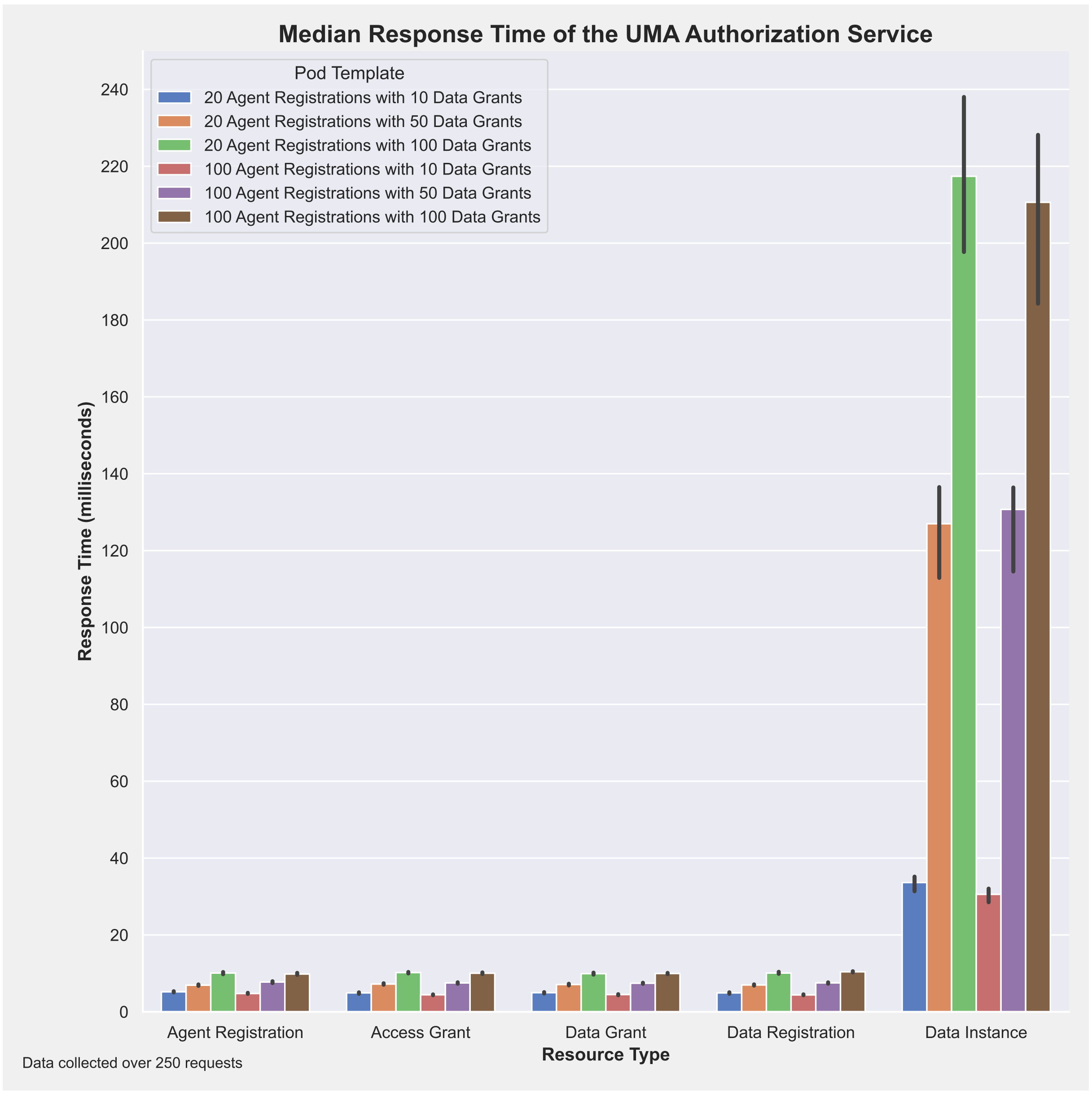


Figure 7.1: Median response times by resource type for a token request to the UMA Authorization Service in different configurations of the Solid Pod. Data collected over 250 requests per configuration, 95% confidence intervals are marked in black. Validated using the 1.0.0 release of the implementation (Node v16.14.0 / Apple M1 Pro / 32GB RAM).

when we compare the 95% confidence intervals we cannot conclude the difference to be statistically significant.

Lastly, the median response times of resource access when using WAC-based authorization were compared to those of UMA-based request authorization in order to assess whether the use of the UMA access token in authorization negatively impacts performance. In figure 7.3 we highlight the total response time of a request authorized with UMA in comparison to a request authorized through Solid OIDC and Web Access Control without intervention of an authorization service. Through the stacked bar plot we also indicate additional latency introduced by the UMA token request. In the UMA Authorization Service a debugging authorization method was used (`AllAuthorizer`), such that any incoming request was instantly authorized and thus a hypothetical, optimal situation was simulated for its response time.

In general, we can conclude that effective resource access when using UMA is somewhat faster due to a reduction in authorization logic that needs to be performed by the Solid server. Nevertheless, a significant overhead is introduced by the ticket retrieval and subsequent token request with the UMA Authorization service, such that the overall median response time is still higher for retrieving the same information.

7.3.3 Potential Optimizations

In our experiments, we witnessed a significant increase in response time due to the introduction of the UMA authorization service as proposed in the Solid-OIDC 0.1.0 draft specification. Moreover, our implementation of authorization logic based on the Solid Application Interoperability specification performed rather poorly in authorizations for the concrete data instances to be used by applications and social agents, in particular for increasing numbers of data registrations. This rather poor scaling behavior leads us to conclude that the current implementation does not allow for bounds on its performance in the light of an increasing amount of data being stored in the Solid Pod.

These performance results have been the topic of discussion in the Solid Authentication and Data Interoperability panels. Based on these discussions as well as the choices and assumptions made in our implementation of the Solid-OIDC 0.1.0 specification and Solid Application Interoperability draft specification, this section proposes three potential optimizations that could be evaluated in future work.

Increased Scope of Access Tokens

A first potential optimization aims to reduce the number of times a client needs to perform the UMA token request flow. By broadening the scope of access tokens from the level of singular resources to collections of resources, the overhead of the token request process could be amortized over a greater number of requests. This could be particularly relevant in the case where clearly defined collections of resources are recognized by the authorization mechanism, as is the case for data instances referenced through a data grant under the SAI specification.

Nevertheless, this approach may be challenging under the Solid-OIDC 0.1.0 specification because it does not define how clients could be made aware of the scope of their UMA 2.0 access token. While the UMA 2.0 grant type[64] does govern the

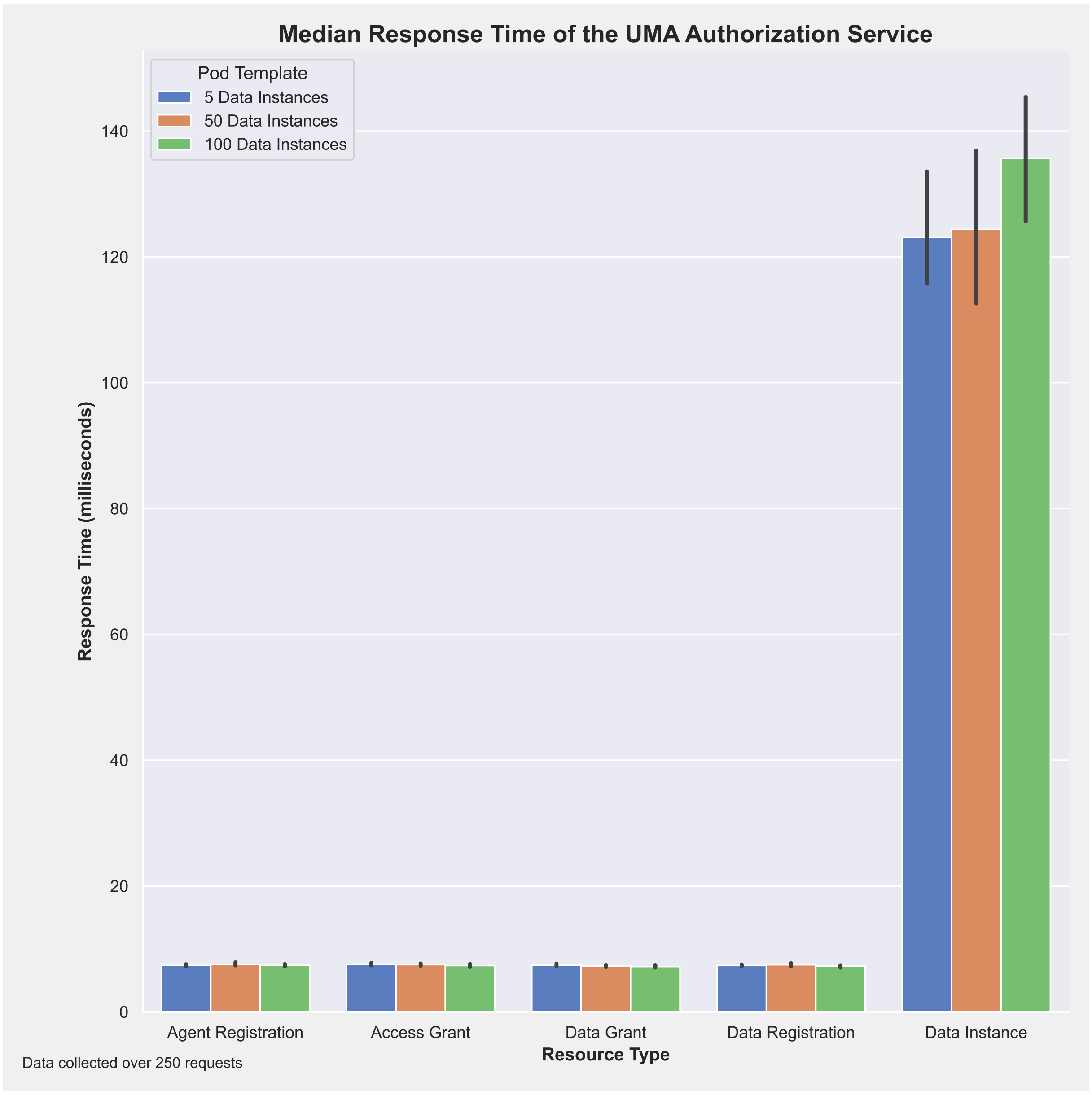


Figure 7.2: Median response times by resource type for a token request to the UMA Authorization Service for varying amounts of Data Instances in the Solid Pod. Data collected over 250 requests per configuration, 95% confidence intervals are marked in black. Validated using the 1.0.0 release of the implementation (Node v16.14.0 / Apple M1 Pro / 32GB RAM).

