Chadwick, David W., Otenko, Sassa and Xu, Wensheng (2005) *Adding Distributed Trust Management to Shibboleth.* **In: NIST 4th Annual PKI Workshop. . pp. 3-14. , Gaithersberg, USA**

Adding Distributed Trust Management to Shibboleth

David Chadwick, Sassa Otenko, Wensheng Xu
University of Kent, Computing Laboratory, Canterbury, England, CT2 7NF

Abstract

This paper analyses the simplicity of the trust model adopted by the Shibboleth infrastructure and describes an enhanced distributed trust model and authorisation decision making capability that can be implemented by using X.509 attribute certificates and a Privilege Management Infrastructure such as PERMIS. Several different combinatorial approaches can be taken, depending upon the trust models adopted by the Shibboleth target and origin sites, and each of these are described. The paper also discusses whether user privacy, which is strongly protected by Shibboleth, is bound to be weakened by the use of X.509 attribute certificates rather than simple attributes, and concludes that this does not have to be the case.

1. Introduction

Shibboleth [1] is a distributed web resource access control system that allows federations to co-operate together to share web based resources. It has done this by defining a protocol for carrying authentication information and user attributes from a home site to a resource site. The resource site can then use the attributes to make access control decisions about the user. The Shibboleth project is run by the Internet2 consortium in the USA, and universities throughout the USA and Europe (at least) are now starting to build experimental services based upon it.

At the heart of Shibboleth is a trust model that allows the members of a federation to cooperate together. This trust model, whilst functional, is somewhat limited. Basically each Shibboleth target resource site trusts each Shibboleth origin (home) site in the federation, so that whatever assertions – authentication or authorisation – are digitally signed by the origin site, they will be believed and trusted by the target site. There is little scope for differentiation between authentication authorities and attribute authorities, or for allowing more sophisticated distribution of trust, such as static or dynamic delegation of authority.

Another limitation of the Shibboleth infrastructure is that it only provides a basic access control decision making capability. Whilst this is adequate for many use cases, it lacks the flexibility and sophistication needed by many applications, for example, to make access control decisions based on role hierarchies or various constraints such as the time of day or separation of duties.

We realised that both these limitations could be addressed by integrating an X.509 Attribute Certificate (AC) Privilege Management Infrastructure (PMI) [3] with Shibboleth. PERMIS [2] is one such infrastructure that has already been successfully integrated into Grid application target sites [4] to support the distributed management of trust. PERMIS incorporates a sophisticated policy controlled RBAC access control decision engine (also called a policy decision point (PDP)). The PERMIS PMI has been used to implement distributed trust management in Shibboleth.

The rest of this paper is structured as follows. Section 2 provides an overview of Shibboleth. Section 3 introduces the more sophisticated distributed trust model that we

wanted to introduce into Shibboleth. Section 4 describes how the trust model can be implemented using an X.509 PMI such as PERMIS. Section 5 describes the different combinations of X.509 ACs, attributes, and the PERMIS PDP that may be integrated with Shibboleth to provide the desired trust models of the Shibboleth target and origin sites. Section 6 discusses user privacy issues and section 7 discusses revocation and performance issues that arise with using X.509 ACs. Finally Section 8 concludes.

2. Overview of Shibboleth

Shibboleth is a web based middleware layer that currently makes use of SAMLv1.1 [5] for encoding some of its messages. When a user contacts a Shibboleth resource site from their browser, requesting access to a particular URL, Shibboleth single sign on and access control takes place in two stages:

– In stage one the resource site redirects the user to their home site, and obtains a handle for the user that is authenticated by the home site
– In stage two, the resource site returns the handle to the attribute authority of the home site and is returned a set of attributes of the user, upon which to make an access control decision.

