S. Tuecke
D. Engert
I. Foster
ANL
V. Welch
U. Chicago
M. Thompson
LBNL
L. Pearlman
C. Kesselman
USC/ISI

Expires: February 2002
August 2001

Internet X.509 Public Key Infrastructure Proxy Certificate Profile

1 Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

2 Abstract

This document forms a certificate profile for Proxy Certificates, based on X.509 PKI certificates as defined in draft-ietf-pkix-new-part1-08.txt (the draft update to RFC 2459), for use in the Internet. The term Proxy Certificate is used to describe a certificate that is derived from, and signed by, a normal X.509 Public Key End Entity Certificate or by another Proxy Certificate for the purpose of providing restricted impersonation within a PKI based authentication system.

3 Table of Contents

4 Introduction

Use of a proxy credential for impersonation is a common technique used in security systems to allow entity A to grant to another entity B the right for B to authenticate with others as if it were A. In other words, entity B is impersonating entity A. This document forms a certificate profile for Proxy Certificates, based on the draft update to RFC 2459, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile" [7].

In addition to simple, unrestricted impersonation, this profile defines a framework for carrying restriction policies in Proxy Certificates, thus allowing a restriction of the rights an impersonating entity is given. Further, when delegating a Proxy Certificate from one entity to another, this profile defines information that can be optionally included in a Proxy Certificate to allow for tracing of the delegation path.

Section 2 provides an overview of the approach. It begins by defining terminology, motivating Proxy Certificates, and giving a brief overview of the approach. It then introduces the notion of a Proxy Authority, as distinct from a Certificate Authority, to describe how end entity signing of a Proxy Certificate is different from end entity signing of another end entity certificate, and therefore why this approach does not violate the end entity signing restrictions contained in the X.509 keyCertSign field of the keyUsage extension. It then continues with discussions of how subject names are used by this impersonation approach, and features of this approach.

Section 3 defines requirements on information content in Proxy Certificates. This profile addresses two fields in the basic certificate as well as five certificate extensions. The certificate fields are the subject and issuer fields. The certificate extensions are subject alternative name, issuer alternative name, key usage, basic constraints, and extended key usage. One new certificate extensions, Proxy Certificate Information, is introduced.

Section 4 defines path validation rules for Proxy Certificates.

Section 5 discusses the relationship of Proxy Certificates to Attribute Certificates.

Section 6 provides commentary on various design choices, open issues, related work, and future directions.

Section 7 discusses security considerations relating to Proxy Certificates.

Section 8 contains the references.

Section 9 contains acknowledgements.

Section 10 contains a log of changes made in each version of this draft.

Section 11 contains contact information for the authors.

This document was written under the auspices of the Global Grid Forum Security Working Group. For more information on this and other related work, see http://www.gridforum.org/security.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

5 Overview of Approach

The goal of this specification is to develop a X.509 Proxy Certificate profile, to facilitate their use within Internet applications for those communities wishing to make use of impersonation within an X.509 PKI authentication based system.

This section provides relevant background, motivation, an overview of the approach, and related work.

5.1 Terminology

This document uses the following terms:

- CA: A "Certificate Authority", as defined by X.509 [7].
- EEC: An "End Entity Certificate", as defined by X.509. That is, it is an X.509 Public Key Certificate issued to an end entity, such as a user or a service, by a CA.
- PKC: An end entity "Public Key Certificate". This is synonymous with an EEC.
- PC: A "Proxy Certificate", the profile of which is defined by this document.
- PA: A "Proxy Authority" is the issuer of a Proxy Certificate, as defined below.
- AC: An "Attribute Certificate", as defined by "An Internet Attribute Certificate Profile for Authorization" [4].
- AA: An "Attribute Authority", as defined in [4].

5.2 Background

Computational and Data "Grids" have emerged as a common approach to constructing dynamic, inter-domain, distributed computing environments. As explained in [6], large research and development efforts starting around 1995 have focused on the question of what protocols, services, and APIs are required for effective, coordinated use of resources in these Grid environments.

In 1997, the Globus Project (www.globus.org) introduced the Grid Security Infrastructure (GSI) [5]. This library provides for public key based authentication and message protection, based on

standard X.509 certificates and public key infrastructure, the SSL/TLS protocol [3], and delegation using proxy certificates similar to those profiled in this document. GSI has been used, in turn, to build numerous middleware libraries and applications, which have been deployed in large-scale production and experimental Grids [2]. GSI has emerged as the dominant security solution used by Grid efforts worldwide.

This experience with GSI has proven the viability of impersonation as a basis for authentication and authorization within Grids, and has further proven the viability of using X.509 Proxy Certificates, as defined in this document, as the basis for that impersonation. This document is one part of an effort to migrate this experience with GSI into standards, and in the process clean up the approach and better reconcile it with existing and recent standards.

5.3 Motivation for Impersonation

A motivating example will assist in understanding the role impersonation can play in building Internet based applications.

Steve is an engineer, who wants to run a set of simulation jobs on idle workstations on his company's Intranet based Grid. From his laptop he wants to invoke the jobs, and then have an agent process running on his desktop workstation monitor the jobs while he is traveling to a conference. As the jobs complete, the agent should automatically archive the results to the company's mass storage system, and after all the jobs are complete it should run a post-processing job which summarizes the simulation results from all of the archived data sets. Later, Steve will reconnect to the agent to get the results for inclusion in a report. Of course, he wants all of this to happen securely on his company's resources, which requires that he initiate all of this using his PKI smartcard.

This scenario requires authentication and delegation in a variety of places:

- Steve needs to be able to mutually authenticate with several remote workstations to start the simulation jobs.

- Steve needs to be able to mutually authenticate with his desktop workstation to start the agent running.

- That agent needs to be delegated the rights to mutually authenticate with the various workstations, in order to monitor the progress of the simulations.

- As simulations complete, the agent needs to move the resulting data from the workstations to the company's mass storage system. In order to perform this move efficiently, it needs to orchestrate a third party data transfer directly between the workstation and the mass storage system. This requires mutual authentication between the agent and the workstations and mass storage system, as well as mutual authentication between the workstations and the mass storage system.

- The agent needs to start the post-processing job, which must be delegated rights to mutually authenticate with the mass storage system in order to retrieve the data.

- When Steve later reconnects his laptop to the network, a program running on the laptop must mutually authenticate with the agent in order to retrieve the summary of results.

Impersonation is a viable approach to solving two (related) problems in this scenario:

- *Single sign-on*: Steve wants to enter his smartcard password (or pin) once, and then run a program that will start all of the simulation jobs and the remote agent. This program needs to be given the rights to be able to perform all of these operations securely, without requiring repeated access to the smartcard or Steve's password.

- *Delegation*: Various remote processes in this scenario need to perform secure operations on Steve's behalf, and therefore must be delegated the necessary rights. For example, the agent needs to be able to authenticate on Steve's behalf with the various workstations and the mass storage system, and must in turn delegate rights to the post-processing job to authenticate on Steve's behalf with the mass storage system.

Impersonation can be used to secure all of these interactions:

- Impersonation allows for the private key stored on the smartcard to be accessed just once, in order to create the necessary impersonation credential, which allows the starter program to impersonate Steve (that is, authenticate as Steve) when starting the various jobs and the agent. Access to the smartcard and Steve's password is not required after the initial creation of the impersonation credential.

- The starter program on the laptop can delegate to the agent the right to impersonate Steve. This, in turn, allows the agent to authenticate to the workstations as if it were Steve in order to start the simulation jobs, and to the mass storage system to archive the data as if it were Steve.

- When the agent starts the post-processing job, the agent can delegate to it the right to impersonate Steve. This allows the post-processing job to authenticate as Steve to the mass storage system in order to gain access to the data sets.

