J. Lopez, I. Agudo, and J. A. Montenegro, "On the deployment of a real scalable delegation service", *Information Security Technical Report*, vol. 12, pp. 139-147, 2007.
http://doi.org/10.1016/j.istr.2007.05.008
NICS Lab. Publications: https://www.nics.uma.es/publications

On the deployment of a real scalable delegation service

Javier Lopez, Isaac Agudo, Jose A. Montenegro

Computer Science Department,
E.T.S. Ingenieria Informatica
University of Malaga, Spain

Abstract

This paper explains the evolution of the concept of delegation since its first references in the context of distributed authorization to the actual use as a fundamental part of a privilege management architecture. The work reviews some of the earliest contributions that pointed out the relevance of delegation when dealing with distributed authorization, in particular we comment on PolicyMaker and Keynote, and also on SDSI/SPKI. Then, we elaborate on Federation as a particular case of delegation, and remark the importance given to federation by the industry. Finally, the paper discusses about privilege management infrastructures, introducing a new mechanism to extend their functionality using advanced delegation services.

1 Introduction

As it is widely known, computer and network security are related to the Internet more than ever before. As a consequence, the use of Internet has brought new requirements and changes to security software. Some of those changes focus on the way users are authenticated by Internet applications and how their rights and privileges are managed.

Therefore, one of the most controversial security services is *Access Control*. Lampson defines access control as the composition of two services, *Authentication* and *Authorization* [12]. Internet applications require distributed solutions for the access control service. Accordingly, authentication and authorization services need to be distributed too.

Email addresses: jlm@lcc.uma.es (Javier Lopez), isaac@lcc.uma.es (Isaac Agudo), monte@lcc.uma.es (Jose A. Montenegro).

In order to achieve a real scalable distributed authorization solution, the *Delegation* service needs to be strongly considered. Delegation is quite a complex concept, both from the theoretical and from the practical point of view. In this sense, the implementation of an appropriate delegation service is becoming a cornerstone of Internet applications since recently.

Because delegation is a concept related to authorization, this paper aims and put into perspective the delegation issues and implications that are derived from a selected group of authorization schemes that have been proposed during the last years as solutions for the distributed authorization problem.

In this work, the group of authorization solutions selected are *KeyNote*, as an evolution of *PolicyMaker*, *SDSI/SPKI* and *Privilege Management Infrastructures* (PMI). We also review two Federation solutions, Microsoft Passport and Shiboleth, as they have become important on the Internet. The reason for selecting these schemes is twofold. On the one hand, in some cases, they are supported by international bodies. This is the case of IETF, that has supported the two first ones through several RFCs. It is also the case of ITU-T, that has proposed and supported the PMIs. It is important to note that PMIs have been supported by the IETF too. On the other hand, schemes selected can be considered as practical solutions, so they can be deployed in the Internet more easily than other more specific approaches based on formalisms (graph theory or logic programming), like [13], [14], [19], [18].

The paper is structured as following. Section 2 covers PolicyMaker and KeyNote solutions, reviewing them briefly. Section 3 analyzes the SPKI/SDSI solution in global, but also the SDSI solution in particular. Section 4 introduces the concept of Federation, and section 5 elaborates on ITU-T PMI, introducing a new mechanism to extend their functionality using advanced delegation services. Finally, section 6 concludes the paper.

2 PolicyMaker and Keynote

Blaze, Feigenbaum and Lacy introduced in [4] the notion of *Trust Management*. In that original work they proposed the PolicyMaker scheme as a solution for trust management purposes. PolicyMaker is a general and powerful solution that allows the use of any programming language to encode the nature of the authority being granted as well as the entities to whom it is being granted. It addresses the authorization problem directly, without considering two different phases (one for authentication and another for access control). PolicyMaker encodes trust in assertions. They are represented as pairs (f, s), where s is the issuer of the statement, and f is a program.

Additionally, PolicyMaker introduces two different types of assertions: *certificates* and *policies*. The main difference between them is the value of the *Source* field. To be more precise, the value is a key for the first one (certificates), and a label for the second one (policies).

It is important to note that, in PolicyMaker, negative credentials are not allowed. Therefore, trust is monotonic; that is, each policy statement or credential can only increase the capabilities granted by others. Moreover, trust is also transitive. This means that if *Alice* trusts *Bob* and, extensively, *Bob* trusts emphCarol, then *Alice* trusts *Carol*. In other words, all authorizations are delegable. Indeed, delegation is implicit in PolicyMaker; thus, it is not possible to restrict delegation capabilities. This is the reason why delegation is uncontrolled in PolicyMaker.

