

Network Working Group  
Request for Comments: 3820  
Category: Standards Track

S. Tuecke  
ANL  
V. Welch  
NCSA  
D. Engert  
ANL  
L. Pearlman  
USC/ISI  
M. Thompson  
LBNL  
June 2004

Internet X.509 Public Key Infrastructure (PKI)  
Proxy Certificate Profile

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2004).

Abstract

This document forms a certificate profile for Proxy Certificates, based on X.509 Public Key Infrastructure (PKI) certificates as defined in RFC 3280, 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 proxying and delegation within a PKI based authentication system.

## Table of Contents

1.	Introduction . . . . .	3
2.	Overview of Approach . . . . .	4
2.1.	Terminology. . . . .	4
2.2.	Background . . . . .	5
2.3.	Motivation for Proxying. . . . .	5
2.4.	Motivation for Restricted Proxies. . . . .	7
2.5.	Motivation for Unique Proxy Name . . . . .	8
2.6.	Description Of Approach. . . . .	9
2.7.	Features Of This Approach. . . . .	10
3.	Certificate and Certificate Extensions Profile . . . . .	12
3.1.	Issuer . . . . .	12
3.2.	Issuer Alternative Name. . . . .	12
3.3.	Serial Number. . . . .	12
3.4.	Subject. . . . .	13
3.5.	Subject Alternative Name . . . . .	13
3.6.	Key Usage and Extended Key Usage . . . . .	13
3.7.	Basic Constraints. . . . .	14
3.8.	The ProxyCertInfo Extension. . . . .	14
4.	Proxy Certificate Path Validation. . . . .	17
4.1.	Basic Proxy Certificate Path Validation. . . . .	19
4.2.	Using the Path Validation Algorithm. . . . .	23
5.	Commentary . . . . .	24
5.1.	Relationship to Attribute Certificates . . . . .	24
5.2.	Kerberos 5 Tickets . . . . .	28
5.3.	Examples of usage of Proxy Restrictions. . . . .	28
5.4.	Delegation Tracing . . . . .	29
6.	Security Considerations. . . . .	30
6.1.	Compromise of a Proxy Certificate. . . . .	30
6.2.	Restricting Proxy Certificates . . . . .	31
6.3.	Relying Party Trust of Proxy Certificates. . . . .	31
6.4.	Protecting Against Denial of Service with Key Generation	32
6.5.	Use of Proxy Certificates in a Central Repository. . . . .	32
7.	IANA Considerations. . . . .	33
8.	References . . . . .	33
8.1.	Normative References . . . . .	33
8.2.	Informative References . . . . .	33
9.	Acknowledgments. . . . .	34
	Appendix A. 1988 ASN.1 Module. . . . .	35
	Authors' Addresses . . . . .	36
	Full Copyright Notice. . . . .	37

## 1. Introduction

Use of a proxy credential [i7] is a common technique used in security systems to allow entity A to grant to another entity B the right for B to be authorized with others as if it were A. In other words, entity B is acting as a proxy on behalf of entity A. This document forms a certificate profile for Proxy Certificates, based on the RFC 3280, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile" [n2].

In addition to simple, unrestricted proxying, this profile defines:

- \* A framework for carrying policies in Proxy Certificates that allows proxying to be limited (perhaps completely disallowed) through either restrictions or enumeration of rights.
- \* Proxy Certificates with unique names, derived from the name of the end entity certificate name. This allows the Proxy Certificates to be used in conjunction with attribute assertion approaches such as Attribute Certificates [i3] and have their own rights independent of their issuer.

Section 2 provides a non-normative 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 Issuer, 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 proxying 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. A new certificate extension, Proxy Certificate Information, is introduced.

Section 4 defines path validation rules for Proxy Certificates.

Section 5 provides non-normative commentary on Proxy Certificates.

Section 6 discusses security considerations relating to Proxy Certificates.

References, listed in Section 8, are sorted into normative and information references. Normative references, listed in Section 8.1, are in the form [nXX]. Informative references, listed in Section 8.2, are in the form [iXX].

Section 9 contains acknowledgements.

Following Section 9, contains the Appendix, the contact information for the authors, the intellectual property information, and the copyright information for this document.

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 BCP 14, RFC 2119 [n1].

## 2. Overview of Approach

This section provides non-normative commentary on Proxy Certificates.

The goal of this specification is to develop a X.509 Proxy Certificate profile and to facilitate their use within Internet applications for those communities wishing to make use of restricted proxying and delegation within an X.509 Public Key Infrastructure (PKI) authentication based system.

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

### 2.1. Terminology

This document uses the following terms:

- \* CA: A "Certification Authority", as defined by X.509 [n2]
- \* 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.

- \* PI: A "Proxy Issuer" is an entity with an End Entity Certificate or Proxy Certificate that issues a Proxy Certificate. The Proxy Certificate is signed using the private key associated with the public key in the Proxy Issuer's certificate.
- \* AC: An "Attribute Certificate", as defined by "An Internet Attribute Certificate Profile for Authorization" [i3].
- \* AA: An "Attribute Authority", as defined in [i3].

