Delegations in the Solid Dataspace: a proxy for rights delegation

Sebastian Schmid[1], Daniel Schraudner[1] and Andreas Harth[1,2]

[1]Friedrich-Alexander-Universität Erlangen-Nürnberg, Nuremberg, Germany
[2]Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany

Abstract

We focus on the problem how delegations in dataspaces building on Solid are realized. For this, we propose to use the Rights Delegation Proxy that shall ensure on the one hand privacy by keeping delegation details hidden, and on the other hand validate delegated actions against defined policies for legitimacy. We showcase our architecture in a scenario for giving a signature for a loan contract, where a natural person is delegated signature rights on behalf of an enterprise.

1. Introduction

Agents may act on the hand towards their own goals [1], but may also have to act on behalf of others, e.g. like a natural person does on behalf of an organization (e.g. as representative or procurator) or an employee on behalf of their superior (e.g. a given task to fulfill). When rights are transferred from one party to another, we call this a delegation or power of attorney [2, 3].

With identifiable agents coming to the Web thanks to Solid [4], data sharing and the creation of organizations and communications are starting to be well defined. At the same time, the proposed concept of Solid dataspaces (SDS) [5] gains momentum, but here, the aspect of delegation among agents is still open. Especially for organizations with complex structures and hierarchies to thrive in dataspaces, organizations need mechanisms to delegate rights.

Delegations refer to agents, e.g. natural persons or legal entities like companies, and can have complex specifications for defined cases, e.g. in business relations for signing on behalf of a company up to a defined sum of money. We define the following roles and parts of a delegation:

- **Affiliate:** an agent that receives transactions
- **Policy:** defined rights that may be exercised in transactions towards an affiliate
- **Delegate:** an agent that acts based on a policy towards an affiliate
- **Delegator:** an agent that defines a policy for a delegate to act in the delegator's name

An example of a delegation is a company SME (delegator) that grants its employee Alice (delegate) the right to sign contracts on its behalf (policy) with BigBank (affiliate). Considering

The 1st Solid Symposium Poster Session, co-located with the 2nd Solid Symposium, May 02 – 03, 2024, Leuven, Belgium
sebastian.schmid@fau.de (S. Schmid); daniel.schraudner@fau.de (D. Schraudner); andreas.harth@iis.fraunhofer.de (A. Harth)
0000-0002-5836-3029 (S. Schmid); 0000-0002-2660-676X (D. Schraudner); 0000-0002-0702-510X (A. Harth)

the requirements delegators have for a delegation, a delegation should be (1) private, so an affiliate shall not necessarily be informed, whether a delegation has happened and who is the delegate, (2) legitimate, with every taken action by the delegate validated against the delegator's policies, and (3) complete, where every initiated action by a potential delegate shall receive a response.

Approaches exist to bring delegations to the Web, with a public online registry, e.g. the German Commercial Registry[1], where companies as delegators publish information on their delegates, and affiliate look up the information at any time. Closer to dataspaces, an approach for the International Data Space (IDS) [6] proposed the use of Verifiable Credentials [7], where the delegator issues Verifiable Credentials for a delegate's claim to act on behalf of the delegate, with the evaluation of the claim done by the affiliate via cryptographic proof. In both cases the delegate's identity is made known to the affiliate to ensure a legitimate delegation.

To describe data access and sharing in Solid, the most basic way uses access rights for resources based on Access Control Lists[2] (ACL) to ensure that specifically stated agents may use the granted rights. Groups of agents can be described by using vCard groups[3], but the hierarchy stays flat to avoid nesting, while group members need to be shared to check for their membership. A currently developed approaches for Solid is the Solid Application Interoperability[4] that uses an Authorization Agent to manage requests of agents based on ShapeTrees. If an agent's intention has to be captured, Open Digital Rights Language[5] or Policy Language for Solid's Metadata-based Access Control (PLASMA) [8] can be used. Overall, considering privacy issues with General Data Protection Regulation [9], a delegate's identity and the delegation are revealed to an affiliate to set the corresponding ACLs, despite the delegate's potential interest to stay hidden.

As we see the SDS approach as promising to realize dataspaces, as Solid builds on Web standards, is easily adopted, interoperable, and inherently decentralized, we propose to extend Solid's architecture on the dataspace application layer, where currently aspects of certification and policy enforcement are lacking [5]: we propose the Rights Delegation Proxy (RDP) as an approach for private and legitimate data sharing and delegations, where the overall architecture shall be compatible with the architecture of the Web and allow automated validation and execution of actions in a delegation. Our implementation of the RDP is available online [6].