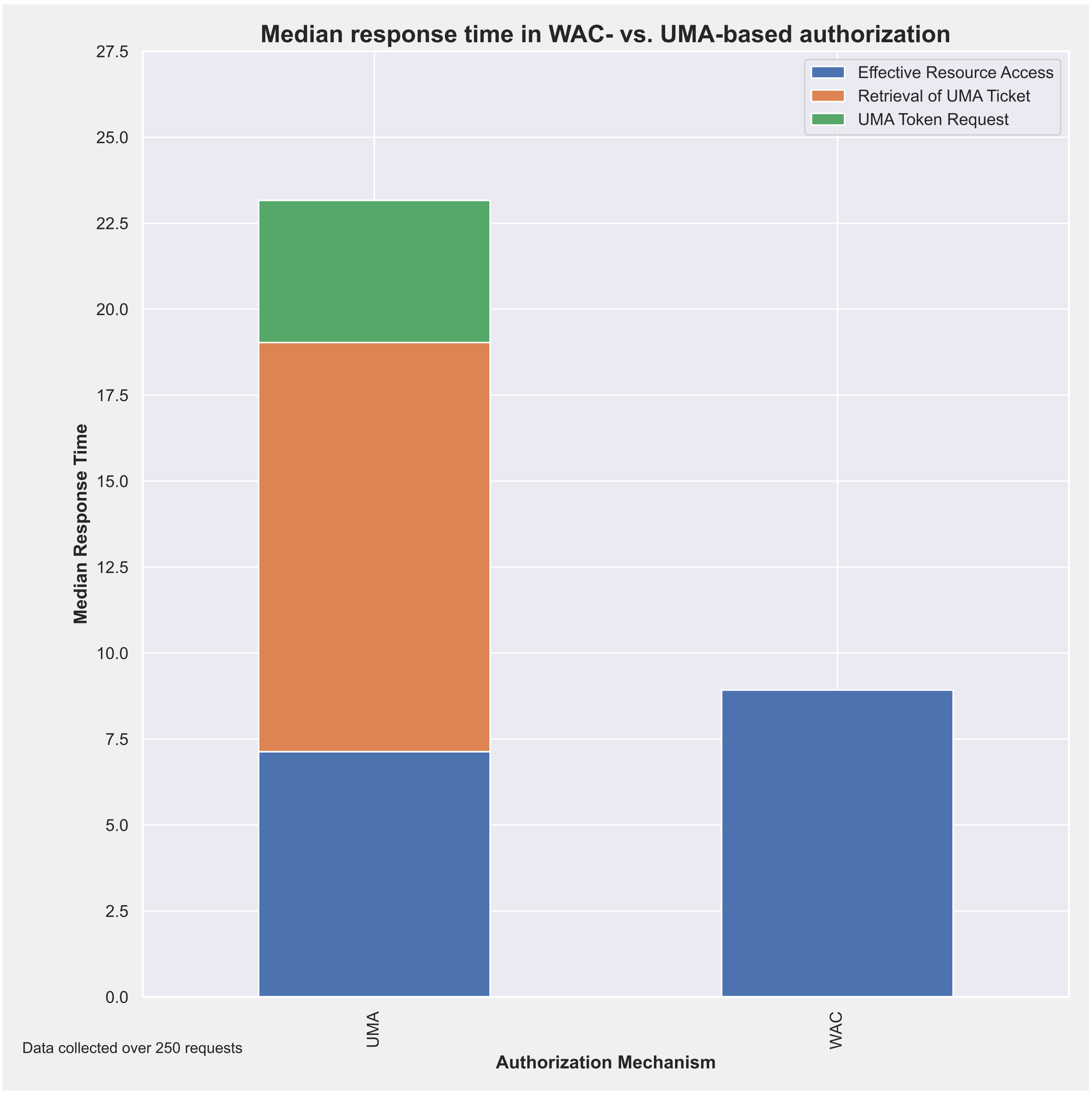


Figure 7.3: Median response times for the different phases of UMA Authorization when compared to WAC-based authorization without UMA AS. Data collected over 250 requests. Validated using the 1.0.0 release of the implementation (Node v16.14.0 / Apple M1 Pro / 32GB RAM).

token request, it does not specifically cover mechanisms like token introspection which could allow for a client to discover the authorization provided by their access token. Therefore, it may not be straightforward to expand the use of these access tokens to collections of resources without modifications to the Solid-OIDC specification.

Additionally, under our implementation the UMA 2.0 authorization service returns access tokens as JSON Web Tokens. These tokens allow for stateless validation of the authorizations of a requesting party. If we choose to expand the scope of access tokens to multiple resources, this would mean that these tokens suddenly have to contain more information and thus become considerably larger to handle. Alternatively, our implementation could be modified to use opaque access tokens, such that token size is not impacted by its scope in terms of resources. However, this would involve additional requests by the Pod server to perform token validation.

Modifications to the Interoperability specification

The SAI draft specification relies heavily on resource containment as the basis for specifying a client's authorization. In order to determine the access modes of a client for a given data instance, we must ensure this instance is part of a data grant for the client. As it currently is not possible for the authorization service to determine which data grants fall in its authorization realm, a traversal must be done for all data grants that exist in the registry set when evaluating SAI-based authorization logic. Inevitably, this causes inefficiency and prevents the authorization service from performing specific optimizations with respect to retrieving its data grants.

Therefore, a proposal has been made to introduce an additional registration in the registry set, dedicated to the UMA authorization service. This registration will reference the data grants that are relevant for the specific authorization service, leading to a more flat resource hierarchy for the authorization service to navigate. In particular with respect to the elevated response times we noticed in the authorization of data instances, this new registration type could enable specific improvements for growing amounts of data grants.

Optimized representations of Authorization Rules

Lastly, the performance of the authorization service could be improved by introducing optimized representations of (cached) authorization rules. Rather than relying on evaluation of the full SAI specification while processing each authorization, as is done in the current implementation, the aim would be to convert the access and data grants to more efficient representations ahead of time. Such a mechanism would of course require a tighter integration between the UMA AS and the Solid Pod than is the case in our implementation. The relative decoupling between these two components currently does provide us with greater flexibility in research and development while incurring this performance penalty. For production applications, however, performance aspects are of far greater importance and thus it would likely make sense to integrate the authorization service and Solid Server more tightly.

8

Conclusion

In this thesis we aimed to answer the question: "Can we rely on data usage policies to implement consent-based data-processing in Solid which conforms to the requirements of the GDPR regulation and is applicable within the constraints of a practical application?". Where we hypothesized that it would be possible to use such semantic data usage policy languages in the context of Solid for a practical, interactive application to achieve compliance with the GDPR regulation.

First, we set out to investigate requirements under the General Data Protection Regulation for data processing activities, as to properly identify the needs which would have to be fulfilled by our solution. In comparing these requirements with the affordances provided by the access control policies that Solid currently relies upon in its discretionary access control mechanism, we concluded that additional means are needed for data controllers to achieve compliance with the stipulations in data protection law. In particular, no details are captured by the Web Access Control specification related to the lawful basis, purpose or relevant time period for the data processing. Moreover, out-of-band mechanisms to enforce compliance, either at the level of the individual application or the individual Pod provider may challenge the open ecosystem that Solid aims to foster.

In order to address the challenges that exist for Solid to meet the standards of data protection law, we identified the field of data usage policies among which ODRL[50] and SPECIAL[56] have previously been discussed in relation to Solid. Recent work[35] shows informational requirements of the GDPR can be properly addressed through a combination of ODRL and the Data Protection Vocabulary. Nevertheless, these proposals have not been able to integrate such data usage policies into Solid in a practical setting.

An important challenge identified in the prior work on data usage policies in Solid, is its resource-centric authorization model and lack of higher level abstractions for resource organization. This complicates the integration of such data usage policies, which typically govern information use at a coarser granularity than individual resources. Solid's recent Application Interoperability draft specification[58] addresses the challenge of allowing different applications in Solid to safely and effectively interoperate over the same data. Consequently, the specification also has to address issues related to resource organization in order for apps and services to discover the data that is relevant to their operation. Data Shapes and Shape Trees have been chosen as solutions for this problem, because they allow developers to define their application's information needs in an unambiguous manner and provide them with the affordance of type safety with respect to resources. The novel component of an authorization agent governs both the data discovery process as well as the authorization procedure for new applications and services, thereby addressing the issues caused by relying on developers to modify relevant authorization rules themselves.

By combining the work of the Data Interoperability Panel with the prior research on data usage policies and their applicability to Solid, we presented a novel reference architecture with the aim of implementing data processing based on the legal ground of consent. In this architecture, an authorization agent governs the technical realm and ensures compliance with data usage policies of the client. These data usage policies are persisted as data processing grants through access management applications operating in a distinct end-user realm. By relying on the authorization agent for enforcement, rather than increasing the complexity of Solid's lower level authorization mechanisms, data usage policies can be evaluated out-of-band from request access using the higher level abstractions for resource organization offered by the Application Interoperability specification. This resolves crucial issues regarding efficiency and practical applicability of such data usage policy languages in Solid.

Nevertheless, in order to truly assess whether our proposed architecture is applicable in a practical setting we also had to evaluate its real-world performance. Due to recent changes in the Solid-OIDC specification[15] this required implementing the novel UMA Authorization Service and integrating it in the Community Solid Server's request flow. The Authorization Service was designed in such a way that it can support the implementation and evaluation of novel authentication and authorization mechanisms through Components.JS dependency injection. These capabilities were used to enable authorization logic based on the Solid Application Interoperability specification, such that its abstractions can in fact be the basis for authorizing client requests. Additionally, an Authorization Agent was developed with the relevant affordances for discovering the data one is authorized to access in a user's Solid Pod as well as for authorizing new access needs based on the evaluation of a data usage policy.

While evaluating our implementation of the technical realm of this architecture, we have to conclude that it currently introduces a request latency that increases significantly with the amount of data to which access is granted for an agent. In particular with respect to data instances, the concrete informational items to which access is governed, this leads to median response times above 200 milliseconds for the case where access to 100 Shape Tree instances is provided to a given client. Whereas Solid's existing Web Access Control mechanism has a complexity that is not directly related to the amount of data which it authorizes access to, we cannot come to the same conclusion with respect to our implementation of authorization based on the Solid Application Interoperability draft specification.

In summary, we presented a framework that integrates ODRL data usage policies in Solid as a layer of abstraction on the Solid Application Interoperability specification. With respect to the informational requirements of the GDPR, prior work[35] allows us to conclude it is possible to adhere to the informational requirements of the GDPR using this approach. Moreover, significant parts of the architecture were implemented in order to partially assess its performance characteristics. Based on the results of this implementation and evaluation, we have to conclude that we cannot answer affirmatively to all subquestions of our initial research question at this point. This is motivated by the fact that poor performance characteristics were observed with growing amounts of data and that we were not able to practically demonstrate the applicability of the ODRL data usage policy in an end-to-end scenario due to time constraints. Nevertheless, our architecture offers important pointers for future research into data usage policies in relation to Solid and our implementation of a modular Authorization Service and Authorization Agent may enable researchers to apply their future work in a practical setting.

8.1 Further Work

While the proposed architecture aims to provide a crucial missing link between the atomic ACL-based authorization mechanism currently used by Solid Pods and the more abstract concepts, and safeguards required by data protection regulations like GDPR, we have identified a number of open challenges that will need to be addressed before such an architecture can become viable in practice.

Firstly, both the Shape Trees and Solid Application Interoperability specifications are still being discussed by the community panel and have only very recently seen their first practical implementations. Outside of the context of this panel, the proposal has not yet gained major traction, which could imply that the specifications may see significant changes before they are finally adopted into the Solid protocol. Another major hurdle that these important building blocks for our proposal face is the complexity of implementing them without breaking compatibility for existing applications and services, while retaining the required semantics for those that do already depend on them. Additionally, the required registries for the Interoperability Specification and the additional metadata necessitated by the use of Shape Trees might prove challenging to consistently maintain within the Solid Pod without imposing additional requirements on its operation.

Secondly, in our limited evaluation of UMA-based authorization, as was introduced in the Solid-OIDC 0.1.0 specification, we noticed a significant performance impact because of the additional requests it requires. This indicates a very real need for assessing the impact of this mechanism more broadly, especially in the context of semantic querying engines that operate over the resources stored in Solid Pods. It is likely that real world applications will necessitate further optimization in how Solid server implementations handle these UMA access tokens.

Additionally, some legal and user-experience challenges related to data processing applications based on Solid were touched upon in this thesis, like the role of the Pod provider as a data controller, the interpretation of purpose limitation in the context of the data re-use enabled by Solid, and how consent could be presented to the end-user in a convenient yet transparent manner. This highlights the need for further investigation into whether and how a Solid-based application or service can fully comply with data processing regulations throughout its lifecycle, not only in terms of the initial authorization we have focused on here but also in terms of legal logging, data minimization, compliance monitoring, etc.