## 2.2. Background

Computational and Data "Grids" have emerged as a common approach to constructing dynamic, inter-domain, distributed computing environments. As explained in [i5], 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](http://www.globus.org)) introduced the Grid Security Infrastructure (GSI) [i4]. 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 [i2], 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 [i1]. GSI has emerged as the dominant security solution used by Grid efforts worldwide.

This experience with GSI has proven the viability of restricted proxying as a basis for 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 proxying. 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.

## 2.3. Motivation for Proxying

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

Steve is an engineer who wants to use a reliable file transfer service to manage the movement of a number of large files around between various hosts on his company's Intranet-based Grid. From his laptop he wants to submit a number of transfer requests to the service and have the files transferred while he is doing other

things, including being offline. The transfer service may queue the requests for some time (e.g., until after hours or a period of low resource usage) before initiating the transfers. The transfer service will then, for each file, connect to each of the source and destination hosts, and instruct them to initiate a data connection directly from the source to the destination in order to transfer the file. Steve will leave an agent running on his laptop that will periodically check on progress of the transfer by contacting the transfer service. 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 the reliable file transfer service to submit the transfer request.
- \* Since the storage hosts know nothing about the file transfer service, the file transfer service needs to be delegated the rights to mutually authenticate with the various storage hosts involved directly in the file transfer, in order to initiate the file transfer.
- \* The source and destination hosts of a particular transfer must be able to mutual authenticate with each other, to ensure the file is being transferred to and from the proper parties.
- \* The agent running on Steve's laptop must mutually authenticate with the file transfer service in order to check the result of the transfers.

Proxying 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 submit all the file transfer requests to the transfer service, and then periodically check on the status of the transfer. 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 file transfer

service needs to be able to authenticate on Steve's behalf with the source and destination hosts, and must in turn delegate rights to those hosts so that they can authenticate with each other.

Proxying can be used to secure all of these interactions:

- \* Proxying allows for the private key stored on the smartcard to be accessed just once, in order to create the necessary proxy credential, which allows the client/agent program to be authorized as Steve when submitting the requests to the transfer service. Access to the smartcard and Steve's password is not required after the initial creation of the proxy credential.
- \* The client program on the laptop can delegate to the file transfer service the right to act on Steve's behalf. This, in turn, allows the service to authenticate to the storage hosts and inherit Steve's privileges in order to start the file transfers.
- \* When the transfer service authenticates to hosts to start the file transfer, the service can delegate to the hosts the right to act on Steve's behalf so that each pair of hosts involved in a file transfer can mutually authenticate to ensure the file is securely transferred.
- \* When the agent on the laptop reconnects to the file transfer service to check on the status of the transfer, it can perform mutual authentication. The laptop may use a newly generated proxy credential, which is just created anew using the smartcard.

This scenario, and others similar to it, is 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.

#### 2.4. Motivation for Restricted Proxies

One concern that arises is what happens if a machine that has been delegated the right to inherit Steve's privileges has been compromised? For example, in the above scenario, what if the machine running the file transfer service is compromised, such that the attacker can gain access to the credential that Steve delegated to that service? 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 proxy by means of policies on the proxy certificates. For example, the machine running the reliable file transfer service in

the above example might only be given Steve's right for the purpose of reading the source files and writing the destination files. Therefore, if that file transfer service is compromised, the attacker cannot modify source files, cannot create or modify other files to which Steve has access, cannot start jobs on behalf of Steve, etc. All that an attacker would be able to do is read the specific files to which the file transfer service has been delegated read access, and write bogus files in place of those that the file transfer service has been delegated write access. Further, by limiting the lifetime of the credential that is delegated to the file transfer service, the effects of a compromise can be further mitigated.

Other potential uses for restricted proxy credentials are discussed in [i7].

## 2.5. Motivation for Unique Proxy Name

The dynamic creation of entities (e.g., processes and services) is an essential part of Grid computing. These entities will require rights in order to securely perform their function. While it is possible to obtain rights solely through proxying as described in previous sections, this has limitations. For example what if an entity should have rights that are granted not just from the proxy issuer but from a third party as well? While it is possible in this case for the entity to obtain and hold two proxy certifications, in practice it is simpler for subsequent credentials to take the form of attribute certificates.

It is also desirable for these entities to have a unique identity so that they can be explicitly discussed in policy statements. For example, a user initiating a third-party FTP transfer could grant each FTP server a PC with a unique identity and inform each server of the identity of the other, then when the two servers connected they could authenticate themselves and know they are connected to the proper party.

In order for a party to have rights of it's own it requires a unique identity. Possible options for obtaining an unique identity are:

- 1) Obtain an identity from a traditional Certification Authority (CA).
- 2) Obtain a new identity independently - for example by using the generated public key and a self-signed certificate.
- 3) Derive the new identity from an existing identity.



