Delegation Issuing Service for X.509

D.W.Chadwick, University of Kent, Canterbury, CT2 7NZ, England

Abstract

This paper describes the concept of a delegation issuing service (DIS), which is a service that issues X.509 attribute certificates on behalf of an attribute authority (typically a manager). The paper defines the X.509 certificate extensions that are being proposed for the 2005 edition of X.509 in order to implement the DIS concept, as well as the additional steps that a relying party will need to undertake when validating certificates issued in this way. The paper also presents our initial experiences of designing a DIS to add to the PERMIS authorization infrastructure. The paper concludes by reviewing some of the previous standards work in delegation of authority and anticipating some of the further standardization work that is still required in the field of privilege management.

1. Introduction

The 2000 edition of X.509 [1] defines a Privilege Management Infrastructure (PMI) based on attribute certificates (ACs). Attribute certificates are very similar to public key certificates (PKCs) but hold privileges (as attributes) instead of public keys. In the X.509 PMI model, the root of the PMI is termed the Source of Authority (SoA), and subordinates are termed Attribute Authorities (AAs). Delegation of Authority passes down the chain from SoA to AA to subordinate AAs in much the same way as the authority to issue public key certificates passes down a PKI certification authority hierarchy from the root CA to subordinate CAs (see Figure 1A). A subordinate AA is given a set of privileges by its superior AA, and may delegate these further to subordinates (AAs or end entities). A subordinate who is not allowed to delegate further is termed an end entity. In the normal situation all privilege holders (AAs and end entities) are allowed to assert the privileges that are delegated to them.

However, in some situations a privilege holder may be allowed to delegate the held privileges to a subordinate, but may not be allowed to assert the privileges itself. An example might be an airline manager who assigns privileges to pilots to fly particular aircraft, but is not allowed to fly the aircraft himself, or a hospital manager who assigns privileges to doctors on duty but is not allowed to assert these privileges himself. Whilst the X.509 standard recognizes this scenario, it offers no support for this in the ACs that can be issued to these AAs i.e. there is no way of signaling to a relying party (RP) that an AC holder may not assert the privileges contained in the AC that it holds. This deficiency needs rectifying.

Work is now progressing towards issuing the 2005 edition of X.509, and another delegation scenario has been identified in the draft amendment [2] to X.509(2000). This concerns the use of a delegation service to issue ACs on behalf of other AAs. The delegation issuing service (DIS) concept recognizes that in some organizational contexts, it might be preferable for a manager (an AA) who wishes to delegate authority to a subordinate, be not empowered to issue the X.509 AC herself, but rather should request a DIS to issue the AC on her behalf.

