I. Agudo, J. Lopez, and J. A. Montenegro, "Attribute delegation in ubiquitous environments", *3rd international conference on Mobile multimedia communications (MobiMedia 07)*, pp. 43:143:6, 2007.
NICS Lab. Publications: https://www.nics.uma.es/publications

Attribute Delegation in Ubiquitous Environments

Isaac Agudo
Computer Science Department
E.T.S. Ingenieria Informatica
University of Malaga,Spain
isaac@lcc.uma.es

Javier Lopez
Computer Science Department
E.T.S. Ingenieria Informatica
University of Malaga,Spain
jlm@lcc.uma.es

Jose A. Montenegro
Computer Science Department
E.T.S. Ingenieria Informatica
University of Malaga,Spain
monte@lcc.uma.es

ABSTRACT

When delegation is implemented using the attribute certificates in a Privilege Management Infrastructure (PMI), this one reaches a considerable level of distributed functionality. However, the approach is not flexible enough for the requirements of ubiquitous environments. Additionally, 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. We also introduce in the paper the concept of Attribute Federation which is responsible for supporting domain attributes and the corresponding ontology.

1. INTRODUCTION

Much has been said about identity management [1] 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 the 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 in charge 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 defining 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 [2, 3, 4, 5, 6] the concepts of *automated trust negotiation* (ATN) and the concept of Attribute Based Access Control (ABAC) are well explained. In these works a new access control model, named *Attribute Based Access Control* (ABAC) is introduced, which defines authorization 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