In this document we describe an approach to option #3, because:

- \* It is reasonably light-weight, as it can be done without interacting with a third party. This is important when creating identities dynamically.
- \* As described in the previous section, a common use for PCs is for restricted proxying, so deriving their identity from the identity of the EEC makes this straightforward. Nonetheless there are circumstances where the creator does not wish to delegate all or any of its rights to a new entity. Since the name is unique, this is easily accomplished by #3 as well, by allowing the application of a policy to limit proxying.

## 2.6. Description Of Approach

This document defines an X.509 "Proxy Certificate" or "PC" as a means of providing for restricted proxying within an (extended) 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. This EEC or PC is referred to as the Proxy Issuer (PI).
- 2) It can sign only another PC. It cannot sign an EEC.
- 3) It has its own public and private key pair, distinct from any other EEC or PC.
- 4) It has an identity derived from the identity of the EEC that signed the PC. When a PC is used for authentication, it may inherit rights of the EEC that signed the PC, subject to the restrictions that are placed on that PC by the EEC.
- 5) Although its identity is derived from the EEC's identity, it is also unique. This allows this identity to be used for authorization as an independent identity from the identity of the issuing EEC, for example in conjunction with attribute assertions as defined in [i3].
- 6) It contains a new X.509 extension to identify it as a PC and to place policies on the use of 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.

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.

## 2.7. Features Of This Approach

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

- \* Ease of integration

- o 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. 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 for authorization purposes.
- o Many existing authorization systems use the X.509 subject name as the basis for access control. Proxy Certificates can be used with such authorization systems without modification, since such a PC inherits its name and rights from the EEC that signed it and the EEC name can be used in place of the PC name for authorization decisions.

- \* Ease of use

- o 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.
- o 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 with proxy rights, but continue to protect the EEC private key in the repository.

- \* Protection of private keys

- o 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.
- o 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

- o When creating a PC, the PI 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 descendants. Further, fine-grained policies can be carried by a PC to even further restrict the operations that can be performed using the PC. These restrictions permit the PI to limit damage that could be done by the bearer of the PC, either accidentally or maliciously.

- o 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.

### 3. 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.

All Proxy Certificates MUST include the Proxy Certificate Information (ProxyCertInfo) extension defined in this section and the extension MUST be critical.

#### 3.1. Issuer

The Proxy Issuer of a Proxy Certificate MUST be either an End Entity Certificate, or another Proxy Certificate.

The Proxy Issuer MUST NOT have an empty subject field.

The issuer field of a Proxy Certificate MUST contain the subject field of its Proxy Issuer.

If the Proxy Issuer certificate has the KeyUsage extension, the Digital Signature bit MUST be asserted.

#### 3.2. Issuer Alternative Name

The issuerAltName extension MUST NOT be present in a Proxy Certificate.

#### 3.3. Serial Number

The serial number of a Proxy Certificate (PC) SHOULD be unique amongst all Proxy Certificates issued by a particular Proxy Issuer. However, a Proxy Issuer MAY use an approach to assigning serial numbers that merely ensures a high probability of uniqueness.

For example, a Proxy Issuer MAY use a sequentially assigned integer or a UUID to assign a unique serial number to a PC it issues. Or a Proxy Issuer MAY use a SHA-1 hash of the PC public key to assign a serial number with a high probability of uniqueness.

### 3.4. Subject

The subject field of a Proxy Certificate MUST be the issuer field (that is the subject of the Proxy Issuer) appended with a single Common Name component.

The value of the Common Name SHOULD be unique to each Proxy Certificate bearer amongst all Proxy Certificates with the same issuer.

If a Proxy Issuer issues two proxy certificates to the same bearer, the Proxy Issuer MAY choose to use the same Common Name for both. Examples of this include Proxy Certificates for different uses (e.g., signing vs encryption) or the re-issuance of an expired Proxy Certificate.

The Proxy Issuer MAY use an approach to assigning Common Name values that merely ensures a high probability of uniqueness. This value MAY be the same value used for the serial number.

The result of this approach is that all subject names of Proxy Certificates are derived from the name of the issuing EEC (it will be the first part of the subject name appended with one or more CN components) and are unique to each bearer.

### 3.5. Subject Alternative Name

The subjectAltName extension MUST NOT be present in a Proxy Certificate.

### 3.6. Key Usage and Extended Key Usage

If the Proxy Issuer certificate has a Key Usage extension, the Digital Signature bit MUST be asserted.

This document places no constraints on the presence or contents of the key usage and extended key usage extension. However, section 4.2 explains what functions should be allowed a proxy certificate by a relying party.

### 3.7. Basic Constraints

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

### 3.8. The ProxyCertInfo Extension

A new extension, ProxyCertInfo, is defined in this subsection. Presence of the ProxyCertInfo extension indicates that a certificate is a Proxy Certificate and whether or not the issuer of the certificate has placed any restrictions on its use.

```
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
    dod(6) internet(1) security(5) mechanisms(5) pkix(7) }
```

```
id-pe OBJECT IDENTIFIER ::= { id-pkix 1 }
```

```
id-pe-proxyCertInfo OBJECT IDENTIFIER ::= { id-pe 14 }
```

```
ProxyCertInfo ::= SEQUENCE {
    pCPathLenConstraint  INTEGER (0..MAX) OPTIONAL,
    proxyPolicy           ProxyPolicy }
```

```
ProxyPolicy ::= SEQUENCE {
    policyLanguage      OBJECT IDENTIFIER,
    policy              OCTET STRING OPTIONAL }
```

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

If a certificate is not a Proxy Certificate, then the proxyCertInfo extension MUST be absent.

