I. Agudo, J. Lopez, and J. A. Montenegro, "Enabling Attribute Delegation in Ubiquitous Environments", *Mobile Networks and Applications*, vol. 13, pp. 398-410, 2008.
http://doi.org/10.1007/s11036-008-0062-4
NICS Lab. Publications: https://www.nics.uma.es/publications

Enabling Attribute Delegation in Ubiquitous Environments

Isaac Agudo, Javier Lopez, Jose A. Montenegro
Computer Science Department, University of Malaga, Spain
{isaac,jlm,monte}@lcc.uma.es

Abstract

When delegation is implemented using the attribute certificates in a Privilege Management Infrastructure (PMI) [2, 11, 4], it is possible to reach a considerable level of distributed functionality. However, the approach is not flexible enough for the requirements of ubiquitous environments. The PMI can become a too complex solution for devices such as smartphones and PDAs, where resources are limited. In this work, we solve the previous limitations by defining a second class of attributes, called domain attributes, which are managed directly by users and are not right under the scope of the PMI, thus providing a light solution for constrained devices. The two classes of attributes are related by defining a simple ontology. While domain attribute credentials are defined using SAML notation, global attributes are defined using X.509 certificates. For this reason, we additionally introduce XSAML so that both kinds of credentials are integrated. We also introduce the concept of Attribute Federation which is responsible for supporting domain attributes and the corresponding ontology.

1 Introduction

Much has been written about identity management [12] and federation. We believe this is an evidence of the need for a distributed solution in response to those issues. When dealing with distributed solutions in the Internet, one realizes about the difficulties to build an infrastructure from scratch. In fact, organizations tend to reuse existing solutions and define interconnection mechanisms to allow interoperability. This problem has been approached using different mechanisms, but we are particularly interested in federations.

In federations, several service providers delegate the identity management to a third party who is responsible of collecting identity information and authenticating users. In this way, service providers rely on the third party to authenticate users and decide, according to the identity of the user, whether the requested service can be provided.

One interesting problem here is that service providers, even if they do not have to perform user authentication, need at least to know the potential users. In the case where the potential users are unknown, access control policies of unknown users need to be defined, what implies relying on the third party also during the access control phase. Hence, there is a distinction between already known users and those not yet known, which makes the definition of access control and authorization policies harder.

Even if the service provider knows all the users, it may be more difficult to rely on identities for the definition of authorization policies than to use attributes for this purpose. Moreover, by using attributes, the use of identities can be avoided entirely in the definition of authorization policies. This is why we should consider an analogous concept to Identity Federation, but using attributes instead of identities for the definition of the authorization rules. In this way, users will not carry out the trust negotiation by themselves but they will request the federation to do it.

In [7, 14, 8, 6, 15] the concepts of *automated trust negotiation* (ATN) and the concept of *Attribute Based Access Control* (ABAC) are introduced. ABAC is an access control model which defines au-

thorization policies based on user's attributes. When using this approach, attributes such as financial or medical data may be sensitive. The ATN implements the mechanisms which avoid disclosure of confidential attributes.

The main contribution of our work is the possibility of moving the ATN from the user side to the attribute Federation side. Thus, in our scenario, users do not have to worry about disclosure of sensitive data, they "instruct" the federation on how to handle the procedure, and allow it negotiate on their behalf. As a result, neither requesters nor service providers have to be involved in the trust negotiation phase before being able to establish a session.

It is important to note that, in ubiquitous environments trust relations are more important than in centralized scenarios and are the foundation for the decision making process in most cases. Also, identities are rarely used due to the dynamic characteristics of those environments. Besides that, the attribute federation becomes more important for ubiquitous devices as they do not usually know their potential neighbors before they enter the device network. Moreover, the more processes we outsource, the less resources are needed in devices using this attribute federation for defining the authorization policies, what is important because when we focus on mobile devices, such as PDAs or cellular phones, saving resources is one of the first priorities. This is another reason why in these kinds of devices the concept of attribute federation becomes attractive.

According to this argumentation, the paper outline is as follows. In section 2, related work is presented from both the theoretical and more practical point of view. Section 3 introduces XSAML as mechanism to interconnect SAML and X.509. In section 4 we introduce the concept of attribute federation and justify its need. Section 5 details some implementation issues and, finally, section 6 provides some conclusions.

2 Related Work

In this section we review some proposals related with the idea of attribute federation. We have focused on one academical research result and on one applied solution.

2.1 RT Framework.

Li et al. proposed logic programming as a way to model authorization and delegation relations [9]. Although they use *Roles* for this purpose, their roles can also be interpreted as attributes, as it is commented in their work. They define a full general framework, RT for *Role Based Trust Management*. It comprised of five different solutions, each of them with different characteristics. Roles can be interpreted as privileges or attributes. The RT Framework defines a partial order in roles, establishing how rights can be inherited.