Mobimedia'07 Month 8, 2007, Nafpaktos, Aitolokarnania, Greece

follows. In section 2, related work is presented from both the theoretical and more practical point of view. In section 3 we introduce the concept of attribute federation and justify its need. Section 4 details some implementation issues and, finally, section 5 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 [7]. 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. $A.R \leftarrow B_1.R_1 \cap B_2.R_2 \cap \cdots \cap B_n.R_n$: 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.*

```
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

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 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 [9] software implements the OASIS SAML v1.1 specification [10], providing a federated Single-SignOn and attribute exchange framework. Liberty Alliance [12, 13] is also a federation solution based on SAML.

Version 2 of SAML, in the Technical Overview (to date in draft 10 [11]), presents an Attribute Federation scenario. They think about 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 [11].

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 which means 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 the 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.

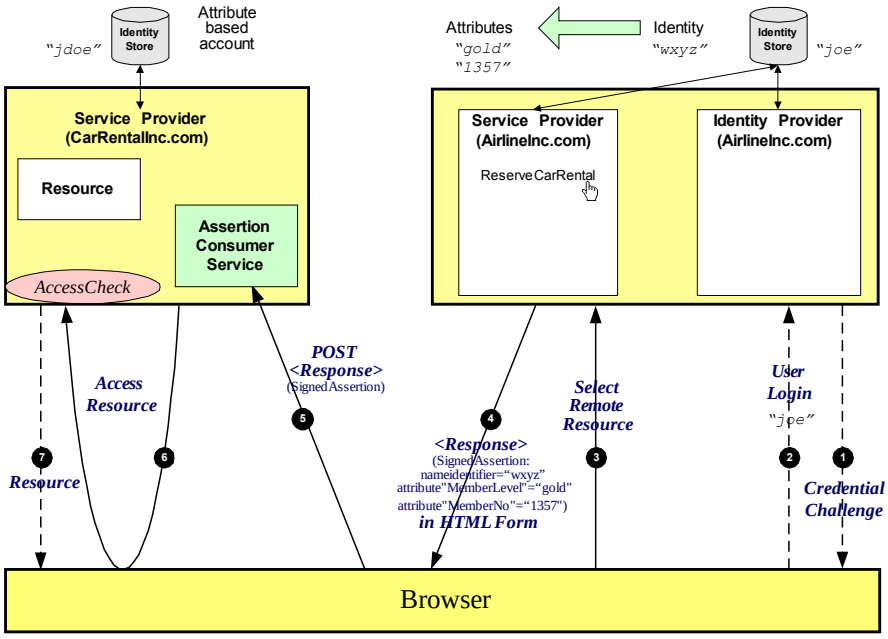

Figure 2: SAML attribute federation scenario

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 the initial one, in the end it has to recover the identity of the requester in order to be able to check the request. So, 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. ATTRIBUTE FEDERATION

In this work 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 think of a University in which Users are either Professors or Students and in which their devices may act as service provider , e.g. providing class notes may be done 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 working at the University.

These types of attributes are named as *Global Attributes* and are defined and issued by Attribute Authorities. The Attribute Authority could be part of an existing infrastructure such as for instance an X.509 (PMI) [8]. 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 and so on. 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 [14] 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 Pro-

fessor1 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 happens 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.

3.1 Attribute Federation Components

In our proposal, the Attribute Federation has two elements, the Attribute Authority (AA) 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 are an analogous concept of Li linked roles [7].

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 3 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.

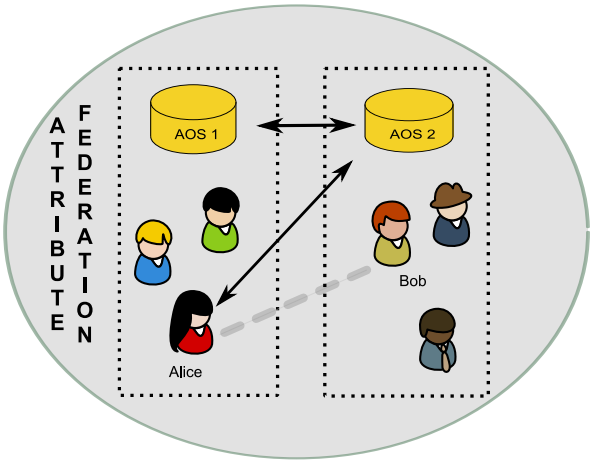

Figure 3: Attribute Federation Scenario

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 an 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 in the AOS confidentially kept.

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 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 4 the steps of the interaction protocol are detailed.

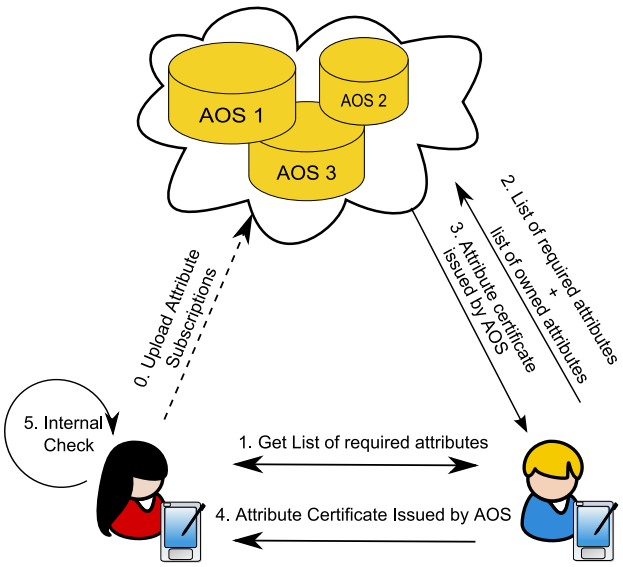

Figure 4: Interaction Protocol

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 to decide 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 register 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.

4. STORING AND UPDATING ATTRIBUTE SUBSCRIPTIONS

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 trust 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 revise a sample scenario depicted in figure 5.

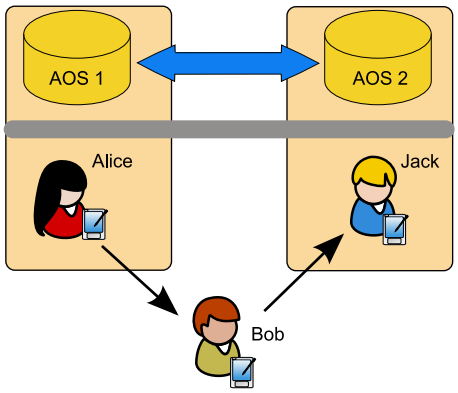

Figure 5: Sample scenario

In this scenario, Alice and Jack offer some services under an attribute federation in which $AOS1$ and $AOS2$ are the respective 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,

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

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

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

If the AOSs only store attribute subscriptions, the only 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 store 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 be accelerated. 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.

5. 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.

The inclusion of local attributes makes necessary to establish a new concept, 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.

6. REFERENCES

[1] 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.

[2] 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.

[3] 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

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

[5] 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.

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

[7] 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.

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

[9] http://shibboleth.internet2.edu/

[10] http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf

[11] http://www.oasis-open.org/committees/download.php/20645/sstc-saml-tech-overview-2-draft-10.pdf

[12] http://www.projectliberty.org/

[13] Andrew Shikiar. Introduction & Overview Talk . Mobile Deployment Workshop 3GSM Barcelona 2007.

[14] 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.