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

#### 3.8.1. pCPathLenConstraint

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

### 3.8.2. proxyPolicy

The proxyPolicy field specifies a policy on the use of this certificate for the purposes of authorization. Within the proxyPolicy, the policy field is an expression of policy, and the policyLanguage field indicates the language in which the policy is expressed.

The proxyPolicy 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 PI that issues a certificate with a proxyPolicy field and the relying party that interprets that field to agree on its meaning, the policy language OID must correspond to a policy language (including semantics), not just a policy grammar.

The policyLanguage field has two values of special importance, defined in Appendix A, that MUST be understood by all parties accepting Proxy Certificates:

- \* id-ppl-inheritAll indicates that this is an unrestricted proxy that inherits all rights from the issuing PI. An unrestricted proxy is a statement that the Proxy Issuer wishes to delegate all of its authority to the bearer (i.e., to anyone who has that proxy certificate and can prove possession of the associated private key). For purposes of authorization, this an unrestricted proxy effectively impersonates the issuing PI.
- \* id-ppl-independent indicates that this is an independent proxy that inherits no rights from the issuing PI. This PC MUST be treated as an independent identity by relying parties. The only rights this PC has are those granted explicitly to it.

For either of the policyLanguage values listed above, the policy field MUST NOT be present.

Other values for the policyLanguage field indicates that this is a restricted proxy certification and have some other policy limiting its ability to do proxying. In this case the policy field MAY be present and it MUST contain information expressing the policy. If the policy field is not present the policy MUST be implicit in the value of the policyLanguage field itself. Authors of additional policy languages are encouraged to publicly document their policy language and list it in the IANA registry (see Section 7).

Proxy policies 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. This document is agnostic to the policies that can be placed in the policy field.

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

- 1) The relying party **MUST** know how to interpret the proxy policy and the request is allowed under that policy.
- 2) If the Proxy Issuer is an EEC then the relying party's local policies **MUST** authorize the request for the entity named in the EEC.
- 3) If the Proxy Issuer is another PC, then one of the following **MUST** be true:
  - a. The relying party's local policies authorize the Proxy Issuer to perform the request.
  - b. The Proxy Issuer inherits the right to perform the request from its issuer by means of its proxy policy. This must be verified by verifying these three conditions on the Proxy Issuer in a recursive manner.

If these conditions are not met, the relying party **MUST** either deny authorization, or ignore the PC and the whole certificate chain including the EEC entirely when making its authorization decision (i.e., make the same decision that it would have made had the PC and its certificate chain never been presented).

The relying party **MAY** impose additional restrictions as to which proxy certificates it accepts. For example, a relying party **MAY** choose to reject all proxy certificates, or **MAY** choose to accept proxy certificates only for certain operations, etc.

Note that since a proxy certificate has a unique identity it **MAY** also have rights granted to it by means other than inheritance from its issuer via its proxy policy. The rights granted to the bearer of a PC are the union of the rights granted to the PC identity and the



inherited rights. The inherited rights consist of the intersection of the rights granted to the PI identity intersected with the proxy policy in the PC.

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 policy that it can be used only to read or write files A and C. Then a trusted attribute authority grants an Attribute Certificate granting the PC the right to read file D. This would make the rights of the PC equal to the union of the rights granted to the PC identity (right to read file D) with the intersection of the rights granted to Steve, the PI, (right to read files A and B) with the policy in the PC (can only read files A and C). This would mean the PC would have the following rights:

- \* Right to read file A: Steve has this right and he issued the PC and his policy grants this right to the PC.
- \* Right to read file D: This right is granted explicitly to the PC by a trusted authority.

The PC would NOT have the following rights:

- \* Right to read file B: Although Steve has this right, it is excluded by his policy on the PC.
- \* Right to read file C: Although Steve's policy grants this right, he does not have this right himself.

In many cases, the relying party will not have enough information to evaluate the above criteria at the time that the certificate path 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 policies, or reject any certificate that contains proxy policies (or that has a parent certificate that contains proxy policies).

#### 4. Proxy Certificate Path Validation

Proxy Certification path processing verifies the binding between the proxy certificate distinguished name and proxy certificate public key. The binding is limited by constraints which are specified in the certificates which comprise the path and inputs which are specified by the relying party.

This section describes an algorithm for validating proxy certification paths. Conforming implementations of this specification are not required to implement this algorithm, but MUST provide functionality equivalent to the external behavior resulting from this procedure. Any algorithm may be used by a particular implementation so long as it derives the correct result.