Moreover, compliance checking with respect to the proposed ODRL profile [50] is a problem that warrants further investigation as generic ODRL policy checking algorithms will be necessary for this capability. This way, we should be able to match an incoming processing request with any existing data sharing preferences of the data subject. Managing the ambiguity that is caused by allowing an end-user's policies to define how generic concepts used in the processing requests map to attributes of the Solid Pod is another challenge that still needs thorough consideration. Furthermore, the trust model between the different entities and services in our architecture needs to be considered in more detail as to identify potential security risks.

Finally, the proposed optimizations in chapter 7 should be evaluated in order to assess whether they can reduce the complexity we observed in authorizing access to data instances. Once the Solid Application Interoperability specification finalizes its definition of the authorization agent, this evaluation could also be combined with an assessment of our proposed interface for linking the data usage policies of the end-user realm with the concrete authorization mechanics of the technical realm using an actual data usage policy language, like the proposed ODRL profile[50].

8.2 Ethical and Societal Reflection

As was highlighted in chapter 2, the European Union considers privacy and data protection to be fundamental, human rights of the individual which should be safeguarded by a requirement of due process and appropriate technical, organizational and legal protections. Whereas legislation like the GDPR significantly strengthened enforcement and explicitly considers the potential threats posed by new technological evolutions, dark patterns and bad practices are still prevalent on the Web and data protections authorities have had to prioritize which infringements they challenge because of their increased responsibilities, and thus workload, under the Regulation.

The Solid project, through its vision of realizing *data governance* for the end-user, offers the building blocks for giving users increased control over their personal data. In particular, with respect to whom can access their data and where it is stored. As we've shown in chapter 4 the existing solutions proposed by the Solid Protocol 0.9 only partially succeed in giving users proper controls over their data. However, through the architecture detailed in chapter 5, we have demonstrated the potential of Solid to increase control over and visibility into data processing for the user as well as the advantage for *data controllers* and *processors* where it could simplify compliance efforts and enable legally mandated protections *by design* and *by default*.

In conclusion this work has investigated the great potential as well as the existing shortcomings of the Solid protocol in relation to data protection, while proposing and implementing solutions which improve compliance. This should aid researchers in investigating new methods for users to express their *data sharing* and *data usage* preferences and evaluate whether services are in compliance when processing data. Furthermore, it has already succeeded in informing and evaluating the specification work happening in the Solid community. Lastly, and most importantly, it shows that Solid can be integrated in an architecture that gives control to end-users, offers necessary affordances with respect to *data protection* law and makes these protections practicable *by design* and *by default* for application developers.

References

[1] S. Zuboff, *The Age of Surveillance Capitalism*. London: Profile Books, 2019.

[2] M. P. Sørensen, "Second modernity," pp. 1–2, Nov. 2019. [Online]. Available: https://doi.org/10.1002/9781405165518.wbeos1357

[3] W. Collins, "Trusts and the origins of antitrust legislation," *Fordham Law Review*, vol. 81, pp. 2279–2348, Apr. 2013.

[4] D. Streitfeld, "How Amazon Crushes Unions," *The New York Times*. [Online]. Available: https://www.nytimes.com/2021/03/16/technology/amazon-unions-virginia.html

[5] S. Hall, "Exxon Knew about Climate Change almost 40 years ago," *Scientific American*. [Online]. Available: https://www.scientificamerican.com/article/exxon-knew-about-climate-change-almost-40-years-ago/

[6] T. Berners-Lee, "Three challenges for the web, according to its inventor," *The Web Foundation*. [Online]. Available: https://webfoundation.org/2017/03/web-turns-28-letter/

[7] R. Sevenhant, J. Stragier, L. De Marez, and D. Schuurman, "imec.digimeter 2021," Apr. 2022. [Online]. Available: https://www.imec.be/sites/default/files/2022-04/IMEC_Digimeterrapport_2021.pdf

[8] M. Nouwens, I. Liccardi, M. Veale, D. Karger, and L. Kagal, *Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–13. [Online]. Available: https://doi.org/10.1145/3313831.3376321

[9] "The GDPR: The Emperor's New Clothes - On the Structural Shortcomings of Both the Old and the New Data Protection Law, author=Winfried Veil," *Consumer Law eJournal*, 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3305056

[10] M. Kretschmer, J. Pennekamp, and K. Wehrle, "Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web," *ACM Trans. Web*, vol. 15, no. 4, jul 2021. [Online]. Available: https://doi.org/10.1145/3466722

[11] R. Verborgh, "Re-decentralizing the Web, for good this time," in *Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*, O. Seneviratne and J. Hendler, Eds. ACM, 2022. [Online]. Available: https://ruben.verborgh.org/articles/redecentralizing-the-web/

[12] S. Capadisli, T. Berners-Lee, R. Verborgh, and K. Kjernsmo, "Solid protocol," Tech. Rep., Dec. 2021. [Online]. Available: https://solidproject.org/TR/2021/protocol-20211217

[13] S. Speicher, J. Arwe, and A. Malhotra, "Linked Data Platform 1.0," Tech. Rep., Feb. 2015. [Online]. Available: https://www.w3.org/TR/ldp/

[14] A. Sambra, H. Story, and T. Berners-Lee, "WebID 1.0," Tech. Rep., Mar. 2014. [Online]. Available: https://www.w3.org/2005/Incubator/webid/spec/identity/

[15] A. Coburn, elf Pavlik, and D. Zagidulin, "Solid-OIDC," Tech. Rep., Mar. 2022. [Online]. Available: https://solidproject.org/TR/2022/oidc-20220328

[16] S. Capadisli and T. Berners-Lee, "Web Access Control," Tech. Rep., Jul. 2021. [Online]. Available: https://solidproject.org/TR/wac

[17] R. Berjon, "Principled Privacy," May 2022. [Online]. Available: https://berjon.com/principled-privacy/

[18] C. J. Hoofnagle, B. van der Sloot, and F. Zuiderveen Borgesius, "The European Union General Data Protection Regulation: What it is and what it means," *SSRN Electronic Journal*, 2018.

[19] "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)." *Official Journal of the European Union*, vol. L 119, 2016-04-27.

[20] A. F. Westin and D. J. Solove, *Privacy and freedom*. Ig Publishing, 2015.

[21] O. M. Reynolds, *Administrative Law Review*, vol. 22, no. 1, pp. 101–106, 1969. [Online]. Available: http://www.jstor.org/stable/40708684

[22] H. Nissenbaum, "Privacy as contextual integrity," *Washington Law Review*, vol. 79, no. 1, pp. 119–157, Feb. 2004.

[23] R. Berjon and J. Yasskin, "Privacy Principles," Tech. Rep., May 2022. [Online]. Available: https://www.w3.org/TR/privacy-principles/#bp-summary

[24] "Charter of Fundamental Rights of the European Union, volume = C 364, year = 2000-12-18," *Official Journal of the European Communities*.

[25] H. Hijmans and C. Raab, *Ethical dimensions of the GDPR*. Edward Elgar, 2018.

[26] M. Kuneva. Keynote Speech - Roundtable on Online Data Collection, Targeting and Profiling. Remarks by European Consumer Comissioner Meglena Kuneva at the Roundtable on Online Data Collection, Targeting and Profiling in Brussels on 31 March 2009. [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/SPEECH_09_156

[27] C. Humby, "Data is the new oil," Nov 2006. [Online]. Available: https://ana.blogs.com/maestros/2006/11/data_is_the_new.html

[28] A. Tauriņš, "Big Data Ownership: Do we Need a New Regulatory Framework?" *Baltic Yearbook of International Law Online*, vol. 18, no. 1, pp. 117 – 133, 2020. [Online]. Available: https://brill.com/view/journals/byio/18/1/article-p117_8.xml

[29] U. Milkau, "The GDPR: Halfway between consumer protection and data ownership rights." *Journal of Digital Banking*, vol. 3, no. 1, p. 7–21.

[30] D. De Bot and T. Haegemans. (2021, Jan.) Data Sharing Patterns as a Tool to Tackle Legal Considerations about Data Reuse with Solid: Theory and Applications in Europe. [Online]. Available: https://lirias.kuleuven.be/retrieve/599839

[31] E. Filtz, S. Kirrane, and A. Polleres, "The linked legal data landscape: linking legal data across different countries," *Artificial Intelligence and Law*, vol. 29, no. 4, pp. 485–539, 2021. [Online]. Available: https://doi.org/10.1007/s10506-021-09282-8

[32] H. J. Pandit, K. Fatema, D. O'Sullivan, and D. Lewis, "GDPRtEXT - GDPR as a Linked Data Resource," in *The Semantic Web*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, Eds. Cham: Springer International Publishing, 2018, pp. 481–495.

[33] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley, and A. Paschke, "LegalRuleML: XML-Based Rules and Norms," in *Rule-Based Modeling and Computing on the Semantic Web*, F. Olken, M. Palmirani, and D. Sottara, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 298–312.

[34] G. G. Karácsony, "Managing personal data in a digital environment - did GDPR's concept of informed consent really give us control?" *International Conference on Computer Law, AI, Data Protection & the Biggest Tech Trens*, 2019. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3452573

[35] B. Esteves and V. Rodríguez-Doncel, "Analysis of Ontologies and Policy Languages to Represent Information Flows in GDPR," 2022.

[36] H. J. Pandit, C. Debruyne, D. O'Sullivan, and D. Lewis, "GConsent - A Consent Ontology Based on the GDPR," in *The Semantic Web*, P. Hitzler, M. Fernández, K. Janowicz, A. Zaveri, A. J. Gray, V. Lopez, A. Haller, and K. Hammar, Eds. Cham: Springer International Publishing, 2019, pp. 270–282.

[37] S. Kirrane, J. D. Fernández, W. Dullaert, U. Milosevic, A. Polleres, P. A. Bonatti, R. Wenning, O. Drozd, and P. Raschke, "A Scalable Consent, Transparency and Compliance Architecture," in *The Semantic Web: ESWC 2018 Satellite Events*, A. Gangemi, A. L. Gentile, A. G. Nuzzolese, S. Rudolph, M. Maleshkova, H. Paulheim, J. Z. Pan, and M. Alam, Eds. Cham: Springer International Publishing, 2018, pp. 131–136.

[38] H. J. Pandit and D. Lewis, "Modelling Provenance for GDPR Compliance using Linked Open Data Vocabularies," in *PrivOn@ISWC*, 2017.

[39] H. J. Pandit, A. Polleres, B. Bos, R. Brennan, B. Bruegger, F. J. Ekaputra, J. D. Fernández, R. G. Hamed, E. Kiesling, M. Lizar, and et al., "Creating a Vocabulary for Data Privacy: The First-Year Report of Data Privacy Vocabularies and Controls Community Group (DPVCG)," Oct 2019.

[40] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner, "Using Semantic Web Technologies for Policy Management on the Web," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'06. AAAI Press, 2006, p. 1337–1344.

[41] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *The Semantic Web - ISWC 2003*, D. Fensel, K. Sycara, and J. Mylopoulos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 402–418.

[42] Å. A. Nyre, "Usage Control Enforcement - A Survey," in *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*, A. M. Tjoa, G. Quirchmayr, I. You, and L. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 38–49.

[43] J. Park and R. Sandhu, "The UCONABC Usage Control Model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, p. 128–174, feb 2004. [Online]. Available: https://doi.org/10.1145/984334.984339

[44] X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu, "A logical specification for usage control," in *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 1–10. [Online]. Available: https://doi.org/10.1145/990036.990038

[45] J. Moffett, M. Sloman, and K. Twidle, "Specifying discretionary access control policy for distributed systems," *Computer Communications*, vol. 13, no. 9, pp. 571–580, 1990, network Management. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0140366490900085

[46] R. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994.

