L. Fritsch, H. Roßnagel, D. Hühnlein (Eds.): Open Identity Summit 17,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 1

Harmonizing Delegation Data Formats[1]

Georg Wagner[2], Olamide Omolola[2], Stefan More[2]

Abstract: Delegations are an integral part of daily transactions. A delegation is the process of authorizing one entity to act on behalf of another. For the delegation to work, the claim that an entity is authorized to act on behalf of another entity needs to be verified. Verifying an analog delegation can be done by checking if the name of a user is on a list of authorized persons. In contrast, in electronic transactions the implementation of a delegation and its verification can be a difficult task. A user logging into a system or signing documents with an electronic signature involving a delegation may be required to show this delegation within the signature. This delegation needs to be discovered and verified during the transaction. Many solutions using eXtensible Markup Language (XML) have been proposed to represent delegations; unfortunately, most of these schemes are designed to be used for one specific domain. In this paper, we propose a delegation scheme that fills this gap by providing a general representation for delegation that can be easily extended to different domains.

Keywords: Electronic transactions, eID, Mandates, Delegations, XML

1 Introduction

Delegating responsibilities or capabilities to another entity is part of daily life. A classic example of delegation is the act of making decisions for a sick child by the parent. Delegations are recognized by the law in our daily transactions. A lawyer can be empowered by the client to carry out certain transactions on behalf of the client under certain conditions. The lawyer can then act with the full powers conferred by the client. In times of growing importance of electronic transactions and the widespread use of electronic signatures, neither the role of delegation has changed nor has its importance diminished.

Electronic delegations can be categorized into the following major use case scenarios [Rö09]

- natural person representing another natural person
- natural person representing a legal entity
- legal entity representing a natural person

[1] The LIGHT^{est} project is partially funded by the European Commission as an Innovation Act as part of the Horizon2020 program under grant agreement number 700321.

[2] Technische Universität Graz, Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie, Inffeldgasse 16, 8010 Graz, lightest@iaik.tugraz.at

- legal entity representing another legal entity

Any other kind of use case scenario can be derived from these fundamental scenarios through delegation chaining.

To represent these scenarios electronically poses a challenge. Different institutions and governments have proposed and implemented various ways of representing delegations. As an example, Austria implemented the use of an XML based scheme to handle delegations within the Austrian MOA-ID scheme [Rö09]. Delegations, which are called electronic mandates by the Austrian government, allow an entity to represent another entity in the context of Austrian's MOA-ID scheme. [NC00, ZAC02, Go05] also implemented delegations in the context of access control to resources either in a standalone environment or in a federated environment. Sang et. al [NC00] and Zhang et. al [ZAC02] both proposed a role delegation model which is specifically adapted to the *Role Based Access Control* (RBAC) environment. Gomi et. al [Go05] proposed a delegation framework for federated identity management systems which transfers user privileges by extending Security Assertion Markup Language (SAML). Gomi et. al's work allows users to manage their own privileges with the aid of a delegation authority.

The mentioned delegation models are usually domain specific, therefore it is often a challenge to re-use the model in other scenarios. Within LIGHT^{est} we need to support delegations in different domains. This does not seem to be the case with any of the previously mentioned delegation models. This makes it harder for us to re-use an already existing delegation model within the project. Further, all delegation models use different field names for same functions. We propose an approach to represent delegations in any kind of use-case scenario and in multiple domains. Our representation is not specific to any implementation format and can be used in any domain to the best of our knowledge.

The rest of this paper is structured as follows. Section 2 discusses the different types of delegations that need to be captured. Section 3 describes our proposed representation scheme and how it can be used. In Section 4 we explain how a delegation is created and Section 5 discusses how verification can be done for our representation. Section 6 concludes the paper and discusses future work.

2 Types of Representation

In total three basic types of delegations are of importance. They are the most basic ones, and act as building blocks for complex scenarios and combinations. The three types are (i) bilateral type, (ii) substitution type, and (iii) delegation type.

Delegations often require three roles: Mandator, Proxy, and Intermediary. The Mandator delegates a capability to the Proxy while the Intermediary acts as a link between Mandator and Proxy in case of an indirect representation. With these three roles in place, the types of delegations can be described as follows:

The **bilateral type** is a direct representation between a Mandator and a Proxy as shown in Figure 1. It requires a Mandator and a Proxy. The Mandator empowers the Proxy directly to represent the Mandator.