2. Example scenario

Consider the following example with the banking institution BigBank and the enterprise SME, both as legal entities, and Alice, a natural person. The three actors are represented as authenticated Solid agents and the delegation from SME to Alice is realized using the Rights Delegation Proxy. The overall flow of messages is shown in Fig. 1.

[1] https://www.handelsregister.de/
[2] https://solidproject.org/TR/wac
[3] https://www.w3.org/2006/vcard/ns#Group
[4] https://solid.github.io/data-interoperability-panel/specification/
[5] ODRL: https://www.w3.org/TR/odrl-model/
[6] https://github.com/wintechis/delegation-proxy

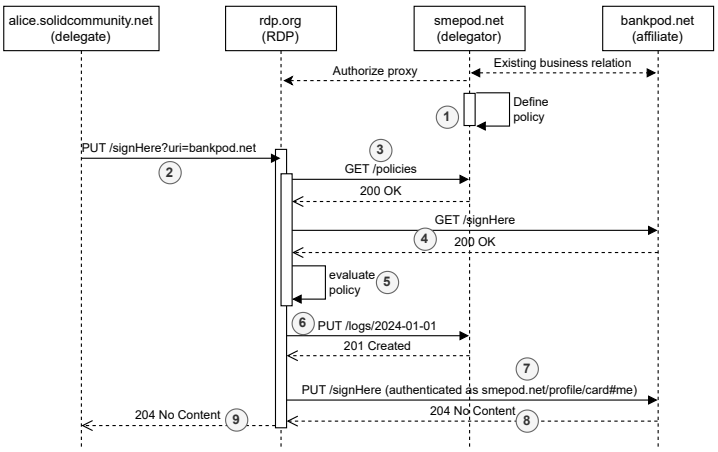

Figure 1: The Rights Delegation Proxy receives a request and checks the policies (steps 1-5). After checking the conditions, the RDP logs and forwards Alice's request, and returns the response (steps 6-9)

Example: SME has to sign a loan offer from BigBank. To receive the signature, BigBank prepared a resource `https://bankpod.net/signHere`. As BigBank expects SME to sign the contract, BigBank created an ACL `signHere.acl` with `acl:Read` and `acl:Write` rights exclusively for SME's WebID `https://smepod.net/profile/card#me`.

SME defines a policy at `https://smepod.net/policies` that Alice, authenticated by her WebID `https://alice.solidcommunity.net/profile/card#me`, has the right to sign loan offers from BigBank (1), where pre- and post-condition are given as Shape Expressions (ShEx)[7] that allow the structural description.

Alice sends an HTTP PUT request to the RDP pointing to `https://bankpod.net/signHere` to give the signature (2). The RDP receives Alice's authenticated request and her WebID. RDP gets (3) the SME's defined policies, executes a preflight request (4) and checks (5) the policies against the result of this request whether Alice has a delegation for `https://bankpod.net/signHere`, and logs the request (6). As Alice fulfills the policy, RDP forwards the request to the specified `https://bankpod.net/signHere` (7), but changes the authentication to SME. At BigBank, the request arrives from SME's WebID. As the ACL is set for SME the signature is created (8) and the response forwarded to Alice (9).

3. Rights Delegation Proxy

We present an overview of the interactions between the Solid agents and the Rights Delegation Proxy during the delegation process. Fig. 2 and 3 show illustrative data used during the delegation with respect to the example scenario in Sec. 2:

1. The delegator defines a policy that states the delegate's rights of transactions towards

[7] https://shex.io/

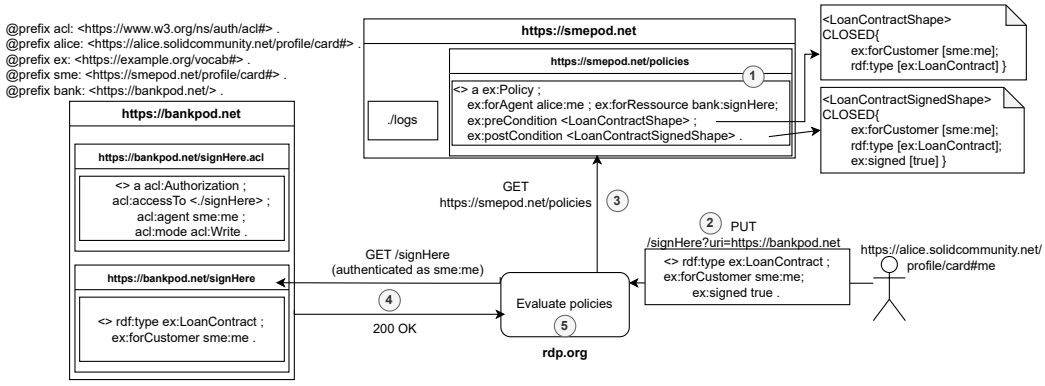

Figure 2: Alice sends a PUT request to the RDP. The RDP retrieves and checks the policies before proceeding (steps 1-5).

the affiliate. We discern pre-conditions and post-conditions that are evaluated, where pre-conditions define how a resource has to look like *before* the delegate may access it, and post-conditions define how a resource has to look like *after* the delegate accessed it.