KeyNote [3] has been proposed and designed to improve two main aspects of PolicyMaker: to achieve standardization and to facilitate its integration into applications. Keynote uses a specific assertion language that is flexible enough to handle the security policies of different applications. Assertions delegate the authorization to perform operations to other principals. As PolicyMaker, KeyNote considers two types of assertions. Also, as in PolicyMaker, these two types of assertions are called *policies* and *credentials*, respectively:

- Policies. This type of assertions does not need to be signed because they are locally trusted. They do not contain the corresponding *Issuer* of PolicyMaker.
- Credentials. This type of assertions delegate authorization from the issuer of the credential, or *Authorizer*, to some subjects or *Licensees* (see later for details).

Assertions are valid or not valid depending on *action attributes*, which are attribute/value pairs like `resouce=="database"` or `access=="read"`

KeyNote assertions are composed of five fields:

- `Authorizer`. If the assertion is a credential, then this field encodes the issuer of that credential. However, if the assertion is a policy, then this field contains the keyword **POLICY**.
- `Licensees`. It specifies the principal or principals to which the authority is delegated. It can be a single principal or a conjunction, disjunction or threshold of principals.
- `Comment`. It is a comment for the assertion.
- `Conditions`. It corresponds to the "program" concept of PolicyMaker, and consists of tests on action attributes. Logical operators are used in order to combine them.
- `Signature`. It is the signature of the assertion. This field is not necessary

for policies, only for credentials.

Further description on how KeyNote uses cryptographic keys and signatures can be found in [5].

Figure 1 shows an example of assertion. It states that an RSA key 12345678 authorizes the DSA keys *abcd*1234 and 1234*abcd* for read and write access on the database.

```
KeyNote-Version: 2
Authorizer: "rsa-hex:12345678"
Licensees: "dsa-hex:abcd1234" || "dsa-hex:1234abcd"
Comment: Authorizer delegates read and write access to
         either of the licensees
Conditions: (resource == "database" &&
            (access == "read") || (access == "write"))
Signature: "sig-rsa-md5-hex:00001234"
```

Fig. 1. **KeyNote assertion**

Given a set of action attributes, an assertion graph is a directed graph with vertex corresponding to principals. An arc exists from principal A to principal B if an assertion exists where the *Authorizer* field corresponds with A, the *Licensees* field corresponds with B and the predicate encoded in the *Conditions* field holds for the given set of action attributes. A principal is authorized, under a given set of action attributes, if the associated graph contains a path from a policy to the principal.

3 SDSI/SPKI

This solution is an unification of two similar proposals, SDSI (Simple Distributed Security Infrastructure) and SPKI (Simple Public Key Infrastructure). SPKI was proposed by the IETF working group and, in particular, by Carl Ellison [7]. SDSI was an alternative design to X.509 for a public-key infrastructure, and it was designed designed by Ronald L. Rivest and Butler Lampson [17].

The SPKI/SDSI certificate format is the result of the SPKI Working Group of the IETF [8]. The main feature of SDSI/SPKI is that its design provides a simple public key infrastructure which uses linked local name spaces rather than a global, hierarchical one. All entities are considered analogous; hence, every principal can produce signed statements.

The data format chosen for SPKI/SDSI is *S-expression*. This is a LISP-like parenthesized expression with the limitations that empty lists are not allowed and the first element in any S-expression must be a string, called the "type" of

the expression. In this section, we detail the SDSI solution and the integrated solution SDSI/SPKI, as the development of the SPKI solution is similar to the integrated solution. The subsections detail the certificates of each proposal and explain how the delegation is implemented.

3.1 SDSI

SDSI establishes four types of certificates: $Name/Value$, $Membership$, $Autocert$ and $Delegation$.

Name/Value Certificates: These certificates are used to bind principals to local names. Every certificate must be signed by the issuer, using his/her public key (figure 2).

Membership Certificates: These are certificates that give to principals the membership to a particular SDSI group.

Autocert Certificates: These are self-certificates, a special kind of certificate. Every SDSI principal is required to have an Autocert (figure 3).