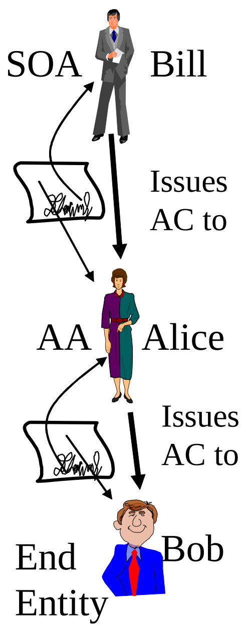

Figure 1A. Normal Delegation of Authority

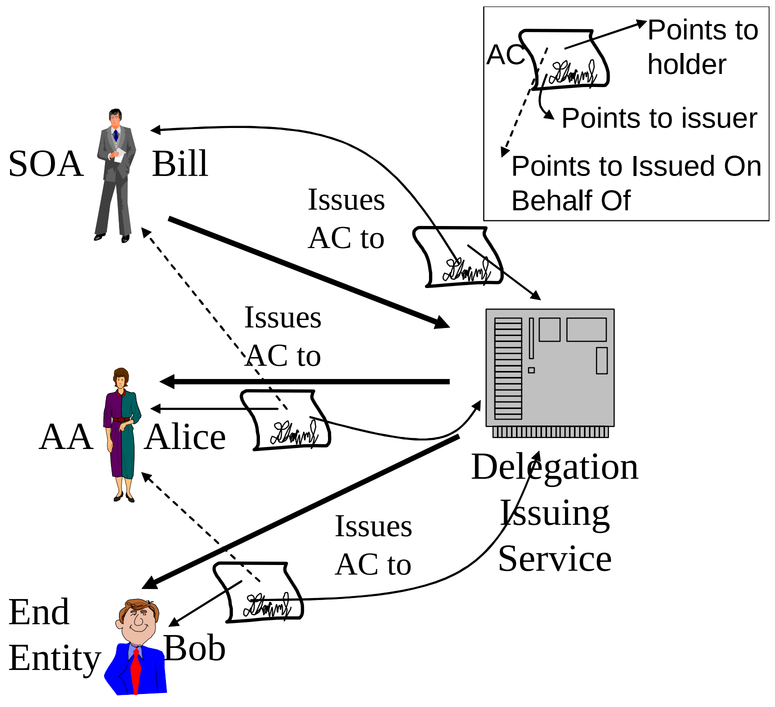

Figure 1B. Delegation of Authority using a Delegation Issuing Service

1.1 Advantages of a DIS

The benefits of using a delegation issuing service instead of AAs issuing X.509 ACs themselves are several. Firstly, the DIS can support a fully secure audit trail and database, so that there is an easily accessible record of every AC that has been issued and revoked throughout the organization. If each manager were allowed to independently issue their own ACs, then this information would be distributed throughout the organization, making it difficult or impossible to collect, being possibly badly or never recorded or even lost. Secondly, the DIS can be provided with the organization's delegation policy, and apply control procedures to ensure that a manager does not overstep her authority by issuing greater privileges to subordinates, or even to herself, than the organization's policy allows. Thirdly, the manager does not need to hold and maintain her own private signing key, which would be needed if the manager were to issue and sign her own ACs. Only the DIS needs to have an AC signing key. This could be a very important feature in organizations that use mechanisms other than PKIs for authentication such as biometrics, user names and passwords, or Kerberos etc. Finally, if the DIS is given its own AC by the SOA, it can replace the (set of) manager's AC(s) in the AC validation chain and therefore decrease the complexity of AC chain validation. The AC chain length would always be two when the DIS issues the ACs to end entities, whereas it would be of arbitrary length when the managers issue the ACs themselves. Also less CRLs will need to be issued – only the DIS will need to issue a CRL rather than each manager. This will further simplify AC chain validation.

1.2 DIS Deployment Models

Two deployment models for the DIS are envisaged in the 2005 draft amendment [2]. In both models the DIS is empowered to issue ACs on behalf of other AAs, by being issued with its own AC by the SoA. This AC confers on the DIS the authority to issue ACs on behalf of other AAs. This empowerment model is similar to the PKI concept of an indirect CRL issuer, whereby an entity is given the authority to issue CRLs on behalf of a CA. In the first DIS deployment model (which we have called the *AC PKI* mode), a privilege holder requests the DIS to issue privileges on its behalf, but the DIS does not actually hold the privileges itself. The AA tells the DIS which privileges to delegate. In the second deployment model, the DIS is actually given the privileges to be delegated by the SoA (we have called this the *PMI* mode). However, the 2005 draft amendment had no mechanisms for implementing either of these deployment models.

In our research and design of a DIS service for PERMIS [5], we have also identified a third deployment model in which the DIS is not given an AC, but has its own PKI key pair for signing the ACs its issues, with empowerment flagged in the public key certificate (we call this the *PKI* mode). The DIS now only needs to authenticate the AAs and issue ACs on their behalf, without validating the contents of the ACs. Furthermore, the users do not need to have their own PKI key pairs. This simplifies the design and deployment of the DIS service, but the downside is that it complicates the process of AC chain validation by the relying parties due to the delegation indirection introduced by the DIS, as described later.

1.3 Disadvantages of a DIS

As mentioned above, in PKI (and AC PKI) modes, AC chain validation is more complex when a DIS issues the ACs. Another potential disadvantage of a DIS is that the single CRL issued by the DIS could get very large, unless distribution points are used. A large CRL can adversely affect the performance of AC chain validation. Further, when cross certification between PMIs takes place, in PMI mode there is a loss of granularity since it has to be the DIS that is cross certified rather than any of the AAs that are trusted. But perhaps the biggest disadvantage of using a DIS for some applications is that the AC signing key should be online and ready to be used to sign ACs when requested. In some highly secure systems this would be unacceptable.

