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

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
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.

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

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

[6] https://w3id.org/dpv#
[7] http://www.w3id.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 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 [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

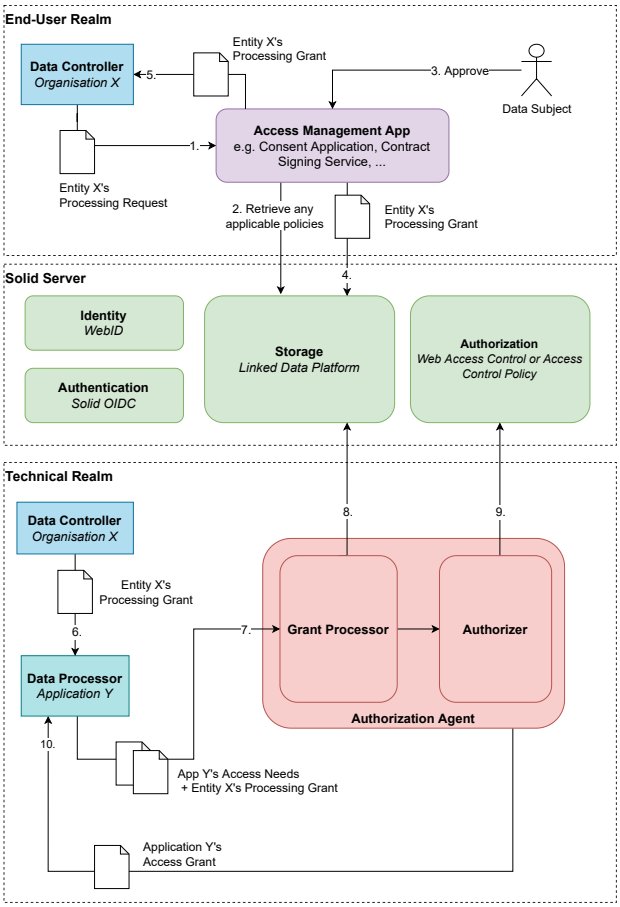

Figure 1: Overview of our proposed architecture, linking an End-User realm governing Data Processing permissions with a Technical realm following the Application Interoperability specification

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

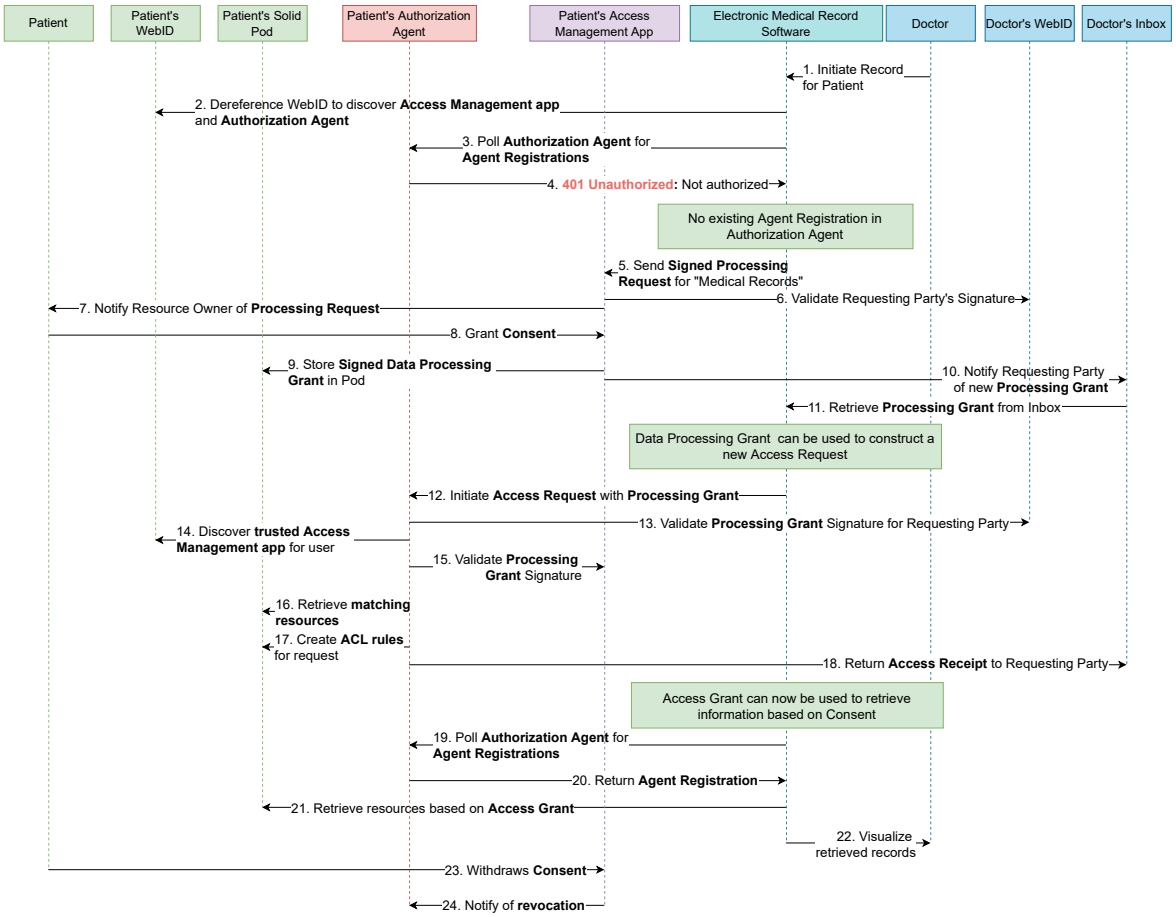

Figure 2: Sequence diagram highlighting the exchanges for the motivating use case of a doctor requesting explicit consent for access to medical records of their patient.

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