The algorithm presented in this section validates the proxy certificate with respect to the current date and time. A conformant implementation MAY also support validation with respect to some point in the past. Note that mechanisms are not available for validating a proxy certificate with respect to a time outside the certificate validity period.

Valid paths begin with the end entity certificate (EEC) that has already been validated by public key certificate validation procedures in RFC 3280 [n2]. The algorithm requires the public key of the EEC and the EEC's subject distinguished name.

To meet the goal of verifying the proxy certificate, the proxy certificate path validation process verifies, among other things, that a prospective certification path (a sequence of  $n$  certificates) satisfies the following conditions:

- (a) for all  $x$  in  $\{1, \dots, n-1\}$ , the subject of certificate  $x$  is the issuer of proxy certificate  $x+1$  and the subject distinguished name of certificate  $x+1$  is a legal subject distinguished name to have been issued by certificate  $x$ ;
- (b) certificate 1 is valid proxy certificate issued by the end entity certificate whose information is given as input to the proxy certificate path validation process;
- (c) certificate  $n$  is the proxy certificate to be validated;
- (d) for all  $x$  in  $\{1, \dots, n\}$ , the certificate was valid at the time in question; and
- (e) for all certificates in the path with a `pCPathLenConstraint` field, the number of certificates in the path following that certificate does not exceed the length specified in that field.

At this point there is no mechanism defined for revoking proxy certificates.

#### 4.1. Basic Proxy Certificate Path Validation

This section presents the algorithm in four basic steps to mirror the description of public key certificate path validation in RFC 3280: (1) initialization, (2) basic proxy certificate processing, (3) preparation for the next proxy certificate, and (4) wrap-up. Steps (1) and (4) are performed exactly once. Step (2) is performed for all proxy certificates in the path. Step (3) is performed for all proxy certificates in the path except the final proxy certificate.

Certificate path validation as described in RFC 3280 MUST have been done prior to using this algorithm to validate the end entity certificate. This algorithm then processes the proxy certificate chain using the end entity certificate information produced by RFC 3280 path validation.

##### 4.1.1. Inputs

This algorithm assumes the following inputs are provided to the path processing logic:

- (a) information about the entity certificate already verified using RFC 3280 path validation. This information includes:
  - (1) the end entity name,
  - (2) the `working_public_key` output from RFC 3280 path validation,
  - (3) the `working_public_key_algorithm` output from RFC 3280,
  - (4) and the `working_public_key_parameters` output from RFC 3280 path validation.
- (b) prospective proxy certificate path of length `n`.
- (c) `acceptable-pc-policy-language-set`: A set of proxy certificate policy languages understood by the policy evaluation code. The `acceptable-pc-policy-language-set` MAY contain the special value `id-ppl-anyLanguage` (as defined in Appendix A) if the path validation code should not check the proxy certificate policy languages (typically because the set of known policy languages is not known yet and will be checked later in the authorization process).
- (d) the current date and time.

#### 4.1.2. Initialization

This initialization phase establishes the following state variables based upon the inputs:

- (a) `working_public_key_algorithm`: the digital signature algorithm used to verify the signature of a proxy certificate. The `working_public_key_algorithm` is initialized from the input information provided from RFC 3280 path validation.
- (b) `working_public_key`: the public key used to verify the signature of a proxy certificate. The `working_public_key` is initialized from the input information provided from RFC 3280 path validation.
- (c) `working_public_key_parameters`: parameters associated with the current public key, that may be required to verify a signature (depending upon the algorithm). The `proxy_issuer_public_key_parameters` variable is initialized from the input information provided from RFC 3280 path validation.
- (d) `working_issuer_name`: the issuer distinguished name expected in the next proxy certificate in the chain. The `working_issuer_name` is initialized to the distinguished name in the end entity certificate validated by RFC 3280 path validation.
- (e) `max_path_length`: this integer is initialized to `n`, is decremented for each proxy certificate in the path. This value may also be reduced by the `pcPathLenConstraint` value of any proxy certificate in the chain.
- (f) `proxy_policy_list`: this list is empty to start and will be filled in with the key usage extensions, extended key usage extensions and proxy policies in the chain.

Upon completion of the initialization steps, perform the basic certificate processing steps specified in 4.1.3.

#### 4.1.3. Basic Proxy Certificate Processing

The basic path processing actions to be performed for proxy certificate `i` (for all `i` in `[1..n]`) are listed below.

- (a) Verify the basic certificate information. The certificate **MUST** satisfy each of the following:

- (1) The certificate was signed with the `working_public_key_algorithm` using the `working_public_key` and the `working_public_key_parameters`.
  - (2) The certificate validity period includes the current time.
  - (3) The certificate issuer name is the `working_issuer_name`.
  - (4) The certificate subject name is the `working_issuer_name` with a CN component appended.
- (b) The proxy certificate MUST have a `ProxyCertInfo` extension. Process the extension as follows:
- (1) If the `pCPathLenConstraint` field is present in the `ProxyCertInfo` field and the value it contains is less than `max_path_length`, set `max_path_length` to its value.
  - (2) If `acceptable-pc-policy-language-set` is not `id-ppl-anyLanguage`, the OID in the `policyLanguage` field MUST be present in `acceptable-pc-policy-language-set`.
- (c) The tuple containing the certificate subject name, `policyPolicy`, key usage extension (if present) and extended key usage extension (if present) must be appended to `proxy_policy_list`.
- (d) Process other certificate extensions, as described in [n2]:
- (1) Recognize and process any other critical extensions present in the proxy certificate.
  - (2) Process any recognized non-critical extension present in the proxy certificate.

