Inter-Pod Credential Exchange Protocol via Linked Data Notifications

Jonas Steinbach[1], Gertjan De Mulder[1], Ben De Meester[1], Beatriz Esteves[1] and Ruben Verborgh[1]

[1]*Ghent University – imec – IDLab, Department of Electronics and Information Systems, Ghent, Belgium*

Abstract

Solid is a set of specifications to describe a decentral Web protocol that enables Personal Data Spaces, empowering individuals to keep control of their personal data, stored in decentralized personal online data stores called Pods. Here, Verifiable Credentials (VC) are a type of data of particular interest, as they allow for cryptographically secure and verifiable digital credentials, which can be used for access and identity management, and also tie into different European Data strategy use cases. However, although the use of VCs within Solid is increasingly receiving attention, there exists no VC exchange protocol within Solid. More specifically, current applications need to rely on implicit agreements for both the transfer destination (i.e. the Web location where the VC should be sent to), and the data format of the messages exchanged. This forces stakeholders to invent their own credential transfer mechanisms, thereby hampering interoperability and adoption. In this paper, we present a VC exchange protocol between Solid Pods with explicit target destination and message format. We propose a working and interoperable protocol using DIDComm for structured messaging primitives in the form of JSON-headers and LDN inboxes as target destinations. LDN inboxes are interoperable with Solid and can be advertised via WebIDs, however, their setup and management of LDN inboxes is difficult, and reliance on WebIDs for inbox discovery might prevent interoperability between systems with different identifiers.

Keywords

Solid, Exchange protocol, Linked Data Notifications, Verifiable Credentials, Personal Data Spaces

1. Introduction

Opportunities for Personal Data Spaces (PDS) emergedue to ongoing effort on Data Spaces [1]: (European) legislative and governmental efforts for more privacy and control of personal data, e.g. through the European Data strategy [2], and laws such as the GDPR [3], and the DGA [4]. One type of PDS is Solid [5]: a set of specifications to describe a decentral Web protocol that builds on top of Linked Data[1]. Solid allows the flow and exchange of different kinds of personal and non-personal data in decentralized personal online data stores, called (Solid) Pods [6].

The Second International Workshop on Semantics in Dataspaces, co-located with the Extended Semantic Web Conference, May 26 – 27, 2024, Hersonissos, Greece

jonas.steinbach@ugent.be (J. Steinbach); gertjan.demulder@ugent.be (G. De Mulder); ben.demeester@ugent.be (B. De Meester); beatriz.esteves@ugent.be (B. Esteves); ruben.verborgh@ugent.be (R. Verborgh)
https://github.com/gertjandemulder (G. De Mulder); https://ben.de-meester.org/#me (B. De Meester)
0000-0003-4440-5037 (J. Steinbach); 0000-0001-7445-1881 (G. De Mulder); 0000-0003-0248-0987 (B. De Meester); 0000-0003-0259-7560 (B. Esteves)

[1]https://solidproject.org/

Of particular interest to PDS and Solid are Verifiable Credentials (VC) [7]: cryptographically secure and verifiable digital credentials that can contain both personal and non-personal (Linked) data. Such verifiable claims can be used for access and identity management (e.g. within Solid [8]), and tie into different use-cases of the European Data strategy (e.g. citizen governance and verifiable diplomas [2, 9, 10]).

However, there exists no transfer protocol that specifies how to transfer VCs in the Solid ecosystem – from, to, and between Solid Pods – without relying on implicit application-dependent agreements. VCs are thus transferred on case-by-case basis. This challenges interoperability and adoption, as new developers and users are prevented from adopting and re-using existing working solutions, and have to resort to out-of-band-communication or reading each others' source code.

All the steps within a VC lifecyle –issuing, holding, and verifying– have been demonstrated to work with Solid [8], but what if there are different systems and implementations, from different vendors and organizations, as is commonplace in decentralized architectures? How do we transfer VCs between Pods? Which data format should be used? Which credential representation? Where should VCs be sent? Where will they be held?

