Authorisation using Attributes from Multiple Authorities

David W Chadwick
Computing Laboratory, University of Kent, UK,
D.W.Chadwick@kent.ac.uk

Abstract

As attribute based authorisation infrastructures such as XACML gain in popularity, linking together user attributes from multiple attribute authorities (AAs) is becoming a pressing problem. Current models and mechanisms do not support this linking, primarily because the user is known by different names in the different AAs. Furthermore, linking the attributes together poses a potential risk to the user's privacy. This paper provides a model and protocol elements for linking AAs, service providers and user attributes together, under the sole control of the user, thereby maintaining the user's privacy. The paper also shows how the model and protocol elements can be implemented using existing technologies, namely relational databases or LDAP directories, and the SAML protocol.

1. Introduction

Enabling local attribute based authorisation, using attributes issued by a remote attribute authority, is possible today using systems such as Shibboleth [1]. In the grid world, this is also possible, using attributes issued by a VO attribute authority [2]. However, a problem arises if a Service Provider (SP) wishes to use attributes issued by multiple Attribute Authorities (AAs) to grant access to its resources. For example, a service provider might have a policy that a member of IEEE and a US university will get access to a premium service; or a medical research database access policy might mandate that a user is a medical practitioner, as certified by the General Medical Council, and an academic at a UK university. Policies such as these are difficult to implement today, primarily because users typically have different identities provided by each AA. A practical example of this problem is now arising as people are experimenting with providing Shibboleth access to Grid services [5]. Users have attributes provided by both their Shibboleth AA (called Identity Provider or IdP) and by their VO AA, and both sets of attributes are needed in order to gain access to the Grid resources. Policy based access control systems such as PERMIS [3] can enforce such policies, but they have the restriction that the user must have the same globally unique LDAP distinguished name at each AA, so that the different attribute certificates can be collected and collated together. Such restrictions on naming are not realistic today, since each AA typically has its own naming rules and name forms. Whilst this restriction might become technically feasible in the future, once national ID cards are commonly in use, and AAs use these identifiers to name the users, privacy concerns are likely to ensure that these names will never be commonly used. Thus we need a scheme whereby a user can have different and unrelated identities at each AA, but if he or she so chooses, can link these identities together so that the resulting pooled set of attributes can be used to gain access to resources from different service providers. Furthermore, due to privacy concerns, we need to tightly control this attribute linking to ensure that no single third party can know which attributes the user has linked together. Some of the consequences of this restriction are:

- when a user links the attributes of several AAs together, the only single person who knows about all these linkings is the user himself,
- a SP should not be able to link a user's request for a service to the identity of the user without the cooperation of the AA,
- a SP should not be able to link together the identity of the user on multiple sequential requests,
- a SP should not be able to determine who the user is on requests that provide attributes from multiple AAs.

We present a solution to the problem presented here. The rest of this paper is structured as follows. Section 2 describes the model for linking AAs, SPs and user attributes together whilst respecting the user's privacy. Section 3 describes the protocols that support the linking of the various entities together, both statically and dynamically. Section 4 indicates how the protocol and model can be implemented in existing technologies, namely relational databases or LDAP directories and the SAML protocol. Finally section 5 concludes and briefly describes our plans for the future.

2. The Linking Model

In the model being proposed here, AAs are statically linked together in pairwise relationships (termed partnerships), to facilitate their users linking their attributes together. Zero, one or more users may statically link together their attributes from an AA partnership. The two AA from a partnership may be in zero, one or more federations, and the number of federations that each is a member of may change dynamically without affecting their partnership. A federation is formed by AAs and SPs joining together for business reasons, and the memberships of these federations can continually change. When a user initiates a session with a federation, the user may choose to dynamically link together her attributes from one or more AA partnerships, and these AAs are then dynamically linked with one or more SPs for the duration of the user session. In order for these linkages to work correctly there are a number of underlying assumptions, described below.

2.1 Underlying Assumptions

The model being proposed relies on the following assumptions being true

1. Each AA and SP has a PKI key pair which is used to identify it to the other participants
2. Each AA can choose its own naming scheme with which to identify its users. It may use a pre-existing global naming scheme, or use its own local one.
3. Users are named according to the chosen naming scheme and their attributes are stored by the AA along with their name.
4. Each AA has access to an Authentication Service which can authenticate users against their name.
5. When an AA issues an attribute certificate or attribute assertion (we treat these terms as being synonymous) on behalf of a user, we make no assumptions about the user identity that is inserted in it. The AA can choose to use the actual name of the user or a pseudonym. It may use the same or different identifiers for the same user in different certificates that it issues.