RT defines several types of credentials, the basic ones are:

1. $A.R \leftarrow D$: This credential can be read as *D has the attribute $A.R$*, or equivalently, *A says that D has the attribute R.*

2. $A.R \leftarrow B.R_1$: This credential can be read as *if B says that an entity has the attribute R_1, then A says that it has the attribute R.*

3. $A.R \leftarrow A.R_1.R_2$: This credential can be read as *if A says that an entity B has the attribute R_1, and B says that an entity D has the attribute R_2, then A says that D has the attribute R.*

4. $A.R \leftarrow B_1.R_1 \cap B_2.R_2 \cap \ldots \cap B_n.R_n$: This credential can be read as *A believes that anyone who has all the attributes $B_1.R_1$, ..., $B_k.R_k$ also has the attribute R.*

RT defines an attribute federation using linked roles. In this way, users can link their attributes to other users' attributes, defining then their authorization policies in terms of other users attributes. However, it is not clear how and where credentials and authorization policies are stored or who defines them.

In the example 2.1 there is a federation between `EOrg.preferred` and `IEEE.member`, so `EOrg` delegates to `IEEE` when making a decision on the `preferred` attribute. This is done by defining a

```
EPub.disct ←
EPub.preferred∩EPub.student
EPub.preferred ← EOrg.preferred
EOrg.preferred ← IEEE.member
EPub.student ← EPub.university.stuID
EPub.university ← ABU.accredited
ABU.accredited ← StateU
StateU.stuID ← Alice
IEEE.member ← Alice
```

Figure 1: RT Federation example

local map in the domain of `EOrg` from attribute `IEEE.member` to attribute `preferred`. Therefore, `EOrg` trusts `IEEE` issuing the attribute `IEEE.member`, and it knows, to some extent, the reasons and implications of the issuance of this attribute. Before definition and federation of an attribute, the defining entity must collect all the information related to the other attribute in the federation.

2.2 SAML

SAML, developed by the Security Services Technical Committee of OASIS, is an XML-based framework for communicating user authentication, entitlement, and attribute information. It is used in many security applications. In particular, the Shibboleth [18] software implements the OASIS SAML v1.1 specification [16], providing a federated Single-SignOn and attribute exchange framework. Liberty Alliance [1] is also a federation solution based on SAML.

Version 2 of SAML presents in its Technical Overview ([17]), presents an Attribute Federation scenario. They conceive attribute federation as a way of passing attributes from one domain to another. In this way, service providers may pass requests to other service providers with attached attributes. The following Attribute Federation scenario is extracted from [17].

1. The user is challenged to supply their credentials to the site *AirlineInc.com*.

2. The user successfully provides their credentials and has a security context with the *AirlineInc.com* identity provider, the user named supplied is John.

3. The user selects a menu option (or function) on the *AirlineInc.com* application what means that the user wants to access a resource or application on
   CarRentalInc.com.

4. The *AirlineInc.com* service provider sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which there is a SAML assertion about user John. The name identifier used in the assertion is an arbitrary value ("wxyz"). The attributes "gold member" and a membership number attribute ("1357") are provided. The name John is not contained anywhere in the assertion.

5. The browser, either due to a user action or via a "-submit", issues an HTTP POST containing the SAML response to be sent to the
   CarRentalInc.com Service provider.

6. The *CarRentalInc.com* service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this and the assertion are correctly validated, a local session is created for user John. This is determined from a combination of the gold member and membership number attributes. It then sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session. An access check is then performed to establish whether the user John has the correct authorization to access the
   CarRentalInc.com web site and the TARGET resource. If the access check is passed, the TARGET resource is then returned to the browser.

In this scenario, the attribute "gold member" and a membership number attribute are passed to the external service provider along with the request. This proposal focuses on including attributes in the single sign on process, but not on how relationships between attributes are established. Moreover, although attributes are passed to the final service provider from

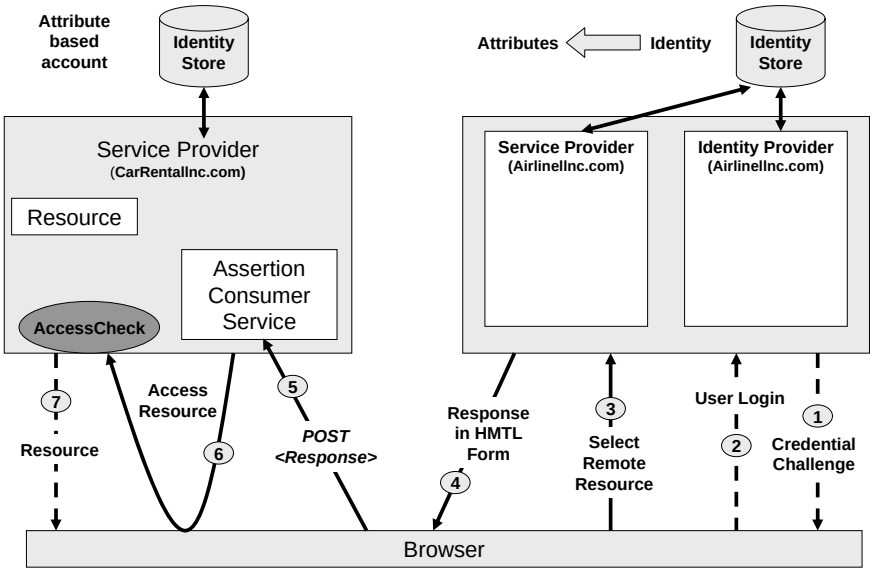

Figure 2: SAML attribute federation scenario

the initial one, in the end it has to recover the identity of the requester in order to be able to check the request. Thus, access control is not based on attributes but on identity; attributes are used only as a means of transporting the identity.