Delegation Certificates: These certificates are the mechanisms for implementing the Delegation in SDSI (figure 3). SDSI provides two types of delegation, based on the structure of the delegation certificate:

(i) A user (issuer) can delegate to someone by adding that person as a member to a group issuer control. A issues a delegation certificate to B. Therefore, B will have the same privileges as the $group1$.
(ii) A user (issuer) can delegate to someone so that this person is able to sign objects of a certain type on the user's behalf. The "certain type" is defined by using the template form.

```
(Cert:
   (Local-Name: user1 )
   (Value:
      (Principal:
        (Public-Key:
        (Algorithm: RSA-with-SHA1 )
        ......
        )))
        (Signed: ...))
```

Fig. 2. **Name-Value certificates**

```
(Auto-Cert:
   (Local-Name: user1 )
     (Public-Key: ....)
     (Description: temporal user)
     (Signed: ...))
```

```
(Delegation-Cert:
   (Template: form )
     (Group: group1 )
     (Signed: ...))
```

Fig. 3. **Autocert and Delegation certificates**

3.2 Integrated Solution, SPKI/SDSI

SPKI/SDSI unifies all types of SDSI certificates into one single type of structure. The SPKI/SDSI certificate contains at least an *Issuer* and a *Subject*, and it can contain validity conditions, authorization and delegation information. Therefore, there are three categories: ID (mapping <name,key>), Attribute (mapping <authorization,name>), and Authorization (mapping < authorization, key >).

The structure of Figure 4 represents the *ID certificate* and the *Authorization Certificate*. The *Attribute Certificate* has the same structure as Authorization Certificates.

```
(cert
    (issuer  <principal>)
    (subject <principal>)
    (valid   <valid>))
```

```
(cert
    (issuer  <principal>)
    (subject <principal>)
    (propagate)
    (tag <tag>)
    (valid ))
```

Fig. 4. **ID and Authorization Certificates**

The field *propagate* is the field used to perform the delegation. As it was desirable to limit the depth of delegation, SPKI/SDSI initially had three options for controlling this: no control, boolean control and integer control. Actually these options have been reduced to boolean control only. In this way, if this field is true, the Subject is allowed by the Issuer to further propagate the authorization.

4 Federations

In this section we analyze some of the most interesting federation solutions that have been developed by different consortium or enterprises. We focus on two significant solutions such as Shibboleth [22], and .Net Passport [21].

These selected solutions represent both educational and enterprise points of view. Shibboleth is the representative for academia solutions, although there are other solutions like PAPI and Athens. On the other hand, we chose .Net Passport as the enterprise representative, although its opponent, Liberty Alliance [20], is growing in popularity, mainly due to the relevance of the partners that conforms the consortium.

The general definition of Federation is the act of establishing a trust relationship between two entities, or in more detail, an association comprising any number of service providers and identity providers. Therefore, Federation should be understood as delegation of services where the service providers delegate the security management to identity providers.

4.1 Microsoft Passport

At the end of 90's, as part of its .NET initiative, Microsoft introduced a set of Web services that implement a so-called "user-centric" application model, and that are collectively referred to as .NET My Services. At the core of Microsoft .NET My Services is a password-based user authentication and Single Sign-In service called Microsoft .NET Passport. The fundamental component of a Federation Solution is Single Sign-In (SSI) Service; therefore, Microsoft .NET Passport could be considered as the first partial Federation Solution.

Microsoft .NET Passport users are uniquely identified with an email address (usually hotmail and MSN accounts) and all participating sites are uniquely identified with their DNS name. A passport account has four parts. The first is a *Passport Unique Identifier* (PUID), assigned to the user when he/she sets up the account. This PUID is a 64-bit number that is sent to the user's site as the authentication credential when a Passport user signs in, being used in representation of the user for the administrative operations. The second is the user profile, containing the user's phone number or email address, user's name and demographic information. The third part of a passport account is the credential information such as the password or security key used for a second level of authentication. The wallet is the fourth element that enables users to digitally store credit card numbers, expiration dates, and billing and shipping addresses.

Passport use a series of cookies to store the authentication information and to assist the sing-in functionality in the user computer. During the early years, there were numerous security failures. The work by Kormann [11] enumerates a series of Passport flaws. The security issues are related to: User Interface, Key management, Cookies and Javascript, Persistent cookies and Automatic credential assignment.

In 2003, IBM, Microsoft, BEA, RSA and Verisign published a competing identity management framework called Web Services Federation Language, or WS-Federation which was intended to be the direct competitor of Liberty [15] although at this moment IBM, BEA, RSA and Verisign are part of the Liberty consortium.

