A Delegation Framework for Liberty

Waleed A. Alrodhan
Information Security Group
Royal Holloway
University of London
Egham, Surrey, TW20 0EX, United Kingdom
Email: W.A.Alrodhan@rhul.ac.uk

Chris J. Mitchell
Information Security Group
Royal Holloway
University of London
Egham, Surrey, TW20 0EX, United Kingdom
Email: C.Mitchell@rhul.ac.uk

***Abstract*—Building support for delegation services into an identity federation system enhances its flexibility and scalability. Users may need to delegate all (or a subset) of their access rights or privileges to other parties in the system. However, the Liberty Alliance, an industry consortium that aims to build open standard-based specifications for identity federation systems, does not include delegation functionality in its specifications. In this paper we propose a delegation framework for Liberty that can be readily integrated into the currently deployed specifications. The framework takes advantage of the trust relationships that exist by definition within the Liberty circles of trust, and is based on extending the use of attribute statements in SAML assertions. The framework is built on SAML 2.0 and the Liberty ID-FF 1.2 single sign-on profiles, and supports both direct and indirect delegation.**

I. Introduction

Identity federation systems are based on the notion of sharing authentication information between an identity provider and service providers within a certain circle of trust. The access rights given to a particular user by a service provider (SP) are determined according to the shared authentication information. Providing support for delegation services in an identity federation system enhances its flexibility and scalability, and builds upon the trust relationship between the identity provider and the service provider. Moreover, most identity federation systems provide reliable approaches for transforming sensitive information about the system users (e.g. using SAML, SOAP and WS-*). These approaches can be used to help support delegation.

The Liberty Alliance Project[1] (henceforth abbreviated to Liberty) is one of the most prominent collaborations (started in December 2001) aiming to build open standard-based specifications for identity federation systems. The Liberty specifications have been adopted by many identity management product vendors, including Sun[2] and Ping Identity[3].

Whilst some identity federation systems, such as Shibboleth[4], support delegation services [4], the Liberty specifications lack delegation support. However, the Liberty frameworks ID-FF and ID-WSF [13], [15] provide a solid basis on which to build customised delegation services. In this paper we extend the use of SAML, an XML-based standard for security assertion specification and exchange, to support delegation services.

The remainder of this paper is organised as follows. Section II provides an overview of delegation and its advantages. In section III we propose a delegation framework for Liberty, and in section IV we discuss possible issues with the proposed framework. Section V highlights related work, and section VI concludes the paper.

II. An Overview of Delegation

In this section we provide a brief introduction to the concept of delegation, and discuss the significance of designing a delegation model for Liberty.

A. Introduction to Delegation

Delegation enables an entity to delegate some or all of its rights to other entities in a specific domain [2]. The notion of delegation is widely used as an effective access control method. For example, many Digital Rights Management (DRM) systems offer delegation services to consumers so that they can delegate their access rights to a protected piece of media to a number of devices. Grid systems provide another example where delegation is widely used; many such systems adopt delegation frameworks to enhance efficiency and scalability [1].

The following definitions for delegation-related terminology are used in this paper:

- **Privilege:** A right to access specific resources or to perform certain tasks. A user may have a number of such privileges.
- **Delegation:** The act of (temporarily or permanently) transferring privileges from one entity to another.
- **Delegator:** The entity that transfers (delegates) all or a subset of its privileges to a Delegatee.
- **Delegatee:** The entity that receives all or a subset of the Delegator's privileges in order to use them on the Delegator's behalf.
- **Delegation Assertion:** An assertion of the correctness and authority for a delegation, issued by a Delegation Authority to a Delegatee.
- **Delegation Authority:** An entity that controls delegation and issues delegation assertions.