In a large distributed open environment stage one has a number of complications. Firstly how does the resource site know where the user's home site is? Secondly, how can the resource site trust the handle that is returned? The answer to these two questions is surprisingly simple, and is part of the Shibboleth trust model. When the user first attempts to access a resource site, he/she is redirected to a Where Are You From? (WAYF) server, that simply asks the user to pick his/her home site from a list of known and trusted home (Shibboleth origin) sites. The target site already has a pre-established trust relationship with each home site, and trusts the home site to

authenticate its users properly. This is facilitated by the exchange of public key certificates or the use of a common trusted root Certification Authority. In the latter case both sites will have been issued with a certificate by the root CA (or one of its subordinates). When a digitally signed SAML message[1] arrives from the home site, such as one containing a user handle, this can be validated and trusted by the resource site.

After the user has picked his/her home site, their browser is redirected to their site's authentication server and the user is invited to log in. If a user is untrustworthy and tries to fool the system by picking a home site to which they do not belong, they will have difficulty authenticating themselves to that site's authentication server, since they won't have any valid credentials. However, if they pick their own home site, they should find authentication is no problem. After successful authentication, the home site re-directs the user back to the resource site and the message carries a digitally signed SAML authentication assertion message from the home site, asserting that the user has been successfully authenticated by a particular means e.g. username/password, Kerberos or digital signature. The actual mechanism used is local to the home site, and the resource site simply has to have a prior agreement with the home site which authentication mechanism(s) will be trusted. If the digital signature on the SAML

---

[1] Note that the connection from the origin server to the target server can also be optionally protected by SSL in Shibboleth, but this is used to provide confidentiality of the connection rather than message origin authentication. In many cases a confidential SSL connection between the origin and the target will not be required, since the handle is obscure enough to stop an intruder from finding anything out about the user, whilst the SAML signature makes the message exchange authentic.

authentication assertion verifies OK, then the resource site has a trusted message providing it with a temporary pseudonym for the user (the handle), the location of the attribute authority at the origin site and the resource URL that the user was previously trying to access. The resource site then returns the handle to the home site's attribute authority in a SAML attribute query message and is returned a signed SAML attribute assertion message. The Shibboleth trust model is that the target site trusts the origin site to manage each user's attributes correctly, in whatever way it wishes. So the returned SAML attribute assertion message, digitally signed by the origin, provides proof to the target that the authenticated user does have these attributes. This message exchange should be protected by SSL if confidentiality/privacy of the returned attributes is required. The attributes in this assertion may then be used to authorise the user to access particular areas of the resource site, without the resource site ever being told the user's identity.

The latest version of the Shibboleth specification has introduced a performance improvement over the earlier versions, by optionally allowing stage one and stage two to be combined together, in that the initial digitally signed SAML message may optionally contain the user's attributes as well as the authentication assertion. It is expected that the Shibboleth software will be upgraded to this during 2005.

Shibboleth has two mechanisms to ensure user privacy. Firstly it allows a different pseudonym for the user's identity (the handle) to be returned each time, and secondly it requires that the attribute authorities provide some form of control over the release of user attributes to resource sites, which they term an attribute release policy. Both users and administrators should have some say over the contents of their attribute release policies. This is to minimise the loss of a user's privacy.

3. An Enhanced Trust Model for Shibboleth

As can be seen from the above overview of Shibboleth, its trust model is sound although rather limited. The model is that the target site trusts the origin site to authenticate its users and to manage their attributes correctly whilst the origin site trusts the target site to provide services to its users. The trust is conveyed using digitally signed SAML messages using target and origin server X.509 key pairs/certificates, configured into the Shibboleth software by their filenames. (Note that the private key files were held unencrypted in the Shibboleth software we were using, so this is a weakness in the implementation if not actually in the trust model.) As each site will typically only have one key pair per Shibboleth system, from the recipient's perspective, there is only a single point of trust per sending Shibboleth system. Although it is not difficult to configure multiple roots of trust into a Shibboleth target site – it is, in fact, a matter of updating one XML file only – the issue is one of being able to use a finer grained distributed trust model, and of being able to use multiple origin site authorities (and private keys) to issue and sign the authentication and attribute assertions.