2.2 The AA Partnership Model

The model assumes that no AA is required to link with any other. They can all freely make the choice whether to link or not. However, when AAs do link together they do so in a pair wise relationship only, termed a partnership. Each AA partnership is unrelated to any other AA partnership that either AA may enter into. The partnership is also independent of any federation that either AA may be a member of. The partnership is cryptographically cemented by the AA pair sharing a strong symmetric key e.g. a 256 AES key. The model does not dictate any symmetric algorithm or key size. The two AAs decide this amongst themselves, according to their requirements. The model does not dictate how this key is generated and shared. Any secure mechanism can be used e.g. Diffie Hellman or encrypting with the public key of the recipient, or manual exchange. The model assumes that this strong secret exists once the partnership has been established, and when the partnership terminates the shared secret will be destroyed. The AA pair may dynamically change their shared secret, but it is outside the scope of the model how or when this is done.

Protocol interactions defined in this paper rely on this strong shared secret, and will carry tokens encrypted with it between the AA pair, either directly, or via one or more intermediaries. Successful decryption of a received token is deemed to be sufficient proof that the token was generated by the other partner in the pair.

When two AAs decide to form a partnership they are doing so for the express purpose of allowing their users to link their attributes together. Each AA is acknowledging that when a linked user retrieves his attributes (either directly or indirectly) from one partner, in order to gain access to a particular federation service, the other partner will also release the linked user's attributes to the same service providing that:

- it receives a token from the service which was freshly generated by its partner, and
- the partners and the service provider are all part of the same federation.

2.3 The User Account Linking Model

Users may individually decide to link together their attributes from two AAs in a partnership. AAs should not automatically and unilaterally link together the attributes of their users, since this could violate the privacy of their users. The only person who has sufficient knowledge (and is therefore authorised) to link together the attributes held under one name in one partner AA, to the attributes held under another name in the other partner AA, is the person who can successfully simultaneously authenticate to each AA using the respective user names used by each AA. How this is achieved is described in section 3.1.

2.4 The AA-SP Federation Model

The model assumes that no AA or SP is required to federate with any other. They can all freely make the choice whether to federate or not, according to business reasons. However, when AAs and SPs do federate together, they do so in groups of any arbitrary size, providing that each group contains at least one AA and one SP. This is conceptually the same as Shibboleth federations and Liberty Alliance circles of trust. When a federation is formed, the members share their public key certificates or root CA certificate(s) so that they can each communicate with the others using digital signatures to authenticate themselves. AAs and SPs may be members of many different federations simultaneously. Each federation must have a unique federation identity, FId, so that each member will know the context of any particular message exchange between themselves. This is because two federation members, e.g. an AA and an SP, may be in several different federations simultaneously, and each federation may be bound by different rules and policies.

3. The Protocols

The protocols comprise a user attributes (UA) linking protocol, a primary AA-SP protocol, and a secondary AA-SP protocol. We do not define a protocol for creating AA partnerships, since there are several pre-existing protocols for exchanging symmetric keys. The UA linking protocol is initiated by a user in order to link together the attributes held under the two different names (or identities) in an AA partnership. The primary AA-SP protocol is directly or indirectly initiated by the user when she wants to gain access to a service provided by the SP. The secondary AA-SP protocol is initiated by the SP when it is prepared to merge the attributes from multiple AAs in order to grant the user access to its resources.

3.1. The UA linking protocol

We assume that a user has identity ID_1 with AA_1 and the unrelated identity ID_2 with AA_2. Each identity has a set of one or more attributes $[A_1, A_2 .. A_n]_i$ associated with it. It is these two sets of attributes that will be linked together by the UA protocol. We further assume that each AA will display on its web site the list of other AAs that it has formed AA partnerships with so that a user can see which of its attribute accounts may be linked together.

Step 1. The user authenticates to AA_1 using identity ID_1 and opens up a session.

Step 2. The user authenticates to AA_2 using identity ID_2 and opens up a session.

The above steps can be implemented in several different ways. We envisage that the simplest way from a user's perspective is to establish an Https (TLS/SSL) connection with each AA from his web browser, using links provided on each AA web site that point to their AA partners. Once the user has two open browser windows, he will input his respective usernames and passwords into each prompt. These are then transferred via encrypted links to each AA.