[1] http://www.projectliberty.org
[2] http://www.sun.com/software/products/identity/standards/liberty.xml
[3] http://www.pingidentity.com
[4] http://shibboleth.internet2.edu

There are two widely discussed models for delegation. The first is the *direct delegation* model, in which a delegator directly delegates all (or a subset) of its privileges to a delegatee, which uses the delegated privileges to perform specific tasks or retrieve information. The second is the *indirect delegation* model, in which a delegator indirectly delegates all (or a subset) of its privileges to a delegatee through one or more agents. In this latter model there will be more than one delegation step before the delegatee can use the delegated privileges. Figures 1 and 2 show these two delegation models.

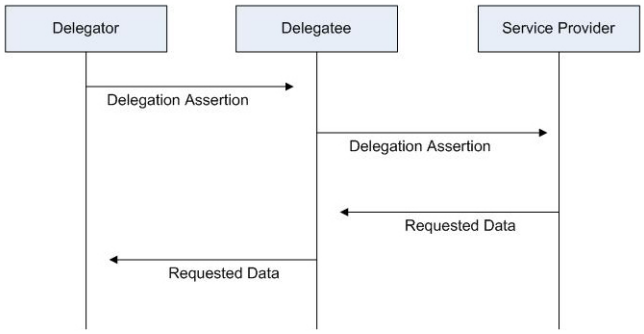

Fig. 1. Direct delegation model

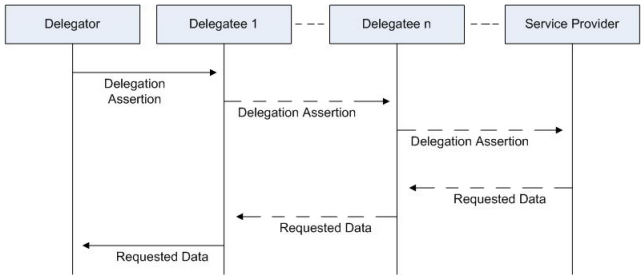

Fig. 2. Indirect delegation model

It merits mentioning here that delegation does not imply authorisation. That is, if the service provider accepts the delegation act (i.e. accepts the delegation assertion), it does not have to grant the requested privileges to the delegatee. It is always up to the service provider to decide whether or not to accept the access requests initiated by the delegatee.

The delegation assertion must explicitly prove the delegator's consent to the delegation. The delegator may also want to impose certain conditions on the delegation (e.g. the delegation validity period, whether or not the delegation assertion is re-delegatable to another entity, the type of information that can be retrieved, etc.); these conditions must be stated in the delegation assertion.

Finally, when required, preserving privacy is a very important issue that needs to be addressed, without affecting the performance or the flexibility of the model.

B. Delegation and Liberty

Although Liberty does not specify a delegation model, supporting delegation would potentially enhance systems adopting the Liberty specifications. As an example, we consider the hypothetical Liberty framework shown in figure 3, in which there are two SPs, an estate agent and a bank, and a single identity provider (IdP) that has a federation agreement with both SPs. Suppose that a user who has accounts with both SPs and with the IdP (and her/his SPs accounts are federated with her/his IdP account), wants to make an offer to purchase a house advertised by the estate agent (SP1) website. In order to approve the proposed purchase, the estate agent must check the financial status of the user to determine whether or not she/he can afford to buy the selected house. Since the website of the bank that holds the user's account is in a circle of trust that includes the estate agent website, both SPs (i.e. the estate agent and the bank) trust the same IdP.

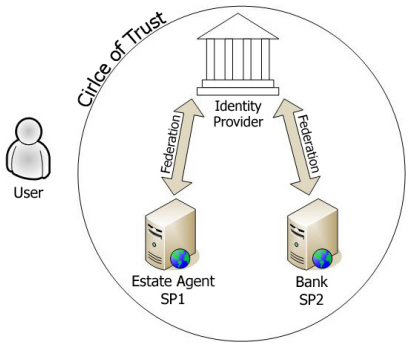

Fig. 3. Liberty model

When using the currently deployed Liberty specifications, which do not support delegation, the only way for the estate agent to get hold of the financial status information of the user, without asking the user to present her/his financial status herself/himself, is for the estate agent to have a predefined trust relationship with the bank; however, this is not a realistic assumption, since each bank may have to deal with a large number of estate agents. However, banks can set up trust relationships with a (relatively small) number of IdPs, that between them have relationships with a large range of estate agents.