In many origin sites a single back end LDAP server is the sole authoritative source for both authentication and attribute information. Typically Shibboleth sites implement stage one by issuing a Bind operation on their LDAP server, using the username and password provided by the user to the web login prompt. If the Bind succeeds, the user has been successfully authenticated against the password stored in the LDAP server. Stage two is implemented

by searching the LDAP server for the attributes stored in the user's entry, and filtering these against the Shibboleth origin's attribute release policy before returning them to the Shibboleth target site as signed SAML attribute assertions. One can see that in such an implementation, and as a consequence of the Shibboleth trust model, the Shibboleth target site has no choice but to make access control decisions based on these attributes, without knowing who actually issued them to the user, whether they are still valid or not, or whether they are even the correct attributes for the particular user, since the user's name is not provided to the target site for privacy reasons. The Shibboleth origin doesn't trust anyone to see the attributes except the trusted targets, but even they are not allowed to see the binding between the attributes and the owner's identity. (The two reasons given for this in the Shibboleth documentation are user privacy and legal requirements for universities to protect a student's privacy). The target site thus has no option but to indirectly trust the contents of the origin site's LDAP server or other attribute repository, since it trusts the origin site directly. One can further see that the origin site has to strongly protect the attributes in its (LDAP) repository, which means that it is probably restricted to centrally administering these, and so would prefer that they do not change that often. Flexibility and distributed management of the attributes is hard to adopt. Dynamic delegation of authority would be even harder to support.

We propose that an enhanced trust model should have the following features.
- Multiple authorities should be able to issue attributes to the users, and the target site should be able to verify the issuer/user bindings. For example, a manager should be able to assign a project leader attribute to an employee under his control.
- The target should be able to state, in its policy, which of the attribute authorities it trusts to issue which attributes to which groups of users. The target site should be able to decide independently of the issuing site which attributes and authorities to trust when making its access control decisions.
- Not all attribute issuing authorities need be part of the origin site. A target site should be able to allow a user to gain access to its resources if it has attributes issued by multiple authorities, for example, a target site holding statistics on medical data may require a user to have an attribute issued by a medical authority as well as one issued by the university that employs the user as a researcher.
- The trust infrastructure should support dynamic delegation of authority, so that a holder of a privilege attribute may delegate (a subset of) this to another person without having to reconfigure anything in the system. For example, a project leader may wish to assign a role of team leader to one of his team members; he should be enabled to do this dynamically by the infrastructure without having to reconfigure the system. The target site should be able, in turn, to state in its policy whether it trusts these delegated attributes or not, regardless of the delegation policy at the user's site.
- The target site should be able to decide if it really does trust the origin's attribute repository (e.g. LDAP server), and if not, be able to demand a stronger proof of attribute entitlement than that conferred by a SAML signature from the sending Web server.
- Finally, the origin site, if it chooses, should be able to use a Privilege

Management Infrastructure, rather than a strongly protected attribute repository, for allocating attributes to its users. This will allow the origin to distribute the management of attributes throughout its site. Nevertheless, the origin site should still be able to communicate with Shibboleth targets as usual by only sending attributes to them, if the targets are happy to trust these.

4. Implementing the Enhanced Trust Model using an X.509 PMI

X.509 attribute certificates (ACs) provide a convenient, standardised and compact representation of attribute assignments, and satisfy several of the above requirements. The basic X.509 attribute certificate construct comprises: the name of the holder of the attributes, the name of the issuing authority, the set of attributes, and the time that they are valid for. An extension field can optionally be inserted to state that the holder is allowed to dynamically delegate (a subset of) these attributes to another user, and the depth to which delegation can take place. The whole AC construct is digitally signed by the issuer (attribute authority), thus providing integrity control and tamper resistance. Multiple attribute authorities can co-exist, either in a hierarchical relationship or as separate independent authorities. Attribute certificates are typically long lived, and after issuance, the ACs need to be stored somewhere for retrieval by the target's policy decision point (PDP), and LDAP repositories at the AA site are a natural choice for this, although web servers, filestores and other repositories can also be used.