1.4 Paper Contents

This paper describes proposed extensions to the 2000 edition of X.509 that can be used to implement the DIS model for delegation of authority, as well as rectify the 2000 deficiency that there is no way to signal that an AA holds a privilege for delegation but is not allowed to assert the privilege. These extensions have recently been included as part of the revised draft amendment to X.509(2000).

The rest of this paper is structured as follows. Section 2 describes the X.509 extension that can be used to signal that a privilege holder is not allowed to assert the privileges that it holds. This corrects the deficiency in the 2000 edition of X.509. Section 3 describes the X.509 extensions that can be used to implement the DIS model. Section 4 describes how relying parties will need to use these new extensions in order to validate ACs issued by a DIS. Section 5 describes how we are implementing the DIS in PERMIS. Section 6 describes related standards work and research in this area, whilst Section 7 describes further standardization work that is still needed to be done in the X.509 PMI framework.

2. No Assertion of Privileges

There are two scenarios where privilege holders may be given privileges in an AC[1], in order to delegate them to other entities, but where they are not allowed to assert the privileges themselves. The first is where a manager is tasked with allocating roles or duties to subordinates, but is not allowed to assert the roles or duties himself. The previous section gave a couple of examples of this scenario, in the shape of an airline manager and a hospital manager. This scenario is represented by Alice in Figure 1A. The second scenario is where a delegation issuing service (DIS) is given a set of privileges to delegate, as directed by the SoA. This is represented by the Delegation Issuing Service in Figure 1B.

We can prevent the holder of these privileges (Alice in Figure 1A and the DIS in Figure 1B) from asserting them by placing a "no assertion" extension into the AC issued to it. This extension will inform all relying parties that understand the extension that the AC holder is not allowed to assert the privileges contained within the AC. This extension obviously needs to be a critical extension, since any relying party that does not understand it, must refuse to accept the AC, rather than simply ignore the extension and allow the privileges to be asserted.

The "no assertion" extension is formally defined in ASN.1 [6] as:

[1] We do not consider it sensible to issue privileges to AAs via the subjectDirectoryAttributes extension of public key certificates, since the AAs would not be allowed to delegate these privileges further by issuing additional PKCs, since they are not a CA.

```
noAssertion EXTENSION ::=  {
        SYNTAX NULL
        IDENTIFIED BY { id-ce-noAssertion }  }
```

where id-ce-noAssertion is an object identifier (OID) assigned in X.509.

If present, this extension indicates that the AC holder cannot assert the privileges indicated in the attributes of the AC. This field can only be inserted into AA ACs, and not into end entity ACs. If present, this extension shall always be marked critical.

3. X.509 Extensions to Support the Delegation Issuing Service

As described in the Introduction, three deployment models have been identified for the DIS, two in [2], in which the DIS is issued with its own AC and we have termed the AC PKI and PMI modes, and one from our own research, termed the PKI mode, in which the DIS does not have its own AC.

3.1 DIS Empowerment

The Delegation Issuing Service (DIS) needs to be empowered by the SoA to issue ACs on behalf of other AAs. This is done by including an "indirect issuer" extension in either the PKC or the AC issued to the DIS by the CA or SoA respectively. The indirect issuer extension serves a similar purpose as the indirect CRL boolean of the issuing distribution point extension in PKI CRLs i.e. it gives authority to the DIS. The indirect issuer extension is formally defined in ASN.1 as:

```
indirectIssuer EXTENSION ::= {
        SYNTAX              NULL
        IDENTIFIED BY       id-ce-indirectIssuer  }
```

where id-ce-indirectIssuer is an OID assigned in X.509.

The indirect issuer extension may be used by the relying party when validating an AC chain to check that the AC issuer was empowered to issue ACs on behalf of other AAs (otherwise anyone with a signing key could issue an AC and say it was authorized by an AA). Alternatively, it may be used by the DIS software at initialization time to check that it is empowered to act as a DIS.

The draft extension to X.509 states that the indirect issuer extension may be placed in either an AC or PKC containing the subjectDirectoryAttributes extension issued to a DIS by an SoA. In our research we have identified that this extension may also be placed in a PKC that does not contain the subjectDirectoryAttributes extension.

The presence of this extension means that the subject AA (the DIS) is authorized by the SoA to act as a proxy and issue ACs that delegate privileges, on behalf of other delegators. This extension is always non-critical, since it does not matter to a relying party if it understands this extension or not when the DIS is acting as a privilege asserter by presenting this to the RP to assert the privileges contained within this certificate. This extension can be used by a RP when validating an AC chain which has the DIS acting on behalf of another AA somewhere in the AC chain (see section 4).