Life would be much easier if a delegation system was available, in which the IdP plays the role of a delegation authority. The user then can simply delegate to the estate agent the ability to get information about her/his financial status from the bank. If this delegation is approved by the IdP, then the bank would accept it, since it trusts the IdP for the purpose of authentication decisions.

The infrastructure to build a delegation framework for Liberty is already in place; for example, Liberty supports a number of data transport and security protocols such as SAML, SOAP and WS-Security. These techniques can be used to transport sensitive information for delegation purposes (notably delegation assertions). However, designing a delegation framework for Liberty is not a straightforward task; many issues need to be addressed, such as representing user consent, handling pseudonyms (i.e. opaque user identifiers), and preserving user privacy. These issues are addressed in the

scheme described in section III.

III. A Delegation Framework for Liberty

In this section we provide a brief introduction to the security assertion markup language (SAML). We then describe the form of the delegation assertion used in the proposed delegation framework. Finally, we propose a delegation framework for Liberty.

A. The Security Assertion Markup Language (SAML)

The security assertion markup language (SAML) is an XML-based standard for exchanging authentication and/or authorisation information between network entities [3].

A *SAML assertion*, as defined in [3], can carry three types of security information (or statements):

1) **Authentication statement**: this indicates whether or not the user has been authenticated, and, if so, it specifies the authentication method used (e.g. password) and the time of the authentication.
2) **Attribute statement**: this contains information about the user (e.g. first name, email address, etc.).
3) **Authorization decision statement**: this contains a recommended access control decision (i.e. whether or not the user should be allowed to access a given resource).

SAML assertion requests and responses need to be mapped to communications protocols for transmission; this process is known as *binding*. The most commonly discussed binding options are to transmit SAML messages over HTTP or SOAP [11]. SAML messages can be protected using the WS-Security protocol, which provides message integrity using XML Signature [6] and data confidentiality using XML Encryption [7] (the use of XML Encryption can be omitted if the messages are carried over a secure channel such as is provided by SSL/TLS or IPSec).

The latest version of SAML is SAML 2.0 [3], which is incompatible in many respects with its predecessor SAML 1.1 [10]. The Liberty specifications have supported SAML 2.0 since 2005.

B. The Delegation Assertion

Before we present the proposed delegation framework, we must describe the structure of the delegation assertion we propose for use in the framework. As stated earlier, our solution is based on extending the attribute statement in the SAML assertion to build a delegation assertion. Although we are using SAML 2.0 to build the delegation assertion, it is also possible to use SAML 1.* to build a similar assertion, although this would require some minor changes to the syntax.

The delegation assertion is issued and signed by the IdP, and should not be issued without explicit consent from the user (or the privilege owner) for the delegation. This consent can be given *online*, by asking the user directly at the time of delegation, or it can be given in advance or *offline* (e.g. by telling the IdP in advance that any delegation request made at a particular time with particular delegatees should always be approved). Since the user ID is included within the assertion, the IdP's signature on the assertion implicitly implies user consent.

As shown in figure 5, in Liberty the IdP and the SPs use different *pseudonyms* (also known as *opaque handles*) for the same user. These pseudonyms are short-term identifiers, used in order to provide unlinkability and thus preserve user privacy. For example, when referring to a user named *Alice*, the IdP would use one pseudonym (xxx, say) when it negotiates with service provider SP1, and a different pseudonym (yyy, say) when it negotiates with another service provider SP2. Moreover, each pair of entities may use two different pseudonyms when referring to the same user; for example, if an IdP uses pseudonym yyy for a user when negotiating with SP2, then SP2 may use a different pseudonym ($y123$, say) for this user when negotiating with the IdP.