[47] B. W. Lampson, "Protection," *SIGOPS Oper. Syst. Rev.*, vol. 8, no. 1, p. 18–24, jan 1974. [Online]. Available: https://doi.org/10.1145/775265.775268

[48] R. Iannella and S. Villata, "ODRL Information Model 2.2," Tech. Rep., Feb. 2018. [Online]. Available: https://www.w3.org/TR/odrl-model/

[49] M. De Vos, S. Kirrane, J. Padget, and K. Satoh, "ODRL Policy Modelling and Compliance Checking," in *Rules and Reasoning*, P. Fodor, M. Montali, D. Calvanese, and D. Roman, Eds. Cham: Springer International Publishing, 2019, pp. 36–51.

[50] B. Esteves, H. J. Pandit, and V. Rodríguez-Doncel, "ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2021, pp. 298–306.

[51] S. Agarwal, S. Steyskal, F. Antunovic, and S. Kirrane, "Legislative Compliance Assessment: Framework, Model and GDPR Instantiation," in *Privacy Technologies and Policy*, M. Medina, A. Mitrakas, K. Rannenberg, E. Schweighofer, and N. Tsouroulas, Eds. Cham: Springer International Publishing, 2018, pp. 131–149.

[52] M. G. Kebede, G. Sileno, and T. Van Engers, "A Critical Reflection on ODRL," in *AI Approaches to the Complexity of Legal Systems XI-XII*, V. Rodríguez-Doncel, M. Palmirani, M. Araszkiewicz, P. Casanovas, U. Pagallo, and G. Sartor, Eds. Cham: Springer International Publishing, 2021, pp. 48–61.

[53] IPTC Rights Expressions Working Group, "IPTC RightsML Standard 2.0," Tech. Rep., Aug. 2018. [Online]. Available: https://iptc.org/std/RightsML/2.0/RightsML_2.0-specification.html

[54] B. A. Bonatti, S. Kirrane, I. Petrova, L. Sauro, and E. Schlehahn, "The SPECIAL Usage Policy Language, V0.1," Tech. Rep., 2018. [Online]. Available: https://specialprivacy.ercim.eu/vocabs

[55] P. A. Bonatti, L. Sauro, and J. Langens, "Representing Consent and Policies for Compliance," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2021, pp. 283–291.

[56] G. Havur, M. Vander Sande, and S. Kirrane, "Greater Control and Transparency in Personal Data Processing," 01 2020, pp. 655–662.

[57] J. Bingham and E. Prud'hommeaux, "Shape Trees Specification," Tech. Rep., Feb. 2022. [Online]. Available: https://shapetrees.org/TR/specification/

[58] J. Bingham, E. Prud'hommeaux, and elf Pavlik, "Solid Application Interoperability," Tech. Rep., May 2022. [Online]. Available: https://solid.github.io/data-interoperability-panel/specification/

[59] E. Prud'hommeau, I. Boneva, J. E. L. Gayo, and G. Kellogg, "Shape Expressions Language 2.1," Tech. Rep., Oct. 2019. [Online]. Available: http://shex.io/shex-semantics/index.html

[60] H. Knublauch and D. Kontokostas, "Shapes Constraint Language (SHACL)," Tech. Rep., Jul. 2017. [Online]. Available: https://www.w3.org/TR/shacl/

[61] S. Villata, L. Costabello, N. Delaforge, and F. Gandon, "Social Semantic Web Access Control?" *Journal on Data Semantics*, vol. 2, 03 2012.

[62] UK Department for Digital, Culture, Media & Sport . (2020, Mar.) Cyber Security Breaches Survey 2020. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020

[63] M. Khonji, Y. Iraqi, and A. Jones, "Phishing Detection: A Literature Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013.

[64] M. Machulak, J. Richer, and E. Maler, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," Tech. Rep., Jul. 2018. [Online]. Available: https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html

[65] M. Bosquet, "Access Control Policy (ACP)," Tech. Rep., Aug. 2021. [Online]. Available: https://solid.github.io/authorization-panel/acp-specification/

[66] L. Debackere, P. Colpaert, R. Taelman, and R. Verborgh, "A policy-oriented architecture for enforcing consent in Solid," in *Proceedings of the 2nd International Workshop on Consent Management in Online Services, Networks and Things*, Apr. 2022. [Online]. Available: https://www2022.thewebconf.org/PaperFiles/88.pdf

[67] M. Sporny and D. Longley, "Data Integrity 1.0," Tech. Rep., Jan. 2022. [Online]. Available: https://w3c-ccg.github.io/data-integrity-spec/

[68] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model v1.1," Tech. Rep., Nov. 2021. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[69] S. Capadisli and A. Guy, "Linked Data Notifications," Tech. Rep., May 2017. [Online]. Available: https://www.w3.org/TR/ldn/

[70] M. Sporny and D. Longley, "Revocation List 2020," Tech. Rep., Apr. 2021. [Online]. Available: https://w3c-ccg.github.io/vc-status-rl-2020/

[71] R. Taelman, J. Van Herwegen, M. Vander Sande, and R. Verborgh, "Components.js: Semantic Dependency Injection," *Semantic Web Journal*, 2022. [Online]. Available: https://linkedsoftwaredependencies.github.io/Article-System-Components/

[72] M. Machulak, J. Richer, and E. Maler, "Federated Authorization for User-Managed Access (UMA) 2.0," Tech. Rep., Jul. 2018. [Online]. Available: https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html

[73] D. Hardt, “User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization,” Tech. Rep., Oct. 2012. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749

[74] M. Jones, N. Sakimura, and J. Bradley, “OAuth 2.0 Authorization Server Metadata,” Tech. Rep., Jun. 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8414

[75] M. Jones, “JSON Web Key (JWK),” Tech. Rep., May 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7517

[76] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” Tech. Rep., May 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

[77] B. Cuenca Grau, S. Kirrane, A. Mileo, and S. Decker, “Access Control and the Resource Description Framework: A Survey,” *Semant. Web*, vol. 8, no. 2, p. 311–352, jan 2017. [Online]. Available: https://doi.org/10.3233/SW-160236

Appendix

Appendix 1: Workshop Paper

This thesis resulted in the publication of the peer-reviewed workshop paper "A Policy-Oriented Architecture for Enforcing Consent in Solid" which was accepted to the 2nd International Workshop on Consent Management in Online Services, Networks and Things (COnSeNT 2022), co-located with the 31st WEB Conference.

The paper discusses an initial version of the policy-based consent framework for Solid this thesis details. Furthermore, it elaborates on the shortcomings of existing Access Control mechanisms and how the Solid Data Interoperability panel has proposed solutions for these issues through its Application Interoperability draft specification.

A Policy-Oriented Architecture for Enforcing Consent in Solid

Laurens Debackere
Laurens.Debackere@UGent.be
IDLab, Department of Electronics and Information Systems, Ghent University – imec
Ghent, Belgium

Pieter Colpaert
Pieter.Colpaert@UGent.be
IDLab, Department of Electronics and Information Systems, Ghent University – imec
Ghent, Belgium

Ruben Taelman
Ruben.Taelman@UGent.be
IDLab, Department of Electronics and Information Systems, Ghent University – imec
Ghent, Belgium

Ruben Verborgh
Ruben.Verborgh@UGent.be
IDLab, Department of Electronics and Information Systems, Ghent University – imec
Ghent, Belgium

ABSTRACT

The Solid project aims to restore end-users' control over their data by decoupling services and applications from data storage. To realize data governance by the user, the Solid Protocol 0.9 relies on Web Access Control, which has limited expressivity and interpretability. In contrast, recent privacy and data protection regulations impose strict requirements on personal data processing applications and the scope of their operation. The Web Access Control mechanism lacks the granularity and contextual awareness needed to enforce these regulatory requirements. Therefore, we suggest a possible architecture for relating Solid's low-level technical access control rules with higher-level concepts such as the legal basis and purpose for data processing, the abstract types of information being processed, and the data sharing preferences of the data subject. Our architecture combines recent technical efforts by the Solid community panels with prior proposals made by researchers on the use of ODRL and SPECIAL policies as an extension to Solid's authorization mechanism. While our approach appears to avoid a number of pitfalls identified in previous research, further work is needed before it can be implemented and used in a practical setting.

CCS CONCEPTS

• **Information systems** → **World Wide Web**.

KEYWORDS

Solid, Consent, Semantic Web, ODRL, Access Control

ACM Reference Format:
Laurens Debackere, Pieter Colpaert, Ruben Taelman, and Ruben Verborgh. 2022. A Policy-Oriented Architecture for Enforcing Consent in Solid. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3487553.3524630

1 INTRODUCTION

The Solid project[1] aims to realize Tim Berners-Lee's vision on decoupling personal data storage from the apps and services that use it, in order to return control and data governance to the user. Ultimately, Solid aims to re-establish a proper balance of power between service providers and their users [28], by providing the latter with the tools to make their own choices in data sharing and storage rather than having their data exist out of sight and out of control. To that end, the Solid community is developing a draft specification for decentralized personal data storage servers, also referred to as *Pods*.

At its core, the Solid Protocol version 0.9 [8] has three crucial building blocks that make up most of its footprint:

(1) Solid implements parts of the **Linked Data Platform** W3C recommendation [23] to allow for read/write-access to the resources stored in a Pod with specific affordances for handling Linked Data.
(2) Solid proposes **WebIDs** [22] and **Solid OIDC** [10] for *identification* and *authentication* purposes respectively. Through these standards, agents can be linked to a decentralized identifier expressing information on them like the agent's trusted identity providers. This allows for authentication between resource and authorization servers that have no prior trust relation.
(3) **Web Access Control** [7] provides the critical controls over sharing of information stored in the Pod. Web Access Control is a cross-domain, decentralized solution for *authorizing* requests using *Access Control Lists* (ACLs) expressed as Linked Data. It identifies both *agents* and *resources* through the use of IRIs. Notably, ACLs can both be defined specifically for a given resource, or be inherited from a parent container.

The EU's General Data Protection Regulation[2] [1] set a major legislative milestone in the realm of data protection and privacy regulation when it entered into force in 2018. It afforded *data subjects* with both transparency and greater control regarding the processing of their personal data by *data controllers* and took new and emerging technologies such as Big Data, AI, and the internet explicitly into account when it was first drafted. While far from perfect [27], it bestows a much greater deal of autonomy upon the

[1]https://solidproject.org
[2]http://data.europa.eu/eli/reg/2016/679/oj

data subject when making decisions regarding the processing of their personal data than has previously been the case.

One of the major shortcomings of the GDPR regulation boils down to the legal basis of consent and how it is typically realized on the Web [16, 18, 27]. In Article 6 of the GDPR, the six possible grounds for lawful data processing are laid out by the legislator, one of these being a *freely given consent* that can be withdrawn by the data subject at any time. The informedness of the data subject when giving their consent is emphasized greatly in the GDPR, meaning that a data subject should be able to accurately assess the consequences of the data processing to which they are consenting. In practice, the way consent is used by many services neither constitutes *consent* nor can it be considered *informed.* Most often the information required by Articles 13 and 14 of the GDPR is hidden away in lengthy privacy policies, which the data subject would have to read in their entirety to fully grasp the impact of their consent.

The prevalence of dark patterns on the internet [19], that are used to obtain the consent of a data subject, highlights a clear issue with respect to how this legal basis is being employed in practice. Today, the act of giving consent in an online setting is mostly a unilateral activity, where the data controller sets out the conditions and the data subject has little impact on what data is being processed and how it is being handled. Solid's model for returning a user's control over their personal data might tip the scales in favor of the data subject when negotiating with a data controller in the context of consent. While in typical online service relationships, a data subject has little negotiating power and consent becomes a take-it-or-leave-it offer more often than not, Solid allows the *data subject* to have a clear overview of what data their Pod contains and granularly control with whom they share it. Therefore, it could bring crucial bilateral protections that consent depends upon in order to be used as intended by the legislator in data processing applications.