3.2 Requesting an AC

When an AA wishes to delegate some of its privileges to a subordinate, and wishes to use the services of a DIS to issue the AC on its behalf, it needs to contact the DIS to request the certificate to be issued. How this communication is achieved is outside the scope of X.509. Some discussion of this is provided later. Assuming this communication is successful, i.e. that the AA is authenticated to the DIS, and is allowed by the DIS's attribute allocation policy to request the AC to be issued, the DIS will issue an AC on behalf of the requesting AA. Thus we need an extension to be inserted into the AC, informing all relying parties that this certificate was issued on behalf of a particular AA. This leads to the requirement for the "issued on behalf of" extension, which is formally defined in ASN.1 below.

```
issuedOnBehalfOf EXTENSION ::= {
        SYNTAX GeneralName
        IDENTIFIED BY id-ce-issuedOnBehalfOf }
```

where id-ce-issuedOnBehalfOf is an OID assigned in X.509.

This extension is inserted into an AC by an indirect issuer. It indicates the AA that has requested the indirect issuer to issue the AC, and allows the delegation of authority chain to be constructed and validated by the relying party if necessary (see section 4).

The GeneralName is the name of the AA who has asked the issuer to issue this AC

The issuer of this AC must have been granted the privilege to issue ACs on behalf of other AAs by an SOA, through the indirectIssuer extension in its AC or PKC.

This extension may be critical or non-critical as necessary to ensure delegation path validation (see next section).

4. Validating Indirect AC chains

The X.509 standard already provides a procedure for validating privilege paths and delegation chains in the standard delegation of authority scenario. This chain is represented by the curved arrows that point to issuers in Figure 1A. This procedure needs to be enhanced when indirectly issued ACs are encountered in the delegation chain, such as those in Figure 1B. As can be seen from the addition of the issuedOnBehalfOf arrows in Figure 1B, the procedure is more complex and more delegation links need to be validated when this extension is marked critical.

Three deployment models have been identified, which we have termed the AC PKI, PMI and PKI modes. In PMI mode, the DIS has been issued with an AC by the SOA, which contains a superset of the attributes that it will issue on behalf of other AAs. This model presents the simplest path validation processing, since the AC chains will always comprise of just two ACs: the end entity's AC signed by the DIS, and the DIS's AC signed by the SOA. The existing standard path validation procedure will work for this AC chain. The RP may safely ignore the issuedOnBehalfOf and indirectIssuer extensions which will be marked as non-critical, since the DIS had full authority to issue the ACs to the end entities even though in reality it was a peer AA that asked for the delegation to be performed. Note that the DIS might not have permission to assert these privileges itself, but that will be signaled separately by the noAssertion extension.

In AC PKI mode, the DIS has an AC containing the indirectIssuer extension, but does not have any of the attributes that it will issue to others. These are held by the AAs that request the DIS to issue the ACs. In this case the issuedOnBehalfOf extension must be set to critical, since the RP will need to validate that the requesting AA had sufficient privileges to delegate to the end entity. If the extension was not set to critical, the RP is likely to compute that the AC chain is invalid since the DIS issuer did not have a superset of the privileges that were allocated to the end entity.

In PKI mode, the DIS does not have an AC, but only has a PKC containing the indirectIssuer extension. In this case the ACs issued by the DIS have to have the issuedOnBehalfOf extension set to critical, since the DIS is incapable of performing any validation of the requesting AA other than authenticating that it is who it says it is. All PMI validation has to be done by the RP. But this is in fact little different to the validation performed in the AC PKI mode, and is if anything slightly simpler since the DIS only has a PKC to be validated and not a PKC and an AC.