4.2 Shibboleth

Shibboleth is a project of Internet2/MACE. The purpose of the proposal is typically to determine if a person using a web browser has the permissions to access a target resource based on information such as being a member of an institution or a particular class. It is implemented by using federated administration. In federated administration, a resource provider usually leaves the administration of user identities and attributes to the user's origin site. Therefore, users are registered only at their origin site, but not at each resource provider. Moreover, the system is privacy preserving in the sense that it does not use identity information. Therefore, it is necessary to associate a handle with the user. This handle stores the security information without exposing the identity of the user. Consequently, Shibboleth is a system for securely transferring attributes about a user, from the user's origin to a resource provider site. Two principal components are in charge in performing the attribute transference, Attribute Authority (AA) in the user side and Shibboleth Attribute Requester (SHAR) on the resource side. These components interchange authorization information by exchanging SAML [6] messages using any shared protocol that supports the required functional characteristics.

5 Privilege Management Infrastructure (PMI)

It is well known that by using an *authentication service* you can prove who you are. Identity certificates (or public-key certificates) provide the best solution to integrate that basic service into most applications developed for the Internet that make use of digital signatures. However, new applications need an *authorization service* to describe what it is allowed for a user to do. In this case, privileges to perform tasks should be considered.

For instance, when a company needs to establish distinctions among their employees regarding privileges over resources, the authorization service becomes important. Different sets of privileges over resources (either hardware or software) will be assigned to different categories of employees. In those distributed applications where company resources must be partially shared through the

Internet with other associated companies, providers, or clients, the authorization service becomes an essential part.

Authorization is not a new problem, and different solutions have been used in the past. However, "traditional" solutions are not very helpful for many of the Internet applications. Those solutions are not easy to use in application scenarios where the use of identity certificates, to attest the connection of public keys to identified subscribers, is a must. In such scenarios, types of independent data objects that can contain user privileges would be of great help. *Attribute certificates* proposed by the ITU-T (International Telecommunications Union) X.509 recommendation [10] provide an appropriate solution, as these data objects have been designed to be used in conjunction with identity certificates.

The use of a wide-ranging authentication service based on identity certificates is not practical unless it is complemented by an efficient and trustworthy mean to manage and distribute all certificates in the system. This is provided by a *Public-Key Infrastructure* (PKI), which at the same time supports encryption, integrity and non-repudiation services. Without its use, it is impractical and unrealistic to expect that large scale digital signature applications can become a reality [16],[1].

Similarly, the attribute certificates framework defined by ITU provides a foundation upon which a *Privilege Management Infrastructure* (PMI) can be built. PKI and PMI infrastructures are linked by information contained in the identity and attribute certificates of every user. The link is justified by the fact that authorization relies on authentication to prove who you are.

Although linked, both infrastructures can be autonomous, and managed independently. Creation and maintenance of identities can be separated from PMI because authorities that issue certificates in each of both infrastructures are not necessarily the same ones. In fact, the entire PKI may be existing and operational prior to the establishment of the PMI.

The last X.509 ITU-T Recommendation [10] establishes four PMI models: (i) *General*, (ii) *Control*, (iii) *Roles* and (iv) *Delegation*. The first one can be considered as an abstract model, while the other ones can be considered as the models for implementation.

The PMI area inherits many concepts from the *Public Key Infrastructure* (PKI) area. In this sense, an *Attribute Authority* (AA) is the authority that assigns privileges (through attribute certificates) to users, and the *Source of Authorization* (SOA) is the root authority in the delegation chain. A typical PMI will contain a SOA, a number of AAs and a multiplicity of *end entities* (EE) [9].

Figure 5 depicts the relation between the entities of a PMI in the Delegation Model. Initially, the Source of Authority assigns or delegates the privilege to Attribute Authorities. These can delegate the privileges to other AAs or to EEs.