This protocol supposes that the requester first signs on some domain prior to accessing the real service provider. In a real attribute federation, the final service provider does not have to be known by the initial service provider. It means that if CarRentalInc.com trusts the attribute gold member, it does not have to let AirLineInc.com know. Only the car rental company is involved and the airline company does not need to be informed. The process of establishing the attribute federation is carried out by the entity who trusts and should be autonomous from the trusted entity.

3 XSAML

Our initial purpose when designing XSAML was to translate the information stored in attribute certificates to SAML code and vice versa. XSAML has an outstanding role in our scheme, bridging the gap between the Global Attributes and Domain Attributes. The Global attributes are defined using X.509 attribute certificates whereas the Domain attributes are defined using SAML. Therefore, XSAML symbolize the glue in our infrastructure because makes possible the translation process between Global and Domain Attributes.

As aforementioned, SAML is defined using XML [10] sentences whereas X509 attribute certificates are defined using ASN.1 [3]. A performance comparison between both technologies was presented in [5], showing that ASN.1 encode, decode and file space is more effective than XML and, therefore, SAML. On the other hand, XML technologies are nowadays more widespread than products based on ASN.1. For this reason, we establish a mixed system using SAML and

attribute certificates technologies.

The structure of X.509 attribute certificate and a brief example of its corresponding ASN1 encoding is shown in figure 3 (for detailed information see [13]).