In this paper, we present a VC exchange protocol between Pods. We especially focus on (i) where and how should VCs be transferred between Solid Pods, and (ii) in which format. After introducing background and related work (Section 2), we specify our approach (Section 3) and implementation (Section 4), finally discussing (Section 5) and concluding on it (Section 6).

2. Background and Related Work

Solid – currently developed by the Solid W3C Community Group[2] being formed into a W3C Working Group – adopts the Linked Data Platform (LDP) specification[3], with each Solid Pod containing its own collection of linked resources (e.g. images, CSV files, and RDF data). However, as neither Solid nor LDP specify rules on *where* to write data, interoperability issues arise [11]. **Identity in Solid** is handled using a WebID: a public accessible URI that dereferences to a WebID Profile Document containing profile (RDF) data[4].

Verifiable Credentials (VCs) have been used and explored in combination with Solid in multiple ways. VCs have been used in Solid for issuance, verification, authentication, and authorization [8, 12]. But to the best of our knowledge, there exists no transfer protocol that specifies how to transfer VCs within the Solid ecosystem, without relying on implicit application-dependent agreements.

Communication in Solid is done via standard HTTP methods, with more detailed notification and exchange protocols built on top. The Solid protocol itself specifies two notification protocols: Linked Data Notifications (LDN) [13] and Solid Notifications[5].

LDN is a W3C recommended permissive and decentralized push-based communication protocol[6], intended for interoperability, and without restraints on possible usage domains. It

[2]https://www.w3.org/community/solid/
[3]https://www.w3.org/TR/ldp/
[4]https://solid.github.io/webid-profile/
[5]https://solidproject.org/TR/notifications-protocol
[6]https://www.w3.org/TR/ldn/

applies the Solid principles – deduplication of data storage and data usage – by treating and persisting notifications as resources. That means any received notification gets saved on the receiving server as a document, which allows notifications to be treated as unique entities, including assignment of a URI. Center to LDN is the *LDN inbox*, which acts as destination and receiving endpoint for all Linked Data notifications, in turn allowing further consumption. LDN does not specify any rules on message type or format. It also does not do any authentication, validation, or verification of messages. In Solid, the LDN inbox gets set via the public-facing WebID profile document[7], allowing its discovery.

The **Solid Notification Protocol** in comparison is a subscription based protocol, intended for receiving update notification on resource changes. It allows clients to listen to resource updates. Unlike LDN, which starts with the discovery of the inbox, a Solid Notification subscription begins with discovering notification services available on a resource. Then, a Subscription Request is sent, and a Notification Channel established. This notification channel can be of varying types, including WebSockets, WebHooks, and LDN[8].

Out of these protocols, LDN is the most generic and modular, evidenced by its ample extensions and adaptions, e.g. for Web Push Notifications [14], or for communication in scholarly use-cases [15] and multi-agent Web communication [16].

The Trust over IP stack describes how to **facilitate trust and exchange VCs between peers** on the Internet [17]. The DIDComm methodology of messaging protocols is responsible for the trusted peer connection and data exchange. DIDComm Messaging [18] provides a set of privacy primitives to support the modular creation of decentral, trustworthy, and secure exchange protocols. It can be used to build higher-order protocols intended for more specialized purposes.

3. Design

When transferring VCs between Solid Pods, current implementations' interaction between multiple parties depends on implicit contracts and out-of-band communication for both the transfer destination (i.e., the location where the VC gets sent too) and the data structure.

Our solution design therefore consists of two components. On the one hand, by using a notification protocol as a way to transfer data, we make the out-of-band agreements explicit. On the other hand, by adhering to a standardized message structure, we make sure that interoperability across applications is retained.