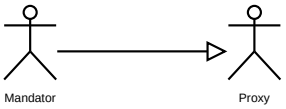

Fig. 1: Bilateral type delegation

The **substitution type** is an indirect representation between a Mandator and a Proxy via an Intermediary, as shown in Figure 2. To create such a type, the Mandator has to empower the Intermediary first. This delegation is a bilateral delegation, with the additional permission from the Mandator that allows the Intermediary to use a substitute. The substitute is then allowed to represent the Mandator. Through substitution the substitute becomes the Proxy.

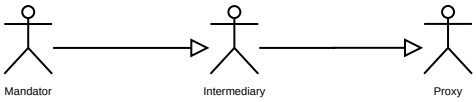

Fig. 2: Substitution type delegation

The **delegation type** is another sort of substitution representation that only differs from the mentioned substitution type in the way a Mandator interacts with the substitute, as shown in Figure 3. The substitute chosen by the Intermediary can become the direct Proxy of the Mandator. The Mandator can interact directly with the substitute without going through the Intermediary again. This is not possible in the substitution type.

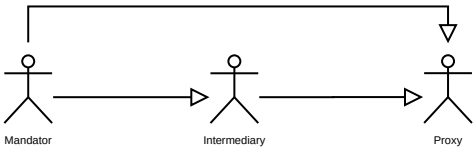

Fig. 3: Delegation type delegation

3 Proposed Representation of Delegations

A problem with current delegation schemes is their domain specific implementation as the delegation schemes provided in [CCJ07], [MW06], and [Rö09] show.

By comparing these different schemes with each other, it can be seen that all delegation schemes separate information in two groups: mandatory and domain specific information. A general representation of these groups is shown in Figure 4.

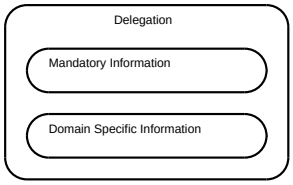

Fig. 4: Generalized delegation scheme

Taking this knowledge into account, we are able to derive the common mandatory information that needs to be contained in a delegation. Those fields are shown in Figure 5. A delegation provides an information block which contains management information. All mandatory information is saved directly in the delegation type. We added the domain specific information in a separate group to create a separation from the mandatory information. The data held by mandatory fields of a delegation are described as follows:

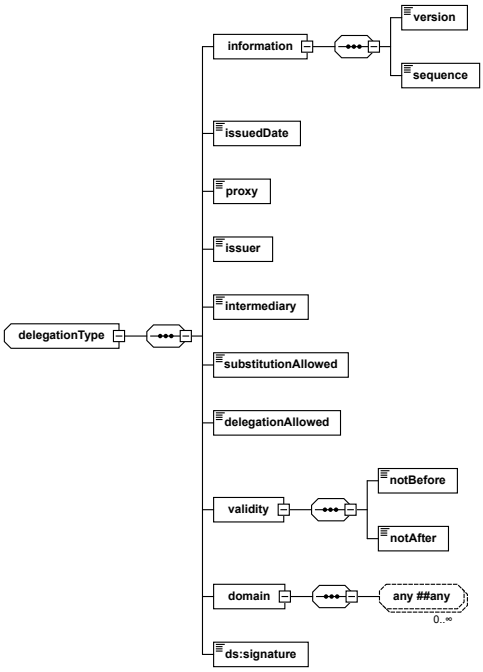

Fig. 5: Fields required for a delegation

information: The information group contains general information about the delegation. The group contains two fields: the file format *version* and the delegations *sequence number*.

version: Contains the scheme version identification of the file. The version identification has the format *major.revision*. Major and revision are both integer values. The version identification is currently set to 1.0 and may change with revision of the file structure.

sequence: Contains the sequence number of the delegation. This number increases with every renewal of the delegation.

issuedDate: Date and time when the delegation was issued. It is encoded as *dateTime*.

proxy: Contains the proxy's verification information. This information is implementation dependent. It can be a fingerprint of the proxy's signature certificate together with an Uniform Resource Locator (URL) at which the fingerprint can be verified, an URL where the name of the proxy can be verified, etc.

issuer: Contains the issuer's verification information. This information is implementation dependent. It can be a fingerprint of the issuer's signature certificate, an URL where the issuer can be verified, etc.

intermediary: Contains the intermediary's verification information. This information is implementation dependent. It can be a fingerprint of the intermediary's signature certificate, an URL where the issuer can be verified, etc.