- When the laptop reconnects to the agent to get the final results, it can perform mutual authentication. The agent will use its delegated impersonation credential in this interaction. The laptop may use a newly generated impersonation credential, which is just created anew using the smartcard.

While this example may seem somewhat contrived, similar applications are already being built today within the Grid community. The Grid Security Infrastructure's single sign-on and delegation capabilities, built on X.509 Proxy Certificates, are being employed to provide authentication services to these applications.

5.4 Motivation for Proxy Restrictions

One concern that arises is what happens if a machine that has been given the right to impersonate Steve has been compromised? Can the attacker now do everything that Steve is allowed to do? A solution to this problem is to allow for restrictions to be placed on the impersonation. For example, the machine running the post-processing job in the example in section 5.3 above might only be given the right to impersonate Steve for the purpose of reading the simulation output files from the mass storage system. Therefore, if that host is compromised, raw simulation data cannot be changed on the mass storage system, new jobs cannot be started, etc.

5.5 Motivation for Proxy Groups

Users will often wish to delegate authority to multiple tasks running on their behalf. These tasks in turn often delegate authority to subtasks that they spawn and so forth. These tasks often then use the delegated credentials to authenticate to each other for purposes of control, synchronization, data transfer, etc.

The user often wishes to federate these tasks to limit their interactions. For example one task may be running on a less secure resource, or one task may be handling highly sensitive data. In these cases the user will not want to allow processes from one task to be able to authenticate to a process that is part of the other task.

While it is in theory possible to implement this functionality as a restriction with Proxy Restrictions, the complexity of interactions of processes in a task often makes enumerating a list of restrictions cumbersome. By assigning processes to groups and then limiting interactions between processes based on the groups this is greatly simplified. Also, by putting this into a separate field, this allows entities needing to evaluate Proxy Groups not to have to evaluate the potentially complex Proxy Restrictions.

5.6 Description Of Approach

This document defines an X.509 "Proxy Certificate" or "PC" as a means of providing for impersonation with an X.509 PKI based authentication system.

A Proxy Certificate is an X.509 public key certificate with the following properties:

1) It is signed by either an X.509 End Entity Certificate (EEC), or by another PC.

2) It can sign only another PC.

3) It has its own public and private key pair, distinct from any other EEC or PC.

4) It has no distinct identity of its own. After a PC is used for authentication, the identity that is used for authorization is that of the EEC that signed the PC. The PC effectively inherits the subject or subjectAltName from its signing EEC.

5) It contains a new X.509 extension to identify it as a PC and to place restrictions on the PC. This new extension, along with other X.509 fields and extensions, are used to enable proper path validation and use of the PC.

The process of creating a PC is as follows:

1) A new public and private key pair is generated.

2) That key pair is used to create a request for a Proxy Certificate that conforms to the profile described in this document.

3) A Proxy Certificate, signed by the private key of the EEC or by another PC, is created in response to the request. During this process, the PC request is verified to ensure that the requested PC is valid (e.g. it is not an EEC, the PC fields are appropriately set, etc).

When a PC is created as part of a delegation from entity A to entity B, this process is modified by performing steps #1 and #2 within entity B, then passing the PC request from entity B to entity A over an authenticated, integrity checked channel, then entity A performs step #3 and passes the PC back to entity B. (Note: There is a related draft that describes how this delegation approach can be incorporated into the TLS protocol [8].)

Path validation of a PC is very similar to normal path validation, with a few additional checks to ensure, for example, proper PC signing constraints. In order to make the appropriate PC(s) and EEC available for path validation, the authentication protocol using the PC (e.g. TLS) may pass the entire PC and EEC chain as part of the authentication protocol.

5.7 Proxy Authority, not Certificate Authority

A common initial reaction against the approach described in this document is, "You are using the end entity certificate (EEC) as a CA!" However, this is not the case. To understand why, one must first understand what a CA does.

In issuing an EEC, a CA performs two primary functions:

1) *Naming*: The CA assigns a (generally unique) "Name" to the end entity to which it issues an EEC. This Name is contained in the subject or subjectAltName field of the issued EEC.

2) *Key to Name binding*: By singing an EEC with the CA's private key, the CA is providing a means to allow an authenticating party to verify that the holder of a particular private key should be associated with (bound to) a particular Name.

In addition, a CA usually has an associated *Registration Authority*, which performs the checks necessary to bind the Name to the real world entity (e.g. person, computer, etc) that is to be the bearer of that Name.

The reason for doing all of this is to allow for authorization decisions to be made, based at least in part on these CA issued Names. In other words, after the public key authentication operation

has determined the Name of the authenticating party, then that Name can be used as the basis for deciding what the entity is allowed to do. (Note: Attribute certificates are discussed below.)

The critical difference between using an EEC to sign a Proxy Certificate, versus using an EEC to sign another EEC, is that a Proxy Certificate does NOT define a new Name. Rather, a Proxy Certificate inherits the name from the EEC that signs it. The next section describes this inheritance in more detail.

In effect, the PC simply provides another route to validating the Key to Name binding that the CA has established with an EEC. A PC allow an alternate Key' to bind to the same Name, optionally with restrictions, with this Key' to Name binding vouched for by the holder of the EEC private key. This allows entity A to give to entity B the ability to establish this binding, and thus allows B to establish itself as a proper bearer of A's Name.

For this reason, we use the term "Proxy Authority", rather than "Certificate Authority", to refer to the issuer of a Proxy Certificates. A Proxy Authority does not perform the Naming function of a Certificate Authority, but rather just a Key to Name binding.

5.8 Names Versus Subjects

In X.509 certificates, the subject (or subjectAltName) is used for two distinct purposes:

1) In an End Entity Certificate, the subject is the Name that the CA has issued, as described in the previous section. This Name is typically used for authorization purposes.

2) In a CA Certificate, the subject is also used for path validation. That is, the issuer field in an EEC or CA Certificate must match the subject field of a CA Certificate, in order for the signing path to be established.

As stated previously, a PC does not have its own Name, but rather it inherits its Name from its signing EEC (or more accurately, from the EEC that signed the first PC in the PC chain). In practice what this means is that the subject field of a PC is only used for purpose #2. The only purpose of the subject field of a PC is to establish the signing path that eventually leads to an EEC.

The implication of this is that after a PC is used for authentication, the PC subject should not be used for authorization. Instead, the PC signing chain should be followed to find the EEC that signed this PC chain, and the subject from that EEC should be used as the identity (or Name) for authorization purposes.

To discourage mistakes in this area, this Proxy Certificate profile defines that the PC subject (actually its subjectAltName) is just a pseudo-randomly generated string. Further, the subject of the EEC is not maintained anywhere in the PC, which forces the authenticating party to properly retrieve the subject from the EEC.

5.9 Features Of This Approach

Using Proxy Certificates to perform delegation has several features that make it attractive:

- Ease of integration
    - Because a PC requires only a minimal change to path validation, it is very easy to incorporate support for Proxy Certificates into existing X.509 based software. For example, SSL/TLS requires no protocol changes to support authentication using a PC, and only small changes to support delegation of a PC [8]. Further, an SSL/TLS implementation requires only minor changes to support PC path validation, and to retrieve the authenticated subject of the signing EEC instead of the subject of the PC.
    - Many existing authorization systems use the X.509 subject name as the basis for access control. Proxy Certificates require no change to such authorization systems, since a PC inherits its name from the EEC that signed it.
- Ease of use
    - Using PC for single sign-on helps make X.509 PKI authentication easier to use, by allowing users to "login" once and then perform various operations securely.
    - For many users, properly managing their own EEC private key is a nuisance at best, and a security risk at worst. One option easily enabled with a PC is to manage the EEC private keys and certificates in a centrally managed repository. When a user needs a PKI credential, the user can login to the repository using name/password, one time password, etc. Then the repository can delegate a PC to the user, but continue to protect the EEC private key in the repository.
- Protection of private keys
    - By using the remote delegation approach outlined above, entity A can delegate a PC to entity B, without entity B ever seeing the private key of entity A, and without entity A ever seeing the private key of the newly delegated PC held by entity B. In other words, private keys never need to be shared or communicated by the entities participating in a delegation of a PC.
    - When implementing single sign-on, using a PC helps protect the private key of the EEC, because it minimizes the exposure and use of that private key. For example, when an EEC private key is password protected on disk, the password and unencrypted private key need only be available during the creation of the PC. That PC can then be used for the remainder of its valid lifetime, without requiring access to the EEC password or private key. Similarly, when the EEC private key lives on a smartcard, the smartcard need only be present in the machine during the creation of the PC.
- Limiting consequences of a compromised key
    - When creating a PC, the PA can limit the validity period of the PC, the depth of the PC path that can be created by that PC, and key usage of the PC and its descendents. Further, fine grained restriction policies can be carried by a PC to even further restrict

the operations that can be performed using the PC. This permits the PA to limit any damage that could be done by the bearer of the PC, either accidentally or maliciously.