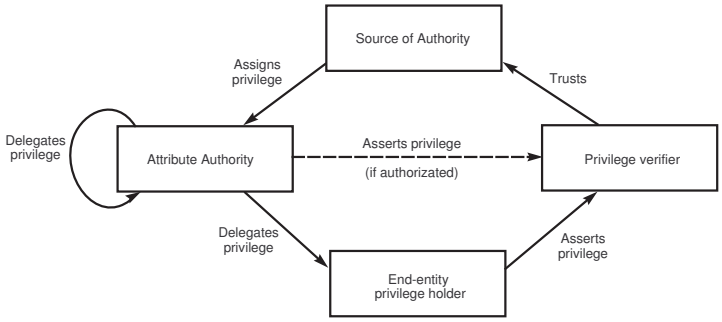


Fig. 5. **PMI Delegation Model**

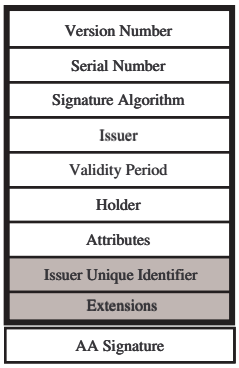
AAs and EEs can use their delegated privileges and present them to the *Privilege Verifier* (PV), that verifies the certification path to determine the validity of the privileges. The difference between AA and EE is that EE can not further delegate the privileges to other entities, becoming the leaves of the tree. The PV must trust the SOA in order to verify the certification path, as they may reside in different domains.

The mechanism (data structure) used to contain the delegation statement(s) is the attribute certificate. Figure 6 shows the structure of the attribute certificate, and how a delegation path is established through a chain of these certificates. The *Extensions* field is used by the authorities to include the delegation policy.

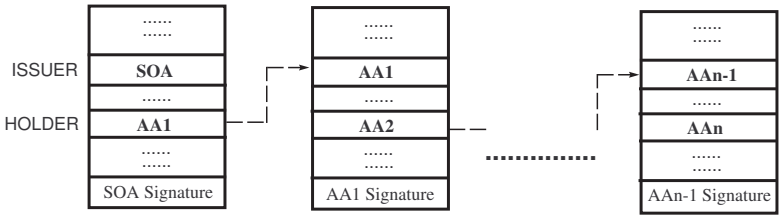
5.1 PMI Extensions

Extensions to X.509 certificates can be done by adding extra information in its extensions field. This field allows us to include additional information into the attribute certificate. Although the X.509 standard provides five predefined extension categories, we focus on the Delegation extension category, which defines different extension fields. Among them, the Recommendation includes:

Authority attribute identifier In privilege delegation, an AA that delegates privileges shall itself have at least the same privilege and the authority to delegate that privilege. An AA that is delegating privilege to another AA or to an end-entity may place this extension in the AA or end-entity certificate that it issues. The extension is a back pointer to the certificate in which the issuer of the certificate containing the extension was assigned its corresponding privilege. The extension can be used by a privilege verifier to



(a) X.509 Attribute Certificate



(b) Delegation Path

Fig. 6. **Delegation Elements in PMI**

ensure that the issuing AA had sufficient privilege to be able to delegate to the holder of the certificate containing this extension.

Different approaches are available to include mechanisms to control delegation in PMI. OpenPMI [23] project is based on the use of X509 Attribute Certificate and therefore is a practical implementation of a PMI. It provides a more complex mechanism to perform the delegation, allowing the possibility to use the extension fields of the attribute certificate to perform a controlled delegation. The proposal makes uses of graphs to model authorization and delegation relationships. The key of the project is to attach extra information to each edge in the graph. In particular, a real number in the interval [0,1] that measures the level of confidence of the issuer on the certificate is included. Moreover, it distinguishes between positive and negative statements. Positive ones grant the right encoded in the certificate and negative ones deny it. The variable "sign" is used for this purpose. It also adds another Boolean variable, "delegation", to define if the certificate can be chained with others, i.e. the attribute can be delegated.

This extension is defined in [2] using ASN.1 (figure 7), based on the Authority attribute identifier one. The new extension determines a sequence between the SOA and the holder. Each sequence includes other sequences, ArcsId, where to include the information of the arcs in the graph, weight of the arc, origin node, and boolean information about statements, delegation and sign.

The destination node must coincide with the serial number of the attribute certificate.