If the ACs are stored in the AA site's LDAP directory or other repository, and transferred from there to the target site's PDP by Shibboleth, then the target site's PDP does not need to indirectly trust the attribute repository or the underlying transport mechanism used to convey them, since it can directly validate the digital signatures on the attribute certificates when it receives them[2]. Furthermore, if the target site's PDP policy is willing to allow dynamic delegation of authority, the PDP can check the attribute certificate chain to ensure that all ACs were properly authorised by their issuing authorities. By using ACs in its authorisation decision making, rather than plain attributes, a target site can support much more sophisticated and finer grained access control policies, for example, by requiring a user to have ACs issued by multiple authorities, from different issuing domains, before they are granted access to particular resources.

The PERMIS X.509 PMI is part of the US NSF Middleware Initiative software release. PERMIS provides a policy controlled role based access control (RBAC) infrastructure, in which the user's roles are stored in X.509 ACs. These ACs are either passed to the PERMIS PDP along with the user's requested action (the push model), or can be fetched from one or more LDAP servers by the PDP (the pull model). The PERMIS PDP then returns a granted or denied response according to the policy in force at that time. The PERMIS policy is written in XML, and is in many respects a simplified alternative to XACML [6], although the PERMIS policy supports dynamic delegation of authority, unlike XACML. The XML policy is itself stored in an X.509 attribute certificate, and is digitally signed by the trusted authority in control of a target resource. This policy certificate is the root of

---

[2] It is the case with ACs that the holder's identity is revealed in the Holder field of the AC. But the Holder field could still be an opaque string, understood by the Issuer at the Origin, and it doesn't have to be understood by the AC verifier at the Target site. See section 6 for a fuller discussion of this issue.

trust for the access control decision making. When the PERMIS PDP is initialised, it is given the name of the trusted authority, and the ID of the policy to use (each policy has a globally unique identifier). PERMIS reads in the policy certificates from the authority's LDAP entry, checks their signatures, and keeps the one with the correct ID. It now has the correct trusted policy with which to make access control decisions. PERMIS thus forms a good basis for demonstrating the distributed management of trust with Shibboleth.

4.1 The PERMIS PDP Policy

The PERMIS policy contains a list of trusted attribute authorities, the set of attributes they are trusted to assign, and the groups of users they can be assigned to. This is called the *role allocation sub-policy* (RAP). Attribute authorities can be distributed worldwide, and can be trusted to issue ACs to users from any domain, according to the RAP. When the PERMIS PDP is passed a set of attribute certificates by Shibboleth, it can determine from the RAP which are trusted to keep, and which should be discarded as untrusted. All the trusted attributes are extracted and stored for later access control decision making.

The PERMIS policy also contains the set of targets that are being protected by this policy, the associated actions that can be performed on them (along with their parameters), and the attributes (or roles) that a user needs in order to be granted the access. In addition, constraints can be placed on these grants, such as, only between 9am and 5pm, or only if the user holds non-conflicting roles[3], or only if the size is less than 3Mbytes etc. This is called the *target access sub-policy* (TAP). When the PERMIS PDP is asked if a user with the

current roles/attributes is allowed to access a particular target resource, it consults the TAP and returns granted or denied based on its contents and the current state of the environment (time of day, resource usage etc.).

Because PERMIS can act in either push or pull mode with attribute certificates, then it is possible for a target site to create a policy that requires a user to have attributes issued by multiple different authorities in different domains, and the PERMIS PDP can then pull these at authorisation time regardless of the origin site that the user has actually authenticated to.

5. Supporting the different trust models of Shibboleth sites

One can immediately see that if Shibboleth and PERMIS are integrated together, then the enhanced distributed trust model that we wish to provide to target and origin sites can be obtained. However, a number of misalignments between PERMIS and Shibboleth need to be addressed first. Either Shibboleth needs to transfer X.509 attribute certificates (ACs) from the origin site to the resource site instead of plain attributes, or PERMIS needs to be modified to accept plain attributes instead of X.509 ACs. In fact, both of these methods have been implemented so as to provide resource sites with the maximum of flexibility. We have modified the Shibboleth origin site to retrieve X.509 ACs from its LDAP directory, and to pass these as text encoded binary attributes within the SAML attribute assertions. This facility should be provided as part of the standard Shibboleth software release during 2005. We have also modified the code that calls the PERMIS PDP to validate the plain attributes from Shibboleth and use these instead of or as well as X.509 ACs.

