User-to-User Delegation in a Federated Identity Environment

HongQian Karen Lu

Gemalto
Austin, Texas, USA
Karen.lu@gemalto.com

Abstract - **Delegation protocols over the Web are mostly used for user-to-machine and machine-to-machine delegations. As more organizations operate in a federated identity environment, user-to-user delegation also becomes a required functionality. User-to-machine or machine-to-machine delegation methods cannot directly apply to user-to-user delegation because human cannot effectively process protocol messages. This paper proposes a new method that allows user-to-user delegations in a federated identity environment. The identity provider (IdP) acts as the delegation authority that manages delegations. Service providers (SPs) in the same environment can use this delegation service, instead of managing delegations individually. The service includes delegation assignment, invocation, and revocation. The method allows service providers to exercise access controls and to decide if the delegator has the right to delegate and if the delegatee should be authorized to perform the requested services. This method is applicable to any access control models.**

Keywords - access control, delegation, federated identity, security.

I. INTRODUCTION

A privilege is the right to perform a certain action to a specific resource or resources; for example, to read (action) a file (resource). Delegation is a process of an identified entity, called a delegator, giving some of the delegator's privileges to another identified entity, called a delegatee. The delegatee receives the privileges to act on behalf of the delegator at a service provider [1][2].

The delegation can be user-to-user (or called person-to-person), user-to-machine, or machine-to-machine. The person-to-person delegation happens often in the physical world. In the digitalized world, a person (a user who uses a computer, an application, or a system) has certain privileges or access rights at a service provider (SP). She may want to give some of her privileges to another user under certain conditions. For example, Alice delegates some of her responsibilities at an SP to Bob while she is out of her office. When a user access services at a SP and the SP needs to access the user's resources at another SP on the user's behalf, the user can authorize a delegation to the first SP, which is a user-to-machine delegation. The machine-to-machine delegation happens similarly among service providers.

The continued increase of online collaborations among organizations and service providers has brought the need of *federated identity* [3]. A federated identity environment consists of an identity provider (IdP) and one or more service providers (SP). The IdP manages user identities and authenticate users. The SPs provide web services and trust the IdP's assertions about the users. Typically, IdP and SPs are different entities and in different domains. This construct, among other things, enables Single Sign-On, which allows a user to use a single set of credentials to login to different SPs through the IdP and login once under certain conditions. This is convenient for both users and SPs, and potentially can provide stronger authentication and, hence, better security as well. Reference [3] provides a good overview about the need and use cases of federated identity, and roles of IdP and SP.

Most research on delegation in a federated identity environment focuses on user-to-machine or machine-to-machine delegations [4]. As the Web becomes the ubiquitous computer, and more organizations, government, and businesses operate in and depend on federated identity environments, user-to-user delegation over the web in such an environment becomes a required functionality. User-to-machine or machine-to-machine delegation methods cannot directly apply to user-to-user delegation because human cannot effectively process delegation protocol messages that may require complex computational and cryptographic operations.

User-to-user delegation has been studied extensively in role-based access control (RBAC) systems [5], and is typically used with a specific SP. In this case, the SP manages its delegation service, which is not an easy task. Furthermore, the delegation should work with different access control models in addition to RBAC.

We propose a new delegation method to address the above issues. The method supports user-to-user delegation service in a federated identity environment. The delegation allows a user to delegate some of her privileges at an SP to another user. The IdP acts as the delegation authority that manages delegations. The SPs use this delegation service, instead of managing delegations individually. The delegation service includes delegation assignment, invocation, and revocation. The SPs need to ensure that delegations are compliant with their access control policies. To facilitate this, the delegation method provides opportunities for SPs to consult their access control engines in order to decide if the delegator has the right to delegate

 ISBN: 978-1-61208-152-6

and if the delegatee should be authorized to perform the requested services. Therefore the method is not tied to any particular access control models.

The interactions between SPs and an IdP follow the SAML 2.0 [6] and XACML [7] standards. We use standard syntax, assertions, protocols, and bindings as much as possible and extend them only as needed.

The rest of the paper is organized as follows. Section II provides some background information and outlines the related work. Section III presents the new user-to-user delegation method in a federated identity environment. Section IV describes the protocols used to support the delegation schemes. Section V discusses issues related to security and implementation. Section VI concludes the paper.