LDN solves the problem of *out-of-band agreements* on transfer destination via its inboxes. In Solid, LDN inboxes can be explicitly advertised via the WebID profile document. As the WebId profile document allows to make statements about the owner of a Solid Pod, we can associate a Solid user identity with the transfer destination by advertising the user's inbox. This means that the minimum required information for a credential transfer is the WebID of the other party: as long as they have set up and advertised their inbox (by setting the `ldp:inbox` triple), notifications can be sent. The LDN inbox acts as receiver for all incoming notifications, handling them as individual entities with their own respective URIs.

[7]https://solid.github.io/webid-profile/#inbox
[8]https://solid.github.io/notifications/ldn-channel-2023

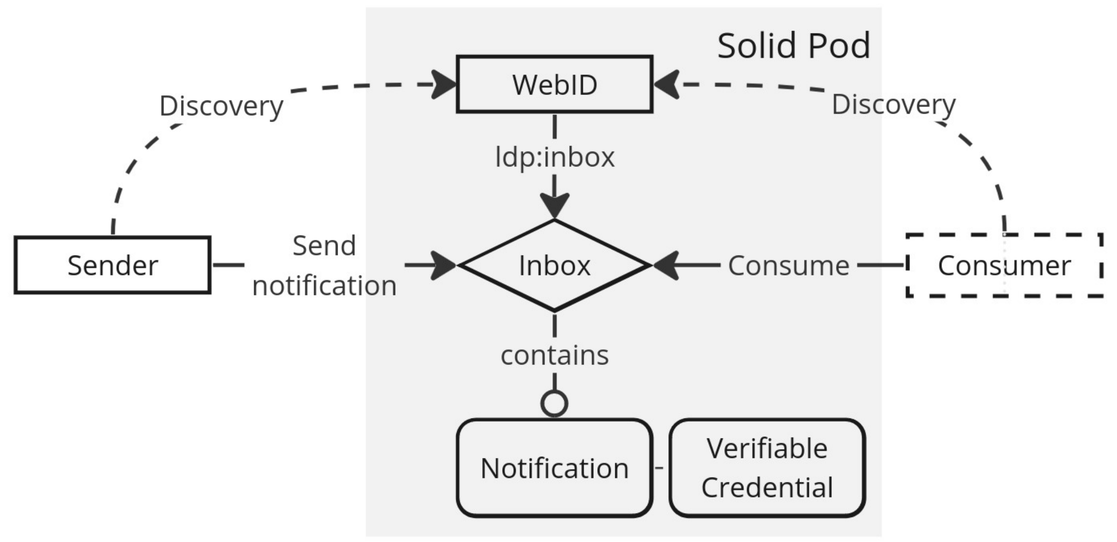


Figure 1: The LDN inbox, discoverable via a link in the WebID profile document, contains Notifications that can be DIDComm envelopes over Verifiable Credentials.

Figure 1 shows an overview of our LDN inbox system. Neither the Sender nor the Consumer need to be using Solid, although the consumption of the inbox (Read, Write, Delete) likely is access controlled[9].

As LDN depends on implicit data structures, interoperable consumption and reuse will be difficult. We therefore build on top of the DIDComm Messaging methodology.

We adapt the DIDComm plaintext message format to form a reusable envelope that wraps around the to-be-transported VC, with the intention of being easy to create, send, and consume on a Solid Pod, in turn enabling common VC ecosystem use-cases, e.g. issuance, holding, and verification. We do so by reusing and introducing a number of DIDComm notification JSON(-LD) headers to structure the notification. Table 1 shows the protocol headers, which can be divided into three categories: Improved inbox handling, data reuse, and data envelope.

By default, handling of the LDN inbox is difficult, as it contains an unstructured collection of resources, and inherently lacks means to identify and find notifications (e.g. notifications get saved with a random UUID as file name). To find a specific notification, inbox consumers need to access and read each notification one-by-one. We reuse DIDComm's *id*, *from*, *to*, and *created_time* to help identify the notification in the inbox and provide inbox consumers with common hooks to look and query for, even though these consumers still need to access and scan each notification in the inbox. The optional *expires_time* can be used to enable automatic cleanup and deletion.