---

[3] Separation of duties is currently being implemented but is not in the current NMI release.

Since it is the target site's resources that are being accessed, we are primarily concerned with the trust that a target site is expected or required to have in the attributes that it receives in order for it to become Shibboleth enabled. An origin site will also have its own preferred trust model for the allocation of attributes, but the target site's trust model must always take precedence since it is the owner of the resources that are being accessed. We can look at trust from two different aspects: the distribution of trust in the attribute issuing authorities (centralised or distributed) and the trustworthiness of an origin site's attribute repository (trusted or not).

Firstly, we consider the distribution of trust. In the simplest case the origin site has a single attribute issuing authority. If the target site trusts the origin site's attribute authority, this authority can issue and sign all the SAML attribute assertions. (This is the standard Shibboleth model.) Alternatively, the origin site may wish to distribute the management of attributes between different trusted authorities in its domain and to allow dynamic delegation of authority. If the target site wishes to distribute its trust to these different authorities, then it can allow (trust) each one of them to issue and sign different attribute assertions, and further decide if it will allow dynamic delegation of authority to take place. Furthermore, in this distributed trust scenario, the target site may be willing to trust, or even require, some attribute authorities that are not even based at the origin site to issue attributes to users. (This is typically the case in today's world when one presents plastic cards from multiple different issuers in order to gain access to a particular resource e.g. access to an airport business lounge may be granted by presenting frequent flyer cards from a number of different airlines or diners clubs.) On the other hand, if the target site is not willing to recognise these multiple authorities, then the origin site will need to (re-)sign all the SAML attribute assertions by the single authority that the target site is willing to trust.

Secondly, we consider the origin site's attribute repository (typically an LDAP server). If either the target or origin site do not trust this to store unprotected attributes securely, then the origin will need to store digitally signed attributes in it, rather than plain attributes. We now consider each combination in turn. Figure 1 pictorially represents each of the trust models shown in the following sections.

5.1 Target trusts origin's attribute repository and origin as a single attribute authority

This is the original Shibboleth trust model and both the target site and origin site will use standard Shibboleth. The origin will store plain attributes in its repository, and pass them in digitally signed SAML messages to the target. The target site may use the standard Shibboleth authorisation mechanism, or optionally, for a finer grained and more refined access control mechanism, use a policy controlled PDP to make decisions. When using the PERMIS PDP for authorisation, the PERMIS target access sub-policy (TAP) is used to say which attributes are needed in order to gain access to the targets, and the (unsigned) attributes from the SAML message are passed to the PERMIS PDP.
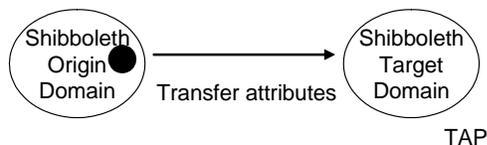
5.1 Standard Shibboleth
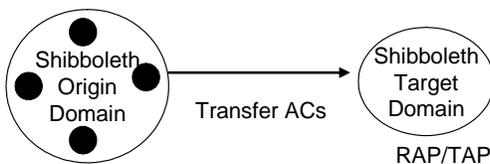
5.2 Multiple ACs at Origin
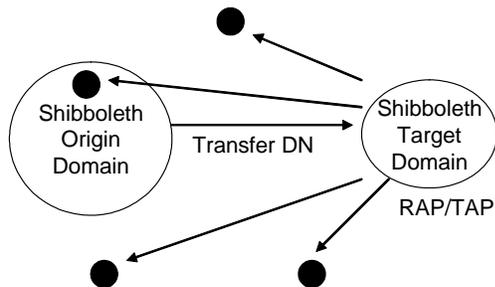
5.3 Pull ACs from multiple AAs
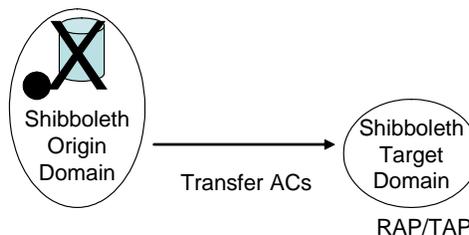
5.4 Untrustworthy attribute repository
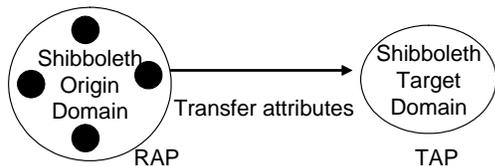
5.5 Multiple AAs at Origin not trusted by Target