II. BACKGROUND AND RELATED WORK

We follow the delegation terminology defined in the reference [4]. A *privilege* is the right to access specific resources or to perform certain tasks. A user may have a number of such privileges. *Delegation* is an act of (temporarily or permanently) transferring privileges from one entity to another. A *delegator* is an entity that transfers (delegates) all or a subset of its privileges to a delegatee. A *delegatee* is the entity that receives the delegator's privileges in order to use them on the delegator's behalf. A *delegation assertion* is an assertion of the correctness and authority for a delegation, issued by a delegation authority to a delegatee. A *delegation authority* is an entity that controls delegation and issues delegation assertions.

A. Access Control and XACML

Access controls are security mechanisms that control how subjects (users, applications, and systems) access and interact with objects (resources, other applications, and systems). Access control includes identification, authentication, authorization, and accountability. There are three main types of access control models: discretionary, mandatory, and role-based. All organizations must have access control policies and implementations to protect their resources and systems from unauthorized access.

Delegation is a mechanism of transferring access privileges from one subject to another. Such a privilege transfer must be authorized and not violate organization/system's access control policy. Therefore, the access control policy should include delegations, and the delegation mechanism should be associated with the access control engine. Research on delegation in access control models have built on and supported this concept. However, recent research on delegation related to the federated identity and machine-to-machine delegations have focused on mechanisms and semantics but failed to address the link between the delegation and access control [2][4][8].

The XACML (eXtensible Access Control Markup Language) is a standard for specifying and communicating access control policies across computer systems (internal or external to an organization) [7]. The current version is XACML 2.0. The XACML 3.0 is working in progress, which includes the concept and process of delegation. The XACML delegation deals with creation of new policies and tracing back trusted policies. We can use the syntax of the XACML 3.0 delegation to support our work.

B. User-to-User Delegation

Delegation in Role-Based Access Control (RBAC) has been studies extensively [5]. The RBAC system manages the delegation based on the access control policies. In doing so, it must answer the following two questions: 1. Is a user (delegator) authorized to delegate a role, privilege, or permission that is available to him? 2. Can a role, privilege, or permission be delegated to a user (delegatee)?

In the context of delegation service in a federated identity environment, service providers manage their own access controls and, hence, the permission to delegate. SP must find answers to the above questions when a user delegates, when a user invokes a delegation, and when a user's privileges have changed. This applies to any access control model, not just RBAC.

Peeters *et al.* [2] described procedures of the delegation, including mandate issuance, acceptance, revocation, and invocation. Furthermore, the paper outlined advanced delegations: transferable delegation and corporate delegation. The paper stays at a pure conceptual level and not at the web application level. The title including "identity federation" is somewhat misleading as the approach has no link to any federated identity method.

C. User-to-Machine Delegation

The Shibboleth System is a SAML 2.0 based, open source software package for web Single Sign-On across or within organization boundaries [9]. The Shibboleth has a solution to the proxy authentication problem: how to authenticate a service to which a user may have authenticated to, and who wishes to invoke another service on the user's behalf [8]? The method uses two Single Sign-On's through the same IdP. The delegation assertion is enabled by the first authentication statement and is built into the second authentication statement. The IdP issues and signs the delegation assertion.

Alrodhan and Mitchell [4] proposed a delegation framework for Liberty Alliance Project. The method extends the attribute statement in the SAML assertion to form a delegation assertion. The IdP issues and signs the delegation assertion with the user (delegator, privilege owner) consent. The Single Sign-On Profiles described in the Liberty ID-FF 1.2 specification provides the base for

 ISBN: 978-1-61208-152-6

this delegation framework. This work is similar to Shibboleth's delegation in the sense that the delegatee (first SP), through the user agent, gets a delegation assertion from the IdP and then presents it to the target (second SP). The differences are in profiles, assertions, and so forth.

OAuth is an open protocol that enables a website to access protected resources from another website on the resource owner's behalf, without requiring the resource owner to disclose his login credentials [10]. As such, OAuth provides a protocol for user-to-machine delegation. A growing number of companies support OAuth, including Twitter, Google, Netflix, Yahoo!, and Facebook.