If either step (a), (b) or (d) fails, the procedure terminates, returning a failure indication and an appropriate reason.

If `i` is not equal to `n`, continue by performing the preparatory steps listed in 4.1.4. If `i` is equal to `n`, perform the wrap-up steps listed in 4.1.5.

#### 4.1.4. Preparation for next Proxy Certificate

- (a) Verify `max_path_length` is greater than zero and decrement `max_path_length`.
- (b) Assign the certificate subject name to `working_issuer_name`.

- (c) Assign the certificate `subjectPublicKey` to `working_public_key`.
- (d) If the `subjectPublicKeyInfo` field of the certificate contains an `algorithm` field with non-null parameters, assign the parameters to the `working_public_key_parameters` variable.

If the `subjectPublicKeyInfo` field of the certificate contains an `algorithm` field with null parameters or parameters are omitted, compare the certificate `subjectPublicKey` algorithm to the `working_public_key_algorithm`. If the certificate `subjectPublicKey` algorithm and the `working_public_key_algorithm` are different, set the `working_public_key_parameters` to null.

- (e) Assign the certificate `subjectPublicKey` algorithm to the `working_public_key_algorithm` variable.
- (f) If a key usage extension is present, verify that the `digitalSignature` bit is set.

If either check (a) or (f) fails, the procedure terminates, returning a failure indication and an appropriate reason.

If (a) and (f) complete successfully, increment `i` and perform the basic certificate processing specified in 4.1.3.

#### 4.1.5. Wrap-up Procedures

- (a) Assign the certificate subject name to `working_issuer_name`.
- (b) Assign the certificate `subjectPublicKey` to `working_public_key`.
- (c) If the `subjectPublicKeyInfo` field of the certificate contains an `algorithm` field with non-null parameters, assign the parameters to the `proxy_issuer_public_key_parameters` variable.

If the `subjectPublicKeyInfo` field of the certificate contains an `algorithm` field with null parameters or parameters are omitted, compare the certificate `subjectPublicKey` algorithm to the `proxy_issuer_public_key_algorithm`. If the certificate `subjectPublicKey` algorithm and the `proxy_issuer_public_key_algorithm` are different, set the `proxy_issuer_public_key_parameters` to null.

- (d) Assign the certificate `subjectPublicKey` algorithm to the `proxy_issuer_public_key_algorithm` variable.

#### 4.1.6. Outputs

If path processing succeeds, the procedure terminates, returning a success indication together with final value of the `working_public_key`, the `working_public_key_algorithm`, the `working_public_key_parameters`, and the `proxy_policy_list`.

#### 4.2. Using the Path Validation Algorithm

Each Proxy Certificate contains a ProxyCertInfo extension, which always contains a policy language OID, and may also contain a policy OCTET STRING. These policies serve to indicate the desire of each issuer in the proxy certificate chain, starting with the EEC, to delegate some subset of their rights to the issued proxy certificate. This chain of policies is returned by the algorithm to the application.

The application MAY make authorization decisions based on the subject distinguished name of the proxy certificate or on one of the proxy certificates in it's issuing chain or on the EEC that serves as the root of the chain. If an application chooses to use the subject distinguished name of a proxy certificate in the issuing chain or the EEC it MUST use the returned policies to restrict the rights it grants to the proxy certificate. If the application does not know how to parse any policy in the policy chain it MUST not use, for the purposes of making authorization decisions, the subject distinguished name of any certificate in the chain prior to the certificate in which the unrecognized policy appears.

Application making authorization decisions based on the contents of the proxy certificate key usage or extended key usage extensions MUST examine the list of key usage, extended key usage and proxy policies resulting from proxy certificate path validation and determine the effective key usage functions of the proxy certificate as follows:

- \* If a certificate is a proxy certificate with a proxy policy of `id-ppl-independent` or an end entity certificate, the effective key usage functions of that certificate is as defined by the key usage and extended key usage extensions in that certificate. The key usage functionality of the issuer has no bearing on the effective key usage functionality.
- \* If a certificate is a proxy certificate with a policy other than `id-ppl-independent`, the effective key usage and extended key usage functionality of the proxy certificate is the intersection of the functionality of those extensions in the proxy certificate and the effective key usage functionality of the proxy issuer.

## 5. Commentary

This section provides non-normative commentary on Proxy Certificates.

### 5.1. Relationship to Attribute Certificates

An Attribute Certificate [i3] 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.