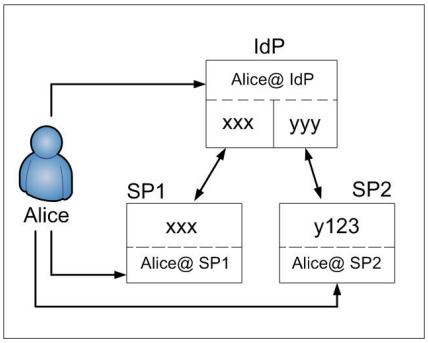

Fig. 4. Pseudonyms in Liberty

Obviously, the IdP will have established pseudonyms for a user with an SP if that user has federated its SP identity to that IdP. The framework requires the user (the delegator) to have federated the delegation assertion target (i.e. the SP that holds the resource to which access is to be delegated) to the IdP that is acting as a Delegation Authority. Hence this IdP will share pseudonyms for the user with the delegation assertion target. To provide a degree of anonymity, the ID of the user included within the assertion is the pseudonym that is understood by the delegation assertion target. This pseudonym is encrypted using the delegation assertion target's public key.

In order for the delegatee to demonstrate its rightful possession of the delegation assertion, SAML 2.0 offers three methods called *confirmation methods* (all of which are also supported by SAML 1.* [12]). We propose here use of the confirmation method known as *holder-of-key*. This method requires the delegatee to include a signature for certain data in the delegation assertion request that it sends to the delegation authority (i.e. the IdP). This signature is verified by the IdP using the delegatee's public key (use of an XML Signature is recommended). The delegatee's signature and public key information are then included in the delegation assertion, to prove that this delegatee is the rightful possessor of the assertion.

Figure 6 shows the structure of the delegation assertion. We extend the `<AttributeStatement>` to include new tags for the purpose of delegation. Given below is a description of

the contents of a delegation assertion.

- <Issuer> : The URI of the IdP that issued the assertion.
- <ds: Signature> : The IdP's signature on the entire assertion.
- <Subject> : This contains the URI of the delegatee, the confirmation method to be used (which in our framework must be holder-of-key) and the confirmation data. It also contains information about the delegatee's key that can be used to verify the confirmation data.
- <Conditions> : This contains constraints that apply to the delegation (e.g. the validity duration of the assertion, and/or the URI of the entity to receive the assertion). In SAML 2.0, this item can be extended to include any required conditions.
- <AttributeStatement> : This includes information about the delegation act in the form of assertion attributes. The defined attributes are:
  - <Delegator> : The pseudonym of the user (or the privilege owner) that is agreed upon between the IdP and the target SP. This attribute is encrypted using the target SP's public key.
  - <Delegatable> : A boolean value that indicates whether or not the assertion is delegatable (i.e. indirect delegation is allowed).
  - <Resource> : The URI where the requested resource is located.
  - <Role> : If the user has multiple access control roles at the target SP system, she/he can use this attribute to specify which role is to be delegated to the delegatee. This attribute is optional.
  - <Description> : A description of the purpose of the delegation. This attribute is optional.
  - <OutputData> : This attribute holds two values. The first is a boolean value that indicates whether or not the result data should be encrypted using the delegatee's public key, and the second value states the type of the result data using WSDL (Web Services Description Language) [5].

An equivalent *Document Type Definition* (or DTD) for the extended <AttributeStatement> would be:

```
  <!DOCTYPE AttributeStatement [
  <!ELEMENT AttributeStatement (Delegator,
Delegatable, Resource,
  Description, Role, OutputData)>
  <!ELEMENT Delegator (♯PCDATA)>
  <!ELEMENT Delegatable EMPTY>
  <!ATTLIST Delegatable Indirect (True|False)
"False">
  <!ELEMENT Resource (♯PCDATA)>
  <!ELEMENT Description (♯PCDATA)>
  <!ELEMENT Role (♯PCDATA)>
  <!ELEMENT OutputData (♯PCDATA)>
  <!ATTLIST OutputData Encrypted (True|False)
"False"> ]>
```

Figure 7 contains an example of a delegation assertion.