The "SAML V2.0 Condition for Delegation Restriction" specifies the expression of delegation information through a SAML Condition extension to address the use cases that a single logical transaction involves one or more intermediate entities (clients or servers) [11]. The SP must evaluate delegates in the condition and should only accept the assertion if it wishes to accept the condition.

D. Resource Sharing

The Liberty Alliance Identity Web Service Framework (ID-WSF) People Service (PS) [12] is a web service that allows users to track and manage people they know online, and allows other web services to query and manage the people list of their users.

The People Service enables "cross-user" interactions that involve more than one user for an online activity. For example, Alice wants to share her photos with Bob at her photo website at which Bob does not have an account. The photo website uses Alice's People Service and Bob's identity provider to identity Bob and lets him to access photos specified by Alice. Such resource sharing may be used for user-to-user delegation in limited situations, but is not designed for such a purpose. It implicitly assumes the discretionary access control (DAC). If Bob also has an account with the SP, the People Service is unnecessary.

III. USER-TO-USER DELEGATION

We propose a new method that supports user-to-user delegation service in a federated identity environment. The delegation allows a user to delegate some of his privileges at a service provider (SP) to another user. The delegation service includes delegation assignment, invocation, and revocation.

In a federated identity environment, service providers trust the identity provider (IdP) to manage user identities and authenticate users [3]. In such an environment, IdP can, in addition, act as the delegation authority that manages user-to-user delegations. The delegator assigns delegations at IdP. The delegatee is to perform the delegated tasks at the specified service provider (SP). The SP obtains delegation assertions from the IdP. Delegations can be revoked either by the delegator or by the SP.

Why do we need a delegation authority? Each individual SP can certainly provide the delegation service by itself without needing a delegation authority. However, tracking and managing delegations is not a trivial task. A valid alternative for service providers is to use a trusted delegation authority [2][4]. From a user perspective, a delegator may want to delegate at more than one service provider. Going to each service provider one by one is not convenient, at least. A delegation authority can solve this problem by providing a common portal for the delegation service.

The service providers' access control models play an important role in the design of this delegation method. Service providers need to ensure that delegations are compliant with their access control policies. For this purpose, the delegation method allows SPs to exercise access controls throughout delegations' life cycles. These mechanisms are independent of SPs' access control models.

A. Delegation Life Cycle

When a delegator delegates to a delegatee, the IdP creates a delegation record, or simply called a delegation. A delegation life cycle starts with the creation and ends with the deletion. The following figure illustrates the life cycle.

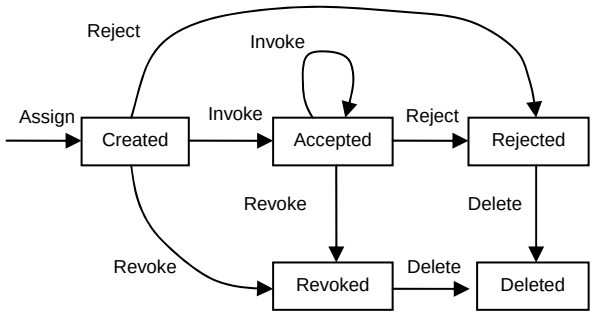

Figure 1. Delegation life cycle.

A delegation record includes the delegator, the service provider, the delegatee, the resources that the delegatee is to access, the actions that the delegatee can do after obtaining the resource, and other things, such as assignment date and time, valid period, delegator's signature, and so on. When the delegator assigns a delegation, the IdP creates a delegation record; the delegation is in the created state. The invocation of the delegation by the delegatee transfers the delegation into the accepted state. Other states illustrated in the figure are self-explainable.

 ISBN: 978-1-61208-152-6

B. Delegation Schemes

The main delegation schemes include assignment, invocation, and revocation.

Assignment

A delegator wants to delegate to a delegatee some tasks to be performed at an SP. The IdP does not know what privileges that the delegator can delegate. The IdP's role is to manage the delegation and to tell the SP that the delegator indeed has delegated certain privileges to the delegatee. The SP needs to make sure that the delegator has these privileges and can delegate them, and the delegatee is authorized to perform the tasks.