In addition to the standard procedural tasks of validating signatures and revocation lists, the relying party will also have to perform the following additional steps.

i) Starting with the end entity's AC, the RP will need to extract the issuer name and look at the critical flag of the issuedOnBehalfOf name.

ii) If the issuedOnBehalfOf extension is marked critical, the RP retrieves the ACs of the issuedOnBehalfOf AA and validates that the AA has a superset of the privilege attributes issued to the end entity and that the ACs have not been revoked. If it does not have sufficient privileges, or they have been revoked, the end entity's AC is rejected. The RP retrieves the certificates (ACs and PKCs) of the issuer and validates that the issuer is an indirect issuer of the SoA (i.e. has the indirectIssuer extension in one of its certificates). If not the end entity's AC is rejected.

iii) If the issuedOnBehalfOf extension is missing or non-critical, the RP retrieves the ACs of the AA (the DIS) and validates that the AA has a superset of the privileges issued to the end entity. If not, the end entity's

AC is rejected.

iv) For each AC of the issuer that contains one or more of the delegated privileges, the RP recurses to step i) for each AC, thereby moving up the delegation chain. This recursion continues until the RP arrives at the AC of an AA that is issued by the trusted SoA(s) who is(are) the root(s) of the PMI. This validates that the privileges were properly delegated.

4.1 Validating the noAssertion extension

If an AA's certificate has the noAssertion extension in it, what is to stop the AA issuing another AC to itself and omitting the noAssertion extension? Clearly there is nothing to stop this from happening. For this reason, SPKI decided (in section 4.2 of [11]) that they were powerless to stop this in their simple certificates that tied authorizations to public keys. However, X.509 has the advantage that AAs are given globally unique names by CAs. Providing an AA is not able to obtain an alias name for itself from the same or another trusted CA then the relying party can check if any AA's AC in a certificate path has the noAssertion extension set, and if it does, apply it also to any subordinate ACs that contain the same holder name. Clearly if an AA is able to obtain totally unrelated aliases from one or more trusted CAs, then the RP is unlikely to know that the AA is asserting privileges that it was not intended to, by using an alias name.

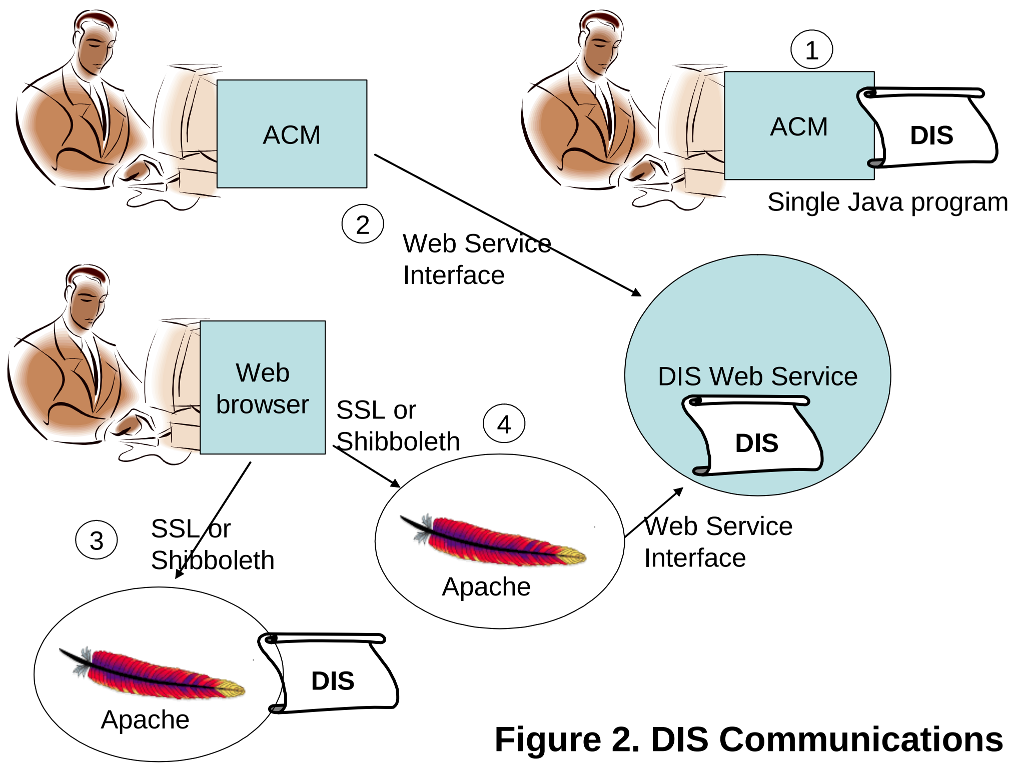

Figure 2. DIS Communications

5. Implementing the DIS

We decided to implement the DIS as part of the PERMIS X.509 PMI, as an aid to implementing dynamic delegation of authority. However, a number of issues needed to be resolved that are not addressed in the proposed extensions to X.509.