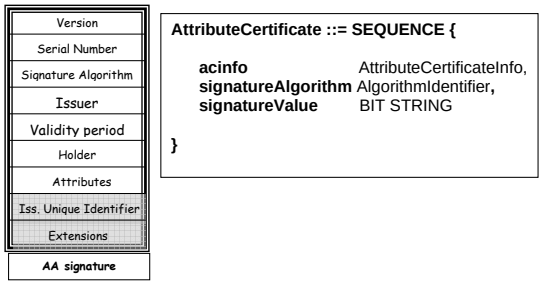

Figure 3: X509 Attribute Certificate and ASN1 representation

Additionally, an example of SAML sentences is shown in figure 4. Furthermore, the relationship between SAML and X509 attribute certificate fields is described in the table 1.

```
<saml:Assertion …>
  <saml:Conditions …/>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier SecurityDomain="snakeoil.edu"
                  Name="Subject1" />     </saml:Subject>
     <saml:Attribute  AttributeName="Pass_Test1_Subject1"
                  AttributeNamespace="http://snakeoil.edu">
       <saml:AttributeValue> Pass_Test1_Subject /saml:AttributeValue>
     </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

Figure 4: SAML example

3.1 Overview of XSAML

Figure 5 shows a brief description of XSAML system. The main design goal has been to facilitate suitability

SAML Assertion	Attribute Certificate
Issuer	Issuer of Attribute
Subject of the Query	Certificate Subject
Validity period	AC Validity period
Attributes	AC Attributes

Table 1: Relationship between SAML and AC fields

to any environment. In this way, when a task depends on the environment, an interface will be created and programmers will be able to add code according to the requirements of the environment in question.

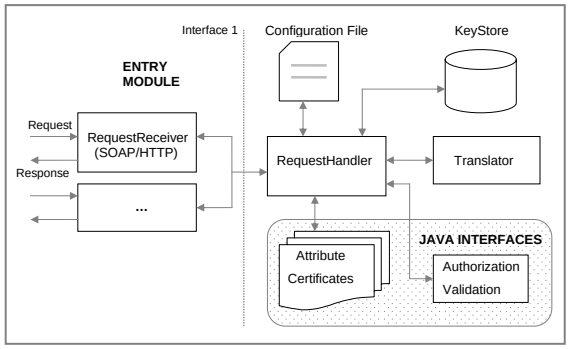

Figure 5: XSAML components

The main components of the proposal are:

- **Entry Module** holds a servlet, named RequestReceiver. This class is implemented to exchange messages with the Service Provider (client) using SOAP over HTTP. Programmers can implement other classes to achieve these tasks. When the RequestHandler processes the request, it sends a response to the Entry Module (in this case RequestReceiver or another class designed).

- **RequestHandler** is the main class for processing the request. It allows programmers to create new classes in the Entry Module; in other words, new ways to exchange messages with the Service Provider.

- Finally, we describe the **java interfaces**. They make the system configurable. These interfaces are used to authorize the requester (interface Authorization), validate requester's certificates (interface validation), and access the attribute certificates (interface AttrCertHandler).

3.2 Integrating XSAML in a system

XSAML must be integrated into a system with specific features. In the above sections we describe how to achieve the full and partial integration of XSAML in a target system.

3.2.1 Full Integration

Figure 6 details the full integration, where XSAML is integrated fully inside the system. XSAML receives SAML Requests from clients. Then, it processes the request exchanging information through the system interfaces. When the response is made, XSAML issues the SAML Response to the client.

As the RequestSender (client) is out of the focus of XSAML, it must be improved before being used. In full integration, programmers only have to implement the interfaces to integrate XSAML. SOAP protocol over HTTP is the communication protocol used to exchange data between XSAML and the clients.

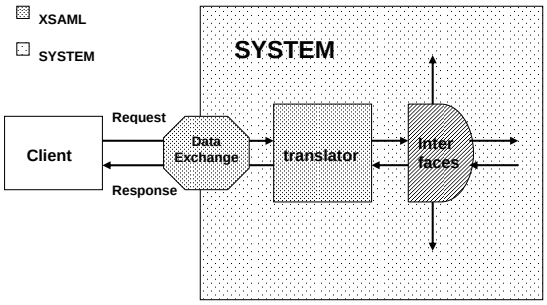

Figure 6: XSAML Full integration

3.2.2 Partial Integration

This scenario shows (figure 7) partial integration. Entry Module classes are not integrated into the system and therefore data exchanging is managed by the system. The system must issue a SAML Request to XSAML and it must process the request through the system interfaces. Finally, the translator XSAML sends a SAMLResponse to the System.

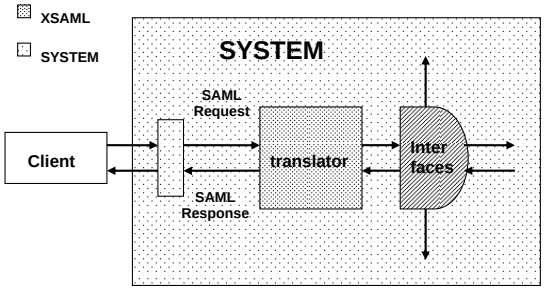

Figure 7: XSAML Partial integration

3.3 Functional Scenarios

Several scenarios can be established by using XSAML technology. In this section, we will show three possible scenarios.

- **Scenario A**: In this scenario, the Service Provider (SP) sends a request with the subject, name and namespace of the attributes (1). The Identity Provider (IdP) issues a request (via AttrCertHandler) to the LDAP server or any attribute certificates repository, specifying the subject, name and namespace of the attributes (2).

  When the AttrCertHandler receives the requested attribute certificates, it sends them to the Identity Provider attached to a namespace (3). Finally, the Identity Provider sends a response to the Service Provider, containing the subject, name, namespace and value of the attributes (4).