Note. At this time neither AA knows that the user has established a secure connection with the other AA, since the two browser sessions are unrelated.

Step 3. The user transfers the same long random secret to each AA across the encrypted link. The simplest way for the user to enact this is to simply cut a long random string from any document currently open on his desktop and paste this into the same empty field in each of the two browser windows that are currently connected to each AA. We suggest a field length of 64 characters is long and strong enough.

Step 4. When an AA receives the long random number, it opens up a service port to listen for incoming messages and displays this URL back to the user, along with the prompt "Please provide this URL to the other attribute authority you wish to link to this one". It also displays the prompt "Please provide the URL of the other attribute authority" along with an empty field for the user to complete.

Step 5. The user gives the URL of AA_1 to AA_2 and the URL of AA_2 to AA_1. The user can simply cut and paste the URL displayed in browser window 1 into browser window 2, and vice versa.

Note. After step 5 has been completed, each AA knows the details of the other AA whom the user has chosen to link his attributes with. Assuming that the user has correctly chosen two AAs who have a partnership with each other, then each AA will be able to identify this partnership and the corresponding strong symmetric key that has been shared between them. Otherwise each AA can terminate the session with the user with an appropriate error message.

Step 6a. (Enacted by both AAs in parallel). As soon as an AA (AA_i) receives the URL input by the user, it sends the following message to that URL

$dateTime_{AAi}$, $nonce_{AAi}$ $\{Hash(LRN), dateTime_{AAi}, nonce_{AAi}, RID_{AAi},\}Enc_{SSK}$

Where
$dateTime_{AAi}$ is the current date and time for AA_i

$nonce_{AAi}$ is a random nonce generated by AA_i. The dateTime and nonce are used to prevent replay of this message.
Hash(LRN) is the SHA1 hash of the long random secret input by the user,
RID_{AAi} is a random identifier for ID_i generated by AA_i and stored along with the record for this user,
$\{..\}\ Enc_{SSK}$ represents encryption of the message enclosed in { } using the strong secret key shared between AA_1 and AA_2.

Note. Since the destination URL has been specifically generated by the receiving AA, it can be sure that only the trusted AA from its partnership will contact it using this URL. Therefore no further identification information is needed in this message. Successful decryption is proof enough that the partner AA is communicating with it, and the dateTime and nonce allow the recipient to confirm that this message is not a replay of a previous communication with the partner AA.

Step 6b. (Enacted by both AAs in parallel). Upon receipt of the message from Step 6a, the receiving AA (AA_j) decrypts the message, and compares the received hash to a hash of the long random number given to it by the user. If they are identical, then AA_j stores the random identifier provided by AA_i (and used by AA_i to identify the user) along with its record for this user.

Note that Steps 6a and 6b may be reversed for one of the AAs i.e. it may receive the message in Step 6b before it sends the message in Step 6a, in which case it sends message 7 instead of message 6a. After step 6 (or 6 and 7) has been completed then both AAs have linked the user's two sets of attributes together and have exchanged new random identifiers to identify this user. Each AA's record for this user will now contain

ID_i, PW_i, $[A_1, A_2 .. A_n]_i$, RID_{AAi}, AA_j, RID_{AAj}

Step 7. (Optional. Only enacted instead of step 6a if step 6b occurred first). AA_j generates its own random identifier RID_{AAj} to identify the user, and sends the following message to AA_i

$\{Hash(LRN), dateTime, nonce, RID_{AAi}, RID_{AAj}\}Enc_{SSK}$

Recovery. In the event of a protocol failure, and one of the messages 6a or 7 not being received by the intended recipient, then the intended recipient should resend message 6a, and wait for message 7 to be returned. In the event that one of the AAs receives message 6a twice, it should respond with message 7.