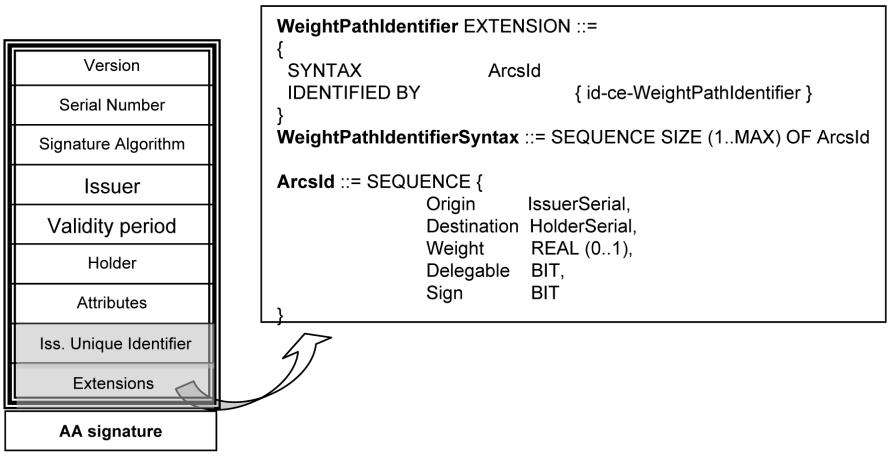


Fig. 7. **Attribute Certificate and Weight Path Identifier Extension**

The main contribution of OpenPMI project is the design of authorization and delegation statements in a graphical way that afterwards can be automatically turned to X509 attribute certificate chains. The example of figure 8 shows the graphical design of delegation statements (normal line) and authorization statements (dashed line) and its equivalent representation using attribute certificates. Each attribute certificate stores, in the extensions field, the information about the graph.

6 Conclusions

Delegation is needed to obtain a real scalable distributed authorization. However, the uncontrolled use of delegation statements can become a security threat because any user could improperly get the same privileges over a resource than the owner of that resource. Therefore, delegation solutions must include a mechanism to control the delegation and to produce appropriate authorizations statements. In this paper, our goal has been to study and put into perspective the delegation implications of a group of schemes that have been proposed as solutions for distributed authorization problems. In *PolicyMaker* and *Keynote* the delegation statement does not exist, and any authorization statement can be delegated again and again without any control. *SDSI* considers three different possibilities to control the delegation, although *SPKI* has reduced it to a boolean condition. Such a boolean parameter is only a modest mechanism to control the depth of the delegation. Federations, included here to show how commercial solutions like Microsoft Passport fit in the Authorization and Delegation picture, are a step ahead PKI but still do not provide full delegation capabilities. On the contrary, the *PMI* solution provides more