If the delegator knows exactly what he can delegate, the assignment becomes simple. The delegator specifies the SP, delegatee, and privileges that he wants to delegate. The IdP creates a delegation record. In many cases, however, the delegator may know what he wants to delegate to a delegatee but is not sure if he can. Then the assignment is more complex. The IdP needs to ask the SP if it can authorize a specified delegation request. The IdP does this by, for example, making an `XACMLAuthzDecisionQuery`, which is specified in the XACML SAML profile. If SP responds with a success, IdP creates a delegation. Otherwise, IdP asks the delegator to make a modification and repeat the process.

In general, a delegator may not know if or what he can delegate to a particular delegatee. In this case, he specifies the SP and the delegatee. The IdP asks the SP about privileges that the delegator can delegate to the delegatee. The SP responds with a list, which may be empty. The IdP asks the delegator to make a selection. When needed, the IdP asks the SP if such delegation can be authorized. If SP responds with a success, IdP creates a delegation record.

Figure 2 illustrates the delegation assignment workflow of the above general case, which includes the following steps. We can easily adopt this workflow to simpler cases.

1. The delegator **A** authenticates to the IdP.
2. **A** selects the SP that he wants the delegatee **B** to access.
3. The IdP finds from the SP the privileges that **A** can delegate to **B**.
4. The IdP presents a list containing those privileges (resources, actions) to **A**.
5. **A** selects privileges to delegate to **B** from the list and other constraints, such as valid time period.
6. The IdP creates a delegation record. (Optionally, IdP asks the SP if such delegation can be authorized before creating a delegation.)
7. The IdP may ask **A** to digitally sign the delegation for non-repudiation.
8. **A** signs the delegation if required.
9. **A** or the IdP informs **B** about the delegation.

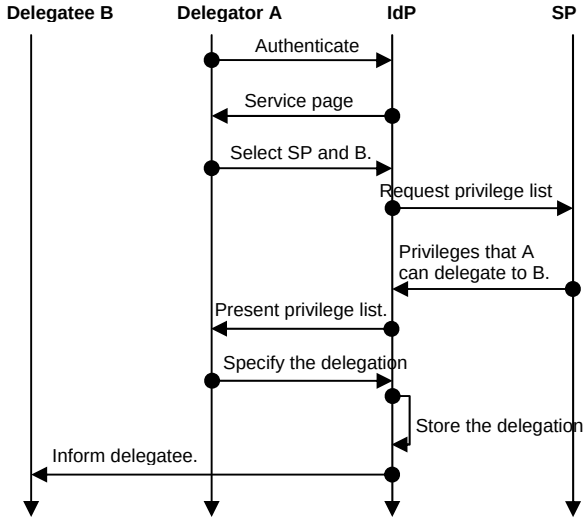

Figure 2. Delegation assignment.

The SP checks with its access control engine to decide what privileges that **A** can delegate to **B** and presents the privilege list, if exist, to IdP. The access control engine makes decisions according to its access control policies.

In practice, a delegator may need an approval from her manager or some other entities in order to delegate. The access control system may not have an automated mechanism for doing so. Then the approval is a physical process. Addressing such issues is outside the scope of this paper.

Invocation

When the delegatee requests to perform a delegated task at the SP, he invokes a delegation. Figure 3 illustrates this process, which consists of the following steps:

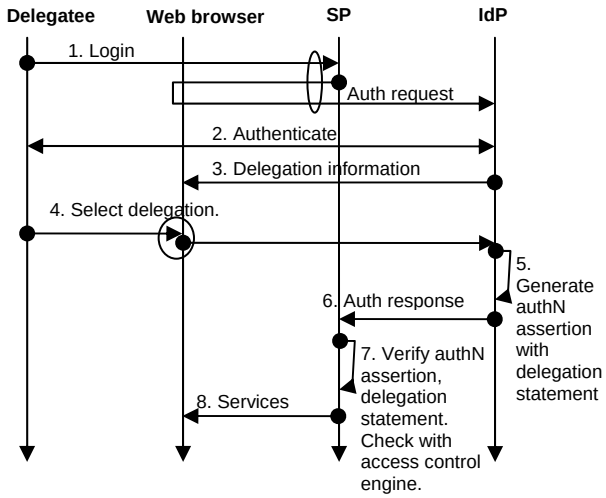

Figure 3. Invocation.