To make the data usable and understandable, we reuse the *type* header to specify the notification type of the credential enveloped in the *body*. The *type* should be descriptive, machine-understandable, actionable, and promote reuse, e.g. by using a shared RDF data vo-

[9] https://solid.github.io/web-access-control-spec/

Table 1
DIDComm protocol headers that are used in the envelope, and their meaning.

Name	Description	Necessity
id	UUID identifying the message.	Required
from	IRI describing the sender.	Required
to	IRI describing the recipient.	Required
type	IRI specifying the notification type or affordance.	Required
created_time	Date when the message was created/sent.	Required
expires_time	Date when the message becomes unusable and should be deleted.	Optional
body	The contents of the message, usually a JSON-serialized VC.	Required

cabulary allowing an inbox consumer to understand how to consume it. For example, the type `https://example.org/IPCEP/transfer` could be used to indicate that this notification is being transferred, and can be consumed as is[10].

4. Implementation

The protocol has been implemented in a research prototype, available on GitHub at https://github.com/SolidLabResearch/inter-pod-credential-exchange-protocol, DOI https://doi.org/10.5281/zenodo.10992060, under the permissive MIT License. It is built on top of the Community Solid Server, a Solid Pod reference implementation[11]. For the issuance, derivation, and verification of the credentials, we use the mature Mattr Linked Data proof suite to create BBS+ Signatures for the VC[12].

Our demo mimics a common VC lifecycle in the Solid ecosystem. It is split into three phases:

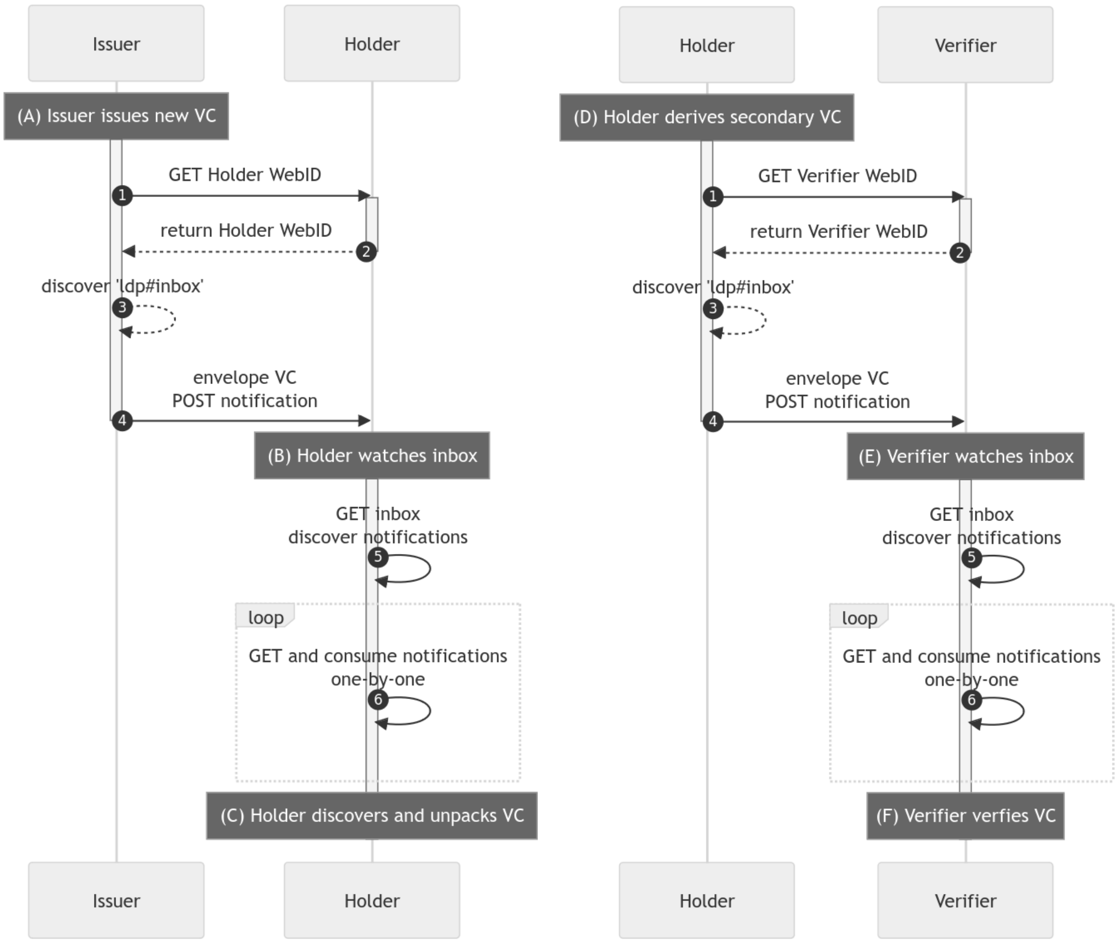
1. The Issuer issues and transfers a new VC to the Holder.
2. The Holder consumes the VC, derives a new VC, and sends it to the Holder
3. The Verifier consumes and verifies the derived VC