Firstly there is no mention of how the communication between the DIS and the AA should be achieved. Clearly the use of a standardized protocol is preferable to a proprietary one. One can envisage that an IETF working group such as PKIX might define a protocol similar to CMP [3], using a request similar to a PKCS#10 message [4]. In the absence of such a standard, in our own research we are proposing to use a Web Services protocol (see ② in Figure 2), and the Java GSSAPI [19] for authenticating the user. The GSS tokens will then be base64 encoded and inserted into SOAP messages. We are also defining a Java API for the DIS (see ① in Figure 2), so that the DIS can be built into other Java programs such as the PERMIS Attribute Certificate Manager (ACM) and called directly by it. In this case user authentication is not necessary. We are also proposing to adopt a 3 tiered model where an Apache server acts as the DIS client, authenticates the AAs via either Apache authentication (e.g. SSL) or Shibboleth (see ④ in Figure 2), and then acts as a proxy for them to the DIS. It would also be possible for Apache to directly call the DIS via our Java API (see ③ in Figure 2).

Secondly there are a number of issues concerned with AC path validation. As pointed out by Knight and Grandy in [18] this can be extremely complex when dynamic delegation of authority is implemented. We want to simplify this process as much as possible. We have already taken the step of not issuing role specification ACs, and instead we store their contents in each target's PERMIS policy read in by its PDP at initialization time. We thus only issue role allocation ACs. Our preferred DIS deployment model is the PMI mode, since the DIS is issued with a role allocation AC containing a superset of the attributes that it can delegate. In this way we limit AC path lengths to two, and if the target policy is willing to trust the DIS as a root (as well as the SoA) then path validation lengths are reduced to just one AC, that of the end user.

In our implementation, the DIS is given an AC containing a full set of privileges, and is configured with its own PERMIS PDP. The PDP is configured with an attribute (or role) assignment policy (RAP) [7], so that it can validate the AA requests. At initialization time the DIS will check that its AC has the indirectIssuer extension in it, otherwise it will fail to start. When an AA asks for an AC to be issued, the DIS will check that the AA is allowed to do this under the RAP policy, and also that the set of attributes requested are a subset of those held by the DIS. Validation against the RAP is done by the existing PERMIS PDP code. It is passed the (unsigned) AC requested by the AA, and it validates the credentials in the AC against the RAP. The only modification needed to PERMIS is to provide it with a null signature validation object that will return *signature valid* to every request to validate the unsigned ACs. If the AC passes the policy, the DIS will check that the requested attributes are a subset of those it holds in its own AC. The task of the RP is now made much simpler, since it only needs to validate 1 or 2 ACs, that of the user issued by the DIS, and optionally that of the DIS issued by the SOA.

Finally we wanted to simplify the use of PMIs in organizations that do not have fully functional PKIs implemented. These organizations, which are in the majority, already have a fully functional user authentication mechanism, and only have a handful of PKCs, e.g. web server certificates. It is for this reason that we have chosen to implement communications between the user and DIS as a three tiered model via an Apache web server as in path ④ in Figure 2. This will allow organizations to use their existing authentication method. One problem that has to be solved is that of proxying, since the DIS will authenticate Apache, Apache will authenticate the user and Apache will then ask for an AC to be issued on behalf of the user. The DIS has to know if Apache is authorized to proxy in this way. We could solve this in a couple of ways. We could configure the details (name address of Apache) into the DIS. Or we could issue Apache with its own AC containing a proxy privilege. When Apache authenticates to the DIS, the DIS will call the PERMIS PDP to validate Apache's AC, and if it has the proxy credential the DIS will allow it to request ACs be issued on behalf of other AAs.