RAP= role assignment policy

TAP=target access policy

= attribute repository

● = AA
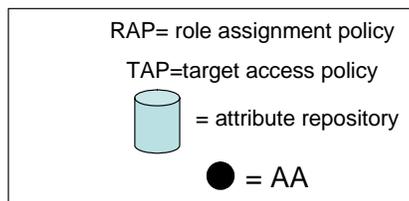
Figure1. Pictorial representation of different trust models

5.2 Origin wishes to distribute attribute assignments and target trusts different attribute authorities at the origin

In this scenario the origin distributes management between multiple authorities and therefore must store attribute certificates in its repository, so that the different attribute authorities can be recognised by the target. The target site uses the role assignment sub-policy (RAP) to describe who it trusts to assign which attributes to whom, and the TAP to determine which attributes are needed in order to access which targets. Note that the target may only trust a subset of the actual attribute authorities at the origin site, according to its RAP, and the policy specification allows for

this. Additionally, the target may allow dynamic delegation of authority at the origin site, by specifying this in the RAP[4]. Shibboleth now fetches attribute certificates from the origin site, rather than plain attributes. Consequently the SAML attribute assertions do no need to be signed, though the link will still need to be SSL encrypted if privacy protection is required. In this scenario the origin's attribute repository may or may not be trusted by either the target or the origin, but this is not an issue since it is storing digitally signed ACs in the repository.

---

[4] Note that the enforcement of dynamic delegation of authority is currently being implemented and will be in a future release of PERMIS.

5.3 Target trusts different attribute authorities at the origin site and elsewhere

In this scenario, the target site wishes to authorise users based on attributes assigned to them by different attribute authorities that are not always co-located with the origin site. In this case, the origin site cannot push all the attributes to the target site (unless the AAs have distributed them to the origin site in the first place, which cannot be guaranteed), so the target will need to operate in pull mode and fetch the ACs that it needs directly from the AAs. The PERMIS PDP can operate in pull mode and fetch all the attribute certificates that are needed from the various distributed (LDAP) repositories. The SAML attribute assertions from the origin site do not need to carry any attribute certificates in this instance. They only need to provide the holder identity of the user, so that the target can know which ACs to retrieve. Of course, each attribute authority will need to let its repository (LDAP server) be accessed by the target site[5]. Once the ACs have been retrieved, the target's PDP will use the RAP to determine which ACs are trusted, and the TAP to determine if the user has the necessary attributes to access the resource.

5.4 Target and/or origin do not trust origin's attribute repository but target trusts origin as a single attribute authority

In this scenario the origin cannot store unsigned attributes in its repository, but rather should store digitally signed attributes in its (LDAP) repository. The exact format of these could be X.509 attribute certificates or (long lived) SAML attribute assertions.

These should all be signed by the same organisational attribute authority that is trusted by the target. Shibboleth will then carry either signed attribute certificates or signed SAML assertions to the target site. (Note that the latter is equivalent to the model in 5.1). When the ACs are handed to the PDP, the RAP will check that they have been issued by the sole origin authority. The TAP is then used to determine if the user has sufficient attributes to be granted access to the target or not. When Shibboleth is transferring attribute certificates in the SAML assertions, the assertions do not need to be signed, though SSL encryption will be needed if privacy protection is required.