For this, three Solid Pods (one for each actor) have been set up, each with a corresponding WebID and LDN inbox. Our proposed exchange protocol is used in the first and second phase to send credentials, and in the second and third phase to receive and consume credentials. The Holder is located in the middle of the flow, with the Issuer and the Verifier not knowing about each other. This is intended to showcase that the protocol is interoperable: sending a message from party A to party B, and then forwarding the same message from party B to party C, with C being able to read and understand it.

To cold-start the demo, we assume that the Issuer already knows the identity of the Holder (WebID), as it is necessary to issue a credential. Similarly, we assume the Holder knows the Verifier (WebID), as they present the claim to them.

Figure 2 shows the demo flow.

[10]Specification of actual type descriptions is out of scope of this paper.
[11]https://github.com/CommunitySolidServer/CommunitySolidServer
[12]https://github.com/mattrglobal/jsonld-signatures-bbs/



(a) Issuer issues and sends new VC (b) Holder derives and sends secondary VC

Figure 2: The credential flow using our proposed protocol. Notifications are uniformly sent across applications and Solid Pods, independent of the role (Issuer, Holder, Verifier) in the use case.

In Figure 2a, the Issuer issues and sends a new VC to the Holder. First, the Issuer issues a new VC (**A**). Then, the Issuer resolves the WebID of the Holder (1-2) to discover the location of the LDN inbox (3). The Issuer envelopes the VC, specifying the correct headers, and POSTs it to the inbox of the Holder (4). The Holder monitors their inbox for changes, by ways not determined in our protocol (**B**). To consume the inbox, the Holder GETs *all* notifications in the inbox (5), and then discovers and consumes them one-by-one (6). As each notification specifies a message type, the Holder can understand if a notification contains a VC, and knows how to unpack it (**C**).

In Figure 2b, the Holder derives and sends a secondary VC to the Verifier. The flow is identical to Figure 2a, except that the Holder derives a VC (**D**), which the Verifier verifies it in the end (**F**).

5. Discussion

LDN proves successful as transport mechanism, allowing us to send notifications from, to, and between Solid Pods. However, neither the setup of the inbox, nor the handling and consumption of it are straightforward.

On the one hand, **setup of the inbox needs to be done manually**, which involves adding the correct triple to the WebID profile document, as well as making sure that the inbox container exists, and that access control (i.e. being readable and writeable) is set. This manual setup is required as Solid Pod implementations, such as the Community Solid Server, do not set up an inbox by default[13]. This is done as safety measure and to combat spam, as a public LDN can be written to by everyone. Hence, our approach does not completely remove implicit agreements: an inbox needs to be (manually) set up.