3.2 The Primary AA-SP protocol

The Primary AA-SP protocol piggybacks on any existing AA-SP protocol such as SAML, Shibboleth, Liberty Alliance etc. One such suggested implementation is given in section 4. The user contacts the SP, authenticates to it[1] in the usual way using an identity that is linked to the primary AA. The SP contacts the primary AA to ask for the user's attributes, and the user indicates to the primary AA which attributes should be returned, as well as which linked AA accounts she wants to be contacted for their sets of attributes to be returned as well. The primary AA returns the primary attributes along with the following set of tokens:

$$AA_1\ \{nonce, timeDate, RID_{AA1}\}Enc_{SSKP1,}\ [AA_2\ \{nonce, timeDate, RID_{AA2}\}Enc_{SSKP2} \ldots AA_i\ \{nonce, timeDate, RID_{AAj}\}Enc_{SSKPj}] \qquad (1)$$

where $\{..\}Enc_{SSKPi}$ represents encryption of the message enclosed in { } using the strong secret key shared between the primary AA and $AA_{i.}$ Conceptually, these tokens are *references* to remote AAs. These tokens inform the SP that the user wishes to use additional attributes that can be obtained from the referenced set of AAs. This set of references can be as long as required by the user. Each element in the set comprises the name of the AA and an encrypted token that should be sent to that AA. The encrypted token contains the unique random identifier used by that AA to identify the user, as well as a nonce and current timeDate stamp. The latter form a dual purpose. Firstly they allow the recipient AA to detect the replay of messages, and secondly they ensure that the SP will never get the same token twice, even though it always contains the same random user id. Thus the SP will not be able to link consecutive user requests together to determine which users have linked attributes in which AAs, unless the primary AA always uses the same identifier to identify the same user with the SP, in which case the SP will know that user x has some attributes in AAs 1 to i, although it wont know the identity of the user with those AAs.

Note that any additional protection of the above set of tokens as they are transferred from the primary AA to the SP, will be enacted in the same way as protection of the primary attributes, since they are carried in the same message. If the primary attributes are transferred in the clear, then the names of the

[1] In Shibboleth the user is redirected to his home site (IdP and AA) where he authenticates and is then redirected back to the SP, who then contacts the AA to ask for the user's attributes.

linked AAs will also be transferred in the clear, but we assume that the latter is less of a privacy issue than the former. If the primary attributes are protected, e.g. via an SSL connection or web services security, then the tokens will be protected in the same way.

3.3 The Secondary AA-SP protocol

It is important to note that attribute merging is a function of the SP. It decides whether it needs to link together user attributes from different AAs or not, in order to grant access to the user. If merging is not necessary, due to the SP's policies, then the SP will simply ignore the set of tokens provided in the primary AA-SP protocol and grant (or deny) access to the user based on her attributes from the primary AA. If however the SP knows that multiple attributes from multiple authorities are necessary in order to grant access, then it will act on the tokens provided in the primary AA-SP protocol, and contact the AAs that it deems to be necessary.

The SP and the primary AA, AA_P, are obviously in the same federation, FId, otherwise the user would not have gotten this far in the transaction. But one or more of the secondary AAs may not be in the same federation as the primary AA and SP, or may be in multiple different federations with them. It is not a requirement of the model that each AA knows which federation(s) its partner AAs are in. Thus it is incumbent upon the SP to notify the secondary AAs of the federation which is currently being utilized. The SP sends a normal attribute request message to each secondary AA (e.g. AA_1) that is in the current federation, FId, and piggybacks the following parameters in that message:

$$\text{FId}, AA_P \{\text{nonce}, \text{timeDate}, RID_{AA1}\}Enc_{SSKP1} \qquad (2)$$

This message informs the recipient AA (AA1) of its partner AA (AA_P). This allows the recipient AA to select the appropriate secret key with which to decrypt the token. Once decrypted, the recipient AA can determine if the message was replayed or not, and if not, use the random identifier RID_{AA1} (that it previously generated) to locate the user's attributes. These are then returned in the normal way in the attribute response message.

Protection of the attribute request and response messages will be performed as usual. No additional requirements are placed on this by the secondary AA-SP protocol. Thus SSL or web services security might be used, as applicable to the application.

The SP will repeat this protocol for each linked AA it deems to be necessary. Once the attributes have been received, the SP can merge them and use them all in making its access control decisions.

4. Implementation

4.1 The Attribute Database

Attributes are normally stored in either relational databases or LDAP directories. The model described above can be implemented in either as follows.

4.1.1. Relational Database

Assuming that there is an existing attribute table containing the rows: userPrimaryKey, AttributeType, AttributeValue, for example:

12345, ID, Fred26
12345, Age, 45
12345, PW, ****
12345, Role, Project Manager
45678, ID, JohnW
45678, Role, Engineer
etc,

then two new tables need to be defined. The first contains details of the AA partnerships, and stores the AA partner names along with the shared secret keys[2]. The second contains details of all the user account linkages, using the tuple: userPrimaryKey, localRID, AA_j, RID_{AAj} , for example:

12345, SA8NOREYS…, cs.kent.ac.uk/atauth, RE7CP2YLZ…
12345, FSZN0TR5CL…, aa.bat.com/sts/issue, SP9DCLBYT…
45678, S97CHWT7A…., cs.kent.ac.uk/atauth, CUA6GL1S..
etc.