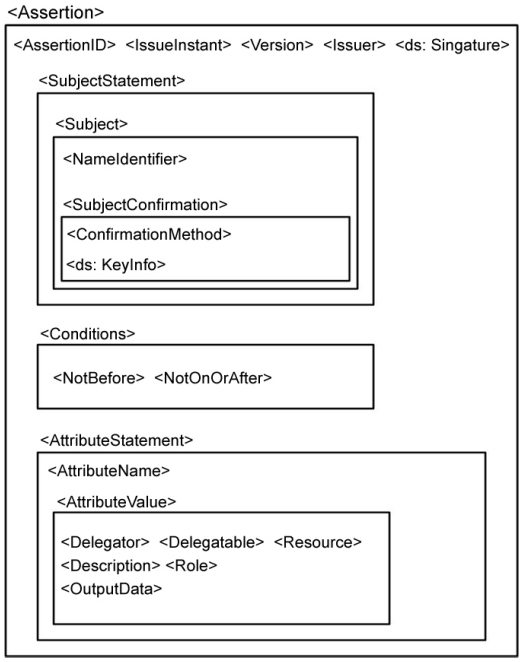

Fig. 5. Structure of the delegation assertion

C. The Delegation Framework

The proposed delegation framework is based on the *Single Sign-On Profiles* described in the Liberty ID-FF 1.2 specifications and adopted by SAML 2.0 [9], [15]. The three relevant profiles are the *artifact* profile, the *POST* profile and the *enabled-client* profile. In this section we describe three variants of the delegation framework for Liberty modelled on these three profiles. We emphasise here that all the message transfers in the three profiles must be carried over a secure channel (e.g. as provided by SSL/TLS, IPSec, SSH, etc.).

1) Delegation Framework Based on the Artifact Profile: The message flow is summarised in figure 8, and involves the following steps:

1) The user requests a service from SP1's web site via a user agent (i.e. a web-browser); the user request includes a delegation request to permit SP1 to obtain certain data from SP2 on the user's behalf (where the user has a pre-existing relationship with SP2).
2) SP1 redirects the user agent to the the IdP's web site, along with a delegation assertion request embedded within an HTML get or post command. This request should include information about the delegatee (i.e. SP1), the target SP (i.e. SP2) and the confirmation data to be included in the token. After receiving the delegation assertion request, the IdP checks whether it has a federation relationship with the target SP for this user. If so, then the IdP checks whether the user has already provided consent to such a delegation (i.e. offline consent). If the user has not, the IdP must, by some means (e.g. an interactive dialogue box), ask the user for consent to the delegation, and whether or not the user

```
<Assertion ID="1234abcd" IssueInstant="2007-07-01T00:22:02Z“ Version="2.0">
<Issuer> .. The URI of the IdP .. </Issuer>
<ds:Signature> .. The IdP's Singature .. </ds:Signature>

<Subject>
<NameID> .. The URI of the Delegatee .. </NameID>
<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
<ds:KeyInfo> .. The Delegatee's key information .. </ds:KeyInfo>
.. Confirmation Data ..
</SubjectConfirmationData>
</SubjectConfirmation>
</Subject>

<Conditions NotBefore="2007-07-05T08:20:05Z"
NotOnOrAfter="2007-07-05T08:24:05Z">
<AudienceRestriction>
<Audience> .. The URI of the assertion receiver .. </Audience>
</AudienceRestriction>
</Conditions>

<AttributeStatement>
<Attribute FriendlyName="Delegator">
<AttributeValue> .. Encrypted pseudonym of the delegator .. </AttributeValue>
</Attribute>
<Attribute FriendlyName="Delegatable">
<AttributeValue>
<Indirect> .. True or false .. </Indirect>
</AttributeValue>
</Attribute>
<Attribute FriendlyName="Resource">
<AttributeValue> .. The URI where the requested resource is located ..</AttributeValue>
</Attribute>
<Attribute FriendlyName="Role">
<AttributeValue> .. The role of the principal .. </AttributeValue>
</Attribute>
<Attribute FriendlyName="Description">
<AttributeValue> .. Optional description of the delegation purpose .. </AttributeValue>
</Attribute>
<Attribute FriendlyName="OutputData">
<AttributeValue>
<Encrypted> .. True or false .. </Encrypted>
.. Type of the result data ..
</AttributeValue>
</Attribute>
</AttributeStatement>
</Assertion>
```