- A compromised PC private key does NOT compromise the EEC private key. This makes a short term, or an otherwise restricted PC attractive for day-to-day use, since a compromised PC does not require the user to go through the usually cumbersome and time consuming process of having the EEC with a new private key reissued by the CA.

See Section 5 below for more discussion on how Proxy Certificates relate to Attribute Certificates.

6 Certificate and Certificate Extensions Profile

This section defines the usage of X.509 certificate fields and extensions in Proxy Certificates, and defines one new extension for Proxy Certificate Information.

6.1 Issuer & Issuer Alternative Name

The Proxy Authority (i.e. the issuer) of a Proxy Certificate MUST be either an End Entity Certificate, or another Proxy Certificate.

If the Proxy Authority Certificate has a non-empty subject field, then the issuer field of the Proxy Certificate MUST contain the subject of the Proxy Authority Certificate.

Otherwise, if the Proxy Authority Certificate has an empty subject field, but non-empty subjectAltName, then the issuer field of the Proxy Certificate MUST be an empty sequence, the issuerAltName MUST be the subjectAltName of the Proxy Authority Certificate, and the issueAltName MUST be critical.

6.2 Subject & Subject Alternative Name

The subject field of a Proxy Certificate MUST be an empty sequence.

The subjectAltName extension of a Proxy Certificate MUST be an otherName, using the impersonationCertName OID (?) and an IA5String (?) containing the name of the Proxy Certificate.

The subjectAltName extension MUST be critical.

The subjectAltName of a Proxy Certificate SHOULD only be used for path validation. As such, the string chosen for the subjectAltName of a Proxy Certificate is arbitrary, but SHOULD be (statistically) unique in order to enable path validation.

6.3 Key Usage

If the issuer certificate includes the keyUsage extension, then the Proxy Certificate MUST include a keyUsage extension, which MAY further restrict the issuer's keyUsage.

If the issuer certificate does not include a keyUsage extension, then the Proxy Certificate MAY include a keyUsage extension to restrict the key usage of the Proxy Certificate.

The keyUsage extension MUST be critical.

If the keyUsage extension is present in a Proxy Certificate, it must conform to the following restrictions:

The keyCertSign bit MUST NOT be asserted.

The following restriction applies to each of these bits: digitalSignature, nonRepudiate, keyEncipherment, dataEncipherment, keyAgreement, cRLSign, encipherOnly, decipherOnly. If this bit in the issuer certificate is not asserted, then this bit in the Proxy Certificate MUST NOT be asserted. If this bit in the issuer certificate is asserted, or if the issuer certificate does not include a keyUsage extension, then this bit in the Proxy Certificate MAY be either asserted or not asserted.

See the commentary in section 6 for more information on the keyCertSign and nonRepudiate bits.

6.4 Extended Key Usage

If the issuer certificate includes the extKeyUsage extension, then:

The Proxy Certificate MUST include an extKeyUsage extension.

Any OID that is contained in the Proxy Certificate's extKeyUsage extension MUST be present in the issuer certificate's extKeyUsage extension.

The Proxy Certificate's extKeyUsage extension MAY omit any OID that is present in the issuer certificate's extKeyUsage.

If the issuer certificate's extKeyUsage extension is critical, then the Proxy Certificate's extKeyUsage MUST be critical.

If the issuer certificate's extKeyUsage extension is not critical, then the Proxy Certificate's extKeyUsage MAY be critical or non-critical.

If the issuer certificate does not include the extKeyUsage extension, then the Proxy Certificate MAY include a extKeyUsage extension to restrict the key usage of the Proxy Certificate. In this case, the extKeyUsage extension MAY be critical or non-critical.

6.5 Basic Constraints

The cA field in the basic constraints extension MUST NOT be TRUE.

6.6 Proxy Certificate Information

Two new extensions, ProxyCertInfo and DelegationTracing, are defined in the following subsections

6.6.1 The ProxyCertInfo Extension

The ProxyCertInfo extension indicates whether or not a certificate is a Proxy Certificate and whether or not the issuer of the certificate has placed any restrictions on its use.

```
id-ce-proxy-cert-info OBJECT IDENTIFIER ::=  { id-ce ?? }

ProxyCertInfo ::= SEQUENCE {
    pC                  BOOLEAN DEFAULT TRUE,
    pCPathLenConstraint  INTEGER (0..MAX) OPTIONAL,
    proxyRestriction     ProxyRestriction OPTIONAL,
    proxyGroup           ProxyGroup OPTIONAL,
    issuerCertHash       Hash OPTIONAL }

ProxyRestriction ::= SEQUENCE {
    policyLanguage       OBJECT IDENTIFIER,
    policy               OCTET STRING }

Hash ::= SEQUENCE {
    hashAlgorithm        AlgorithmIdentifier,
    hashValue            BIT STRING }

ProxyGroup :: = SEQUENCE {
    proxyGroupName       OCTET STRING,
    proxyGroupAttached   BOOLEAN DEFAULT TRUE }
```

If a certificate is a Proxy Certificate, then the proxyCertInfo extension MUST be present, the pC field MUST be TRUE, and this extension MUST be marked as critical.

A Proxy Certificate MUST NOT be used to sign an End Entity Certificate or a CA Certificate.

If a certificate is not a Proxy Certificate, then the proxyCertInfo extension MAY be present, and MAY appear as a critical or non-critical extension. In this case, if this extension is present, then the pC field MUST be FALSE.

The ProxyCertInfo extension consists of one required and four optional fields, which are described in detail in the following subsections.

1.1.1.1. pC

As described above, the pC field indicates whether or not the certificate is a proxy certificate: if the certificate is a proxy certificate, the pC field MUST be TRUE; otherwise, the pC field MUST be FALSE.

1.1.1.2. pCPathLenConstraint

The pCPathLenConstraint field, if present, specifies the maximum depth of the path of Proxy Certificates that can be signed by this End Entity Certificate or Proxy Certificate. A pCPathLenConstraint of 0 means that this certificate MUST not be used to sign a Proxy Certificate. If the proxyCertInfo extension is not present, or if the pCPathLenConstraint is not present, then the proxy path length is unlimited.

1.1.1.3. proxyRestriction

The proxyRestriction field, if present, specifies restrictions on the use of this certificate. If the certificate is not a Proxy Certificate (i.e, if the pC field is FALSE), then the proxyRestriction field MUST NOT be present.

An unrestricted proxy is a statement that the PA wishes to delegate all its authority to the bearer (i.e., to anyone who has that proxy certificate and can prove possession of the associated private key). Proxy restrictions are used to limit the amount of authority delegated, for example to assert that the proxy certificate may be used only to make requests to a specific server, or only to authorize specific operations on specific resources.

Within the proxyRestriction, the policy field is an expression of policy, and the policyLanguage field indicates the language in which the policy is expressed.