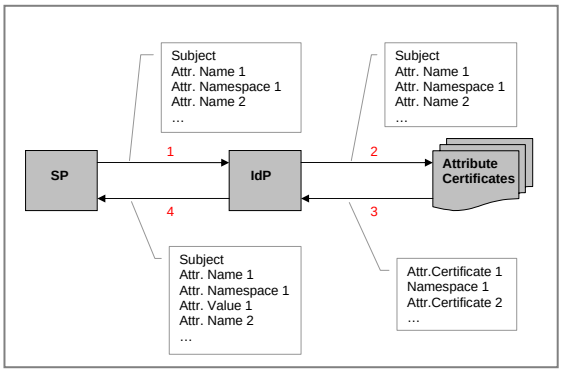

Figure 8: Scenario A

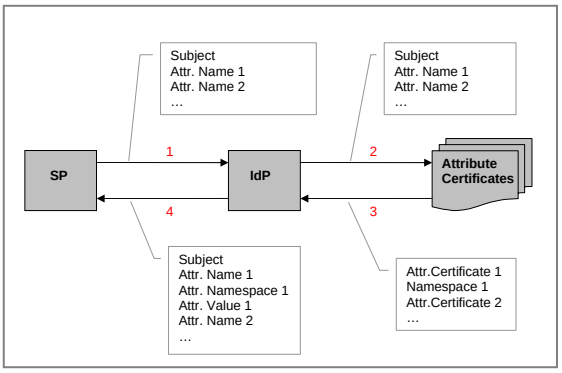

Figure 9: Scenario B

- **Scenario B**: In this scenario, the Service provider sends a request with the subject and the attribute name (1). The Identity Provider issues a request (via AttrCertHandler) to the LDAP server or any attribute certificates repository, specifying the subject and the attribute name. (2).

  When the AttrCertHandler receives the requested Attribute Certificates, it sends them to the Identity Provider attached to a namespace (3). Finally, the Identity Provider sends a response to the Service Provider, containing the subject, name, namespace and values of the attributes (4).

- **Scenario C**: The Service Provider sends a request with the subject and the attribute name (1). The Identity Provider issues a request (via AttrCertHandler) to the LDAP server specifying the subject and the name of the attributes (2).

  When the class AttrCertHandler receives the requested Attribute Certificates, it sends them to the Identity Provider (3). Finally, the Identity Provider sends a response to Service Provider, containing the subject, name and values of the attributes (4).

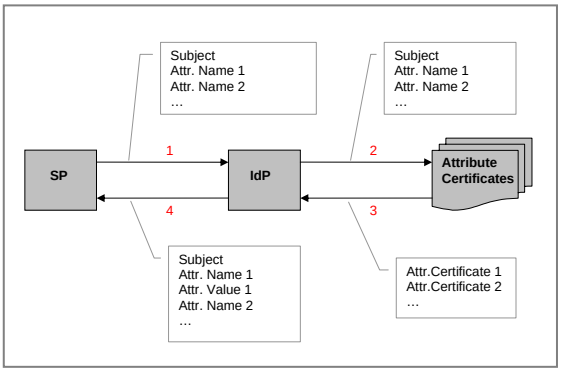

Figure 10: Scenario C

4 Attribute Federation

In this section, a new concept is introduced, Attribute Federation (AF). AF is required when several ubiquitous service providers share a common context. The concept introduced is explained using the following scenario.

Let us suppose a University in which Users are either Professors or Students and in which their devices may act as service providers, e.g. providing class notes, maybe performed either by Students or by Professors. In this case, the University defines some generic attributes with or without parameters, e.g. Student, Enrolled(Subject), Professor,

Teach(Subject) and so on, that help characterizing entities at the University.

These types of attributes are named as *Global Attributes* and are defined and issued by Attribute Authorities. The *Attribute Authority* (AA) could be part of an existing infrastructure such as for instance an X.509 (PMI) [13]. The definition of this type of Attribute must be reduced because in ubiquitous scenarios the global infrastructure should only be used in specific situations. The university is in charge of defining and issuing those attributes to Students and Professors, therefore each university acts as an AA inside the PMI.

Once those basic attributes have been issued, some Professors may need to define new attributes such as,
$Pass\_Test1\_Subject1$ or $Pass\_Global\_Subject1$. These specific attributes can be used to control student progress, access to money grants, awards, access to other subjects, etc. These attributes are named as *Domain Attributes*. The Domain Attributes are defined locally and its meaning is limited to the domain in which they were defined. These type of attributes make the system more dynamic because they do not require the same verification process of Global Attributes, which could be a very expensive process.

In some cases, there are relationships between attributes, i.e. $Pass\_Global\_Subject1$ may imply $Pass\_Test1\_Subject1$ in the case where passing the first test is a requirement for passing the subject. Then, along with the attributes, there is a partial order that defines a simple ontology [19] in the domain of the attribute manager. An ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts. It is used to reason about the objects within that domain. This ontology encodes the relationships between the attributes in the system. Moreover, there may be relationships between attributes in different domains, e.g. different Professors may issue different attributes. In order to make a reference to the attribute domain we use the dot notation. For instance, $Prof1.Pass\_Test1\_Subject1$ is an attribute created by Professor1 and managed by him which states that the first test of Subject1 has been passed. This attribute on its own has only a local meaning in the domain of Professor1.