When the AA receives an SP request for a user's attributes, and the user indicates that linked attributes from one or more AAs should also be returned, then it looks up the user's primary key in the new table, matches the AAs with those requested by the user, and constructs message (1) from this data and the secret keys of the partners.

4.1.2. LDAP Directory

The AA partnership secret keys could be stored in LDAP by creating an LDAP entry for each AA partner,

[2] For security reasons the AA may decide to store the partner secret keys in another place, e.g. in a file encrypted to a strong password known to the AA administrator.

and storing the secret keys in these, or they could be stored elsewhere e.g. in a file encrypted by a strong password known to the AA administrator.

There are two ways of implementing user account linkages in LDAP. In method 1, each user's entry contains a set of subordinate entries, each containing the details of one user account link. A new LDAP schema is needed for this type of entry, comprising an object class (accountLink) and three attribute types (localRID, aa, remoteRID). In method 2, a new complex attribute type (accountLink) is defined whose values comprises a concatenation of the three string values localRID+aa+remoteRID.

4.2. The Primary AA-SP protocol using SAML

SAML is expressly designed with extension points to allow new protocol elements to be added. In the Global Grid Forum draft "Use of SAML for OGSI Authorisation" [6], there is already a SubjectAttributeReferenceAdvice element, which specifies that "the designated attributes associated with the specified subject may be obtained from the referenced URI". We therefore propose to use this element to carry the AA-SP protocol. The element contains the URI of the AA, and one or more saml:AttributeDesignators. The latter comprises an attribute namespace (a URI) and an attribute name. We define the following namespace: http://sec.cs.kent.ac.uk/namespaces/attributes
and attribute name: encryptedIdentifier. The base64 encoding of {nonce, timeDate, RID_{AA1}}Enc_{SSKP1} is used as the attribute name.

4.3. The Secondary AA-SP protocol in SAML

We use the SAML AttributeQuery element to carry the Secondary AA-SP protocol. This element comprises an optional Resource element and one or more AttributeDesignators. The Resource element is used to hold the Federation identifier, FId, and the Attribute Designator is copied from the same field of the Primary AA-SP protocol.

5. Conclusions

In this paper we have defined a secure way of linking user attributes together from multiple attribute authorities, in such a way that only the user is aware of the linkages, and the AAs can only become aware of the linkages by conspiring or collaborating together. By linking attributes together in this way, the user can then gain access to resources whose access control policies require possession of attributes from multiple AAs. Even though the service provider is responsible for collecting the linked attributes, it is not able to discover which user has them, without the collaboration of the involved AAs. In this way the user's privacy is protected. We have defined a general model for linking the user attributes together, the cryptographic protocol elements that are needed to securely establish the linkings, and we then indicate how the model and protocols can be implemented using existing technologies, namely relational databases or LDAP directories, and the SAML protocol.

Our future plans are to standardise this model and protocol, and add the standardised version to the Shibboleth open source software.

6. References

[1] S. Cantor. "Shibboleth Architecture, Protocols and Profiles", Working Draft 02. 22 September 2004, see http://shibboleth.internet2.edu/

[2] Alfieri R, et al. "VOMS: an authorization system for virtual organizations", 1st European across grids conference, Santiago de Compostela. Available from: http://grid-auth.infn.it/docs/VOMS-Santiago.pdf; 13-14 February 2003.

[3] D. W. Chadwick, A. Otenko, E. Ball. "Role-based access control with X.509 attribute certificates". *IEEE Internet Computing*, March-April 2003, pp.62-69.

[4] OASIS. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", 15 January 2005

[5] Tom Barton, Jim Basney, Tim Freeman, Tom Scavo, Frank Siebenlist, Von Welch, Rachana Ananthakrishnan, Bill Baker, Kate Keahey. "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy". To be presented at NIST PKI Workshop, April 2006.

[6] Von Welch, Rachana Ananthakrishnan, Frank Siebenlist, David Chadwick, Sam Meder, Laura Pearlman. "Use of SAML for OGSI Authorization", Aug 2005, Available from https://forge.gridforum.org/projects/ogsa-authz