Proxy restrictions impose additional requirements on the relying party, because only the relying party is in a position to ensure that those restrictions are met. When making an authorization decision based on a proxy certificate, it is the relying party's responsibility to verify that the requested authority is compatible with all restrictions in the PC's certificate path. In other words, the relying party MUST verify that the following three conditions are met:

1) If the PC includes a proxy restriction, then the relying party knows how to interpret the policy expressed in the PC's restriction, and the request is allowed under that policy.

2) If the PA is an EEC, then the relying party's local policies authorize the request for the entity named in the EEC.

3) If the PA is another PC, then conditions (1), (2), and (3) are met for the PA.

If these conditions are not met, the relying party MUST either deny authorization or ignore the PC entirely when making its authorization decision (i.e., make the same decision that it would

have made had the PC never been presented). Note that this verification MUST take place regardless of whether or not the PC itself contains restrictions, as other PCs in the signing chain may contain conditions that must be verified.

The relying party MAY impose additional restrictions as to what proxy certificates it accepts. For example, a relying party may choose to reject all proxy certificates, or to accept only those proxy certificates that include delegation tracing information, or to accept proxy certificates only for certain operations, etc.

The rights granted to the bearer of a PC will, then, be (at most) the intersection of the set of rights granted to the entity named in the EEC in the PC's certificate path, and the sets of rights authorized by the policies in each proxyRestriction that appears in the certificate path. For example, imagine that Steve is authorized to read and write files A and B on a file server, and that he uses his EEC to create a PC that includes the restriction that it can be used only to read or write files A and C. At most, the rights granted to the bearer of that PC will be the right to read and write file A -- a request to read file B, for example, would be rejected because it would be incompatible with the proxy restriction, and a request to read file C would be rejected because the file server's local policies do not grant Steve any access to file C. If that PC were then used to create a new PC that includes the restriction that it can be used only to read files, then the bearer of that new PC would have, at most, the right to read file A.

In many cases, the relying party will not have enough information to evaluate the above criteria at the time that the certificate itself is validated. For example, if a certificate is used to authenticate a connection to some server, that certificate is typically validated during that authentication step, before any requests have been made of the server. In that case, the relying party MUST either have some authorization mechanism in place that will check the proxy restrictions, or reject any certificate that contains proxy restrictions (or that has a parent certificate that contains proxy restrictions).

1.1.1.4. proxyGroup

The ProxyGroup field provides a method of assigning a PC to a group that serves as a method to limit a PC's ability to do self-authentication (authentication with entities authenticating with a PC derived from the same EEC as the original party). This field, if present, assigns the PC to a subgroup, identified by the proxyGroupName field, of the group of it's Proxy Authority.

The proxyGroupAttached field indicates whether this subgroup is attached to it's parent group in terms of the trust model. If a subgroup is attached, proxies in the subgroup (and it's descendants) are considered trusted for self-authentication by proxies in the parent group (and it's ancestors). If a subgroup is detached then proxies in the subgroup (and it's descendants) are considered untrusted for self-authentication by proxies in the parent group (and it's ancestors).

The Proxy Certificate group namespace is hierarchical, with the namespace being defined by the End Entity Certificate. In other words, two Proxy Certificates having the same group name is only meaningful if they both have the same EEC at the root of their signing chain.

The EEC is always considered to be in the group that is the root of the namespace. Each Proxy Certificate in a chain can then be in a subgroup of the PA that issued it. The full group name of a Proxy Certificate is the sequence of subgroup names in ProxyCertInfo extensions starting in the signing chain starting with the EEC.

If two parties are doing self-authentication, not only should they verify that they each have a PC derived from the same EEC, but they should make sure that the groups of their PCs are compatible. Compatibility is defined as being in groups that are a direct attached ancestors or descendants of each other. E.g. a parent and an attached child group are compatible, but siblings groups are not.

If a certificate is not a proxy certificate, the proxyGroup field MUST NOT be present.

1.1.1.5. issuerCertHash

The issuerCertHash field, if present, is used during path validation to ensure that each Proxy Certificate Path (the subset of a PC's certificate path that starts at an End Entity Certificate and ends at the PC) is unique. In other words, if certificate N+1 in a certificate path is a Proxy Certificate, then issuerCertHash is used to verify that certificate N is actually the PA that issued it and not some other certificate with the same name and public key. Without this field, if a PA were to issue two different proxy certificates (P1 and P2) with the same subjectAltName and public key but different proxy restrictions or validity time constraints, then the path validation algorithm would accept a path in which P2 appeared as the issuer of a certificate that had really been issued by P1.

This field consists of the following two subfields:

- hashAlgorithm MUST be identical to the PA's signatureAlgorithm.
- hashValue MUST be identical to the PA's signatureValue.

This field MUST be present if the pC field is TRUE.

6.6.2 The DelegationTrace Extension

The DelegationTrace extension is used to provide information about the identity of the Acceptor of a Proxy Certificate and, in some cases, to demonstrate that the Acceptor has agreed to accept the Proxy Certificate. If a Proxy Certificate does not include policy extensions, the Acceptor's agreement to "accept" that certificate is not an agreement to accept any additional responsibilities, such as safeguarding the Proxy Certificate's private key.

If the DelegationTrace extension is present, then the certificate MUST be a Proxy Certificate: the ProxyCertInfo extension MUST also be present, and the ProxyCertInfo.pC field MUST be TRUE. The DelegationTrace extension MAY be present in any proxy certificate, and SHOULD be present in any Proxy Certificate whose issuer is a Proxy Certificate in which the DelegationTrace extension is present. This extension SHOULD NOT be marked critical.

```
id-ce-delegation-trace OBJECT IDENTIFIER ::=  { id-ce ?? }

DelegationTrace ::= CHOICE {
    x509          [0]  X509DelegationTrace }

X509DelegationTrace ::= SEQUENCE {
    agreedCertInfo      AgreedCertInfo,
    x509AcceptorInfo    X509AcceptorInfo }

AgreedCertInfo ::= SEQUENCE {
    ignoredExtensions   SEQUENCE OF OBJECT IDENTIFIER,
    certSubsetHash      Hash }

X509AcceptorInfo ::= SEQUENCE {
    acceptorSig         Signature,
    acceptorName        Name,
    acceptorAltName     GeneralName OPTIONAL,
    acceptorCertHash    Signature }

Signature ::= SEQUENCE {
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }
```

The DelegationTrace extension consists of information regarding the certificate's Acceptor, in a format appropriate for the mechanism that was used by the Acceptor to authenticate to the Proxy Authority. Currently, the only format defined is X509DelegationTrace, which is intended for use when that authentication took place using X.509 certificates, or when the Acceptor and the PA are the same entity.

The X509DelegationTrace structure is used to verify that, at the time the Proxy Certificate was issued, the Acceptor had agreed to accept it. This structure consists of two required fields: the agreedCertInfo field, which contains hashes of some information related to the certificate, and the acceptorInfo field, which contains the Acceptor's signature of the agreedCertInfo, plus additional information that can be used by a relying party to verify the Acceptor's signature. These fields are described in detail in the following two subsections.

1.1.1.6. agreedCertInfo

The agreedCertInfo field is used to describe the proxy certificates that an Acceptor is willing to accept. It consists of these subfields:

- ignoredExtensions: a list of OIDs. The presence of an OID in this list is an indication that the presence, absence, or value of an extension with this OID in a certificate will not affect the Acceptor's willingness to accept the certificate.

- certSubsetHash: a hash of a TBSCertificate structure representing a certificate that the Acceptor is willing to accept.

  When verifying this extension, the relying party should construct a TBSCertificate structure identical to the current certificate's tbsCertificate field, minus the DelegationTrace extension and any extensions listed in ignoredExtensions; the hash of that structure should be equal to certSubsetHash.