substitutionAllowed: Contains information if a substitution type is allowed. The field is encoded as *boolean* value.

delegationAllowed: Contains information if a delegation type is allowed. The field is encoded as *boolean* value.

validity: The validity group contains information in which time-period the delegation is valid. For this purpose it contains the date and time the delegation becomes valid, and it also contains the date and time when the delegation expires.

notBefore: Date and time the delegation becomes valid. The field is encoded as *dateTime*.

notAfter: Date and time of the expiration of the delegation. The field is encoded as *dateTime*.

domain: This group contains a list of domain specific fields. To identify the given domain the delegation is operated under, the two attributes name and version are used. They define which domain specific fields are used in this group.

ds:signature: Signature of the delegation by the issuer. This is used to verify the correctness of the data provided in this document and as a proof that the data has not been tampered with.

3.1 Example Delegation Representation

In this example, Alice and Bob are in a business relation. Alice wants Bob to represent her and act as her proxy in case of purchasing. Bob is not allowed to choose a substitution for himself, as Alice only allows this bilateral delegation. Bob is only allowed to act on behalf of Alice during June 15th 2017 and October 6th 2017. The delegation is created by Alice on May 05th 2017 at 23:59:59. The following listing shows this example of a bilateral type delegation represented in XML. XML is chosen, as the LIGHT^{est} project is going to use XML to represent delegations. It should be easy to adopt this to any other data representation format, e.g. JSON. The domain is the purchasing domain, which is not shown here as the fields are not part of this proposal and will be created within the LIGHT^{est} project. In the LIGHT^{est} project a delegation will be created and thus signed by the Delegation Provider.

```
<?xml version="1.0"?>
<delegation version="1.0">

    <!-- Mandatory Information -->
    <issuedDate> 2017-05-14T23:59:59 </issuedDate>
    <proxy> Bob </proxy>
    <issuer> Alice </issuer>
    <intermediary />
    <substitutionAllowed>false</substitutionAllowed>
    <delegationAllowed>false</delegationAllowed>
    <validity>
        <notBefore> 2017-06-15T00:00:00 </notBefore>
        <notAfter> 2017-10-06T23:59:59 </notAfter>
    </validity>

    <domain name="purchase" version="0">
       ...
    </domain>

    <ds:signature>
    </ds:signature>

</delegation>
```

4 Creating a delegation

In the context of LIGHT^{est} the creation of a delegation is done using the delegation publisher software [BL16]. The software provides an interface to fill in the different fields such as the proxy, issuer, substitutionAllowed, delegationAllowed. In addition, the validity and

domain specific fields are filled in. The information group and the issuedDate are filled automatically by software. The creation of the different types of delegations differ because of the difference in actors. The different processes are described in subsequent paragraphs.

Bilateral delegations are quite trivial to create compared to the other types of delegations. To create a bilateral delegation, the user fills in the proxy's verification information, the issuer field; the user sets the substitutionAllowed field and the delegationAllowed field to false. The user also specifies the validity information and adds other domain specific fields if necessary.

Substitution type delegations are usually created in two main steps. The first step is to create a bilateral delegation from the mandator to the intermediary with the only difference that the substitutionAllowed field is set to true. In addition the intermediary chooses a substitute, the later proxy, for this delegation.

Delegation type can be created with two approaches. In the first approach, the mandator creates a bilateral delegation to the intermediary. However, the proxy field is left blank while adding the verification information of the intermediary to the intermediary field, as well as the field delegationAllowed is set to true. Next the intermediary needs to inform the mandator that the intermediary has chosen a substitute for himself and that the intermediary has created a delegation from the intermediary to the final proxy. Once the intermediary informs the mandator about this, the mandator then updates the data he has to reflect the information of the proxy and create the certificate for the proxy. This certificate is then issued to both the intermediary and the proxy. The second approach is almost the same as the first but in this approach, the mandator waits for the intermediary to first create the delegation to the proxy before the mandator then creates the delegation to the proxy while filling in all the fields as in the first approach.

5 Verifying a Delegation

The verification process checks the data used in the delegation for authenticity. The user signing a document claims that he is empowered to do so. Without a verification of this claim, it cannot be confirmed that the delegation is allowed or authorized. In the context of LIGHT^{est}, we only have a link to the delegation. In order to verify that this link is correct, and that the answer is coming from the correct server, DNS with DNSSEC extension is used [BL16].