1.1 Motivation

While Solid has the potential to become a major driver for realizing the vision of true explicit consent as a legal basis for data processing as it was envisioned by legislators, several technical shortcomings still exist. As described earlier, Solid's current access control mechanisms, while suited for simple use cases, lack interpretability for average users and only capture very limited information on the identity of the parties involved, the data being exchanged, and the purpose and legal basis of this exchange. Furthermore, only limited analysis of how data sharing patterns and required legal safeguards can be implemented in Solid has happened so far [11].

The core idea of using *policies* in modeling and enforcing security and data privacy requirements for the Semantic Web has been the subject of prior work [5, 15]. Some extensions to the access control mechanisms in Solid based on the use of policy languages have already been proposed, such as the use of ODRL policies [12, 14] or the SPECIAL policy language[3] [13]. While these address some concerns with regard to interpretability and flexibility raised above, they also inherit or worsen some of the flaws of Solid's ACL mechanism. Issues include poor interpretability due to rule inheritance, increased runtime complexity of the authorizing process, and limited abstractions for identifying resources. Furthermore, the process by which a data controller requests the explicit consent and how this consent is then materialized in the new ACL policies need to be considered in order to address the current shortcomings of Solid's ACL-based authorizations in a data processing context.

There is a distinction between the technical and end-user perspectives when using explicit consent as the basis for accessing resources in the data subject's Solid Pod. Whereas end-users need to understand *what data* they are sharing, *with whom*, for *what purpose* and *in which ways* this data is to be processed, a developer should not have to consider how their interactions map to these user-interpretable concepts. Rather we want developers to interact with the existing technical concepts from the Solid specification while having the Solid Pod or an intermediary validate whether these interactions are covered by a prior consent (or perhaps even some other legal basis). Therefore, we will define an architecture allowing for the decoupling of the legal and end-user interactions regarding consent from the technical interactions that were authorized by it.

Our contributions through this paper can be summarized as follows:

- identifying the shortcomings of Solid's existing Access Control mechanism and how it is typically employed by developers when implementing a legal concept such as consent;
- presenting a framework reconciling end-user and legal requirements for data processing with Solid's existing access control model.

Section 2 provides background information on the state of the art regarding Solid's authorization mechanism and briefly introduces concepts and standards that will be used throughout the rest of this paper. Section 3 details our proposed architecture, its interaction patterns and primary data structures. Section 4 applies our architecture to a motivating use case highlighting how explicit consent can be effectively implemented. Section 5 summarizes the reasoning behind this architecture, how it compares to previous proposals, and provides a brief interpretation of our findings. Section 6 discusses further work needed for the proposal to become viable in practice.

2 BACKGROUND

2.1 Authorization in Solid

Solid's primary mechanism for authorizations is the Web Access Control (WAC) specification [7]. It employs the ACL ontology[4] to express *access modes* applicable to some resource for an agent, where both the agent and the resource are identified using IRIs. WAC supports four access modes in its rules, namely:

Read Allowing for full or partial read operations on resources.

Write Allowing for write operations on resources, i.e., create, update, or delete.

Append Allowing for append operations on resources, i.e., to add information to the resource but not remove any.

Control Allowing for read and write operations on the resource's *associated ACL resource.* This permits the grantee to delegate or revoke access to the resource.

[3]https://ai.wu.ac.at/policies/policylanguage/

[4]http://www.w3.org/ns/auth/acl

Notably, these access modes are broad and do not map well to the more common CRUD[5] permission model [29]. Also, some of these access modes will align poorly with user expectations: e.g., what does it mean to have Append permissions over a container of resources?

Furthermore, WAC uses an inheritance mechanism to determine which ACL resource is the effective ACL governing some resource or container resource in the Pod. While this inheritance mechanism might be reasonably easy to understand for developers, to an unaware end-user, this behavior can be counter-intuitive or even lead to unintended information disclosure. For example, when a user grants an app access to a container, they implicitly grant access to all data transitively contained within, including any new resources that are added after the user granted access.

The use of IRIs to both identify resources and agents might also contribute to poor user experience and lead to security breaches. For example, users might perceive an analogy between how they would typically manage a photo collection in a filesystem on a computer, and how pictures are stored in a folder in one's Pod. That way, an end-user could have some understanding of what kind of data is being shared, as they can easily open the files and look at their contents. However, the analogy falls short when it comes to *structured* data, which is commonly persisted as Linked Data in the Solid Pods. In this case, resource IRIs do not necessarily have meaning, and the organization of resources can be chosen arbitrarily by application developers. A similar concern is applicable to agent IRIs: How do I know my doctor's IRI is actually `https://nhs.gov.uk/id/123#me`? According to the UK Government's Department for Digital, Culture, Media & Sport's 2020 Cyber Security Breaches Survey [2] phishing attacks are one of the most common type of breaches experienced by UK businesses. Being just ordinary IRIs in the context of ACL rules, WebIDs suffer the same risk of being used in phishing attacks, where very similar looking WebIDs could be constructed that open the doors of your Pod to malicious actors. Detection mechanisms for phishing IRIs have been proposed, however these fall largely in the realm of heuristics.

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.