1. The delegatee logs in to the SP, which redirects the authentication to the IdP.
2. The IdP authenticates the delegatee.
3. The IdP finds and presents delegation(s) at the SP to the delegatee.
4. The delegatee selects one or more delegations.
5. The IdP generates an authentication assertion for the delegatee with a delegation attribute statement specifying the delegation(s).
6. The IdP sends the authentication assertion to the SP.
7. The SP verifies the authentication assertion and the delegation statement. The SP consults with its access control engine for both the delegator and delegatee.
8. If all is well, The SP present services to let the delegatee to perform the delegated tasks.

The IdP generates an authentication assertion in response to the authentication request from the SP. The subject in the assertion is the delegatee. The assertion in addition includes an attribute statement about the delegation. The following code snippet illustrates an example in the form of an SAML 2.0 assertion.

```
<Assertion>
  <Issuer> … URI of the IdP … </Issuer>
  <ds:Signature> IdP's signature </ds:Signature>
  <Subject> Information on delegatee </Subject>
  <Conditions>
  <AuthnStatement> // authentication statement
  <AttributeStatement>
    <Attribute Name="Delegation">
       <AttributeValue>
          <Delegator>
          <Delegatee>
          <Privilege> // one or more
             // description, services, resources,
actions, and so forth.
          </Privilege>
       </AttributeValue>
    </Attribute>
  </AttributeStatement>
</Assertion>
```

The service provider processes the delegation information. For example, SP verifies the following:

1. The delegation is in the valid period.
2. The service request is specified in the statement.
3. The requester is the delegatee specified in the statement.
4. The signature of the assertion is valid and the certificate is not revoked.
5. The delegator is authorized to perform the delegated privileged task.
6. The delegator is authorized to delegate the privileged task.
7. The delegatee is authorized to perform the delegated task.
8. Other optional constraints are met.

If any of the verification steps fails, the SP denies the requested service from the delegatee. The checking on authorizations is necessary because conditions may have changed since the last time that the SP queried the access control engine regarding the delegator and the delegatee when the IdP requested the privilege list in setting up the delegation. If any of the authorization checking fails, the SP revokes the delegation.

Revocation

Previous research suggested using certificate revocation mechanisms, Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP), to handle delegation revocation [2]. Both CRL and OCSP are known for their complexities of maintaining the list of revoked certificates. We propose a simpler delegation revocation approach that does not require maintaining a revocation list nor require a separate query on the delegation status.

As described earlier, IdP is the delegation authority that manages delegation records. SP gets the delegation statement from IdP as a part of an authentication assertion. The SPs should not accept delegations from anyone else. SP can store the delegations for audit purposes, but should not reuse them. The delegation assertion is always dynamically acquired. Therefore, SP does not need to check for the delegation status.

A delegation revocation can be initiated by the delegator or by the SP. After receiving a revocation request, IdP authenticates and verifies the request. If the request is authentic and verifiable, IdP removes the delegations involved from its delegation database. (IdP may keep the revoked delegations for auditing purpose.)

The delegation situation is very different from that of SSL certificate. Typically SP receives a SSL certificate from a third party. Before using it, SP would check to see if the certificate is still valid by consulting with CRL or using an OCSP service. With our delegation approach, SP obtains the delegation assertion from the delegation authority, IdP. Therefore, SP can verify the assertion and does not need to ask IdP for the validity of the assertion, and IdP does not need to provides a service for such purpose.

(a) Revocation by Delegator

The delegator can revoke a delegation at IdP. This involves the following steps:

1. The delegator **A** logs in to the IdP.
2. **A** revokes his delegation to the delegatee **B**.
3. IdP removes **A**'s delegation to **B** from its record.
4. **A** or IdP informs **B** about the revocation of the delegation.