6. Related Research

Some of the earliest standardization work for attribute certificates and attribute certificate issuing servers was undertaken by ECMA in the late 80's and early 90's. This lead to the publication of ECMA Standard 219 [9] which specifies a model for distributed authentication and access control. The Privilege Attribute Certificates (PACs) described therein are a forerunner of the attribute certificates later standardized in X.509. A Privilege Attribute Server (PA-Server) is responsible for issuing PACs to users, and is similar in concept to the DIS described in this paper. However, to support delegation of authority between principals, new PACs are not issued to the delegatees (as in this paper) but rather the PA-Server provides the initial user with a PAC that contains one of more embedded Protection Values (PVs) that can be used for subsequent delegation. A PV is a hash of a secret Control Value (CV). The user is separately issued with the corresponding CVs. When a user wishes to delegate authority to another user or server, the latter is provided with the PAC and the appropriate CV (sent confidentially, of course). The delegate then presents the PAC to the target along with the CV. The target validates that the hash of the CV corresponds to a PV in the PAC, and if so allows the delegate to have the appropriate delegated access on behalf of the user. Different delegates can be given different CVs which authorize different subsets of the privileges contained in the PAC to different sets of target resources. The EC SESAME project [8] implemented a subset of ECMA Standard 219 and this was eventually rolled out into several commercial products from the project's partners. The disadvantage of the ECMA scheme is that the user has to know in advance of requesting the PAC what delegation is required, since this is built into the PAC at the time of its issuance.

ECMA Standard 219 supports both symmetric and asymmetric encryption for protection of the PACs, since it supports both Kerberos V5 [10] and digital signatures for user authentication to the authentication server prior to contacting the PA-Server. Interestingly, X.509 decided to standardize on only asymmetric encryption for its certificates, whereas Microsoft Windows decided to adopt Kerberos and symmetric encryption for its tokens when allowing users to gain access to multiple Windows services.

The Simple Public Key Infrastructure (SPKI) [11] IETF working group, whose work eventually merged with the Simple Distributed Security Infrastructure (SDSI) [12] of Ron Rivest, defined three types of certificate which mapped names, authorizations and group names respectively to public keys. Authorization certificates can be further delegated, and this is indicated by a Boolean flag set by the issuing delegator. The delegator can set the Boolean as desired, except that if the Boolean is already false in the authorization certificate delegated to him/her then it cannot be switched back to true and be trusted. It therefore would be easy to apply the DIS concept and service to SPKI using the PMI mode deployment model, i.e. where the DIS is delegated an authorization certificate with the Boolean set to true. However it would break the theory of SPKI to implement either of the two PKI mode deployment models since these require the issuedOnBehalfOf extension to be present in the delegatee's certificate, and this would mean that the certificates are no longer *simple* according to SPKI's definition.

One feature included in SPKI that is not formally in the X.509 standard, is a rights language for expressing authorization policies. Consequently PERMIS defined its own policy language, written in XML [7]. SPKI uses S-expressions. X.509 has left it to other standards, e.g. the ISO Rights Expression Language [20] to specify the policies. This means that the policy rules by which a DIS operates are not specified in X.509.

Proxy certificates, defined initially by the Globus grid software developers, and later published as an IETF proposed standard [13], use a different model for delegation of authority. In this model a user, who is the subject of a public key certificate (and defined as an end entity by the X.509 standard), issues his own PKC (called a proxy certificate) to the public key of his grid job which has previously generated its own key pair. Validating the proxy certificate of course breaks the standard X.509 certificate path validation rules, since an end entity is not allowed to act as a CA. To rectify this, a critical extension (proxyCertInfo) is added to the proxy certificate to indicate the fact. The extension can also carry information about which privileges are being delegated, i.e. none, all or a subset, the latter being defined in an application specific way. Grid applications and users could use the DIS framework described here as an alternative to the latter, and ask the DIS to issue ACs to their grid jobs that contain a subset of the privileges contained in the user's AC. We plan to demonstrate this feature in due course, since PERMIS is already integrated with Globus toolkit [14].

More recently the work on Shibboleth [15] has implemented a limited mechanism for delegation of authority. In this case a target site delegates to the user's home site the task of authenticating and assigning attributes to the user. The user's privileges are returned to the target site in the form of a SAML Attribute Statement [16] signed by the home site. In research described in another paper at this conference [17], we have extended the Shibboleth delegation model by integrating it with PERMIS and X.509 ACs. The DIS will then be able to be used by home sites to delegate privileges even further.