# Your doctor has Read & Write Access to your Medical Records
<#records> a acl:Authorization;
        acl:agent <https://nhs.gov.uk/id/123#me>;
        acl:accessTo <./MedicalRecords/>;
        acl:mode acl:Read, acl:Write.
```

Listing 1: Example ACL resource

Let us illustrate the mechanics of WAC using an example that will return throughout this paper; a doctor is requesting access to the medical records of their patient. If we were to realize this type of interaction pattern with Web Access Control, the physician would have to persist an ACL resource governing the medical records of the patient in the patient's Pod, as shown in Listing 1. Through this ACL, the doctor (identified by their WebID `https://nhs.gov.uk/id/123#me`) obtains read and write permissions on the patient's medical records. Note that the choice of a container named "MedicalRecords" to retain your medical information is a completely arbitrary one, such that the interpretability of this ACL rule could be considerably worse if the developers of these medical record applications made arbitrarily different naming choices. Also, the Solid protocol currently does not define how the doctor should request for their patient to grant these rights. Having the interpretability and modification of an ACL rule depend fully on implementation choices of the developer is not a desired behavior for an authorization system, let alone one that aims to maximize end-user control.

2.2 The Data Privacy Vocabulary

The Data Privacy Vocabulary[6] (DPV) [20] is a vocabulary that attempts to translate concepts and requirements related to the processing of personal information under data processing and privacy regulations, like GDPR, into classes and properties that can be used as Linked Data. It is structured to be extendable with concepts and requirements for specific jurisdictions, like the DPV-GDPR extension[7] that defines the GDPR-specific rights and legal bases concerning data processing.

2.3 Prior proposals for improving authorization in Solid

The Open Digital Rights Language (ODRL) [14] is a language for expressing policies that define permitted and prohibited actions over some entities. An ODRL profile and algorithm was proposed [12] as an extension of the existing ACL-mechanism used by Solid Pods to authorize requests. Furthermore, obligations and constraints can be imposed upon these actions. The proposed ODRL profile[8] enables the use of concepts from the Data Privacy Vocabulary in order to define policies that relate to data processing over some resources. The proposal also contextualizes the use of such policies for materializing complex data sharing preferences and legal bases for processing like informed consent. The authors [12] do highlight some significant challenges with their proposal, such as the efficiency of compliance checking with these ODRL policies, especially when used in a heterogeneous, decentralized architecture and combined with an inheritance mechanism, as well as the privacy risks associated with making these policies publicly accessible.

While evaluating different technical approaches that support the enforcement of legal rights given to data subjects under data protection regulation like GDPR, an assessment [13] was made of the affordances with respect to data governance provided by Solid and the SPECIAL project[9] in comparison to the current defacto standard of data subjects giving very broad consent to processing. In the evaluation of Solid in relation to data protection regulations it was found that Solid's current ACL-based solution for authorization falls short when trying to implement solutions adhering to the strict regulatory requirements set forth. Firstly, because of its poor user experience caused by issues like the lacking interpretability of access mode and resource identifiers for non-technical users, risk

[5]Acronym for Create, Read, Update, Delete.

[6]https://w3id.org/dpv#
[7]http://www.w3.org/dpv/dpv-gdpr#
[8]https://w3id.org/oac/
[9]The SPECIAL project was a research project that aimed to deliver technologies to reconcile big data applications with the necessary regulatory compliance with respect to data processing. It delivered user interfaces for consent and processing transparency as well as ontologies for the logging by data processing applications and for modeling data usage policies of both data subjects and data controllers that are machine verifiable. (https://specialprivacy.ercim.eu)

of phishing attacks due to the use of IRIs to identify agents, and the security concerns that arise from inherited ACL rules. Secondly, because ACLs fail to capture important concepts under data protection regulations that define what type of information is being shared, how that data will be processed and for what purpose, and which legal basis is used to warrant this processing. And lastly because the burden of modifying these ACL rules is currently delegated to application developers themselves, thus contradicting the original goals of returning control back to the end-user as developers have unlimited authority when modifying ACL rules and could resort to the dark patterns that have haunted modern-day implementations of consent on the web.

A layered, decentralized architecture for combining SPECIAL and Solid was also proposed and compared to these other approaches [13]. The concrete mechanics of the policy exchange and negotiation are left as future work by the authors, however their evaluation provides a good insight into the existing limitations of ACL based authorization when confronted with complex data processing applications.

2.4 Solid's Data Interoperability Panel

The Data Interoperability Panel within the Solid Community Panels[10] was started with the goal of standardizing the mechanics by which multiple applications can interoperate over the same data safely and effectively. In the process they try to increase user awareness and interpretability of the data stored in a Pod, by abstracting away complexities such as resource organization that are currently not governed by the Solid protocol, to finally enable multiple agents to safely and effectively interoperate over the same data. Most importantly, they aim to tackle these hurdles while preserving the fundamentals of the Solid protocol as it exists today.

In the context of the panel, two significant proposals have begun to take shape over the past year, namely the *Shape Trees* [3] and the Solid Application Interoperability [4] draft specifications. The former builds upon the existing specifications of RDF[11] and data shapes [17, 21], which respectively provide us with the foundations for interoperability through unambiguous identifiers (IRIs) and a structural schema against which individual RDF graphs can be validated. Where these existing specifications fall short however is in modeling complex resource hierarchies. Consider for example the organization of a collection of medical records that takes form in a Solid Pod where developers have relative freedom in both resource naming and the use of containers to gather their data. A Shape Tree defines structural constraints for a tree of resources in any ecosystem that has a notion of containers[12]. For each container, it allows shape constraint to be imposed on the contained resources. Shape Trees themselves can also contain other Shape Trees giving form to tree hierarchies (for example medical records as a whole may consist of medical images, prescriptions, bills, reports, etc.). The major strength of Shape Trees is that they can unambiguously define resource organization in a Pod and provide a higher-level abstraction that can be more easily understood by end-users. This way, Shape Trees guide applications and users by determining where data should be written to and where it can be read from. The modeling of related resource collections in this manner allows us to perform operations such as authorization, data migration and validation on this higher abstraction level as well. Especially in the context of authorization, defining rules at the level of Shape Trees rather than individual resources reduces complexity, the likelihood of errors and allows us to relate these higher-level conceptual resource aggregations to legal concepts such as Data Categories.

The Solid Application Interoperability (SAI) draft specification [4] leverages these proposed Shape Trees to standardize concrete mechanics by which applications and agents request access to information in a Solid Pod, the way by which they locate the concrete instances of the Shape Trees, and how they can interoperate over these. Up until now most of the specifics of these different operations were left open to individual application developers by the Solid specification, complicating interoperability over the same data. In the context of this paper, the standardizing of access requests is of specific importance, and will be used as a building block in our proposal. The SAI specification introduces the concept of an Authorization Agent as a service linked to an agent's WebID that manages the data under their control. It is tasked with processing access requests for the agent, managing previously granted permissions, and recording the concrete instances of Shape Trees through a collection of registries. While the specification is still under discussion by the panel, and some aspects of the mechanics of the authorization agent have not yet been fully defined or are deliberately being left open for implementation, we will be using many of the core concepts it sets forth in our proposal.

2.5 Linked Data Integrity & Authentication

The Data Integrity 1.0 draft community report [25] is a recent proposal by the W3C's Credentials Community Group, with the aim of providing authentication and data integrity capabilities to Linked Data resources through the use of mathematical proofs such as digital signature algorithms. It details a vocabulary for describing proof types, verification methods and algorithms. The origins of this work are to be found in the W3C's recommendation of the Verifiable Credentials Data Model [26], a data model that can be used to assert specific claims on a subject (such as a degree, driver's license, etc.) and which should be accompanied by a cryptographic proof that can assert their authenticity and integrity. These techniques will provide us with the necessary building blocks, in terms of authentication and accountability, we need to realize our proposed authorization architecture.

3 ARCHITECTURE

Our architecture splits out the implementation of consent as a legal basis for accessing personal data in the Solid Pod into two domains, where policies stored in the subject's Solid Pod form an interface between these different realms:

(1) On the one hand, the *end-user domain* is governed by a so-called *Access Management Application* which is tasked

[10]The Solid specification is drafted by different community panels, each focused on specific issues or domains that are relevant to Solid like authentication, authorization or data interoperability.
[11]The Resource Descriptor Format, core data model used in Semantic Web technologies to construct Linked Data resources.
[12]Solid builds upon the Linked Data Platform specification which governs the semantics of a container resource.

with validating the data processing request coming from the responsible data controller against applicable legal requirements, end-user data sharing preferences and, if the processing request is approved, storing it as a Processing Grant in the data subject's Solid Pod.

(2) On the other hand, the *technical domain* uses the *Authorization Agent*, as proposed by the Solid Application Interoperability specification, to handle concrete access requests made by applications and other agents in terms of Shape Trees, Data Shapes and ACL access modes. The interface between the two realms is formed by Processing Grants which are generated by the Access Management App and persisted in the agent's Solid Pod.

For authentication and identification of the different actors in the architecture we depend on the WebID [22] and Solid OIDC [10] specifications that are defined within the Solid Protocol version 0.9 [8]. In the following paragraphs we will be expanding upon both the Access Management App, Authorization Agent and the proposed concepts of Processing Requests and Processing Grants used to bind these two services.

3.1 End-User Realm: Access Management App

The Access Management App is used by Data Controllers to obtain the necessary approval for the Data Processing they are requesting for some personal data categories and processing actions in fulfillment of a processing purpose that was allowed for through a specific legal basis. Once it has received a Data Processing Request, the Access Management App will first verify if the request is admissible and will attempt to match it against any explicit data sharing preferences the user might have in their Pod that can lead to an automatic granting of the request. If no preferences turn out to explicitly match the request, the data subject must be polled for their explicit consent. Once a Processing Request is granted, it is stored as a Processing Grant in the Solid Pod and delivered to the inbox [9] of the Data Controller.

3.2 End-User Realm: Processing Requests and Processing Grants

Whenever a Data Controller (Requesting Party) wants to obtain permissions for performing some data processing on the data subject, it will be constructing a Processing Request. This Request is constructed based upon a proposed ODRL profile [12] and concepts from the Data Privacy Vocabulary. An example request for medical records based upon explicit consent is shown in Listing 2. The request details handling of the personal data, in terms of legal basis (in the case of this paper we will only consider explicit consent), data controller, and specific permissions that will be needed in the context of the processing.

Each permission specifies what personal data categories it concerns as a target, what actions it needs to perform on this data, and constraints on the purpose or output of the processing. Other constraints could be envisioned as well, like technical measures used in the processing and associated risks, however these haven't been explored in the current proposal.

The *Data Processing Request* itself is presented to the Access Management App accompanied by a Data Integrity Proof that

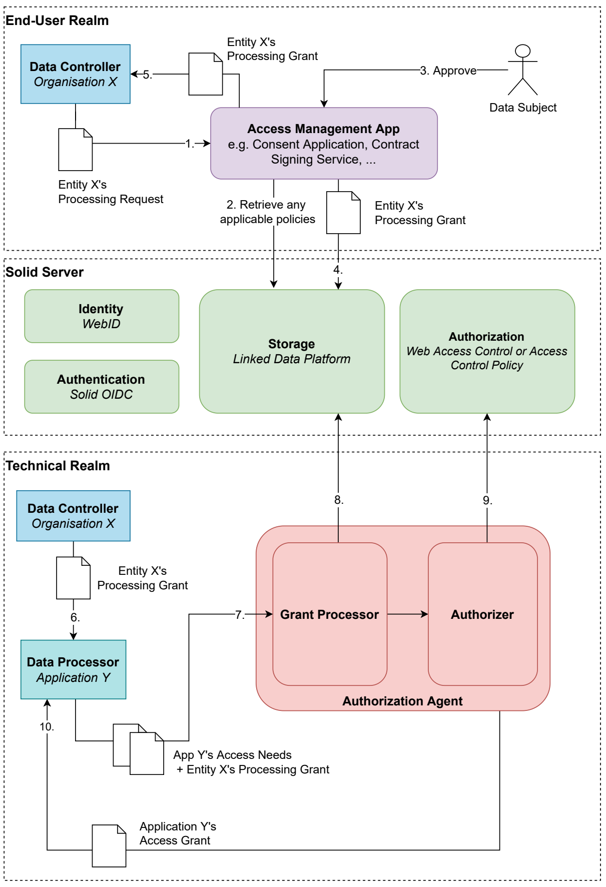


Figure 1: Overview of our proposed architecture, linking an End-User realm governing Data Processing permissions with a Technical realm following the Application Interoperability specification

was generated by the Data Controller, this way the provenance and integrity of the request can be validated. Through this signing mechanism, the risk of spam or other malicious attacks with respect to the Access Management App and Processing Request procedures could be reduced, for example by assigning different trust levels to issuer services that can be used by Data Processors to sign their request based upon requirements like identity validation or regulatory compliance.

Finally, a *Processing Grant* is constructed from the Processing Request by first completing the legal basis, i.e., consent, with any other necessary attributes that were either gathered in interaction with the data subject or in an automated manner by the access management app. Thereafter any permissions that have not been witheld will be removed from the Grant, the RDF graph is supplemented with a Revocation Status attribute conforming to the W3C Revocation List 2020 specification [24] for revoking Data Integrity Proofs such that the Access Management App can revoke the Processing Grant at a later time, and a Data Integrity Proof will be created and signed by the Access Management App to indicate that the legal

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/>.
@prefix dpv: <http://www.w3.org/ns/dpv#>.
@prefix cert: <http://www.w3.org/ns/auth/cert#>.
@prefix oac: <https://w3id.org/oac/>.

@prefix : <https://example.com/#>.

:medicalRecordsConsent a odrl:Policy, dpv:PersonalDataHandling;
    odrl:profile oac:;
    dpv:hasLegalBasis [
        a dpv:Consent;
    ];
    dpv:hasDataController <https://example.com/id/doctor#me>;
    odrl:permission [
        a odrl:Permission;
        odrl:assignee <https://example.com/id/doctor#me>;
        odrl:target dpv:HealthRecord, dpv:Prescription, dpv:
      HealthHistory;
        odrl:action dpv:Collect, dpv:Consult, dpv:Analyse, dpv:Alter;

        odrl:constraint [
            odrl:leftOperand oac:Purpose;
            odrl:operator odrl:isA;
            odrl:rightOperand :MedicalConsultation
        ]
    ].
```

Listing 2: Example Unsigned Processing Grant Request

requirements for the data processing to be approved are fulfilled. The signed Processing Grant can be seen as an instruction for the authorization agent to provision certain types of information to a Data Controller and its designated processors.

3.3 Technical Realm: Authorization Agent

The Authorization Agent is largely based upon the proposed Solid Application Interoperability specification in terms of its semantics and API, which is still under discussion by the panel and thus might be subject to changes. This is enabled by the fact that mechanics of the authorization agent are largely left open to implementation, such that additional authorization checks can be executed between the *Access Needs* being presented to the authorization agent and the delivery of a so-called Access Grant that specifies the concrete data that has been elected for sharing with the application.

In fact, the only modification to the authorization agent interface that we are proposing in this paper is that a Processing Grant should accompany the Data Processor's access needs when access is being requested. This way the authorization agent can link the access request being made by the application or service, acting as a Data Processor for the Data Controller, to a valid legal basis for data processing. It then becomes the task of a Grant Processor module in the Authorization Agent to match the specified Processing Grant to the Processor's Access Needs in terms of Data Needs (Shape Trees) and Access Modes. The latter confronts us with the need for an unambiguous equivalence relation between the abstract definitions in the Processing Grant and their technical counterparts in the Access Needs.

Finally, once the *Grant Processor* has determined that the Data Processor's request actually matches our initial Processing Grant, it can proceed with an Authorizer that is tasked with modifying the atomic access control rules applicable to the instances of the Shape Trees that were specified in the service's Data Needs. Once this process has ended, an Access Grant is returned to the Data Processor and the necessary registrations are added to the Pod.

3.4 Auxiliary Rules & Policies

While the ODRL-based processing request and processing grant may suffice for defining the data processing that is being requested and approved on a business-level, it is insufficient for the authorization agent to relate these with the technical access needs specified by a Data Processor like an application. The semantic gap here is twofold, on the one hand we need to unambiguously define what data in the Pod falls under the approved processing and on the other hand we must know what actions on this data are permitted.

Firstly, the abstract data categories used to specify the personal data being shared under the approved data processing activities must be related to concrete technical data type information. As was elaborated upon in the background section, the combination of data shapes and shape trees as a mechanism for defining resource collections and their structure allows us to delimit conceptually related resources in the Pod like medical records, pictures, notes, etc. Through an additional set of rules that is configured by the data subject in their Solid Pod, a so-called Data Category Equivalence policy, we link the technical resource type information provided by Data Shapes and Shape Trees to Personal Data Categories as they are specified under DPV and used in the ODRL profile.

As higher level abstractions are used to define the actions that the processing allows for, we must also relate the Processing categories from the DPV with the Access Modes as they are used in both Solid's Access Control mechanism and the technical Access Needs specified by the Data Processor. These can be defined by the subject as Processing Access Needs, which are stored as an additional set of rules in the Pod.

Furthermore, while not elaborated upon in this paper, the proposed ODRL profile [12] was devised with the concept of data sharing preferences which allowed for the data subject to also express more complex data processing activities that could automatically be permitted to some requesting party based on purpose, data and processing categories. Such policies could also be persisted in the Solid Pod besides these previously noted equivalence relations and the concrete processing grants that flow from them.

4 EXAMPLE USE CASE

In this section we will be illustrating our proposed architecture through the motivating use case of a doctor looking to access the medical records of a patient stored in their Solid Pod based on an explicit, informed consent. We assume no previous consent or authorizations were given over the patient's Electronic Medical Records (EMR). Figure 2 provides a complete sequence diagram, highlighting the relevant exchanges that are initiated by the physician and their electronic patient record application.