In the University scenario, the University must provide a server to store the particular attributes defined by its members and also the relationships between them.

As it occurs in many Universities, some subjects may share some topics. Let us suppose for example that the first part of subject1 is equivalent to the second part of Subject2. In this case, the professor of subject2 (Professor2) may rely on the previously defined attributes and define an attribute relationship stating that the first test of subject1 implies passing the second test of subject2. In this way, Professor2 gives the previous attributes defined by professor1 a new meaning outside its context, by uploading the following relation to the university server:

$$Prof1.Pass\_Part1\_Subject1 \rightarrow Prof2.Pass\_Part2\_Subject2$$

This can be also represented using the partial order symbol,

$$Prof2.Pass\_Part2\_Subject2 \leq Prof1.Pass\_Part1\_Subject1$$

In this case, Professor2 is delegating his authorization on passing the second part to Professor1. Then, a student who is trying to convince the University that he has passed the second test of subject2 may show his identity details to the University so it could ask professor2, or may show professor1's attribute stating that he has passed the first test of subject1 together with the attribute relationship

$$Prof2.Pass\_Part2\_Subject2 \leq Prof1.Pass\_Part1\_Subject1$$

signed by professor2. So, the process can be carried out without involving Professor2.

Although Domain and Global attributes are different concepts, they can also be related using a partial order. An AA may decide to transform a Domain attribute into a Global attribute by defining an order relation of the form $AA.Global1 \leq Entity1.Attribute1$. In this way, an AA can delegate some attributes to other entities in the federation by simply linking their attributes with its own. By doing so, Domain attributes could get a global scope that reach anyone who trusts the AA.

We call this process *Attribute Globalization*. On the other hand, individuals can also use global attributes in the definition of their authorization policies, $Entity1.Attribute1 \leq AA.Global1$ is also a valid attribute relationship. We call it an *Attribute Localization*.

In our example, when the University needs to make Professor Decisions "official", it can introduce this relationship in the University Server,

$$University1.Pass\_Subject1 \leq Prof1.Pass\_Global\_Subject1$$

Then, Professor1 is elected as the coordinator of Subject1.

4.1 Attribute Federation Components

In our proposal, the Attribute Federation has two elements, the Attribute Authority and the Attribute Ontology Server (AOS), which can be composed of many subordinated AOS as we will describe in the next section. The AA manages Global Attributes whereas the AOS manages the Domain Attributes. Therefore, the main element that the attribute federation introduces, in contrast with traditional privilege management infrastructures, is the AOS, which is in charge of:

1. Managing Domain Attributes.
2. Storing relationships over Domain Attributes in the system.
3. Checking attribute relationships based on the stored ontology, i.e. performing the trust negotiation autonomously.

Every entity is registered to an AOS which is in charge of checking the relationships over attributes. Registered users trust the AOS not to disclose their attribute relationships neither to cheat them adding fake relationships.

Users may store new relationships in their AOS and/or check whether a relationship exists or not in any of the AOS of the Federation. The kind of relationships a user can upload to the AOS are of the form $User1.Attribute1 \leq User2.Attribute2$, where $User1$ is the ID of the user that is uploading the subscription. We call them attribute subscription and we read it as "attribute $User1.Attribute1$ is subscribed to attribute $User2.Attribute2$". Attribute subscriptions are transitive in the sense that if $Attr1$ is subscribed to $Attr2$ and $Attr2$ is subscribed to $Attr3$, then we can infer that $Attr1$ is subscribed to $Attr3$. Attribute subscriptions is an analogous concept of Li linked roles [9].