7. Further Work

As indicated above, a protocol for interactions between an AA and a DIS will need to be standardized so that requests to issue ACs can be made in a standard manner. This request-response protocol may be similar to the PKIX CMP protocol, but it need not be, since proof of possession of the private key is not essential (indeed one of the motivations for having a DIS is that the users may not have their own key pairs). In many scenarios AAs may not be PKI users, but rather may use Kerberos, biometrics or symmetric tokens for authentication. In this case the AAs are computationally unable to issue X.509 ACs so will need to use the services of a DIS to issue the ACs on their behalf. But they will be unable to sign those requests to the DIS. In this case a web services interface like the one we propose to use may be more appropriate, with the AA using a web browser to interact with the DIS via a web server, and perhaps authenticating with a username and password over an SSL link. Whatever protocol is standardized, it will need to be flexible enough to cater for the different environments in which it may be used.

Practical experience of working with X.509 PMIs is only just beginning. Most organizations who are experimenting with PMIs use them internally initially. They define their own privilege attributes and therefore the relying parties and SoAs have a common understanding of both the semantics and syntax of these privilege attributes. However, as organizations move towards role based or attribute based access controls, and federations between organizations, including the formation of virtual organizations, they will find that the attributes and roles they have defined are different from those in use by their federation partners. When this starts to occur, organizations will not want to re-issue ACs to users from the federated organizations, but rather will wish to understand and use the ACs that have already been issued. This will require cross certification between PMIs, the mapping of role allocation policies between organizations and constraints on what foreign users may asserts in home domains. It is anticipated that this work will form the bulk of the standardization activity for the sixth edition of X.509.

Acknowledgements

The authors would like to thank the UK JISC for funding this work under the DyVOSE project, and the NIST PC members who robustly reviewed this paper and made some constructive comments that helped to improve it.

References

[1] ISO 9594-8/ITU-T Rec. X.509 (2000) The Directory: Public-key and attribute certificate frameworks

[2] ISO SC 6 N12596 “Proposed Draft Amendment on Enhancements to Public-Key and Attribute Certificates”, Geneva output, March 2004

[3] C. Adams, S. Farrell. “Internet X.509 Public Key Infrastructure Certificate Management Protocols”. RFC 2510, March 1999

[4] Kaliski, B., "PKCS #10: Certificate Request Syntax, Version 1.5." RFC 2314, March 1998.

[5] 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.

[6] ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation

[7] D.W.Chadwick, A. Otenko. “RBAC Policies in XML for X.509 Based Privilege Management” in Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002), May 7-9, 2002, Cairo, Egypt. Ed. by M. A. Ghonaimy, M. T. El-Hadidi, H.K.Aslan, Kluwer Academic Publishers, pp 39-53.

[8] T.Parker, D.Pinkas. “Sesame V4 Overview”, Issue 1, Dec 1995. Available from https://www.cosic.esat.kuleuven.ac.be/sesame/html/sesame_documents.html

[9] Standard ECMA-219 "Authentication and Privilege Attribute Security Application with Related Key Distribution Functions" Parts 1, 2 and 3, December 1994.

[10] J. Kohl, C. Neuman. “The Kerberos Network Authentication Service (V5).” RFC

1510, Sept 1993.
[11] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. “SPKI Certificate Theory”. RFC 2693, Sept 1999.
[12] Ron Rivest and Butler Lampson, "SDSI - A Simple Distributed Security Infrastructure [SDSI]", See <http://theory.lcs.mit.edu/~cis/sdsi.html>.
[13] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson. “Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile”. RFC3820, June 2004.
[14] David W 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
[15] Scot Cantor. “Shibboleth Architecture, Protocols and Profiles, Working Draft 02, 22 September 2004, see http://shibboleth.internet2.edu/
[16] OASIS. “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) v1.1”. 2 September 2003. See http://www.oasis-open.org/committees/security/
[17] David Chadwick, Sassa Otenko, Wensheng Xu. “Adding Distributed Trust Management to Shibboleth” presented at NIST 4th Annual PKI Workshop, Gaithersberg, USA, April 19-21 2005
[18] Knight, S., Grandy, C. “Scalability Issues in PMI Delegation”. Pre-Proceedings of the First Annual PKI Workshop, Gaithersburg, USA, April 2002, pp67-77
[19] Charlie Lai, Seema Malkani. “Implementing Security using JAAS and Java GSS API” Presentation from 2003 Sun JavaOne conference. See http://java.sun.com/security/javaone/2003/2236-JAASJGSS.pdf
[20] ISO/IEC 21000-5:2004, “Information technology — Multimedia framework (MPEG 21) — Part 5: Rights Expression Language [REL]”. 2004