1.1.1.7. x509AcceptorInfo

The x509AcceptorInfo field consists of a signature, using the private key associated with the Acceptor's certificate, of the agreedCertInfo field, plus additional information that the relying party may use to identify the Acceptor.

Note that the Acceptor's certificate is not the newly-issued proxy certificate; rather, it is an X.509 certificate already held by the Acceptor at the time of delegation. If the issuer and Acceptor are the same entity, then the Acceptor's certificate SHOULD be the Issuer's certificate. If the Acceptor sent a certificate request to the issuer over a channel that was authenticated using an X.509 certificate, then the Acceptor's certificate SHOULD be the certificate that the Acceptor used to authenticate to the issuer.

The x509AcceptorInfo field consists of these subfields:

- acceptorSig is a signature, using the private key associated with the Acceptor's certificate, of the agreedCertInfo field.

- acceptorName is the subject name from the Acceptor's certificate.

- acceptorAltName is the subjectAltName from the Acceptor's certificate. If acceptorName is null, this field MUST be present and non-null.

- acceptorCertHash is a copy of the signature from the Acceptor's certificate: acceptorHash.hashAlgorithm and acceptorHash.hashValue must be identical to the signatureAlgorithm and signatureValue from the Acceptor's certificate.

7 Certificate Path Validation

[TBD: Consider changing this section to add a second phase to path validation for PC validation, rather than modifying the existing path validation to accommodate the entire chain.]

The Certificate Path Validation algorithm described in Section 6 of draft-ietf-pkix-new-part1-08 [7] must be modified to accommodate Proxy Certificates. Changes are needed to:

1) check the generalized signing chains involving CAs, End Entity Certificates, and Proxy Certificates;

2) handle the use of subjectAltName and issuerAltName in the certificate path;

3) handle the iCPathLenConstraint in the proxyCertInfo extension.

4) check the key usage and extended key usage extensions.

5) handle the issuerCertHash in the proxyCertInfo extension.

Changes to section 6.1.2, Initialization:

(j) This step defines the working_issuer_name to be a distinguished name. However, because a PC uses the issuerAltName, the working_issuer_name variable needs to be generalized to accommodate not just a distinguished name, but any of the valid issuerAltName/subjectAltName types.

(new) working_certificate_type: This can be one of CA, EEC, or PC. A certificate type of CA is determined by the basicConstraints extension or as verified out-of-band. A certificate type of PC is determined by the proxyCertInfo extension. Otherwise, the certificate type is EEC.

(new) valid_pc_key_usage & pc_key_usage_criticality: These are used to verify that the key usage of a PC is a subset of the key usage of the certificate that signed that PC, and that the criticality of this extension never diminishes. These variables are not initialized or used until the first EEC or PC is encountered in the path validation algorithm with this extension.

(new) valid_pc_ext_key_usage & pc_ext_key_usage_criticality: These are used to verify that the extended key usage OIDs of a PC is a subset of the extended key usage OIDs of the certificate that signed that PC, and that the criticality of this extension never diminishes. These variables are not initialized or used until the first EEC or PC is encountered in the path validation algorithm with this extension.

(new) working_issuer_hash_algorithm & working_issuer_hash_value: These are used to verify that, if certificate N+1 is a Proxy Certificate, then certificate N is the certificate that issued that proxy. These variables are not used until the first EEC or PC is encountered in the path validation algorithm with the proxyCertInfo extension.

Changes to section 6.1.3, Basic Certificate Processing:

(a)(4) The comparison of the certificate issuer name with the working_issuer_name must be generalized to support comparison between any of the valid issuerAltName types.

(a)(new) The certificate type is CA and the working_certificate_type is CA, or the certificate type is EEC and the working_certificate_type is CA, or the certificate type is PC and the working_certificate_type is EEC or PC.

(b) & (c) This step checks the Name Constraints defined by the CA. However, since a PC does not define a new Name, these checks should be skipped if the certificate type is PC (as specified in a proxyCertInfo extension).

(new) If certificate type is PC, and valid_pc_key_usage has been initialized, then verify that:

(1) all bits that are asserted in the keyUsage extension of the certificate are also asserted in the valid_pc_key_usage;

(2) if pc_key_usage_criticality is true, then the keyUsage extension is critical

(new) If certificate type is PC, and valid_pc_ext_key_usage has been initialized, then verify that:

(1) all OIDs that are in the extKeyUsage extension in the certificate are also in the valid_pc_ext_key_usage;

(2) if pc_ext_key_usage_criticality is true, then the extKeyUsage extension is critical.

(new) If certificate type is PC, then verify that:

(1) proxyCertInfo.issuerCertHash is present.

(2) proxyCertInfo.issuerCertHash.hashAlgorithm is equal to working_issuer_hash_algorithm.

(3) proxyCertInfo.issuerCertHash.hashValue is equal to working_issuer_hash_value.

Changes to section 6.1.4, Preparation for Certificate i+1:

(c) Adjust this to assign the subjectAltName to working_issuer_name, if the subject is empty. This is done to accommodate the use of subjectAltName and issuerAltName by PCs.

(k) This step verifies that the certificate is a CA certificate. However, it is not general enough to support a PC. So change this step to simply assign the certificate type to the working_certificate_type. The necessary CA, EEC, and PC signing constraints check has been added to the Basic Certificate Processing section above.

(m) This step resets the max_path_length if pathLenConstraint is present in the certificate. This needs to be generalized to support pCPathLengthConstraint from the proxyCertInfo extension, as follows:

Reset max_path_length as follows:

(1) If certificate type is CA, and pathLenConstraint is present in the certificate and is less than max_path_length, then set max_path_length to the value of pathLenConstraint.

(2) If certificate type is EEC, and pCPathLenConstraint is not present in the certificate, then set max_path_length to n.

(3) If certificate type is EEC, and pCPathLenConstraint is present in the certificate, then set max_path_length to the value of pCPathLenConstraint.

(4) If certificate type is PC, and pCPathLenConstraint is present in the certificate and less than max_path_length, then set max_path_length to the value of pCPathLenConstraint.

(n) Since keyCertSign is currently defined to be equivalent to being a CA, this check needs to be changed to accommodate PCs, as follows: If certificate type is CA, and a key usage extension is present and marked critical, verify that the keyCertSign bit is set.

(new) If certificate type is EEC or PC, and the key usage extension is present, then set valid_pc_key_usage to keyUsage, and set pc_key_usage_criticality to the keyUsage criticality.

(new) If certificate type is EEC or PC, and the extended key usage extension is present, then set valid_pc_ext_key_usage to extKeyUsage, and set pc_ext_key_usage_criticality to the extKeyUsage criticality.

(new) Assign the certificate signatureAlgorithm to working_issuer_hash_algorithm, and assign the certificate signatureValue to working_issuer_hash_value.

At this point we have no plans for a PA (that is, an EEC or PC) to revoke the PCs that it has issued. If this feature is needed in the future, the CRL Distribution Point extension can be used in the PA certificates to locate a CRL.

8 Relationship to Attribute Certificates

An Attribute Certificate [4] can be used to grant to one identity, the holder, some attribute such as a role, clearance level, or alternative identity such as "charging identity" or "audit identity". This is accomplished by way of a trusted Attribute Authority (AA), which issues signed Attribute Certificates (AC), each of which binds an identity to a particular set of attributes. Authorization decisions can then be made by combining information from the authenticated End Entity Certificate providing the identity, with the signed Attribute Certificates providing binding of that identity to attributes.

There is clearly some overlap between the capabilities provided by Proxy Certificates and Attribute Certificates. However, the combination of the two approaches together provides a broader spectrum of solutions to authorization in X.509 based systems, than either solution alone. This section seeks to clarify some of the overlaps, differences, and synergies between Proxy Certificate and Attribute Certificates.