Fig. 6. Example of delegation assertion

wishes the result to be encrypted. Once user consent has been obtained, the IdP creates a delegation assertion and give it a “reference” string (known as a SAML artifact).

3) The IdP directs the user agent to SP1’s web site, along with the assertion reference embedded within an HTML get or post command.
4) SP1 sends a SAML assertion request that includes the reference received in step 3.
5) The IdP sends the delegation assertion to SP1.
6) SP1 forwards the delegation assertion to the target SP (i.e. SP2). Steps 4, 5 and 6 are conveyed over SOAP. If the delegation is approved by SP2, and the delegator (i.e. the user) is authorised to access the requested data (or perform the requested task), then SP2 proceeds with the requested action.
7) SP2 sends the resulting data to SP1 (SP2 will first encrypt the resulting data using SP1’s public key, if requested to do so in the delegation assertion).
8) SP1 grants the user the requested service (or sends the requested data to the user).

2) Delegation Framework Based on the POST Profile: The message flow is summarised in figure 9. In the POST profile there is no direct communication between the delegatee (i.e. SP1) and the delegation authority (i.e. the IdP).

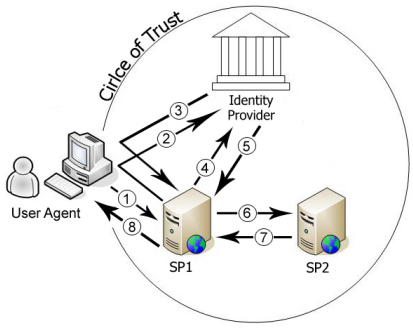

Fig. 7. Delegation framework based on the artifact profile

The first two steps of this profile are identical to the first two steps of the previous profile. However, in step 3, instead of sending an assertion reference to SP1 (via the user agent), the IdP sends the delegation assertion itself to SP1, after embedding it within an HTML form and redirecting the user agent to SP1’s URI. Steps 4, 5 and 6 are identical to steps 6, 7 and 8 of the previous profile.

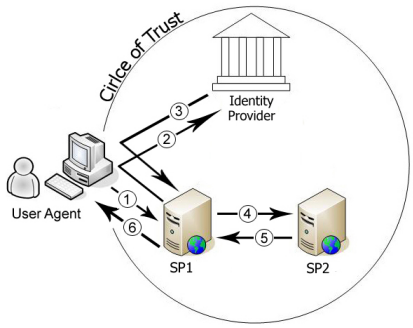

Fig. 8. Delegation framework based on the POST profile

3) Delegation Framework Based on the enabled-client Profile: The message flow is summarised in figure 10. This profile requires the user agent to be SAML-enabled (i.e. to be capable of understanding and processing SAML assertion requests and responses); this can be achieved by installing appropriate software components on the user machine.

The first step is similar to the first step of the previous profiles. In step 2, SP1 sends a delegation assertion request (including the confirmation data) to the user agent, and in step 3 the user agent forwards the assertion request to the IdP. In step 4, the IdP sends the delegation assertion to the user agent, after approving the request and obtaining user consent. Steps 3 and 4 are conveyed over SOAP. In step 5, the user forwards the delegation assertion to SP1 by embedding it within an HTML form. Steps 6, 7 and 8 of this profile are identical to steps 6, 7 and 8 of the profile described in III-C1.