5.5 Origin wishes to distribute trust to multiple authorities, but target does not recognise them

In this scenario the target wishes to run standard Shibboleth but the origin wishes to distribute the management of attributes to different AAs i.e. to run its own PMI, with all the advantages this brings such as dynamic delegation of authority. The origin will be creating and storing attribute certificates in its AC repository signed by multiple distributed attribute authorities. However, because the target wishes to run standard Shibboleth, and wants a single point of trust at the origin, these ACs cannot be passed to the target. Therefore the origin site should run a PDP with its own RAP to validate that the ACs are issued in accordance with its own policy. This will validate the stored attribute certificates, extract the attributes that are trusted and pass these to the local Shibboleth origin server for transfer in signed SAML attribute assertions to the target. The target site can then run the standard Shibboleth authorisation module, or for finer grained control can run its own PDP and TAP, as in 5.1, to determine if the user is to be granted access or not.

---

[5] Note that if a site's firewall prevents the LDAP protocol from passing through, there are several http to ldap gateways available that allow the firewall to be tunnelled through on port 80.

6 User Privacy Issues

One of the limiting factors of X.509 attribute certificates (ACs) is that they bind the attributes to the holder, and if the holder is identified by their real name in the AC e.g. {CN=David Chadwick, OU=Computing Laboratory, O=University of Kent, C=GB} then the user's privacy is (at least partially) lost. There are a number of solutions to this problem. X.509 ACs allow holders to be identified in a number of different ways. Firstly, they can be identified by a distinguished name (DN). However, this DN does not need to be the real name of the holder or indeed in any way be similar to the holder's real name. It can be a pseudonym rather than their real name e.g. {CN=123456789}, or even a group name e.g. {CN=Programmer, OU=Computing Laboratory, O=University of Kent, C=GB}. This opaque name only needs to have meaning to the issuing site. The mapping between the user's login/authentication identity and AC holder identity would be performed at authentication time by the origin site's authentication server. It is important to note that the binding between the pseudonym in the AC and the authentication name of the human user is handled not by normal PKI registration procedures, but by the origin authentication system, so that the target site's trust in user authentication has to be placed in the origin site's systems and not in a trusted third party CA. Further, the use of pseudonyms or group names will make it much more difficult for independent AC issuers to participate in distributed trust management, since they will need to liaise with the origin site to know which opaque names have been given to which users.

The difference between using a pseudonym and a group identity is that in the former case the target site would be able to profile the user, without knowing the real physical identity of the user. With a group identity the target site would only be able to profile the whole group, and would not be able to differentiate between different group members, or know how many members were in the group.

Secondly, the holder can be identified indirectly by reference to their X.509 public key certificate. In this case the attribute certificate holds the serial number and issuer name of the user's public key certificate e.g. {x509serialNumber=123456 + x509issuer = {OU=Some CA, O=Some Org, C=US}. The limitations of this method are that the user must be PKI enabled, which of course, many are not; and that, depending upon the contents of the user's public key certificate, the user might be identified via this.

Finally, the holder can be identified indirectly by reference to the hash of a public key that they hold. This is now effectively a random number, giving good privacy protection. The user can prove ownership of the attribute certificate by digitally signing a challenge provided by the origin authentication server, which can then provide this AC to the target site. The restrictions are that the user needs to be using some form of asymmetric cryptography, has generated their own private/public key pair, has created a self signed certificate with a random DN and does not have a corresponding X.509 public key certificate identifying him/her. The main limitation from a privacy perspective is that the target site can profile the user, without knowing the actual identity of the user, since the same public key hash is used each time.

In all these cases there is a trade-off between the "degree of anonymity" and the "quality of issuance". At one extreme we have dynamically generated Shibboleth short lived signed SAML attribute assertions that

provide anonymity but require a trusted directory to store the user's attributes. At the other extreme we have long lived ACs where each attribute authority can issue its own attributes in a controlled manner, but without any privacy protection. At various points in the middle we have long lived ACs with various forms of privacy protection (pseudonyms, group names and public key identifiers) where the AA or authentication system maps the user's name into a privacy protected one.