8.1 Types of Attribute Authorities

For the purposes of this discussion, Attribute Authorities, and the uses of the Attribute Certificates that they produce, can be broken down into two broad classes:

1) *End entity AA*: An End Entity Certificate may be used to sign an AC. This can be used, for example, to allow an end entity to delegate some of its privileges to another entity.

2) *Third party AA*: A separate entity, aside from the end entity involved in an authenticated interaction, may sign ACs in order to bind the authenticated identity with additional attributes, such as role, group, etc. For example, when a client authenticates with a server, the third party AA may provide an AC that binds the client identity to a particular group, which the server then uses for authorization purposes.

This second type of Attribute Authority, the third party AA, works equally well with an EEC or a PC. For example, Proxy Certificates can be used to delegate the EEC's identity to various other parties. Then when one of those other parties uses the PC to authenticate with a service, that service will receive the EEC's identity via the PC, and can apply any ACs that bind that identity to attributes in order to determine authorization rights. There would appear to be great synergies between the use of Proxy Certificates and Attribute Certificates produced by third party Attribute Authorities.

However, the uses of Attribute Certificates that are granted by the first type of Attribute Authority, the end entity AA, overlap considerably with the uses of Proxy Certificates as described in the previous sections. Such Attribute Certificates are generally used for delegation of rights from one end entity to others, which clearly overlaps with the stated purpose of Proxy Certificates, namely single sign-on and delegation.

8.2 Delegation Using Attribute Certificates

In the motivating example above, PCs are used to delegate Steve's identity to the various other jobs and agents that need to act on Steve's behalf. This allows those other entities to authenticate as if they were Steve, for example to the mass storage system.

A solution to this example could also be cast using Attribute Certificates that are signed by Steve's EEC, which grant to the other entities in this example the right to perform various operations on Steve's behalf. In this example, the starter program, the agent, the simulation jobs, and the post-processing job would each have their own EECs. Steve's EEC would therefore issue ACs to bind each of those other EEC identities to attributes that grant the necessary privileges allow them to, for example, access the mass storage system.

However, this AC based solution to delegation has some disadvantages as compared to the PC based solution:

- All protocols, authentication code, and identity based authorization services must be modified to understand ACs. With the PC solution, protocols (e.g. TLS) likely need no modification, authentication code needs minimal modification (e.g. to perform PC aware path validation), and identity based authorization services need no modification.

- ACs need to be created by Steve's EEC, which bind attributes to each of the other identities involved in the distributed application (i.e. the agent, simulation jobs, and post-processing job). This implies that Steve must know in advance which other identities may be involved in this distributed application, in order to generate the appropriate ACs which are signed by Steve's ECC. On the other hand, the PC solution allows for much more flexibility, since parties can further delegate a PC without a priori knowledge by the originating EEC.

There are many unexplored tradeoffs and implications in this discussion of delegation. However, reasonable arguments can be made in favor of either an AC based solution to delegation or a PC based solution to delegation. The choice of which approach should be taken in a given instance may depend on factors such as the software that it needs to be integrated into, the type of delegation required, and religion.

8.3 Propagation of Authorization Information

One possible use of Proxy Certificates is to carry authorization information associated with a particular identity.

The merits of placing authorization information into End Entity Certificates (also called a Public Key Certificate or PKC) have been widely debated. For example, Section 1 of "An Internet Attribute Certificate Profile for Authorization" states:

> "Authorization information may be placed in a PKC extension or placed in a separate attribute certificate (AC). The placement of authorization information in PKCs is usually undesirable for two reasons. First, authorization information often does not have the same lifetime as the binding of the identity and the public key. When authorization information is placed in a PKC extension, the general result is the shortening of the PKC useful lifetime. Second, the PKC issuer is not usually authoritative for the authorization information. This results in additional steps for the PKC issuer to obtain authorization information from the authoritative source.
>
> For these reasons, it is often better to separate authorization information from the PKC. Yet, authorization information also needs to be bound to an identity. An AC provides this binding; it is simply a digitally signed (or certified) identity and set of attributes." ([4], Section 1)

Placing authorization information in a PC mitigates the first undesirable property cited above. Since a PC has a lifetime that is mostly independent of (always shorter than) its signing EEC, a PC becomes a viable approach for carrying authorization information.

The second undesirable property cited above is true. If a third party AA is authoritative, then using ACs issued by that third party AA is a natural approach to disseminating authorization information. However, this is true whether the identity being bound by these ACs comes from an EEC (PKC), or from a PC.

There is one case, however, that the above text does not consider. When performing delegation, it is usually the EEC itself that is authoritative (not the EEC issuer, or any third party AA). That

is, it is up to the EEC to decide what authorization rights it is willing to grant to another party. In this situation, including such authorization information into PCs that are generated by the EEC seems a reasonable approach to disseminating such information.

8.4 Proxy Certificate as Attribute Certificate Holder

In a system that employs both PCs and ACs, one can imagine the utility of allowing a PC to be the holder of an AC. This would allow for a particular delegated instance of an identity to be given an attribute, rather than all delegated instances of that identity being given the attribute.

However, the issue of how to specify a PC as the holder of an AC remains open.

9 Commentary

This section provides commentary on various design choices, open issues, related work, and future directions for Proxy Certificates.

9.1 keyCertSign Bit in the Key Usage Basic Extension

This Proxy Certificate profile does not change the definition of the keyCertSign bit of the keyUsage extension. draft-ietf-pkix-new-part1-08 states:

> "The keyCertSign bit is asserted when the subject public key is used for verifying a signature on public key certificates. If the keyCertSign bit is asserted, then the cA bit in the basic constraints extension (section 4.2.1.10) MUST also be asserted."

Nor does this Proxy Certificate profile contradict this keyCertSign definition, since a Proxy Certificate is not an end entity public key certificates, as discussed in section 2 above.

9.2 nonRepudiate Bit in the Key Usage Basic Extension

One alternative for the nonRepudiate bit is that it MUST NOT be asserted. It seems, on the surface, and impersonation and non-repudiation are at odds with one another. However, this decision is postponed until further discussion with others who are more familiar with the use of this bit.

9.3 Subject Name of a Proxy Certificate

The subject name of a PC is only used for path validation. This PC profile uses a randomly generated subjectAltName to provide a (statistically) unique subject name for the PC.

Another possibility for naming the PC is to use a subject field that is derived from the subject of the PA. In fact, this is the approach taken in the current Grid Security Infrastructure implementation.

For example, the PC subject field could be the EEC subject field, extended with the addition of a new AttributeType and Value component of proxyLevel:nnnn where proxyLevel is a new AttributeType, and nnnn is the depth of the PC signing path. The issuer field would contain the subject field of the PA that signed the PC. In this scheme the path validation process would check that the subject and issuer names match up the chain and the proxyLevel values increase by one at each subsequent delegation.

One advantage of this approach is that some current implementations of path validation, such as OpenSSL-0.9.6, do not support the use of subjectAltName and issuerAltName. Thus for practical purposes it is arguably better to use the subject name and the proxyLevel:nnnn scheme.

A disadvantage of this approach is that it is reliant on the DN convention used by the subject field. This limits Proxy Certificates such that they can only be used for EECs that use the subject field. If an EEC instead uses subjectAltName, with a null subject field, then this approach does not work. For this reason, this approach was rejected for this Proxy Certificate profile.

9.4 Carrying Along the End Entity Subject

Another suggestion was to include the subject of the signing EEC as an informational field in the PC. This would allow an authorizing process to use only information in the final PC in the chain to determine identity, and not need to walk the chain in order to find out the subject (or subjectAltName) of the EEC that the PC is derived from.