#### 5.1.1. 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, unrestricted 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. Additionally PC with policies could be used to selectively deny the binding of ACs to a particular proxy. An AC could also be bound to a particular PC using the subject or issuer and serial number of the proxy certificate. 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.

#### 5.1.2. Delegation Using Attribute Certificates

In the motivating example in Section 2, PCs are used to delegate Steve's identity to the various other jobs and entities 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 reliable file transfer service and all the hosts involved in file transfers, 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 minimal modification (e.g., possibly to find the EEC name and to check for any proxy policies).

- \* 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 the file transfer service, the hosts transferring files). 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 other factors.

#### 5.1.3. 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" [i3] 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."

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 for the purpose of delegation.

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.

#### 5.1.4. 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. An AC could be bound to a particular instance of a PC using the unique subject name of the PC, or its issuer and serial number combination.

Unrestricted PCs issued by that PC would then inherit those ACs and independent PCs would not. PCs issued with a policy would depend on the policy as to whether or not they inherit the issuing PC's ACs (and potentially which ACs they inherit).

While an AC can be bound to one PC by the AA, how can the AA restrict that PC from passing it on to a subsequently delegated PC? One possible solution would be to define an extension to attribute certificates that allows the attribute authority to state whether an issued AC is to apply only to the particular entity to which it is bound, or if it may apply to PCs issued by that entity.

One issue that an AA in this circumstance would need to be aware of is that the PI of the PC that the AA bound the AC to, could issue another PC with the same name as the original PC to a different

entity, effectively stealing the AC. This implies that an AA issuing an AC to a PC need to not only trust the entity holding the PC, but the entity holding the PC's issuer as well.

## 5.2. Kerberos 5 Tickets

The Kerberos Network Authentication Protocol (RFC 1510 [i6]) 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 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 proxying in a Kerberos based system, so can a PC can be delegated to allow for proxying 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 (Key Distribution Center), 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.

## 5.3. Examples of usage of Proxy Restrictions

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

#### 5.3.1. Example use of proxies without 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. He would inform each host of the unique subject name of the PC given to the other host. When the servers establish the data channel connection to each other, they use these delegated credentials to perform authentication and verify they are talking to the correct entity by checking the result of the authentication matches the name as provided by Steve.

#### 5.3.2. Example use of proxies with Restrictions

Steve wishes to delegate to a process the right to perform a transfer of a file from host H1 to host H2 on his behalf. Steve would delegate a PC to the process and he would use Proxy Policy to restrict 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. Note that these delegated PCs would inherit the restrictions of their parents, though this is not relevant to this example. As in the example in the previous Section, each host would be provided with the unique name of the PC given to the other server.

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 based on the restrictions in the PC that the process used to authenticate with them (in addition to any local policy they have). 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. When setting up the data channel the servers would again verify the names resulting from the authentication match the names provided by Steve as in the example in the previous Section.

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.

#### 5.4. Delegation Tracing

A relying party accepting a Proxy Certificate may have an interest in knowing which parties issued earlier Proxy Certificates in the certificate chain and to whom they delegated them. For example it may know that a particular service or resource is known to have been

compromised and if any part of a Proxy Certificate's chain was issued to the compromised service a relying party may wish to disregard the chain.

A delegation tracing mechanism was considered by the authors as additional information to be carried in the ProxyCertInfo extension. However at this time agreement has not been reached as to what this information should include so it was left out of this document, and will instead be considered in future revisions. The debate mainly centers on whether the tracing information should simply contain the identity of the issuer and receiver or it should also contain all the details of the delegated proxy and a signed statement from the receiver that the proxy was actually acceptable to it.

#### 5.4.1. 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). An 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.

## 6. Security Considerations

In this Section we discuss security considerations related to the use of Proxy Certificates.

### 6.1. Compromise of a Proxy Certificate

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 PC path validation checks, 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, and within the bound of the restriction policy carried by the PC. Therefore, one common way to limit the misuse of a compromised PC is to limit its validity period to no longer than is needed, and/or to include a restriction policy in the PC that limits the use of the (compromised) PC.

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.

## 6.2. Restricting Proxy Certificates

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. Once a key usage or extended key usage has been removed, the path validation algorithm ensures that it cannot be 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.

## 6.3. Relying Party Trust of Proxy Certificates

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. 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, etc.

The relying party should also be aware that since the policy restricting the rights of a PC is the intersection of the policy of all the PCs in its certificate chain, this means any change in the certificate chain can effect the policy of the PC. Since there is no mechanism in place to enforce unique subject names of PCs, if an issuer were to issue two PCs with identical names and keys, but different rights, this could allow the two PCs to be substituted for each other in path validation and effect the rights of a PC down the chain. Ultimately, this means the relying party places trust in the entities that are acting as Proxy Issuers in the chain to behave properly.

#### 6.4. Protecting Against Denial of Service with Key Generation

As discussed in Section 2.3, one of the motivations for Proxy Certificates is to allow for dynamic delegation between parties. This delegation potentially requires, by the party receiving the delegation, the generation of a new key pair which is a potentially computationally expensive operation. Care should be taken by such parties to prevent another entity from performing a denial of service attack by causing them to consume large amount of resource doing key generation.