IV. Additional services

In this section we discuss a number of possible issues with the proposed delegation framework.

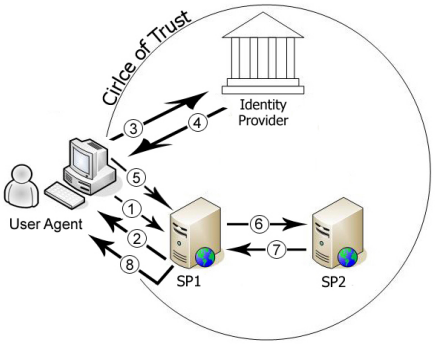

Fig. 9. Delegation framework based on the enabled-client profile

A. Indirect Delegation

Although the delegation scheme described in section III-C supports direct delegation, the framework can also support indirect delegation.

Figure 11 shows how the delegation assertions are handled in the indirect delegation case. As shown in the figure, the delegation assertion is forwarded to the new delegatee, which then forwards it to the IdP in order to obtain a new delegation assertion. The IdP first checks whether the delegation is delegatable or not, and, if so, checks whether the user has consented to delegation to the new delegatee (in the case of indirect delegation, user consent must be obtained in advance). If (and only if) all the checks succeed, the IdP issues the new delegatee with a revised delegation assertion.

The above process can be applied iteratively if further transfers of a delegation are required.

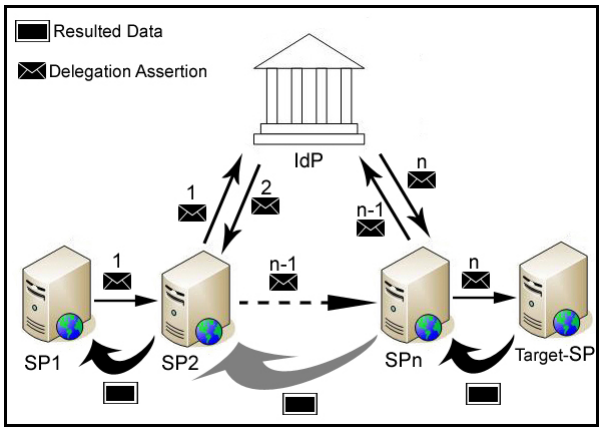

Fig. 10. Handling delegation assertions in the indirect delegation model

B. Delegation Assertion Revocation

Another important issue is revocation of a delegation assertion. The user may at any time wish to revoke a delegation assertion, e.g. for security or operational reasons.

To meet this requirement, a *revocation list* can be implemented, containing details of all delegation requests subsequently withdrawn by the users concerned. Each IdP acting as a delegation authority will be responsible for publishing and updating such a list. A user wishing to revoke a delegation request makes a revocation request (which can be an XML-based message signed using XML signature) to the appropriate IdP. The IdP will authenticate the request, and, if the authentication succeeds, will add the revoked assertion to the revocation list. If this addition is made to the scheme, then a target SP will be required to check the status of every assertion before accepting it.

C. Privacy

The level of privacy provided by the proposed framework depends on the privacy features provided by the Liberty's currently deployed single sign-on framework. The unlinkability property is preserved by using the same opaque handles (i.e. pseudonyms) as are used in Liberty in an encrypted form within a delegation assertion. Moreover, explicit user consent for the delegation process is mandatory, and the resulting data can be encrypted if the user so requests.

V. Related Work

In this section we consider previous work relevant to the proposed delegation framework.

The notion of extending the attribute statements in SAML in order to facilitate delegation has been discussed previously by a number of authors (see, for example, [8], [14]). However, unlike the framework proposed here, previous schemes are generic (i.e. not designed specifically for Liberty), do not assume trust relationships between the system parties, and do not take into consideration the existing communication profiles for SAML 2.0 and Liberty. There are also a number of differences in the delegation assertion structure and the message flow model between the solution described in this paper and the solutions described in [8], [14].