This approach was rejected for the following reasons:

- It would be easy to spoof this informational field. For example, a PC with an informational subject of "Steve" could be used to create a PC with an informational subject set to "Doug". This leaves us with two alternatives:
    - We can augment the path validation to check that this informational field of the PC is the same as in the signing PC or EEC. But this is not desirable, as it complicates the path validation.
    - But if we do not validate this field, we cannot trust the contents of this informational field. So then there is no point in including this informational field.
- Upon closer examination, there is a lot of information in the certificate chain that may be needed during authorization, such as the number of levels of delegation, the CA (or multiple levels of CAs) who signed the original EEC, the constraints and keyUsage values of the signing EEC, possibly Certificate Policies associated with CAs or IAs. All of these require essentially the same amount of work as retrieving the subject of the EEC that signed the PC. So why threat the EEC subject specially by including it in an information field?

In the end, just including the EEC subject name does not seem to be sufficiently useful to justify the addition of another field and the work of verifying that name during the path validation.

Therefore, to determine the identity of a PC for authorization purposes, the subject of the EEC must be retrieved directly from the EEC in the signing chain. This approach also has the beneficial side effect of further stressing that a Proxy Certificate has no identity of its own, but rather inherits it from its signing EEC.

9.5 Specifying Proxy Restrictions

The proxyRestriction field in the proxyCertInfo extension does not define a policy language to be used for proxy restrictions; rather, it places the burden on those parties using that extension to define an appropriate language, and to acquire an OID for that language (or to select an appropriate previously-defined language/OID). Because it is essential for the PA that issues a certificate with a proxyRestriction field and the relying party that interprets that field to agree on its meaning, the policy language OID must correspond to a policy language, not just a policy grammar.

Several different approaches were considered regarding how to limit the use of a PC for specific authorization purposes. One of these approaches was to include a list the specific rights granted by the PC (perhaps along with conditions associated with those rights), either as a separate extension or as part of proxyCertInfo. This list of rights would define the subset of the issuer's rights to be granted to the PC holder. But the parties using that extension would still be responsible for ensuring that both the PA and relying party agreed on the meanings of the access rights and conditions appearing in the restriction.

Another possible approach is to embed an Attribute Certificate (signed by the EEC issuing the PC) within a PC, which would define a subset of the issuer's attributes to be associated with the PC holder.

9.6 Proxy Restrictions vs. Proxy Rights

The proxyRestriction field in the proxyCertInfo extension defines restrictions on the use of the proxy certificate; if that field is not present, the proxy is unrestricted.

Another approach would be to require that each proxy certificate explicitly list the rights that it grants.

9.7 Site Information in Delegation Tracing

In some cases, it may be desirable to know the hosts involved in a delegation transaction (for example, a relying party may wish to reject proxy certificates that were created on a specific host or domain). The DelegationTrace extension could be modified to include the PA's and Acceptor's IP addresses; however, IP addresses are typically easy to spoof, and in some cases the two parties to a transaction may not agree on the IP addresses being used (e.g., if the Acceptor is on a host that uses NAT, the Acceptor and the PA may disagree about the Acceptor's IP address).

Another suggestion was, in those cases where domain information is needed, to require that the subject names of all End Entities involved (the Acceptor(s) and the End Entity that appears in a PC's certificate path) include domain information.

9.8 Delegation Tracing vs. Usage Tracing

Delegation tracing provides information about whom a certificate was delegated to, but it does not provide any information about who actually used the certificate. That is, if Entity A delegates a certificate to Entity B, and then Entity C somehow acquires the certificate and private key and delegates to Entity D, and so on:

A delegates PC1 to B
C delegates PC2 to D
E delegates PC3 to F
G uses PC3

In this diagram, A has used A's identity certificate to create proxy certificate PC1 and delegate it to B. C has (somehow) acquired PC1 and its private key, and used it to sign PC2 and delegate PC2 to D. E has acquired PC2 and its private key, and used it to sign PC3 and delegate PC3 to F. Finally, G has acquired a copy of PC3 and its private key, and used it to authenticate to some relying party.

If the relying party wishes to audit who has been involved in the use of this certificate, it can determine A's identity (by using the certificate chain), and G's identity (by requirint that anyone using a proxy certificate also present an identity certificate).

If each proxy certificate includes a DelegationTracing extension, the relying party has the identities B, D, and F available to it -- but it has no indication that C or E were involved. Another approach towards auditing the usage of a certificate would be to provide a usage tracing extension that would include the issuer's signature of the certificate (using the issuer's identity certificate); this would make the identities C and E (but not B, D, or F) available to the relying party.

9.9 Contents of X509AcceptorInfo

The X509AcceptorInfo field contains a signature using the Acceptor's private key, plus some additional information that a relying party can use to identify the Acceptor's certificate. There have been various suggestions about how much additional information should be included in this field, ranging from simply including the Acceptor's subject name (or subjectAltName) to including all certificates used by the issuer when doing path validation on the Acceptor's certificate.

Currently, the X509AcceptorInfo field contains the Acceptor's name (or subjectAltName) and the signature from the Acceptor's certificate. This is enough information to uniquely identify a certificate, but in itself does not necessarily convey any meaningful information about the Acceptor's identity (especially if the Acceptor certificate is itself a Proxy certificate). Another

approach would be to include the sequence of names from a valid certificate path for the Acceptor's certificate.

9.10 Certificate Policies Extension

One could imagine some interesting things to do with the Certificate Policies extension. For example:

- One could define policies for creation of a Proxy Certificate. For example, was the PC created locally or remotely?

- An alternate approach to defining restricted Proxy Certificates would be use the Certificate Policies extension to carry the OIDs of various Proxy Certificate Policies. For example, a Proxy Certificate policy might state that the PC can only be used within a limited scope of machines, or for a limited set of uses.

9.11 Kerberos 5 Tickets

The Kerberos Network Authentication Protocol (RFC 1510 [9]) is a widely used authentication system based on conventional (shared secret key) cryptography. It provides support for single sign-on via creation of "Ticket Granting Tickets" or "TGT", and support for delegation of impersonation rights via "forwardable tickets".

Kerberos 5 tickets have informed many of the ideas surrounding X.509 Proxy Certificates. For example, the local creation of a short-lived PC can be used to provide single sign-on in an X.509 PKI based system, just as creation of short-lived TGT allows for single sign-on in a Kerberos based system. And just as a TGT can be forwarded (i.e. delegated) to another entity to allow for impersonation in a Kerberos based system, so can a PC can be delegated to allow for impersonation in an X.509 PKI based system.

A major difference between a Kerberos TGT and an X.509 PC is that while creation and delegation of a TGT requires the involvement of a third party (the Kerberos Domain Controller), a PC can be unilaterally created without the active involvement of a third party. That is, a user can directly create a PC from an EEC for single sign-on capability, without requiring communication with a third party. And an entity with a PC can delegate the PC to another entity (i.e. by creating a new PC, signed by the first) without requiring communication with a third party.

The method used by Kerberos implementations to protect a TGT can also be used to protect the private key of a PC. For example, some Unix implementations of Kerberos use standard Unix file system security to protect a user's TGT from compromise. Similarly, the Globus Toolkit's Grid Security Infrastructure implementation of Proxy Certificates protects a user's PC private key using this same approach.

Looking at developments with Kerberos 5 tickets also can inform us about potential future directions for Proxy Certificates. For example:

- Kerberos tickets have two simple mechanisms for allowing their use to be restricted: a time period during which the ticket is valid (the "starttime" and "endtime" fields of a ticket), and a host address which restricts the host on which the ticket may be used (the "caddr" field of a ticket). An X.509 PC also has a validity period, but does not have a host restriction field, though it could be easily added via an X.509 extension. While these particular restrictions have a variety of limitations and problems, they points toward a future of more general restriction policies that might be included in a PC and/or Kerberos 5 ticket.

- The Microsoft implementation of Kerberos 5 has (not without controversy) used the "authorization-data" field in the Kerberos ticket to encode authorization information into the ticket. A similar approach could be taken with X.509 Proxy Certificates, by encoding the authorization information into an X.509 extension in a PC. This approach allows for a user's normal, long-lived identity certificate to be used to create a short-lived authorization certificate that can be delegated as necessary. Merits of this approach versus Attribute Certificates are discussed in Section 5.