We show how the claim of a delegation can be verified. To do so, we assume a signed document, a verifier that can find a signature and detect that this signature has been created by a proxy. The verifier then extracts the information from the signature and finds the link where the delegation is published. The extracted address is queried and the answering DNS servers DNSSEC key is verified for correctness. Now that the information about the delegation is available at the verifiers side, the verifier has to verify the delegation.

With the available information, the verifier has to determine which type of delegation is claimed. The type can be one of the three types described in Section 2. To determine the type of delegation the verifier checks if the intermediary field is empty. If it is empty the type of the delegation can only be bilateral. In any other case it must be either a substitution or delegation type. Both substitution or delegation type are very similar to each other, thus the substitutionAllowed and delegationAllowed fields are evaluated. With these properties we can identify the type of delegation.

The first step of the verification checks the signature on the delegation and if the data has not been tampered with. If somebody has been tampering with the data, all other checks are useless as the data has been corrupted. This check is the same for all types of delegation, regardless if it is bilateral, substitution, or delegation.

Bilateral: For a bilateral type delegation, the data of the proxy needs to be verified. The verifier must know if the mandator has issued a delegation to the proxy. Therefore, the signature fingerprint saved in the proxy field is checked against the signature of the document. Further, all other parameters relevant (notBefore, notAfter, etc.) as well as the domain specific parameters are checked in order to make the decision if the delegation is valid.

Substitution: For a substitution type delegation, the data of the proxy and the intermediary needs to be verified. The verifier first queries the substitutionAllowed field. If this field is true, the verifier queries the issuers DNS to find out the intermediary of this delegation and compares it to the information saved in the intermediary field. If the issuer has given a delegation to the intermediary, then the information about the intermediary would match. Next the intermediaries DNS is queried to find out the proxy. Further, all other parameters relevant are checked to make the decision if the delegation is correct.

Delegation: The delegation type is verified in the same way as the bilateral delegation, with the only difference that the intermediary field is filled out and the delegationAllowed is set to true. The intermediary field is filled out during the creation of the delegation as this type starts as an substitution type delegation, where the intermediary can choose an substitute for himself. Therefore, this field is filled. This field can later also be used for logging purposes and to show that the intermediary has chosen the proxy as substitute.

6 Conclusion and Future Work

In this paper, we presented a novel way of representing delegations based on the aggregation of other previous delegation schemes. Our representation is intuitive and also powerful enough to be used in different domains. Furthermore, the representation can be used to reflect the three main types of delegations i.e., bilateral, substitution and delegation type which is often overlooked in the previous works.

This representation will be used in the LIGHT^{est} project to represent delegations and as the main information source when publishing and querying delegations. We plan to combine

this representation together with records stored in DNS, pointing to the location of the delegation. This method should allow everyone to publish own delegations. In the future, we intend to create tools that create delegations using this representation. We further plan to provide the domain specific data to make use of the domain field within the delegation representation.

References

[BL16] Bruegger, B. P.; Lipp, P.: LIGHT^{est} - A Lightweight Infrastructure for Global Heterogeneous Trust Management. In: Open Identity Summit 2016, 13.-14. October 2016, Rome, Italy. pp. 15–26, 2016.

[CCJ07] Cho, Y.; Cho, S.; Jin, S.: A Unified User Consent Acquisition and Delivery Mechanism for Multi-Source User Data Integrated Service. In: 2007 IEEE International Symposium on Consumer Electronics. IEEE, jun 2007.

[Go05] Gomi, H.; Hatakeyama, M.; Hosono, S.; Fujita, S.: A delegation framework for federated identity management. In: Proceedings of the 2005 workshop on Digital identity management - DIM '05. ACM Press, 2005.

[MW06] Ma, M.; Woodhead, S.: Authentication delegation for subscription-based remote network services. Computers & Security, 25(5):371–378, jul 2006.

[NC00] Na, S.; Cheon, S.: Role delegation in role-based access control. In: Proceedings of the fifth ACM workshop on Role-based access control - RBAC '00. ACM Press, 2000.

[Rö09] Rössler, T.: Empowerment through Electronic Mandates – Best Practice Austria. In: Software Services for e-Business and e-Society, pp. 148–160. Springer Berlin Heidelberg, 2009.

[ZAC02] Zhang, L.; Ahn, G.; Chu, B.: A role-based delegation framework for healthcare information systems. In: Proceedings of the seventh ACM symposium on Access control models and technologies - SACMAT '02. ACM Press, 2002.