A number of other approaches to delegation have been discussed, such as the *grid delegation protocol* of Ahsant, Basney and Mulmo [1], which is designed for grid applications and uses the WS-Trust protocol. The Wohlgemuth-Müller scheme [16] is built on proxy-based PKI services.

Finally, we mention the Sun One Identity Server, that implements the Liberty specifications. This product, produced by Sun[5], supports delegation services. However, the details of its delegation system are proprietary and do not appear to have been published.

VI. Conclusion

In this paper we have proposed a delegation framework for Liberty based on SAML 2.0 and the Liberty ID-FF 1.2 single sign-on profile. The framework extends the attribute statements defined for a SAML assertion to include delegation related information. The framework is designed to support both direct and indirect delegation. The framework can be readily integrated with the currently deployed Liberty specifications, and it preserve the privacy protection measures in the original framework.

[5]http://www.sun.com

We have also discussed a number of related issues, including an approach to support revocation of delegations.

REFERENCES

[1] M. Ahsant, J. Basney, and O. Mulmo. Grid delegation protocol. Technical Report YCS-2004-380, University of York, Department of Computer Science, July 2004.

[2] O. L. Bandmann, B. S. Firozabadi, and M. Dam. Constrained delegation. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 131–140. IEEE Computer Society, 2002.

[3] S. Cantor, J. Kemp, R. Philpott, and E. Maler (editors). Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. OASIS Standard Specification, OASIS Open.

[4] S. Cantor (editor). SAML 2.0 single sign-on with constrained delegation, October 2005. http://shibboleth.internet2.edu/shibboleth-documents.html.

[5] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) — version 1.1, March 2001. The World Wide Web Consortium (W3C).

[6] D. Eastlake, J. Reagle, and D. Solo (editors). XML-Signature syntax and processing, February 2002. The World Wide Web Consortium (W3C).

[7] D. Eastlake and J. Reagle (editors). XML-Encryption syntax and processing, December 2002. The World Wide Web Consortium (W3C).

[8] H. Gomi, M. Hatakeyama, S. Hosono, and S. Fujita. A delegation framework for federated identity management. In *DIM '05: Proceedings of the 2005 workshop on Digital identity management*, pages 94–103, New York, NY, USA, 2005. ACM Press.

[9] J. Hughes, S. Cantor, J. Hodges, F. Hirsch, P. Mishra, R. Philpott, and E. Maler (editors). Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. OASIS Standard Specification, OASIS Open.

[10] E. Maler, P. Mishra, and R. Philpott (editors). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1, September 2003. OASIS Standard Specification, OASIS Open.

[11] N. Mitra and Y. Lafon (editors). SOAP — version 1.2, April 2007. The World Wide Web Consortium (W3C).

[12] R. Monzillo, C. Kaler, A. Nadalin, and P. Hallem-Baker (editors). Web Services Security: SAML Token Profile 1.1, February 2006. OASIS Standard Specification, OASIS Open.

[13] J. Tourzan and Y. Koga (editors). Liberty ID-WSF web services framework overview — version: 1.1. Liberty Alliance Project.

[14] J. Wang, D. D. Vecchio, and M. Humphrey. Extending the security assertion markup language to support delegation for web services and grid services. In *IEEE International Conference on Web Services (ICWS 2005)*, volume 1, pages 67–74. IEEE Computer Society, 2005.

[15] T. Wason (editor). Liberty ID-FF architecture overview — version: 1.2. Liberty Alliance Project.

[16] S. Wohlgemuth and G. Müller. Privacy with delegation of rights by identity management. In *Proceedings of the Emerging Trends in Information and Communication Security, International Conference (ETRICS 2006)*, number 3995 in LNCS, pages 175–190. Springer, 2006.