9.12 Examples of usage of Proxy Groups and Restrictions

This section gives some examples of Proxy Certificate usage and some examples of how Proxy Restrictions and Proxy Groups can be used to restrict Proxy Certificates.

9.12.1 Example One: Use of proxies without Groups or Restrictions

Steve wishes to perform a third-party FTP transfer between two FTP servers. Steve would use an existing PC to authenticate to both servers and delegate a PC to both hosts. When the servers establish the data channel connection to each other, they use these delegated credentials to perform self-authentication and secure the channel.

9.12.2 Example Two: Use of proxies with Groups

Steve wants to again perform a third-party FTP transfer and he wants to use Proxy Groups to provide extra security. As in the previous example, Steve would use his existing PC to authenticate to both servers. However when he delegates PCs to the servers he would assign both PCs to the same, detached subgroup. The servers use these delegated credentials to authenticate each other over the data channel, each verifying the other’s PC is in a compatible group.

The proxy groups in the above example provide two forms of protection. First since each server verifies the Proxy Group of the other server, they have assurance they are interacting with another task that Steve has intended them to interact with. Second it provides a limited form of restriction in case one of the delegated PCs is stolen.

9.12.3 Example Three: Use of proxies with Groups and Restrictions

Steve wishes to delegate to a process the right to perform a third-party transfer of a file on his behalf. Steve would delegate a PC to the process and he would use Proxy Restrictions to limit the delegated PC to two rights – the right to read file F1 on host H1 and the right to write file F2 on host H2.

The process then uses this restricted PC to authenticate to servers H1 and H2. The process would also delegate a PC to both servers, placing both PCs in the same detached subgroup. Note that these delegated PCs would inherit the restrictions of their parents, though this is not relevant to this example.

Now when the process issues the command to transfer the file F1 on H1 and to F2 on H2, these two servers perform an authorization check, in addition to any local policy they have, based on the restrictions in the PC that the process used to authenticate with them. Namely H1 checks that the PC gives the user the right to read F1 and H2 checks that the PC gives the user the right to write F2.

The extra security provided by these restrictions is that now if the PC delegated to the process by Steve is stolen, its use is greatly limited.

The servers would then check the proxy groups when setting up and authenticating each over the data channel as explained in Example Two.

10 Security Considerations

A Proxy Certificate is generally less secure than the EEC that issued it. This is due to the fact that the private key of a PC is generally not protected as rigorously as that of the EEC. For example, the private key of a PC is often protected using only file system security, in order to allow that PC to be used for single sign-on purposes. This makes the PC more susceptible to compromise.

However, the risk of a compromised PC is only the misuse of a single user's privileges. Due to the path validation checks made on a PC, a PC cannot be used to sign an EEC or PC for another user.

Further, a compromised PC can only be misused for the lifetime of the PC. Therefore, one common way to limit the misuse of a compromised PC is to limit their validity periods to no longer than is needed.

In addition, if a PC is compromised, it does NOT compromise the EEC that created the PC. This property is of great utility in protecting the highly valuable, and hard to replace, public key of the EEC. In other words, the use of Proxy Certificates to provide single sign-on capabilities in an X.509 PKI environment can actually increase the security of the end entity certificates, because creation and use of the PCs for user authentication limits the exposure of the EEC private key to only the creation of the first level PC.

The pCPathLenConstraint field of the proxyCertInfo extension can be used by an EEC to limit subsequent delegation of the PC. A service may choose to only authorize a request if a valid PC can be delegated to it. An example of such as service is a job starter, which may choose to reject a job start request if a valid PC cannot be delegated to it. By limiting the pCPathLenConstraint, an EEC can ensure that a compromised PC of one job cannot be used to start additional jobs elsewhere.

An EEC or PC can limit what a new PC can be used for by turning off bits in the Key Usage and Extended Key Usage extensions. However, once a key usage or extended key usage has been removed, the path validation algorithm ensures that it cannot added back in a subsequent PC. In other words, key usage can only be decreased in PC chains.

The EEC could use the CRL Distribution Points extension and/or OCSP to take on the responsibility of revoking PCs that it had issued, if it felt that they were being misused.

The relying party that is going to authorize some actions on the basis of a PC will be aware that it has been presented with a PC, and can determine the depth of the delegation and the time that the delegation took place and any entities through which the PC was delegated (if the optional DelegationTrace extension is included in the PCs in the cert chain). It may want to use this information in addition to the information from the signing EEC. Thus a highly secure resource might refuse to accept a PC at all, or maybe only a single level of delegation, or maybe only a PC that has not been delegated through a untrusted host, etc.

11 References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.

[2] Butler, R., D. Engert, I. Foster, C. Kesselman, and S. Tuecke, "A National-Scale Authentication Infrastructure," *IEEE Computer*, vol. 33, pp. 60-66, 2000.

[3] Dierks, T. and C. Allen, "The TLS Protocol, Version 1.0," RFC 2246, January 1999.

[4] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization," Internet Draft draft-ietf-pkix-ac509prof-06.txt, January 2001.

[5] Foster, I., C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," presented at Proceedings of the 5th ACM Conference on Computer and Communications Security, 1998.

[6] Foster, I., C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, 2001.

[7] Housley, R., W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," Internet Draft draft-ietf-pkik-new-part1-08.txt (update to RFC 2459), July 2001.

[8] Jackson, K., S. Tuecke, and D. Engert, "TLS Delegation Protocol," Internet Draft draft-ietf-tls-delegation-00.txt, 2001.

[9] Kohl, J. and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993.

12 Acknowledgments

We are grateful to numerous colleagues for discussions on the topics covered in this paper, in particular (in alphabetical order, with apologies to anybody we've missed): Joe Bester, Randy Butler, Keith Jackson, Stephen Kent, Bill Johnston, Marty Humphrey, Sam Meder, Clifford Neuman, Gene Tsudik.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38 and DE-AC03-76SF0098; by the Defense Advanced Research Projects Agency under contract N66001-96-C-8523; by the National Science Foundation; and by the NASA Information Power Grid project.

13 Change Log

draft-ietf-pkix-impersonation-00 (February 2001)

Initial submission.

draft-ietf-pkix-proxy-00 (July 2001)

Renamed to "Proxy Certificate", from "Impersonation Certificate", due to overwhelming feedback from IETF and GGF.

Added proxyRestriction field to ProxyCertInfo extension.

Added delegationTrace field to ProxyCertInfo extension.

Updated to agree with draft-ietf-pkix-part1-08.

draft-ietf-pkix-proxy-01 (August 2001)

Changes related to delegation tracing: removed delegationTrace field from ProxyCertInfo extension, created DelegationTrace extension, added and modified commentary sections related to delegation tracing.

Added issuerCertHash to proxyCertInfo extension and to the path validation section.

draft-ietf-pkix-proxy-02 (February 2002)

Added concept of proxy group.

Updated section on keyCertSign bit to reflect draft-pkix-new-part1-07.

14 Contact Information

Steven Tuecke
Distributed Systems Laboratory
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
Phone: 630-252-8711
Email: tuecke@mcs.anl.gov

Doug Engert
Argonne National Laboratory
Email: deengert@anl.gov

Ian Foster
Argonne National Laboratory & University of Chicago
Email: foster@mcs.anl.gov

Von Welch
University of Chicago
Email: welch@mcs.anl.gov

Mary Thompson
Lawrence Berkeley National Laboratory
Email: mrthompson@lbl.gov

Laura Pearlman
University of Southern California, Information Sciences Institute
Email: laura@isi.edu

Carl Kesselman
University of Southern California, Information Sciences Institute
Email: carl@isi.edu