2. The delegate makes an HTTP request to the RDP, where the accessed path is equal to the resource at the affiliate and the query contains the host of the affiliate's URI.
3. The RDP receives the request, checks that the delegate's WebID is authenticated, and extracts the affiliate's URI as well as the path to the web resource. The RDP looks up suiting policies at the delegator depending on the WebID or web resource.
4. The RDP does a "preflight GET request" to the requested resource.
5. The RDP evaluates if the delegate's request is valid concerning the policies. Therefore the preflight response is checked against the pre-condition. To check a post-condition, the RDP evaluates if the message body, which contains the to-be-expected resource state, adheres to the post-condition. If a condition does not hold, the RDP responds with *403 Forbidden*.
6. After checking the request, the RDP logs the result as well as time, content, accessed resource, and requesting WebID at a location defined by the delegator.
7. The RDP authenticates as delegator to forward the checked request to the resource as specified by the delegate in the query string.
8. The affiliate responds to the RDP, and the RDP logs the response.
9. The RDP sends the affiliate's response to the delegate and concludes the flow of messages for the initiated request.

4. Requirements for roles in delegation

Finally, we analyse the requirements for the roles of affiliate, delegator, and delegate to take part in the RDP system. Note that we do not make assupmtions in our system on how a delegate is informed about the delegation coming from a delegator, or how the range of rights shall be defined between affiliate and delegator.

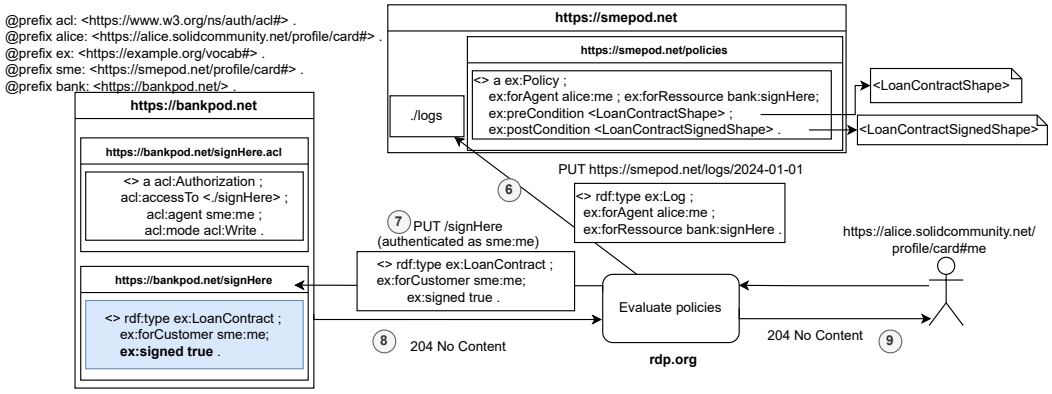

Figure 3: The policies are met, so the RDP forwards Alice's request as SME. After the affiliate applied the request (marked in blue), the RDP sends the response to Alice (steps 6-9).

Affiliate As the affiliate is aware of the existing business relation, the affiliate knows the delegator's identity and thus the WebID. With the WebID, the affiliacte can define an appropriate ACL including permissions for the Web resource on on Solid Storage that shall be accesses by the delegator.

Delegator As the delegator is aware of the business relation with the affiliate, too, the delegator knows the location where the affiliate stores the shared Web resource and ideally the rights that are granted by the affiliate. To define a delegation, the delegator has to know the delegate's Web ID and define policies that associate the delegate with the affiliate, e.g. via explicitly stating the Web resource as permitted for the delegate. The rights granted by the policy may include all or a subset of the delegator's rights with respect to the affiliate - of course rights that go further than the one's granted by the affiliate will not have an effect, if the affiliate did not grant them. To realize the actions, the delegator has to use and authorize an RDP, e.g. hosted as its own service or by a provider, and configure the RDP with the locations on where to find policies and store logs, e.g. the delegator's own Soldi storage.