By using attribute subscriptions, users can rely on some other user attributes when defining their own authorization policies, but they can not force other users to rely on their own attributes. In order to preserve the privacy of the attribute ontology, a check for a relation of the form $User1.Attribute1 \leq User2.Attribute2$ is only answered to a user owning attribute $User2.Attribute2$.

In figure 11 Alice contacts Bob in order to retrieve the attributes needed to use the service she wants to use. This can be done contacting Bob directly or by checking some other public service used by Bob to publish his authorization policies, e.g. a static web page.

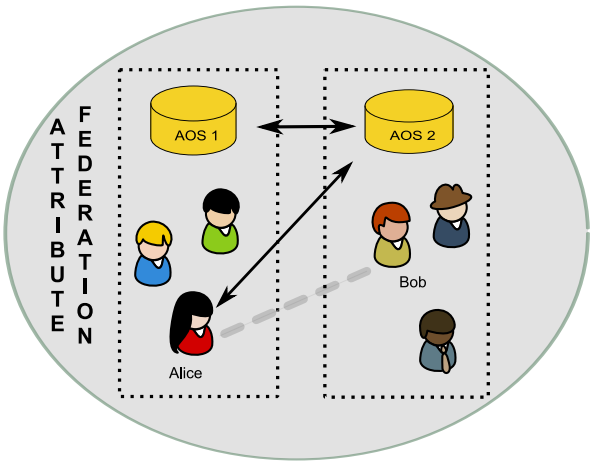

Figure 11: Attribute Federation Scenario

Once Alice knows the required attributes, she looks for attributes that may be related to her own. If the

authorization policy is confidential, Alice will only get a specific domain attribute (in the domain of Bob) linked with the service she wants to access (e.g. $Bob.service1$), and the policy will remain confidentially in the AOS.

Then, she contacts the respective AOS in order to check whether there is a real relation between the candidate attributes. This checking is done by Alice, while Bob is not involved. Once Alice has found a relation between one of her attributes and one of the required attributes, she is ready to initiate a session with Bob in order to use the service. In this way, Bob is only slightly involved in the protocol because he only has to locally check the response of the AOS instead of having to check an attribute matching for all the users trying to use his web service.

In figure 12 the steps of the interaction protocol are detailed:

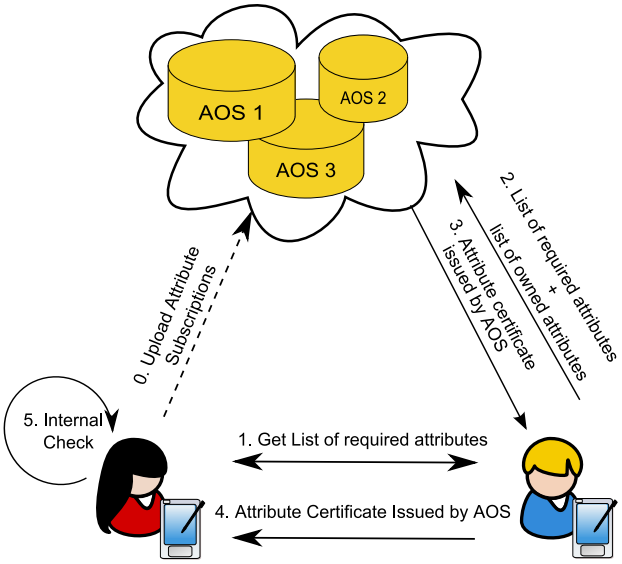

Figure 12: Interaction Protocol

0. Some time before the request, $User2$ sends its attribute subscriptions to the Attribute Federation. This is usually done at the initialization of the services offered by $User2$.

1. In the initial step, $User1$ gets a list with the attributes needed to use the services of $User2$ either by sending an authorization request to $User2$ or by checking them in some repository. In any case, after step 1, $User1$ knows which attributes he needs for being able to make use of the desired services.

2. Then, $User1$ sends the required attributes together with the owned attributes to the attribute federation. Normally, only a few attributes are sent to the federation but the task of deciding which attributes are sent to the federation depends on the context of the request. Anyway, in the worst case it could try with all the attributes.

3. When the federation receives a request, it tries to find a chain of attribute subscriptions so that it could be proved that the attributes requested by $User2$ are subscribed to attributes owned by $User1$. In the case where there is any matching, the federation returns an attribute certificate stating that the required attributes are subscribed to at least one of the attributes provided by $User1$, so $User1$ is entitled to use all the matching attributes. This certificate is issued by the AOS which $User2$ is registered to because it is the only one he trusts.

4. At this stage, $User1$ is able to prove to $User2$ that the requested attributes are subscribed to the attribute it owns without having to send any attribute. So, $User1$ sends the signed attribute certificate to $User2$.

5. $User2$ verifies the signature of the certificate to check if it becomes from its AOS and allows $User1$ to make use of its services in the case where everything is in order.

5 Storing and Updating attribute subscriptions

We try to adopt a syntax as close as possible to the one used in SAML, so our definitions become fairly

integrated with it. In particular attributes are represented easily in SAML as they use the same philosophy than we do.

In SAML, the Attribute element has two XML attributes, namely AttributeName and AttributeNamespace. This idea of namespaces allows organizations to have security attribute names without the fear of any collision between names defined by different naming services. The first one can be identified with our AttributeID and the second one reference, in some way, the user who defines the attribute. So, the following SAML attribute definition is equivalent to the attribute $UserID.AttributeID(Value)$.