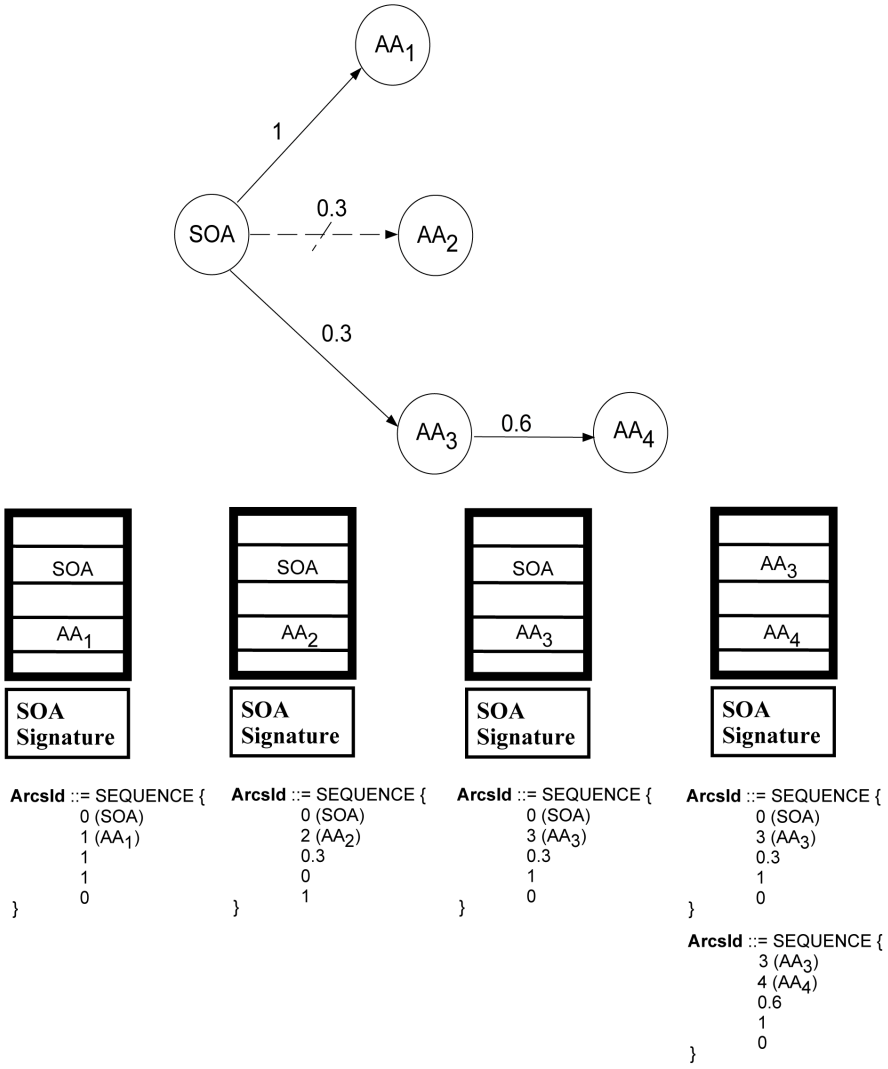


Fig. 8. **Design of Certificates and its corresponding certificate chains**

complex mechanisms to perform the delegation, allowing the possibility to use the extension fields of the attribute certificate for a controlled delegation.

References

[1] Adams, C., Lloyd, S. "Understanding Public-Key Infrastructure: Concepts, Standards and Deployment Considerations", New Riders, 1999

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

[3] Blaze, M., Feigenbaum, J., Ioannidis, J. and Keromytis, A. 1999. "The KeyNote

Trust-Management System Version 2." *RFC 2704*.

[4] Blaze, M., Feigenbaum, J. and Lacy, J. 1996. "Decentralized Trust Management." In *IEEE Symposium on Security and Privacy*. IEEE Computer Society Press pp. 164–173.

[5] Blaze, M., Ioannidis, J., and Keromytis, A. 2000. "DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System." *RFC 2792*.

[6] Cantor, S., Kemp, J., Philpott, R., Maler, E (2005). Security Assertion Markup Language (SAML V2.0). OASIS. Retrieved March 15, 2005, from http://docs.oasis-open.org/security/ saml/v2.0/

[7] Ellison, C., Frantz, B. and Lacy, J. 1996. "Simple public key certificate". *Internet Draft* `http://world.std.com/~cme/spki.txt`

[8] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B. and Ylonen, T. 1999. "SPKI Certificate Theory", *RFC 2693*.

[9] Farrell, S. and Housley, R. 2002. "An Internet Attribute Certificate Profile for Authorization", *RFC 3281*.

[10] ITU-T X.509, ISI/IEC 9594-8, Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. 08/2005.

[11] Kormann, D., Rubin, A 2000. Risks of the Passport Single Signon Protocol. Computer Networks, Elsevier Science Press, volume 33, pages 51-58, 2000

[12] Lampson, B. 2004. "Computer security in the real world." *IEEE Computer Society Press* 37(6):37,46.

[13] Li, N., Mitchell, J.C. and Winsborough, W.H. 2002. "Design of a Role-Based Trust Management Framework." In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press pp. 114–130.

[14] Li, N., Grosof, B. and Feigenbaum, J. 2003. "Delegation logic: A logic-based approach to distributed authorization." *ACM Trans. Inf. Syst. Secur.* 6(1):128–171.

[15] Liberty Alliance Project White Paper (2003). Liberty Alliance & WS-Federation: A Comparative Overview. Retrieved October 14, 2003, from http://www.liberty.org Rivest, R. and Lampson, B. 1996.

[16] Nash,A., Duane,W., Joseph,C., Brink, D."PKI: Implementing and Managing E-Security", McGraw-Hill, 2001

[17] "SDSI - A Simple Distributed Security Infrastructure." *Working document*, Presented at CRYPTO '96 Rumpsession.

[18] Ruan, C. and Varadharajan, V. 2004. "A Weighted Graph Approach to Authorization Delegation and Conflict Resolution." In *ACISP 2004*. Vol. 3108 of *Lecture Notes in Computer Science*, Springer pp. 402–413.

[19] Varadharajan, V., Ruan, C. and Yan Zhang. 2003. "A Logic Model for Temporal Authorization Delegation with Negation." In *6th International Information Security Conference, ISC*. Vol. 2851 of *Lecture Notes in Computer Science*, Springer pp. 310–324.

[20] http://www.liberty.org

[21] http://www.passport.com

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

[23] http://openpmi.sourceforge.net