Also, **notifications get placed in one inbox container without any order**, using a generated UUID as file name. There are no default means for filtering and finding resources. To find a specific notification, we must loop through the whole (unpaginated) notification collection and inspect every message individually.

On the other hand, as the inbox constitutes a central container that stores an unstructured and unprotected collection of notifications, it will be **accessed by different consumers applications that all share the same set of rights to move, change, read or delete notifications**. This makes it possible for a negligent or malicious consumer to move or delete notifications that are otherwise being relied on. This can be partially mitigated with access controls, for example by restricting consumers to read-only, as well as preventing change and deletion of notifications, but usability and privacy concerns remain.

We can summarize above findings into following recommendations for LDN support in Solid: (i) provide an option to enable and disable an (LDN) inbox within pod management software, (ii) provide methods for filtering and pagination of notifications, and (iii) allow for more granular access and usage controls.

We now continue a more high-level discussion on functionality of our proposed approach.

As exchanges can take place across ecosystem boundaries, **we do not allow addressing multiple recipients in the *to* header**. Sending to multiple recipients should be done separately, as otherwise the WebID of every recipient will be disclosed.

To stay extensible and re-usable for a wide range of targeted purposes, **we do not set any rules on how to consume the notification and contained credential**. Instead, our protocol offers the *type* header to specify how to consume the enveloped data, for example through use of semantic vocabularies.

Taking a step back, the LDN design depends on a properly set up inbox (which provides no failsafe except for HTTP error codes), and relies on WebIDs for discovery. Although WebIDs are the default identifier within Solid, they do not yet integrate well with other identifiers used in the the VC ecosystem, such as Decentralized Identifiers[14].

[13] https://github.com/CommunitySolidServer/CommunitySolidServer/issues/515
[14] https://www.w3.org/TR/did-core/

6. Conclusion and Future Work

In this work, we designed an application-level inter-Pod credential exchange protocol and implemented it in a research prototype on top of the Community Solid Server. We apply LDN in combination with DIDComm Messaging primitives to solve the problem of implicit contracts when transferring VCs in the Solid ecosystem.

The combination of LDN and DIDComm Messaging were demonstrated succesfully, but we find that LDN has weaknesses that hamper ease of use, especially filtering and finding notifications in the inbox. An LDN inbox intermediary could help manage the different notifications. Similarly, there is a need for more granular access control for messages, as well as usage policies and logs, to show how notifications are consumed and processed in the LDN inbox.

Looking at our transfer protocol, we currently use a plain-text message body. This suffices for our purposes, and also has the benefit of improved clarity and ease of debugging. Nevertheless, a next step should be secure encapsulation, e.g. encrypted and signed DIDComm Messaging envelopes, to help with verifiability, tamper-evidence, and provenance. Similarly, our protocol could be extended to interoperate with other DIDComm protocols, for example by supporting different identifiers to discover the LDN inbox.

Although being a first step in inter-Pod credential exchange, we believe our protocol has the potential to solve a real need and can be extended to broader requirements, relevant for the GDPR and the DGA [19].

Acknowledgments

The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10), and imec ICON project SHARCS (Agentschap Innoveren en Ondernemen project nr. HBC.2022.0543).

References

[1] J. Theissen-Lipp, M. Kocher, C. Lange, S. Decker, A. Paulus, A. Pomp, E. Curry, Semantics in Dataspaces: Origin and Future Directions, in: Companion Proc. ACM Web Conf. 2023, WWW ’23 Companion, Association for Computing Machinery, New York, NY, USA, 2023, pp. 1504–1507. doi:10.1145/3543873.3587689.
[2] European Comission, A European strategy for data, 2020.
[3] European Union, Regulation 2016/679 (General Data Protection Regulation), 2016.
[4] European Union, Regulation 2022/868 (Data Governance Act), 2022.
[5] S. Van Damme, P. Mechant, E. Vlassenroot, M. Van Compernolle, R. Buyle, D. Bauwens, Towards a Research Agenda for Personal Data Spaces: Synthesis of a Community Driven Process, in: M. Janssen, C. Csáki, I. Lindgren, E. Loukis, U. Melin, G. Viale Pereira, M. P. Rodríguez Bolívar, E. Tambouris (Eds.), Electron. Gov., Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 563–577. doi:10.1007/978-3-031-15086-9_36.