(b) Revocation by Service Provider

When a user's privilege**s** are reduced or removed, the service provider should find out if there are outstanding delegations relevant to this user. If so, SP needs to examine

 ISBN: 978-1-61208-152-6

each of the delegations to see if they are still valid. For example, if the user was a delegator and he no longer has privilege to delegate the task, or if the user was a delegatee and he no longer has the privilege to perform the delegated task, then SP sends a revocation request to IdP to revoke the delegation. IdP then informs the delegator and the delegatee. Figure 4 illustrates the process. Section 4 will provide more details on the delegation query request and response, and delegation revocation request and response.

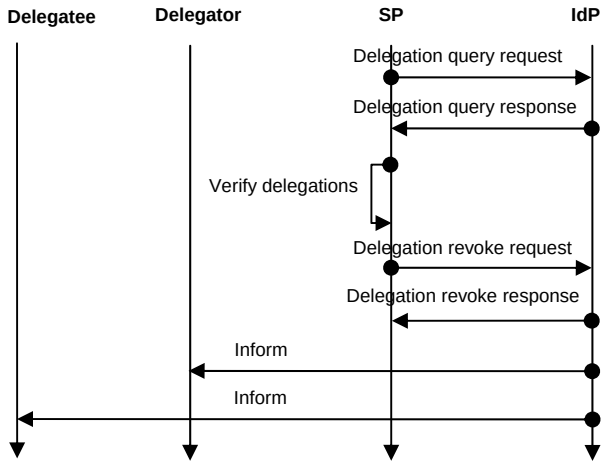

Figure 4. Revocation by the service provider.

(c) Cleanup by Identity Provider

IdP cleans up its delegation repository periodically. For delegations that have not been activated for a while or just for any delegations, IdP can make XACML authorization decision query to SP. If the response is negative, IdP can remove the delegation. This avoids using any SAML extension for delegation revocation.

C. Acceptance and Rejection

The delegatee can either accept or reject a delegation. The present scheme does not allow partial acceptance. The delegatee examines the delegation from the received information or the IdP provides a service for the delegatee to do so. The act of the delegatee requesting to perform the delegated task at the SP is a form of delegation acceptance. IdP may also provide a service for the delegatee to explicitly accept the delegation.

If the delegatee rejects the delegation, he can inform the delegator, who may either modify the delegation or revoke the delegation at IdP. The IdP may also provide a service for the delegatee to reject a delegation. Revocation of acceptance can be done the same way as rejection.

IV. PROTOCOLS

Delegation assignment, query, invocation, and revocation all require communications between SP and IdP. We use SAML 2.0 assertions [6] as the message exchange format and extend as needed. SAML protocols and bindings are used to transport the delegation messages.

A. Request and Response

SAML protocol is a request and response protocol. The requester sends a request, and the responder processes the request and sends a response.

B. Attribute Query

The SAML 2.0 attribute query `<AttributeQuery>` is used for querying attributes of a subject. We use it to query privileges that a delegator (subject) can delegate to a delegatee, and existing delegations for a delegator or delegatee. The response is an attribute assertion or query status.

Query Privileges

During the delegation assignment, IdP asks SP what privileges that the delegator can delegate to the delegatee. The XACML policy query `<XACMLPolicyQuery>` specified in the XACML SAML profile can serve this purpose. In response, the SP sends an XACML policy assertion that contains the requested information. The following code snippet illustrates the query.

```
<xacml-samlp:XACMLPolicyQuery>
  <saml:Issuer>
  <ds:Signature>
  <Attribute ID>
  <Attribute IssurInstant>
        … …
  <xacml-context:Request>
    <xacml:Attributes>
        <Attribute name=”user”>
          <AttributeValue>
             id or other attributes of delegator
          </AttributeValue>
        </Attribute>
        <Category
        name=”urn.oasis:names.tc:xacml:3.0:attribu
te-category:delegate”>
    </Attributes>
    <xacml:Attributes>
        <Attribute name=”user”>
          <AttributeValue>
             id or other attributes of delegatee
          </AttributeValue>
        </Attribute>
        <Category
name=”urn.oasis:names.tc:xacml:3.0:attribute-
category:delegated:urn:oasis:names:tc:xacml:3.0:su
bject-category:access-subject”>
    </Attributes>
        <xacml:Attributes>
        <Category
name=”urn.oasis:names.tc:xacml:3.0:attribute-
category:delegate”>
        <Category
name=”urn.oasis:names.tc:xacml:3.0:attribute-
category:delegated:urn:oasis:names:tc:xacml:3.0:su
bject-category:resource”>
        <Category
name=”urn.oasis:names.tc:xacml:3.0:attribute-
category:delegated:urn:oasis:names:tc:xacml:3.0:su
bject-category:action”>
    </Attributes>
```