```
<Attribute
   AttributeName=AttributeID
   AttributeNamespace=UserID>
   <AttributeValue>Value</AttributeValue>
</Attribute>
```

From the $UserID$ descriptor we should be able to derive a pointer to the AOS in which the subscriptions of this attribute are stored. Then an domain attribute credential is a regular SAML attribute assertion, where the attribute element is defined as before.

An attribute subscription is an attribute statement in which the subject is also an attribute, then the subscription

$$UserID1.AttributeID1(Value1) \leq \ldots$$

$$\ldots UserID2.AttributeID2(Value2)$$

can be represented in a SAML alike notation using the following code,

```
<AttributeStatement>
    <Subject>
        <Attribute
             AttributeName=AttributeID2
             AttributeNamespace=UserID2>
             <AttributeValue>Value2</AttributeValue>
        </Attribute>
    </Subject>
    <Attribute
        AttributeName=AttributeID1
        AttributeNamespace=UserID1>
        <AttributeValue>Value1</AttributeValue>
    </Attribute>
</AttributeStatement>
```

This assertion has to be issued and signed by the user to which the Attribute namespace $UserID1$ points.

As mentioned, an attribute federation can be composed of many AOSs. In this way, attribute subscriptions are spread over the Federation instead of being stored in a central server. Each entity in the federation uses a unique AOS to store its attribute subscriptions, so finding attribute subscription chains involves communication between different AOSs. A trust relationship between AOSs in the Federation is needed, so each AOS trusts other AOS answers to its requests. Ideally, each AOS stores only its own attribute subscriptions, but for performance reasons it may be desirable that each AOS stores a cache of most requested attribute subscriptions connected to their own. Let us revise a sample scenario depicted in figure 13.

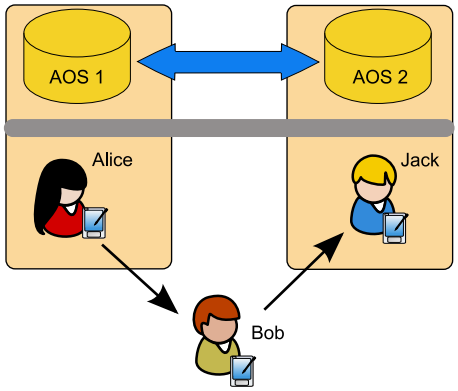

Figure 13: Sample scenario

In this scenario, Alice and Jack offer some services under an attribute federation in which $AOS1$ and $AOS2$ are the corresponding attribute ontology servers for Alice and Jack. Alice has issued some attributes to Bob, as they are friends. Alice has issued Bob an attribute certificate for attribute $Alice.Friend$. Later on, Bob tries to access some of Jack resources. In particular, one of the resources is granted to Jack friends, i.e. owners of attribute $Jack.Friend$. Bob then has the choice to ask the federation if attribute $Jack.Friend$ is subscribed to

attribute *Alice.Friend*.

To do so, Bob has to first prove to the federation that he owns attribute *Alice.Friend* and then send the requested attributes he thinks are connected to the one presented, in the example *Jack.Friend*. The answer is positive when the federation finds a valid chain of attribute subscriptions,

$$Jack.Friend \leq ID\_1.Attribute\_1 \leq \ldots$$

$$\ldots \leq ID\_n.Attribute\_n \leq Alice.Friend$$

and proves to Bob that it exists by sending him an attribute certificate that he could send back to Jack.

If the AOSs stores only attribute subscriptions, the unique way to start searching for this chain is contacting $AOS2$ and asking for all the attributes *Jack.Friend* is subscribed to. Then, the process is repeated for those attributes until attribute *Alice.Friend* is eventually reached.

We can include some redundancy in the data stored in order to achieve a better performance. When a new attribute subscription is uploaded to the federation, the corresponding AOS not only stores the attribute subscription but also notifies (in the case the two attributes in the subscription do not belong to the same AOS) to the other AOS involved, that a new subscription to one of its attributes is going to be added to the federation. In this way, the notified AOS stores for a given attribute, not only its attribute subscriptions but also the set of attributes subscribed to it. When doing this, the search for an attribute federation chain can be done in a bidirectional way.

In fact, the storage of this extra information can be left as optional but the notification should be done in order to allow those AOSs interested in storing the information to do so. Then, if the target attribute belongs to an AOS that stores this extra information, the search can speed up. For the sake of consistency, when an attribute subscription is removed, a notification should be also sent to the AOS of the other attribute, so it can remove this attribute from its records.

6 Conclusions

In this work, we establish a division between Global and Domain attributes. Global attributes are defined in large Infrastructure as X.509 PMI, and are well known attributes. On the other hand, local decisions demand a different approach, based on the definition of local attributes, that we name Domain Attributes. These attributes are not directly managed by using the Infrastructure. So, by using domain attributes, we avoid using the underlying infrastructure which can be computationally expensive in several environments as mobile devices. Moreover, Domain attribute credentials are defined using SAML, an emerging technology based on XML.

The inclusion of domain attributes makes necessary to establish a new concept, called Attribute Federation. The concept is similar to Identity Federation but applied to Authorization sentences, avoiding identity information exchange. In each Federation, a new element has been defined named Attributed Ontology Server (AOS), in which entities store their attributes and the relationships between them, i.e. attribute subscriptions, to simplify the authorization process. The AOS is independent from the underlying PMI and can be easily distributed. It allows linking attributes from different users and reuse them in the definition of authorization policies. In this way, we divide the authorization process in two layers, that is, relation between users implemented by using attribute certificates, and relation between attributes implemented by using attribute subscriptions.

References

[1] S. Landau and J. Hodges. A Brief Introduction to Liberty, August 2002.

[2] Isaac Agudo, Javier Lopez, Jose A. Montenegro A Graphical Delegation Solution for X.509 Attribute Certificates ERCIM News. SPECIAL THEME: Security and Trust Management No. 63, October 2005, pp. 33-34. ISSN: 0926-4981

[3] B. Kaliski. A Layman's Guide to a Subset of ASN.1, BER, and DER, 1993.

[4] Isaac Agudo, Javier Lopez and Jose A. Montenegro A representation model of trust relationships with delegation extension. In *3rd International Conference on Trust Management, iTrust 2005*, volume 3477 of *Lecture Notes in Computer Science*, pages 116 – 130. Springer, 2005.

[5] D. Mundy and D. Chadwick An XML alternative for perfomance and security: ASN.1. In *IEEE IT Professional*, 6(1) pages 30–36. IEEE Computer Society Press, 2004.

[6] E. Yuan and J. Tong. Attributed based access control (ABAC) for web services. In IEEE International Conference on Web Services (ICWS'05), pages 561–569, 2005.

[7] William H. Winsborough, Kent E. Seamons and Vicki E. Jones. Automated trust negotiation. In DARPA Information Survivability Conference and Exposition. volume I, pages 88–102, IEEE Press. January, 2000.

[8] William H. Winsborough, Jay Jacobs. Automated Trust Negotiation in Attribute-based Access Control DISCEX (2) 2003: 252-

[9] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.

[10] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maeler and F. Yergeau. Extensible Markup Language (XML) 1.0. Fourth Edition. W3C Recommendation. 16 August 2006

[11] Isaac Agudo, Javier Lopez and Jose A. Montenegro Graphical Representation of Authorization Policies for Weighted Credentials In 11th Australasian Conference on Information Security and Privacy. (ACISP'06), pp. 383-394. LNCS 4058, Springer. Melbourne, Australia. July 2006.

[12] Arnaud Sahuguet, Stefan Brands, Kim Cameron, Cahill Conor, Aude Pichelin, Fulup Ar Foll, Mike Neuenschwander. Identity management on converged networks: a reality check. In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 747-747.

[13] ITU-T Recommendation X.509. X509.Information technology Open systems interconnection. The Directory: Public-key and attribute certificate frameworks, March 2000.

[14] Kent E. Seamons, Marianne Winslett and Ting Yu. Limiting the disclosure of access control policies during automated trust negotiation . In Proceedings of the Symposium on Network and Distributed System Security, (NDSS'01). February, 2001

[15] William H. Winsborough, Ninghui Li. Safety in automated trust negotiation ACM Trans. Inf. Syst. Security 9(3): 352-390 (2006)

[16] J. Hughes. SAML technical overview. Oasis. Document id sstc-saml-tech-overview-1.1-cd, 2004.

[17] J. Hughes. SAML technical overview. Oasis. Document id sstc-saml-tech-overview-2.0-draft-03, 2005.

[18] M. Erdos and S. Cantor. Shibboleth-Architecture DRAFT v05, May 2002.

[19] Thomas Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies Vol. 43, Issues 5-6, November 1995, p.907-928.