[6] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, T. Berners-Lee, A Demonstration of the Solid Platform for Social Web Applications, Proc. 25th Int. Conf. Companion World Wide Web - WWW 16 Companion (2016) 223–226. doi:10.1145/2872518.2890529.

[7] M. Sporny, D. Longley, D. W. Chadwick, Verifiable Credentials Data Model v1.1 – W3C Recommendation, Technical Report, W3C, 2022.

[8] C. H.-J. Braun, T. Käfer, Attribute-based Access Control on Solid Pods using Privacy-friendly Credentials, in: International Conference on Semantic Systems, 2022.

[9] A. Preukschat, D. Reed, Self-Sovereign Identity, Manning Publications, 2021.

[10] J. Sedlmeir, R. Smethurst, A. Rieger, G. Fridgen, Digital Identities and Verifiable Credentials, Bus Inf Syst Eng 63 (2021) 603–613. doi:10.1007/s12599-021-00722-y.

[11] R. Dedecker, W. Slabbinck, J. Wright, P. Hochstenbach, P. Colpaert, R. Verborgh, What's in a Pod? A knowledge graph interpretation for the Solid ecosystem, in: 6th Workshop Storing Querying Benchmarking Knowl. Graphs QuWeDa ISWC 2022, volume 3279, CEUR, 2022, pp. 81–96.

[12] C. H.-J. Braun, V. Papanchev, T. Käfer, SISSI: An Architecture for Semantic Interoperable Self-Sovereign Identity-based Access Control on the Web, in: Proc. ACM Web Conf. 2023, WWW '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 3011–3021. doi:10.1145/3543507.3583409.

[13] S. Capadisli, A. Guy, C. Lange, S. Auer, A. Sambra, T. Berners-Lee, Linked Data Notifications: A Resource-Centric Communication Protocol, in: E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler, O. Hartig (Eds.), The Semantic Web, volume 10249, Springer International Publishing, Cham, 2017, pp. 537–553. doi:10.1007/978-3-319-58068-5_33.

[14] C. H.-J. Braun, T. Käfer, Web Push Notifications from Solid Pods, in: Web Eng. 22nd Int. Conf. ICWE 2022 Bari Italy July 5–8 2022 Proc., Springer-Verlag, Berlin, Heidelberg, 2022, pp. 487–490. doi:10.1007/978-3-031-09917-5_41.

[15] P. Hochstenbach, R. Verborgh, H. Van De Sompel, Using Event Notifications, Solid and Orchestration for Decentralizing and Decoupling Scholarly Communication (2023).

[16] J.-P. Calbimonte, D. Calvaresi, M. Schumacher, Multi-agent Interactions on the Web Through Linked Data Notifications, in: F. Belardinelli, E. Argente (Eds.), Multi-Agent Systems and Agreement Technologies, volume 10767, Springer International Publishing, Cham, 2018, pp. 44–53. doi:10.1007/978-3-030-01713-2_4.

[17] M. Davie, D. Gisolfi, D. Hardman, J. Jordan, D. O'Donnell, D. Reed, The Trust over IP Stack, IEEE Comm. Stand. Mag. 3 (2019) 46–51. doi:10.1109/MCOMSTD.001.1900029.

[18] S. Curren, T. Looker, O. Terbu, DIDComm Messaging Specification v2.1, Technical Report, 2023.

[19] M. Florea, B. Esteves, Is Automated Consent in Solid GDPR-Compliant? An Approach for Obtaining Valid Consent with the Solid Protocol, Information 14 (2023) 631. doi:10.3390/info14120631.