 ISBN: 978-1-61208-152-6

```
  </Request>
  <Attribute name="ReturnPolicyIdList">
    <AttributeValue>true</AttributeValue>
  </Attribute>
        … …
</XACMLPolicyQuery>
```

In response, SP sends a <samlp:Response>, which contains an XACMLPolicy assertion that has a statement of the type xacml-saml:XACMLPolicyStatementType. This statement contains policies that the query requested.

Query Delegations

When a user's privileges have been removed or reduced, the SP should examine all outstanding delegations associated with this user, either as a delegator or as a delegatee. For this purpose, the SP sends a SAML query request <AttributeQuery> to the IdP and the IdP responds with an attribute assertion containing relevant delegation statements, if they exist. The following code snippet illustrates the query.

```
<samlp:AttributeQuery>
  <saml:Issuer>
  <ds:Signature>
  <Attribute ID>
  <Attribute IssuerInstant>
        … …
  <Subject>
    <NamdID id=… delegator or delegatee's id…>
              … …
  </Subject>
  <Attribute name="Delegation">
        … …
</AttributeQuery>
```

The IdP responds with an assertion containing one or more attribute statements about the delegations.

C. Authentication Request

The authentication request is the standard SAML 2.0 <AuthnRequest>. When sending an authentication request, the SP does not know anything about the delegation. The delegatee selects the delegation at the IdP during authentication.

D. Delegation Revocation Request

When access privileges of a user have changed and existing delegations are no longer valid, the SP sends a delegation revocation request to the IdP to revoke relevant delegations. While neither SAML 2.0 nor XACML 2.0/3.0 specified revocation, we can follow the SAML syntax to define it. The delegation revocation request and response are similar to authentication request and response, in which the SP sends a request to the IdP; the IdP fulfills the request and sends an assertion in response. The request can take the following form.

```
<DelegationRevokeRequest>
  <Issuer>
  <ds:Signature>
        … …
  <Subject>
  <Attribute name="Delegation">
    <Delegator>
    <Delegatee>
    <Resource>
        … …
  </Attribute>
        … …
</DelegationRevokeRequest>
```

The elements in <Delegation> are optional. The request can have the following rules:

1. If no <Resource> is specified, SP requests to revoke all delegations associated with <Delegator>, <Delegatee>, or both.
2. If none of <Delegator> and <Delegatee> exists, SP requests to revoke all delegations associated with the subject and <Resource>.
3. If there is <Delegator> but no <Delegatee>, SP requests to revoke all delegations that the subject is a delegator and delegated <Resource> to any delegatee.
4. If there is <Delegatee> but no <Delegator>, SP requests to revoke all delegations that the subject is a delegatee and was delegated for <Resource> by any delegators.
5. If there are both <Delegator> and <Delegatee>, SP requests to revoke all delegations that associated to <Delegator>, <Delegatee>, and <Resource>. The subject can be a delegator or delegatee.

The response from the IdP contains the status of the revocation.

E. Bindings

A transport binding is a mapping from SAML messages to a communication protocol. The delegation statement comes as a part of an authentication assertion. Therefore, the delegation takes on whatever binding that the authentication process uses. For example, the authentication can use SAML 2.0 Web browser SSO profile [13]. The corresponding bindings include HTTP redirect, HTTP POST, and artifact bindings.

V. DISCUSSIONS

This section discusses some issues related to security and implementations. Other technical details for providing delegation services are dependent on the specifics of the environment, access control policies, security level, and so forth, which is outside the scope of this paper.

The proposed user-to-user delegation scheme can use existing federated identity frameworks and protocols, such as SAML 2.0 and XACML, as its foundation. The established trust relationships in a federated identity environment enable the SPs to trust and use the IdP as the delegation authority.