In addition to protecting the identity of the AC holder, the AC issuer name may also be protected in the same ways as above. Note that the name of the SOA may be privacy protected or pseudonymised in some way, but the target PDP will need to know who this name actually belongs to if it is to be configured as a root of trust. The privacy of the embedded attributes may be protected by encryption, as described in [3] and [7]. Different attributes can be encrypted for different target sites. The main disadvantage of encrypted attributes is that the issuer needs to know in advance, when creating the AC, who the targets are going to be. This of course may not always be possible with relatively long lived ACs, in which case SSL encryption of the communications link is a better option for attribute privacy.

7 Revocation and Performance Issues

The signed SAML assertions of Shibboleth do not need to be revoked due to their short life times. Attribute certificates on the other hand are expected to have a relatively long life time, according to the privileges/attributes being allocated. For example, one might expect a "student" attribute certificate to be valid for an entire academic year. Whilst signed SAML attribute assertions have the performance overhead of requiring a digital signature per

message sent by the origin site, long lived ACs may have the overhead of requiring revocation list processing, depending upon how they are stored and distributed. If the ACs are stored in a repository under the control of the issuer, and are retrieved from there by either the Shibboleth origin or target sites, or directly by the target's PDP, then a revocation list may be avoidable providing the issuer deletes the ACs when they need to be revoked, and third parties are not able to surreptitiously write them back again. In this way the revoked ACs will not be available to the PDP when either it or the Shibboleth components try to retrieve them. If on the other hand the ACs are not stored in a repository under the control of the issuer, for example, they are distributed directly to their holders, then standard attribute certificate revocation lists (ACRLs) will be needed, and the issuer will need to periodically update them, in exactly the same way as for public key certificate CRLs. The PDP will need to ensure that it has a current ACRL when validating the ACs that have been presented to it. This will cause some performance overhead at the target site. Short lived ACs on the other hand do not need ACRLs to be published, just as the short lived SAML assertions do no require them. Whilst short lived ACs do not have the distributed trust management benefits of long lived ones (one cannot expect human managers to issue ACs to their staff daily, whilst automated issuing servers have the same trust related problems as existing Shibboleth implementations), they do have a significant performance benefit over signed XML messages [8] [9], so they might still be worthy of consideration in this respect.

8 Conclusions

We have shown how a distributed, finer grained and more functional trust model can be added to Shibboleth, to increase the

latter's flexibility and authorisation decision making capabilities. We have further shown how the model can support target and origin sites using different combinations of centralised and distributed trust models, and different assumptions concerning the trustworthiness of the origin's attribute repository. We have implemented this distributed trust model in Shibboleth by combining it with the PERMIS authorisation infrastructure and X.509 attribute certificates. Finally we have argued that user privacy does not need to be compromised *per se* by using long lived X.509 attribute certificates instead of short lived digitally signed SAML attribute assertions, although it is certainly more difficult to fully protect a user's privacy in the former case.

8 Acknowledgements

9 References

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

[2] 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

[3] ISO 9594-8/ITU-T Rec. X.509 (2001) The Directory: Public-key and attribute certificate frameworks

[4] David Chadwick, Sassa Otenko, Von Welch. "Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure". Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Windermere, UK, 15-18 September 2004

[5] OASIS. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1", 2 September 2003

[6] OASIS. "eXtensible Access Control Markup Language (XACML)" v1.0, 12 Dec 2002, available from http://www.oasis-open.org/committees/xacml/

[7] S. Farrell, R. Housley. "An Internet Attribute Certificate Profile for Authorization". RFC 3281, April 2002

[8] Mundy, D. and Chadwick, D.W., "An XML Alternative for Performance and Security: ASN.1", IEEE IT Professional, Vol 6., No.1, Jan 2004, pp30-36

[9] "Fast Web Services," P. Sandoz and colleagues, Aug 2003, available from http://java.sun.com/developer/technicalArticles/WebServices/fastWS/