The exchange starts with a discovery phase *(steps 1–2)* where the application aims to determine which Access Management Application and Authorization Agent the patient has elected to use through their WebID. Once it has been determined that no previous registration exists for the EMR application with the Authorization

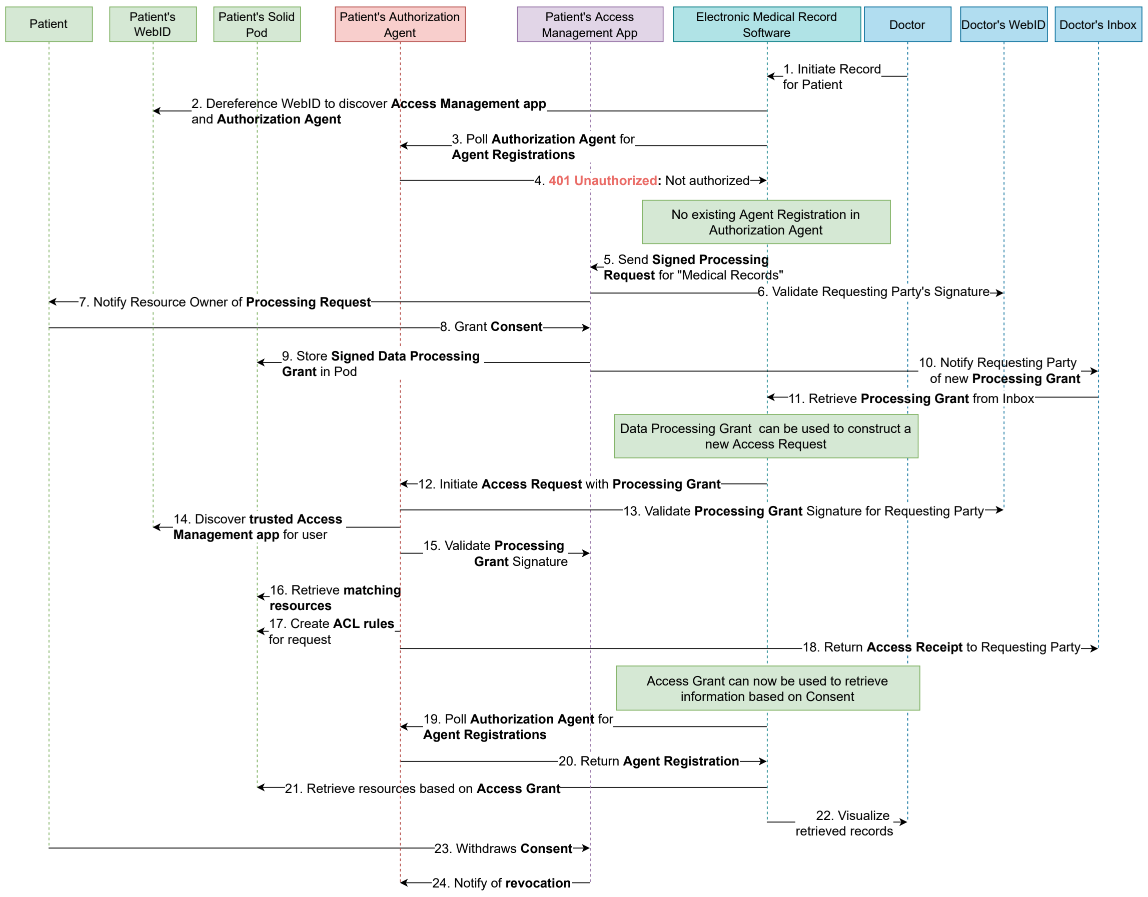

Patient
Patient's WebID
Patient's Solid Pod
Patient's Authorization Agent
Patient's Access Management App
Electronic Medical Record Software
Doctor
Doctor's WebID
Doctor's Inbox
1. Initiate Record for Patient
2. Dereference WebID to discover Access Management app and Authorization Agent
3. Poll Authorization Agent for Agent Registrations
4. 401 Unauthorized: Not authorized
No existing Agent Registration in Authorization Agent
5. Send Signed Processing Request for "Medical Records"
6. Validate Requesting Party's Signature
7. Notify Resource Owner of Processing Request
8. Grant Consent
9. Store Signed Data Processing Grant in Pod
10. Notify Requesting Party of new Processing Grant
11. Retrieve Processing Grant from Inbox
Data Processing Grant can be used to construct a new Access Request
12. Initiate Access Request with Processing Grant
13. Validate Processing Grant Signature for Requesting Party
14. Discover trusted Access Management app for user
15. Validate Processing Grant Signature
16. Retrieve matching resources
17. Create ACL rules for request
18. Return Access Receipt to Requesting Party
Access Grant can now be used to retrieve information based on Consent
19. Poll Authorization Agent for Agent Registrations
20. Return Agent Registration
21. Retrieve resources based on Access Grant
22. Visualize retrieved records
23. Withdraws Consent
24. Notify of revocation

Figure 2: Sequence diagram highlighting the exchanges for the motivating use case of a doctor requesting explicit consent for access to medical records of their patient.

Agent *(steps 3–4)*, a new Processing Request will be initialized and transferred to the patient's access management application *(step 5)*. After validation and explicit consent *(steps 6–8)*, a signed Processing Grant is created by the Access Management Application, stored in the patient's Pod and delivered to the physician's inbox *(step 9–10)*. Subsequently an Access Request for the patient's Authorization Agent can be constructed by the EMR application based on its Access Needs defined in terms of Shape Trees (in this case, shape trees relevant to the patient's medical records), and accompanied by the physician's Processing Grant *(step 12)*. After validation of the Processing Grant by the Authorization Agent *(steps 13–15)*, it is converted into ACL-rules for the instances of the Shape Trees mentioned in the Access Needs *(steps 16–17)*. An Access Receipt[13] is then returned to the physician's inbox *(step 18)*, finally allowing for the EMR application to visualize the patient's medical records *(step 19–22)*. If the patient subsequently chooses to withdraw their consent through the Access Management App *(step 23)*, the app will modify a Revocation List in order to revoke the Processing Grant that was initially provided and notify the Authorization Agent *(step 24)*.

5 DISCUSSION

One of the major departures from previous proposals is that this architecture aims to separate the problem of reconciling technical authorization with the legal requirements for data processing into two distinct domains. On the one hand, there is an end-user realm where the user is presented with requests for data processing in terms of processing actions happening on more abstract data categories, and where an end-user can determine explicit data sharing preferences. On the other hand, there is a technical realm where Solid's proposed Application Interoperability Specification governs how application developers can gain access to resources in an agent's Solid Pod, once a proper legal basis for processing has been established. The concept of Data Processing Grants that are verifiable through the W3C's Data Integrity Specification for Linked Data form the link between these two distinct domains, combined with policies that relate the meaning of Data Categories and Processing Actions to technical concepts that can be used by the Solid Pod.

[13]Notification referencing an Access Grant

Whereas previous solutions aimed to integrate business concepts into Solid's existing authorization mechanisms, for example through expanding upon Access Control with purpose and policy concepts, our proposal retains the current semantics and mechanics of Web Access Control and uses a layered architecture on top of it to introduce business concepts. While this does not solve the core limitations of the ACL mechanism as were mentioned in previous sections, like inheritance concerns and over-permissioning, it prevents us from importing these same issues into the higher level concepts of Access Grants and Data Processing Grants. Furthermore our architecture aims to reduce assumptions made on the supported features by the Solid server by externalizing the concepts of an Access Management App and an Authorization Agent such that these can be separate services a user chooses to add to their WebID and link to their Solid Pods. Additionally, an explicit dependency on the use of Web Access Control can be avoided in this framework, as long as a sufficient mapping from the business domain to the authorization mechanism supported by the Solid Pod exists for the Authorization Agent to perform. Especially in the light of the recent proposal of the Access Control Policy (ACP) language [6] as an alternative to Web Access Control, which addresses a number of the shortcomings we have highlighted about the latter, this decoupling is a desirable property.

Some of the highlighted challenges in the ODRL proposal [12] have been addressed here, while others will necessitate further consideration. With respect to efficiency, by introducing the Authorization Agent as an intermediary for providing the technical access, we avoid the necessity of matching policies for each HTTP request on a resource in the Pod and convert this into a negotiating step that theoretically should only occur if the access needs or the processing grant of the application have been modified. Moreover, in our proposal, policies do not show the hierarchical inheritance that is common for ACL resources thus compliance checking does not need to take this into account. Still, it might be the case that complex data sharing preferences used by the access management app or elaborate processing grants could lead to an unacceptable time complexity when used in practice. By introducing the access management app as a dynamic negotiator in the flow we avoided the need for public policies to be advertised by the Solid Pod, which addresses important privacy concerns with a policy based solution (i.e., what if anyone can see that you have shared your medical records for the purpose of a past treatment with a psychiatrist). Specific legal issues raised in the ODRL proposal [12] remain largely applicable to our proposal as well, i.e., the legal implications of user choice enabled by Solid's novel approach to data governance, the necessity of awareness of the applicable jurisdiction and its requirements for data processing activities and whether data sharing preferences, as were briefly touched upon, indeed constitute a form of consent.

While we have focused largely on the problem space of implementing explicit consent as a legal basis throughout this proposal, there could be room for enabling other legal bases to be enforced by the access management app as well. For example when processing is requested on the basis of a contractual obligation, the Access Management App could retrieve the contract from the subject's Pod and validate it against the identity of the requesting party for the Data Processing.

6 CONCLUSIONS & FUTURE WORK

While the proposed architecture aims to provide a crucial missing link between the atomic ACL-based authorization mechanism currently used by Solid Pods and the more abstract concepts, and safeguards required by data protection regulations like GDPR, it still has a number of open challenges that will need to be addressed before such an architecture can be practically implemented and used by developers and end-users.

Firstly, both the Shape Trees and Interoperability Specifications are still being discussed by the community panel and have only very recently seen their first practical implementations. Outside of the context of this panel, the proposal has not yet gained major traction, which could imply that the specifications might see significant changes before they are finally adopted into the Solid protocol. Another major hurdle that these important building blocks for our proposal face is the complexity of implementing them without breaking compatibility for existing applications and services, while retaining the required semantics for those that do already depend on them. Also the required registries for the Interoperability Specification and the additional metadata necessitated by the use of Shape Trees might prove challenging to consistently maintain within the Solid Pod without imposing additional requirements on its operation.

Secondly, due to the fact that our authorization mechanism ultimately still depends on the enforcement of Web Access Control rules by the Solid Pod we are again subject to its limitations outside the context of the proposed granting procedure. This means that if a single agent or application has obtained multiple processing grants from the data subject for the same Shape Trees, we cannot effectively differentiate between them when a request to a concrete resource comes in. One could ask the question whether this differentiation in context even matters at that point, as it would depend on the honesty of the client. However from a legal perspective this distinction could matter, and might need to be logged and recorded somewhere. This highlights the need for further investigation into whether and how a Solid-based application or service can fully comply with data processing regulations throughout its lifecycle, not only in terms of the initial authorization we have focused on here but also in terms of legal logging, data minimization, compliance monitoring, etc.

Also, compliance checking with respect to the proposed ODRL profile [12] is a problem that warrants further investigation as generic ODRL policy checking algorithms will be necessary. This way, we should be able to match an incoming processing request with any existing data sharing preferences of the data subject. Managing the ambiguity that is caused by allowing an end-user's policies to define how generic concepts used in the processing requests map to attributes of the Solid Pod is another challenge that still needs thorough consideration. Furthermore, the trust model between the different entities and services in our architecture needs to be considered in more detail as to identify potential security risks. Finally, the technical overhead imposed by this solution on query efficiency should be further analyzed as well, given that the negotiation process introduces asynchronicity to the process of accessing resources in a Solid Pod.

ACKNOWLEDGMENTS

The authors would like to thank Justin Bingham, Tom Haegemans, Sabrina Kirrane, and Eric Prud'hommeaux for giving their insights and feedback regarding this work. This research is supported by *SolidLab Vlaanderen* (Flemish Government, EWI and RRF project VV023/10). Ruben Taelman is a postdoctoral fellow of the Research Foundation – Flanders (FWO) (1274521N).

REFERENCES

[1] 2016-04-27. 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). *Official Journal of the European Union* L 119 (2016-04-27).