 ISBN: 978-1-61208-152-6

When deploying the delegation service, securing the communications between IdP and SPs is important, especially for services involving high value or high security transactions. At the web application level, this can be achieved by using HTTPS and TLS/SSL with mutual authentication and strong cipher suites. It allows each party to know for sure whom it is talking to, and ensures the integrity and confidentiality of the communications.

Digitally signing all delegation statements, queries, requests, and responses is also important. These signatures provide authenticity and non-repudiation. The SPs should not reuse delegation statements because the situation may have changed since a delegation was issued.

The delegation statement in the authentication assertion provided by the IdP is not an authorization of delegation. Instead the IdP vouches that the delegator indeed has delegated some tasks to the delegatee. It is the service provider's responsibility to consult its access control engine to decide if the delegatee should be authorized to receive the requested services, and to record the transactions.

This delegation scheme requires that the delegatee has an account at the IdP, because the IdP needs to be able to identify and authenticate the delegatee. The service providers' access control policies dictate whether the delegatee needs an account at their websites. For example, if the SP has a mandatory access control policy, the delegatee needs an account at the SP because the access control is managed by the SP's system. For another example, if the SP has a discretionary access control policy, that is, it lets the user to decide permissions regarding to *his* resources, such as data, at the SP. Then the SP may not need to know the identity of the delegatee.

VI. CONCLUSIONS

This paper proposes a new delegation method that enables user-to-user delegations in a federated identity environment. This method allows service providers (SPs) to use the delegation service, instead of managing delegations individually. The service providers can exercise access controls and decide if the delegator has the right to delegate and if the delegatee should be authorized to perform the requested services. This method is applicable to any access control models because service providers control the access to their resources.

ACKNOWLEDGEMENT

The author would like to thank Dr. Ksheerabdhi Krishna and Mr. Kapil Sachdeva for their help to this work.

REFERENCES

[1] X. Huysmans and B. Van Alsenoy, editors, "Identity management for eGovernment, Annex I. Glossary of terms (v1.07), IDEM project, 2007, https://projects.ibbt.be/idem/uploads/media/2007-12-27.idem.glossary.v1.07.pdf [accessed: July 18, 2011].

[2] R. Peeters, *et al*. "Cross-context delegation through identity federation," Proc. of SIG on Biometrics and Electronic Signature, 2008. http://www.cosic.esat.kuleuven.be/publications/article-1156.pdf [accessed: July 18, 2011].

[3] OASIS, "Security assertion markup language (SAML) v2.0 technical overview," editors: N. Ragouzis *et al*., committee draft 02, March 25, 2008, http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf [accessed: July 18, 2011].

[4] W. Alrodhan and C. Mitchell, "A delegation framework for liberty," Proc. of the 3rd Conference on Advances in Computer Security and Forensics, 10-11 July 2008, Liverpool, UK, http://www.isg.rhul.ac.uk/cjm/adffl.pdf [accessed: July 18, 2011].

[5] J. Crampton and H. Khambhammettu, "Delegation in role-based access control," Lecture Notes in Computer Science, volume 4189, 2006, pp. 174-191.

[6] OASIS, "Assertions and protocols for the OASIS security assertion markup language (SAML)," v2.0, http://saml.xml.org/saml-specifications [accessed: July 18, 2011].

[7] OASIS, "eXtensible access control markup language (XACML)," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml [accessed: July 18, 2011].

[8] Internet 2 – Shib-uPortal, https://spaces.internet2.edu/display/ShibuPortal/Home [accessed: July 18, 2011].

[9] Shibboleth, http://shibboleth.internet2.edu/ [accessed: July 18, 2011].

[10] OAuth, http://oauth.net/ [accessed: July 18, 2011].

[11] OASIS, "SAML V2.0 Condition for delegation restriction," Committee draft 01, 10 March 2009, http://wiki.oasis-open.org/security/SAML2DelegationCondition [accessed: July 18, 2011].

[12] Project Liberty, "Liberty ID-WSF people service specification," v 1.0, http://projectliberty.org/liberty/content/download/890/6246/file/liberty-idwsf-people-service-v1.0.pdf [accessed: July 18, 2011].

[13] OASIS, "Profiles for the OASIS security assertion markup language (SAML)," v2.0, http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf [accessed: July 18, 2011].

 ISBN: 978-1-61208-152-6