Delegate As recipient of the delegator's policy, the delegate ideally (but not necessarily) knows which rights are granted by the delegator. Again, if the delegate would try to use rights that go further than the one's defined in the policy, the RDP would refuse to perform the action. To realize the delegation and act on behalf of the delegator, the delegate has to know how and where to act, that is the delegate needs to know the affiliate's Web resource, e.g. via the URI, and the URI of the RDP to realize the action authorized as delegator, so that the action can be executed at the affiliate.

5. Conclusion

We propose the RDP as a medium between the agents delegator, delegate, and affiliate. As all delegated actions go through the RDP, which authenticates as the delegator, the RDP is a

component with high responsibility. As a consequence, we shift the power over actions to the delegator by having exclusive control over policies, while shifting responsibility away from the affiliate, who has to know only the delegator. The delegate powers are limited to exist only in the defined policies, so the RDP solves the privacy problem straightforwardly: with authentication as delegator (like a Solid App), there is no difference of the action's origins towards an affiliate, while the delegate's identity is obfuscated.

As of now, the RDP is a centralized component that manages all incoming delegated requests, which poses a bottleneck and single point of failure [10]. Handling a single request, however, is independent of other occurring requests such that multiple instances of an RDP may be run in parallel. Here, a load balancer may distribute incoming requests to several RDP instances.

Policy implementation is subject to the applied use case, but we see huge potential for complex, custom policies for large organizations to be possible: to evaluate the pre- and post-conditions e.g. ShEx can define the expected structural data of resource, or SPARQL[8] ASK queries can be used instead. If the conditions in the query are met (as with shapes), the delegate request is forwarded. Extension to use more complex workflows are also possible, e.g. to use Business Process Model and Notation (BPMN) to be close to processes in organizations, e.g. contracts may only be signed after an accountant agreed, or ontologies like WiLD [11] to represent and monitor workflows similarly.

Acknowledgments

This work is partially funded by the German Federal Ministry of Education and Research via the MANDAT project (FKZ 16DTM107A).

[8] https://www.w3.org/TR/sparql11-overview/

References

[1] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall Press, USA, 2009.

[2] M. M. Hughes, Remedying financial abuse by agents under a power of attorney for finances, Marquette Elder's Advisor 2 (2012) 39. URL: https://api.semanticscholar.org/CorpusID:37730572.

[3] W. C. Schmidt, Supported decision-making proxy decision-making : A legal perspective, 2015. URL: https://api.semanticscholar.org/CorpusID:53399327.

[4] S. Capadisli, T. Berners-Lee, R. Verborgh, K. Kjernsmo, Solid Protocol, 2021. URL: https://solidproject.org/TR/protocol.

[5] S. Meckler, R. Dorsch, D. Henselmann, A. Harth, The Web and Linked Data as a Solid Foundation for Dataspaces, in: Companion Proceedings of the ACM Web Conference 2023, WWW '23 Companion, Association for Computing Machinery, New York, NY, USA, 2023, p. 1440–1446. URL: https://doi.org/10.1145/3543873.3587616. doi:10.1145/3543873.3587616.

[6] S. Steinbuss, et al., IDS Reference Architecture Model 4. Technical Report, 2024. URL: https://github.com/International-Data-Spaces-Association/IDS-RAM_4_0?tab=readme-ov-file.

[7] H. Meyer zum Felde, M. Kollenstart, T. Bellebaum, S. Dalmolen, G. Brost, Extending actor models in data spaces, in: Companion Proceedings of the ACM Web Conference 2023, WWW '23 Companion, Association for Computing Machinery, New York, NY, USA, 2023, p. 1447–1451. URL: https://doi.org/10.1145/3543873.3587645. doi:10.1145/3543873.3587645.

[8] B. Esteves, H. Pandit, Using patterns to manage governance of solid apps, in: Proceedings of the 14th Workshop on Ontology Design and Patterns (WOP 2023) co-located with the 22nd International Semantic Web Conference (ISWC 2023), CEUR, 2023, p. 43–55. URL: https://ceur-ws.org/Vol-3636/paper5.pdf.

[9] Parliament and Council of the European Union, Regulation (EU) 2016/679 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), 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[10] R. de Lemos, et al., Software Engineering for Self-Adaptive Systems: A Second Research Roadmap, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 1–32.

[11] T. Käfer, A. Harth, Specifying, monitoring, and executing workflows in linked data environments, in: D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, E. Simperl (Eds.), The Semantic Web – ISWC 2018, Springer International Publishing, Cham, 2018, pp. 424–440.