[2] UK Department for Digital, Culture, Media & Sport . 2020. *Cyber Security Breaches Survey 2020.* Retrieved February 6, 2022 from https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020

[3] Justin Bingham and Eric Prud'hommeaux. 2022. *Shape Trees Specification.* Technical Report. Retrieved February 6, 2022 from https://shapetrees.org/TR/specification/

[4] Justin Bingham, Eric Prud'hommeaux, and elf Pavlik. 2021. *Solid Application Interoperability.* Technical Report. Retrieved February 6, 2022 from https://solid.github.io/data-interoperability-panel/specification/

[5] Piero A. Bonatti, Sabrina Kirrane, Iliana M. Petrova, and Luigi Sauro. 2020. Machine Understandable Policies and GDPR Compliance Checking. *CoRR* abs/2001.08930 (2020). arXiv:2001.08930 https://link.springer.com/article/10.1007/s13218-020-00677-4

[6] Matthieu Bosquet. 2021. *Access Control Policy (ACP).* Technical Report. Retrieved February 6, 2022 from https://solid.github.io/authorization-panel/acp-specification/

[7] Sarven Capadisli and Tim Berners-Lee. 2021. *Web Access Control.* Technical Report. Retrieved February 3, 2022 from https://solidproject.org/TR/wac

[8] Sarven Capadisli, Tim Berners-Lee, Ruben Verborgh, and Kjetil Kjernsmo. 2021. *Solid Protocol.* Technical Report. Retrieved February 6, 2022 from https://solidproject.org/TR/2021/protocol-20211217

[9] Sarven Capadisli and Amy Guy. 2017. *Linked Data Notifications.* Technical Report. Retrieved February 6, 2022 from https://www.w3.org/TR/ldn/

[10] Aaron Coburn, elf Pavlik, and Dmitri Zagidulin. 2022. *Solid-OIDC.* Technical Report. Retrieved February 7, 2022 from https://solid.github.io/solid-oidc/

[11] Dirk De Bot and Tom Haegemans. 2021. *Data Sharing Patterns as a Tool to Tackle Legal Considerations about Data Reuse with Solid: Theory and Applications in Europe.* https://lirias.kuleuven.be/retrieve/599839

[12] Beatriz Esteves, Harshvardhan J. Pandit, and Víctor Rodríguez-Doncel. 2021. ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW).* 298–306. https://doi.org/10.1109/EuroSPW54576.2021.00038

[13] Giray Havur, Miel Vander Sande, and Sabrina Kirrane. 2020. Greater Control and Transparency in Personal Data Processing. 655–662. https://doi.org/10.5220/0009143206550662

[14] Renato Iannella and Serena Villata. 2018. *ODRL Information Model 2.2.* Technical Report. Retrieved February 6, 2022 from https://www.w3.org/TR/odrl-model/

[15] Lalana Kagal, Tim Finin, and Anupam Joshi. 2003. A Policy Based Approach to Security for the Semantic Web. In *The Semantic Web - ISWC 2003*, Dieter Fensel, Katia Sycara, and John Mylopoulos (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 402–418.

[16] Gergely G. Karácsony. 2019. Managing personal data in a digital environment - did GDPR's concept of informed consent really give us control? *International Conference on Computer Law, AI, Data Protection & the Biggest Tech Trens* (2019). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3452573

[17] Holger Knublauch and Dimitris Kontokostas. 2017. *Shape Expressions Language 2.1.* Technical Report. Retrieved February 6, 2022 from https://www.w3.org/TR/shacl/

[18] Michael Kretschmer, Jan Pennekamp, and Klaus Wehrle. 2021. Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web. *ACM Trans. Web* 15, 4, Article 20 (jul 2021), 42 pages. https://doi.org/10.1145/3466722

[19] Midas Nouwens, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal. 2020. *Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376321

[20] Harshvardhan J. Pandit, Axel Polleres, Bert Bos, Rob Brennan, Bud Bruegger, Fajar J. Ekaputra, Javier D. Fernández, Roghaiyeh Gachpaz Hamed, Elmar Kiesling, Mark Lizar, and et al. 2019. Creating a Vocabulary for Data Privacy: The First-Year Report of Data Privacy Vocabularies and Controls Community Group (DPVCG). https://doi.org/10.1007/978-3-030-33246-4_44

[21] Eric Prud'hommeau, Iovka Boneva, Jose Emilio Labra Gayo, and Gregg Kellogg. 2019. *Shape Expressions Language 2.1.* Technical Report. Retrieved February 6, 2022 from http://shex.io/shex-semantics/index.html

[22] Andrei Sambra, Henry Story, and Tim Berners-Lee. 2014. *WebID 1.0.* Technical Report. Retrieved February 4, 2022 from https://www.w3.org/2005/Incubator/webid/spec/identity/

[23] Steve Speicher, John Arwe, and Ashok Malhotra. 2015. *Linked Data Platform 1.0.* Technical Report. Retrieved February 6, 2022 from https://www.w3.org/TR/ldp/

[24] Manu Sporny and Dave Longley. 2021. *Revocation List 2020.* Technical Report. Retrieved February 6, 2022 from https://w3c-ccg.github.io/vc-status-rl-2020/

[25] Manu Sporny and Dave Longley. 2022. *Data Integrity 1.0.* Technical Report. Retrieved February 6, 2022 from https://w3c-ccg.github.io/data-integrity-spec/

[26] Manu Sporny, Dave Longley, and David Chadwick. 2021. *Verifiable Credentials Data Model v1.1.* Technical Report. Retrieved February 6, 2022 from https://www.w3.org/TR/vc-data-model/

[27] Winfried Veil. 2018. The GDPR: The Emperor's New Clothes - On the Structural Shortcomings of Both the Old and the New Data Protection Law. *Consumer Law eJournal* (2018). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3305056

[28] Ruben Verborgh. 2022. Re-decentralizing the Web, for good this time. In *Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*, Oshani Seneviratne and James Hendler (Eds.). ACM. https://ruben.verborgh.org/articles/redecentralizing-the-web/

[29] Serena Villata, Luca Costabello, Nicolas Delaforge, and Fabien Gandon. 2012. Social Semantic Web Access Control? *Journal on Data Semantics* 2 (03 2012). https://doi.org/10.1007/s13740-012-0014-9

Appendix 2: Performance Evaluation of UMA Authorization Service

The table below summarizes a performance evaluation of the **response times (in milliseconds)** for a token request with the UMA Authorization Service. For this evaluation 250 authorization requests were randomly made for each of the different resource types defined by the Solid Application Interoperability specification[58] and for differing numbers of agent registrations and data grants.

The experiment was run on NodeJS v16.14.0, with an Apple M1 Pro processor and 32GB RAM, using the 1.0.0 release of our UMA Authorization Service and SAI Authorizer module. The Community Solid Server v4.0.1 release was used, with both the AS and CSS running locally on the machine. Data was generated synthetically, using the Python-script included in the repository.

#Agent Registrations	**#Data Registrations**	**Resource Type**	Mean	Standard Deviation	Minimum	Median	75th Percentile	99th Percentile	Maximum
20	10	Access Grant	5,12	1,34	3,13	4,92	5,76	9,96	14,63
20	10	Agent Registration	5,89	3,93	2,88	5,20	5,83	25,32	37,96
20	10	Data Grant	5,31	2,70	3,21	4,98	5,59	16,21	36,59
20	10	Data Instance	34,39	13,16	10,20	33,64	41,86	72,59	104,75
20	10	Data Registration	7,40	30,28	2,68	4,91	5,67	32,71	480,81
20	50	Access Grant	7,57	2,55	4,51	7,22	8,03	14,87	37,12
20	50	Agent Registration	8,90	24,38	4,60	6,93	7,99	18,42	389,64
20	50	Data Grant	9,04	27,25	5,00	7,10	7,93	17,02	437,12
20	50	Data Instance	120,60	60,40	15,69	126,98	157,29	211,14	648,95
20	50	Data Registration	9,83	29,19	4,80	6,98	7,80	50,95	459,58
20	100	Access Grant	14,29	36,99	7,46	10,21	11,39	51,86	425,98
20	100	Agent Registration	10,88	4,24	7,39	10,08	11,47	23,70	54,56
20	100	Data Grant	14,59	36,96	7,27	9,93	11,05	60,47	420,99
20	100	Data Instance	214,72	98,60	29,66	217,38	288,18	438,03	566,67
20	100	Data Registration	16,56	51,27	7,01	10,08	11,29	251,42	513,31
100	10	Access Grant	8,78	63,31	2,61	4,43	5,17	12,74	1005,14
100	10	Agent Registration	5,01	1,63	2,61	4,77	5,42	11,94	19,41
100	10	Data Grant	4,99	4,22	2,67	4,49	5,06	12,26	62,86
100	10	Data Instance	30,97	11,47	9,86	30,56	38,08	55,59	113,90
100	10	Data Registration	4,72	2,58	2,46	4,40	5,06	8,31	42,11
100	50	Access Grant	8,04	2,38	4,91	7,51	8,56	18,78	24,82
100	50	Agent Registration	19,16	97,81	4,72	7,75	8,95	338,60	1143,02
100	50	Data Grant	12,35	63,03	4,67	7,44	8,46	32,57	1002,74
100	50	Data Instance	127,03	69,48	20,97	130,71	164,13	221,25	890,60
100	50	Data Registration	12,59	70,28	5,13	7,52	8,71	20,00	1118,50
100	100	Access Grant	14,78	66,24	7,26	10,04	11,13	26,23	1056,96
100	100	Agent Registration	24,23	123,12	7,16	9,83	10,99	587,89	1185,42
100	100	Data Grant	14,50	63,48	7,12	9,96	10,91	24,75	1012,74
100	100	Data Instance	222,95	153,32	30,71	210,61	282,48	852,17	1366,28
100	100	Data Registration	11,76	6,70	6,80	10,40	11,55	49,04	73,51

The table below summarizes a performance evaluation of the **response times (in milliseconds)** for a token request with the UMA Authorization Service. For this evaluation 250 authorization requests were randomly made for each of the different resource types defined by the Solid Application Interoperability specification[58] for 10 agent registrations, 50 data grants per agent and an increasing number of data instances per grant.

The experiment was run on NodeJS v16.14.0, with an Apple M1 Pro processor and 32GB RAM, using the 1.0.0 release of our UMA Authorization Service and SAI Authorizer module. The Community Solid Server v4.0.1 release was used, with both the AS and CSS running locally on the machine. Data was generated synthetically, using the Python-script included in the repository.

# Data Instances	**Resource Type**	Mean	Standard Deviation	Minimum	Median	75th Percentile	99th Percentile	Maximum
5	Access Grant	10,32	29,17	4,72	7,56	8,43	42,70	463,02
5	Agent Registration	11,86	38,11	5,02	7,38	8,27	53,02	449,40
5	Data Grant	9,94	30,87	4,75	7,44	8,28	21,52	493,32
5	Data Instance	125,25	57,54	20,93	123,08	166,98	246,97	552,08
5	Data Registration	7,97	3,37	4,92	7,37	8,36	20,99	46,37
50	Access Grant	8,31	4,66	4,72	7,50	8,52	38,68	46,89
50	Agent Registration	10,01	29,65	4,69	7,54	8,55	17,96	473,95
50	Data Grant	9,67	27,76	4,58	7,31	8,25	33,18	442,82
50	Data Instance	126,49	55,23	24,04	124,35	169,96	233,34	283,31
50	Data Registration	11,48	41,69	5,23	7,49	8,40	31,70	490,42
100	Access Grant	10,57	28,26	5,10	7,35	8,23	40,93	333,08
100	Agent Registration	10,60	32,73	5,05	7,39	8,20	18,39	379,26
100	Data Grant	7,64	3,03	4,85	7,20	8,25	13,57	49,14
100	Data Instance	137,88	68,84	25,70	135,67	187,50	270,18	669,78
100	Data Registration	11,01	37,73	4,94	7,27	8,22	34,83	435,44