A general guideline would always to perform authentication of the delegating party to prevent such attacks from being performed anonymously. Another guideline would be to maintain some state to detect and prevent such attacks.

#### 6.5. Use of Proxy Certificates with a Central Repository

As discussed in Section 2.7, one potential use of Proxy Certificates is to ease certificate management for end users by storing 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. and the repository would then delegate a PC to the user with proxy rights, but continue to protect the EEC private key in the repository.

Care must be taken with this approach since compromise of the repository will potentially give the attacker access to the long-term private keys stored in the repository. It is strongly suggested that some form of hardware module be used to store the long-term private keys, which will serve to help prevent their direct threat though it may still allow a successful attacker to use the keys while the repository is compromised to sign arbitrary objects (including Proxy Certificates).



## 7. IANA Considerations

IANA has established a registry for policy languages. Registration under IETF space is by IETF standards action as described in [i8]. Private policy languages should be under organizational OIDs; policy language authors are encouraged to list such languages in the IANA registry, along with a pointer to a specification.

OID	Description
---	-----
1.3.6.1.5.5.7.21.1	id-ppl-inheritALL
1.3.6.1.5.5.7.21.2	id-ppl-independent

## 8. References

### 8.1. Normative References

- [n1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [n2] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

### 8.2. Informative References

- [i1] Butler, R., Engert, D., Foster, I., Kesselman, C., and S. Tuecke, "A National-Scale Authentication Infrastructure", IEEE Computer, vol. 33, pp. 60-66, 2000.
- [i2] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
- [i3] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.
- [i4] Foster, I., Kesselman, C., Tsudik, G., and S. Tuecke, "A Security Architecture for Computational Grids", presented at Proceedings of the 5th ACM Conference on Computer and Communications Security, 1998.
- [i5] Foster, I., Kesselman, C., and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, 2001.
- [i6] Kohl, J. and C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

- [i7] Neuman, B. Clifford, "Proxy-Based Authorization and Accounting for Distributed Systems", In Proceedings of the 13th International Conference on Distributed Computing Systems, pages 283-291, May 1993.
- [i8] Narten, T. and H. Alvestrand. "Guidelines for Writing an IANA Considerations Section in RFC", RFC 2434, October 1998.

## 9. Acknowledgments

We are pleased to acknowledge significant contributions to this document by David Chadwick, Ian Foster, Jarek Gawor, Carl Kesselman, Sam Meder, Jim Schaad, and Frank Siebenlist.

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): Carlisle Adams, Joe Bester, Randy Butler, Keith Jackson, Steve Hanna, Russ Housley, Stephen Kent, Bill Johnston, Marty Humphrey, Sam Lang, Ellen McDermott, Clifford Neuman, Gene Tsudik.

We are also grateful to members of the Global Grid Forum (GGF) Grid Security Infrastructure working group (GSI-WG), and the Internet Engineering Task Force (IETF) Public-Key Infrastructure (X.509) working group (PKIX) for feedback on this document.

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.

## Appendix A. 1988 ASN.1 Module

```

PKIXproxy88 { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
    proxy-cert-extns(25) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

-- IMPORTS NONE --

-- PKIX specific OIDs

id-pkix OBJECT IDENTIFIER ::=
    { iso(1) identified-organization(3)
      dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

-- private certificate extensions
id-pe OBJECT IDENTIFIER ::= { id-pkix 1 }

-- Locally defined OIDs

-- The proxy certificate extension
id-pe-proxyCertInfo OBJECT IDENTIFIER ::= { id-pe 14 }

-- Proxy certificate policy languages
id-ppl OBJECT IDENTIFIER ::= { id-pkix 21 }

-- Proxy certificate policies languages defined in
id-ppl-anyLanguage OBJECT IDENTIFIER ::= { id-ppl 0 }
id-ppl-inheritAll OBJECT IDENTIFIER ::= { id-ppl 1 }
id-ppl-independent OBJECT IDENTIFIER ::= { id-ppl 2 }

-- The ProxyCertInfo Extension
ProxyCertInfoExtension ::= SEQUENCE {
    pCPathLenConstraint ProxyCertPathLengthConstraint
                        OPTIONAL,
    proxyPolicy ProxyPolicy }

ProxyCertPathLengthConstraint ::= INTEGER
ProxyPolicy ::= SEQUENCE {
    policyLanguage OBJECT IDENTIFIER,
    policy OCTET STRING OPTIONAL }

END

```

## Authors' Addresses

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

Von Welch  
National Center for Supercomputing Applications  
University of Illinois

EMail: vwelch@ncsa.uiuc.edu

Doug Engert  
Argonne National Laboratory

EMail: deengert@anl.gov

Laura Pearlman  
University of Southern California, Information Sciences Institute

EMail: laura@isi.edu

Mary Thompson  
Lawrence Berkeley National Laboratory

EMail: mrthompson@lbl.gov

## Full Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

