The Rights Delegation Proxy: An Approach for Delegations in the Solid Dataspace

Abstract
We propose the Rights Delegation Proxy to check and execute delegations in dataspaces building on the Social Linked Data project (Solid). The Rights Delegation Proxy ensures privacy by keeping delegation details hidden and validates delegated actions against policies for legitimacy. We show our implemented architecture in a loan contract scenario, where a person signs a contract on behalf of a company with a bank. Additionally, we analyze our architecture for privacy and legitimacy using formal models.

1. Introduction

Agents act in their environments to reach defined goals, often their own [1]. Agents, however, may also have to act on behalf of others, e.g. natural persons on behalf of organizations (e.g. as representative or procurator), employees on behalf of the superiors (e.g. task to fulfill), and colleagues on behalf of their fellows (e.g. as a substitute). We call the transfer of rights and responsibilities from one party to another a delegation or power of attorney [2, 3].

As the Social Linked Data project (Solid) started to bring identifiable agents to the Web [4], the concept of data sharing and the creation of organizations and communications are starting to be well defined. But while the idea of Solid dataspaces (SDS) gains momentum [5], the aspect of delegation among agents is still open. For organizations with hierarchies and sub-organizations to thrive in dataspaces, organizations need rights to be delegated along complex structures.

Delegations refer to agents, e.g. natural persons or whole legal entities like companies, and 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 requirements delegators have for a delegation, we want a delegation to have:

R1 **Privacy**: An affiliate shall not necessarily be informed, whether a delegation has happened and who is the delegate.

R2 **Legitimacy**: Every taken action by the delegate shall be validated against the policies as defined by the delegator and only approved actions may happen.

The 1st Solid Symposium Poster Session, co-located with the 2nd Solid Symposium, May 02 – 03, 2024, Leuven, Belgium

R3 **Completeness**: Every initiated action by a potential delegate shall receive a response.

A delegation process in SDS may use access rights for resources based on Access Control Lists[1] (ACL). While ACLs ensure that specified agents may use the granted rights, only small extensions for more complex agent structures are possible by using vCard groups[2]. With groups, the hierarchy has to stay flat to avoid nesting, while the members of a group need to be shared with external affiliates to check if agents are group members. The ACL approach is even less attractive when considering privacy issues (cf. General Data Protection Regulation [6]): a delegate's identity and the delegation are revealed to an affiliate who has to set the corresponding ACLs, despite the delegate's potential interest to stay hidden. Solid in its current state implements a solution based on ACLs on a resource- and WebID-bound level, which is insufficient for delegations. The possibility to declare rights for groups of WebIDs lacks the sophistication to mirror more complex hierarchies. All in all, current solutions lack an elaborated mechanism to ensure a GDPR-conformant solution that separates the identities of delegator and delegate, as well as the possibility of keeping the delegation itself hidden.

As we see the SDS approach as promising to realize dataspaces at all, because of the SDS foundation on Web standards, easy adoption, interoperability, and inherently decentralized architecture, 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 inherently decentralized style of the Web and allow automated validation and execution of actions in a delegation. Our implementation of the RDP is available online [3].

2. Example scenario

Two legal entities, the banking institution BigBank and the enterprise SME, and the natural person Alice interact, all represented as authenticated Solid agents (see also Fig. 1).

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)[4] 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) and checks (4) the SME's defined policies if Alice has

[1]https://solidproject.org/TR/wac
[2]https://www.w3.org/2006/vcard/ns#Group
[3]https://anonymous.4open.science/r/rights-delegation-proxy/
[4]https://shex.io/

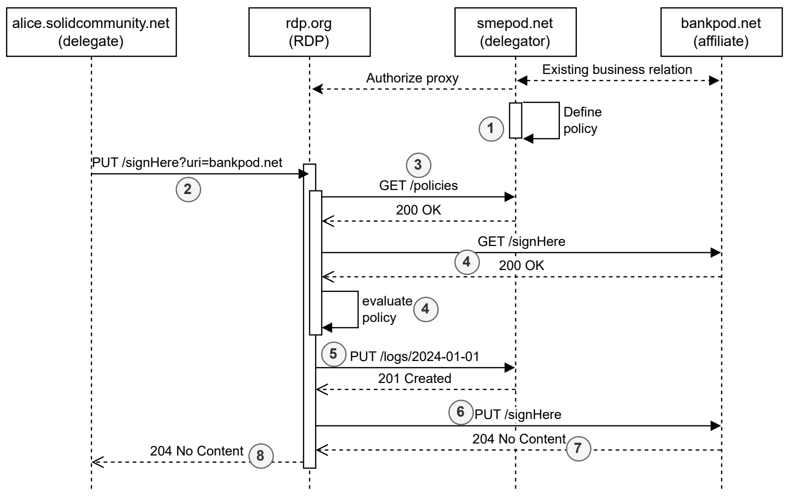

Figure 1: The Rights Delegation Proxy receives a request and checks the policies (steps 1-4). After checking the conditions, the RDP logs and forwards Alice's request, and returns the response (steps 5-8)

a delegation for `https://bankpod.net/signHere`, and logs the request (5). As Alice fulfills the policy, RDP forwards the request to the specified `https://bankpod.net/signHere` (6), 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 (7) and the response forwarded to Alice (8).

In the example, the delegation from SME to Alice is realized using the Rights Delegation Proxy. From BigBank's perspective, only SME is involved, but while the legal body of SME could still not act on its own, Alice initiated the action which resulted in an accepted offer.

3. Rights Delegation Proxy

We present an overview of the interactions between the Solid agents and the Rights Delegation Proxy during the delegation process (cf. Fig. 1 for illustration).

1. The delegator defines a policy that states the delegate's rights of transactions towards 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 evaluates if the delegate's request is valid concerning the policies. To check a pre-condition, the RDP does a "preflight GET request" to the requested resource and evaluates if the response matches 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*.

5. 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.
6. The RDP authenticates as delegator to forward the checked request to the resource as specified by the delegate in the query string.
7. The affiliate responds to the RDP, and the RDP logs the response.
8. The RDP sends the affiliate's response to the delegate and concludes the flow of messages for the initiated request.

We used ProVerif [7], a protocol verification tool with formal models in applied π-calculus, to prove the RDP's correctness with respect to the requirements R1-R3. Our proof is available online [5] where we show that:

- The affiliate will never get to know a delegates's name.
- For all messages a delegate sends to the affiliate, the delegator's policies have been validated.
- For all messages a delegate sends to the affiliate, the delegate will receive a response afterwards.

4. 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 [8]. 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[6] ASK queries can be used instead. If the specified conditions in the query are met (as with shapes), the delegate request is forwarded. As organizations often use Business Process Model and Notation (BPMN) to describe more complex workflows (e.g. a contract may only be signed after an accountant agrees), ontologies like WiLD [9] can be used to represent and monitor workflows similarly and be validated by the RDP with a0 privacy-respecting and secure delegation mechanism that builds directly on existing Web standards and Solid.

[5]https://anonymous.4open.science/r/rights-delegation-proxy/
[6]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] 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.

[7] B. Blanchet, Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif, Foundations and Trends® in Privacy and Security 1 (2016) 1 – 135. URL: https://inria.hal.science/hal-01423760. doi:10.1561/3300000004.

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

[9] 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.